Report No. UIUCDCS-R-82-1081

# NOTESFILE REFERENCE MANUAL

by

Raymond B. Essick IV
Rob Kolstad

February 14, 1983
(Revised: October 20, 1985)
(Printed: May 27, 1992)

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
1304 W. SPRINGFIELD AVENUE
URBANA, ILLINOIS 61801-2987

Supported in part by NASA Project NAS-1-138

## 1 Introduction.

Notesfiles support computer managed discussion forums. Discussions can have many different purposes and scopes: the notesfile system has been designed to be flexible enough to handle differing requirements.

Each notesfile discusses a single topic. The depth of discussion within a notesfile is ideally held constant. While some users may require a general discussion of personal workstations, a different group may desire detailed discussions about the I/O bus structure of the WICAT 68000 (a particular workstation). These discussions might well be separated into two different notesfiles.

Each notesfile contains a list of logically independent notes (called base notes). A note is a block of text with a comment or question intended to be seen by members of the notesfile community. The note display shows the text, its creation time, its title, the notesfile's title, the author's name (some notesfiles allow anonymous notes), the number of ''responses'', and optionally a ''director message''. Each base note can have a number of ''responses'': replies, retorts, further comments, criticism, or related questions concerning the base note. Thus, a notesfile contains an ordered list of ordered lists. This arrangement has historically been more convenient than other proposals (e.g., trees were studied on the PLATO (trademark of Control Data Corporation) system).

The concept of a notesfile was originally implemented at the University of Illinois, Urbana-Champaign, on the PLATO system. The UNIX (trademark of Bell Laboratoris) notesfile system includes these ideas with adaptations and enhancements made possible by the UNIX environment.

The UNIX notesfile system was designed and implemented by Ray Essick at the University of Illinois, Urbana-Champaign. It provides users with the abilities to read notes and responses, write notes and responses, forward note text to other users (via mail) or other notesfiles, save note text in their own files, and sequence through a set of notesfiles seeing just new text. Each notesfile has a set of ''directors'' who manage the notesfile: they delete old notes, compress the file when needed, grant and restrict access to the notesfile, and set different notesfile parameters (e.g., title, ''director message'', policy note, whether notes' authors can be anonymous). Some notesfiles contain correspondence from other computers. Like the UNIX ''USENET'', notes and responses are exchanged (often over phone lines) with remote machines. The notesfile system provides automatic exchange and updating of notes in an arbitrarily connected network.

This document details the use of notesfiles from invocation through intersystem notes exchanges. The last chapter summarizes the entire set of commands for easy reference. An appendix contains detailed checklists for the installation of a notesfile system.

## 2 Using Notesfiles.

The notesfile system is invoked with a single command line. Most notesfile commands require only a single character (like the vi editor). Those that require more than one character are terminated by a carriage return.

### 2.1 Invocation.

Invoke the notesfile system with:

notes [ -sxi ] [-a subsequencer] [ -t termtype ] [ -f nfile ] [ topic1 ] [ topic2 ... ]

The topic list (e.g., topic1) specifies the notesfiles to read. Invoking the notes system with NO arguments yields a list of some available topics. When more than one topic is specified, the user encounters each topic sequentially (i.e., topic2 is entered upon completion of topic1).

The -s switch activates the ''notesfile sequencer'' which is discussed in section 2.8. Specify ''-x'' to use the extended sequencer. The ‘‘-i’’ flag selects yet another sequencing mode. The ‘‘-a’’ option specifies a particular subsequencer. This allows several users sharing a signon to maintain their own sequencing timestamp information.

The -t option directs the notesfile system to use ‘‘termtype’’ as the user’s terminal type, overriding the TERM shell variable.

The -f option directs the notesfile system to read the contents of the file ‘‘nfile’’ for a list of notesfiles to read. See section 2.3 (‘‘The -f Option’’) for more information on the format of this file.

## 2.2 Notesfile Names and Wildcards.

Notesfiles can be specified in several ways. The most common way is to merely give the name of the notesfile, such as ‘‘general’’. These notesfiles typically reside in the directory ‘‘/usr/spool/notes’’. Notesfiles may also be specified by their complete pathname; thus you could also refer to ‘‘general’’ by its full pathname ‘‘/usr/spool/notes/general’’. Using complete naming, notesfiles can be placed anywhere in the filesystem. This allows ‘‘private’’ notesfiles to be stored in personal directories.

The notesfile system supports pattern matching for names in the same manner as the shell. By using the shell meta-characters ‘‘*’’, ‘‘?’’, ‘‘[’’ and ‘‘]’’, the user can specify a number of notesfiles with a single entry. To read all the notesfiles that pertain to unix, enter the following line (the quotes are required to protect the metacharacters from interpretation by the shell):

notes ‘‘*unix*’’

There are several ways to read the notesfiles test1, test2, test3 and test4:

notes test1 test2 test3 test4
notes ‘‘test?’’
notes ‘‘test[1234]’’

Entries can also be eliminated from the list of notesfiles to look at. By prefixing a notesfile name (possibly containing wildcard characters) with a ‘!’, the notesfiles are excluded from the list to be examined. If one wished to look at all of the ‘‘test’’ notesfiles except test3, one could specify:

notes ‘‘test?’’ !test3

If you use the c shell, you will have to escape the ‘!’, the history character:

notes ‘‘test?’’ \!test3

These features are available from the normal entry (notes) and the automatic sequencer entry (see section 2.8). Most notesfile programs recognize this format. Among those which do not are programs which must receive exactly one notesfile name.

## 2.3 The -f Option.

The ‘‘-f’’ option of the notesfile system specifies a file of notesfile names to read. The file consists of lines containing notesfile names:

nfgripes
net.unix-wizards
net.general
fa.telecom

The names start at the left margin; they are indented here for readability. Wildcard characters (‘‘*’’, ‘‘?’’, ‘‘[’’, and ‘‘]’’) are acceptable in this context. Full names such as ‘‘/usr/spool/notes/general’’ are also accepted. Notesfiles can be eliminated through the ‘‘!’’ feature as described in section 2.2. The sequencer mode can be changed (see section 2.8) by inserting a line of the form:

```
-s
```

Again, this starts at the left margin. The ‘‘s’’ can be any of: ‘‘s’’, ‘‘x’’, ‘‘i’’, or ‘‘n’’. When a line of this form is read from the file, the sequencer mode is set to the corresponding mode: The normal ‘‘s’’equencer, the e‘‘x’’tended sequencer, the ‘‘i’’ndex sequencer, and ‘‘n’’o sequencer.

To always enter nfgripes, micronotes, and bicycle while only entering the networked notesfiles ‘‘net.*’’ when new notes are present, one might use ‘‘notes -f myfile’’ with this ‘‘myfile’’:

```
-x
nfgripes
micronotes
bicycle
-s
net.*
```

## 2.4 General.

Almost all notesfile commands consist of exactly one character (no carriage return). Only commands that are longer than one character require a terminating carriage return (currently, choosing a note to read is the only non-single character command).

The commands were chosen to be easy to remember. Upper case forms of commands usually function like their lower case counterparts but with some additional feature or power (i.e., ‘‘w’’ writes a response, ‘‘W’’ includes the current displayed text in the response).

Some commands are available almost everywhere in the notesfile system. These include those for help, exiting, forking a shell, and making a comment for the suggestion box.

### 2.4.1 Help.

Typing ‘‘?’’ anywhere will list the available options in an abbreviated format.

### 2.4.2 Exiting.

Type ‘‘q’’ (‘‘quit’’) to leave the current notesfile. Capital ‘‘Q’’ leaves the current notesfile and refrains from entering your last entry time into the sequencer table (see section ‘‘The Sequencer’’). The notesfile system proceeds to the next topic in the invocation list. The ‘‘k’’ and ‘‘K’’ keys function exactly as ‘‘q’’ and ‘‘Q’’.

Use control-D (‘‘signoff’’) to leave the notesfile system completely (without updating entry time information). The ‘‘z’’ command (which functions only when reading notes or responses or when on the index page) behaves similarly to control-D: the user exits the notesfile system immediately, but unlike control-D, updates the entry time information for the current notesfile.

### 2.4.3 Shells.

Fork a shell at any time by typing ‘‘!’’ (just like many other Unix programs).

### 2.4.4 Comments & Suggestions.

Type capital ‘‘B’’ (‘‘suggestion Box’’) while on the index page or reading notes to make a comment or suggestion about the notesfile program. Your suggestion will be stored in another notesfile reviewed frequently by the notesfile system manager.

## 2.5 The Index Page.

When the notes system is invoked without the -s option, the user sees an index of the most recent notes. A sample page is shown below:

| Workstation Discussion | | 2:03 pm Jan 4, 1982 | |
|---|---|---|---|
| 12/9/81 | 2 Stanford SUN | 4 | horton |
| | 3*WICAT 68000 | | kolstad |
| | 4 M68000 | 1 | horton |
| | 5 Dolphin | 3 | duke!johnson |
| 12/10 | 6 CDC Standalone | 1 | smith |
| | 8 IBM Personal Computer | | henry |
| | 9 Personal computers harmful? | 8 | Anonymous |
| | 10 Ethernet interfaces 3 mhz? | 23 | essick |
| | 11 Requirements for uiucdcs | 10 | botten |
| 1/1/82 | 12 Happy New Year! | 5 | mjk |

The upper left corner shows the notesfile’s title. In this example, the notesfile discusses personal workstations. The current time and date are displayed in the upper right corner. Approximately ten note titles are displayed (if available). More notes are displayed on longer screens (such as the Ann Arbor Ambassador). Each note is displayed with its date (if different from the previous date), note number, title, number of responses (if any), and author. The first note above was written by user ‘‘horton’’ on December 9th, is entitled ‘‘Stanford SUN’’ and has four responses. Note 7 has been deleted for some reason (by either its author or a notesfile director). Note 5 was written by user ‘‘johnson’’ whose signon resides on the ‘‘duke’’ system. Note 9 was written by an author who preferred to remain unidentified. Notes with director messages (sometimes denoting importance) are displayed with a ‘‘*’’ next to the note number (see note 3 above).

From the index page the user may:

- Scroll the index forward or backward.
- Read a note.
- Write a note.
- Go to the next unread note.
- Search for notes or responses after a specific date/time.
- Search for keywords within notes’ titles.
- Search for notes/responses by a specific author.
- Go to another notesfile.
- Consult the notesfile’s archive.
- Read the policy note.
- Check on anonymous and networked status.
- Register a complaint/suggestion about notesfiles.

- Fork a shell.
- Exit the notes program.
- Invoke notesfile director options (if the user is a director).

### 2.5.1 Scrolling the Index Page.

Scroll the index page by:

| | |
|---|---|
| +, <return>, <space> | forward one page |
| * | forward to the most recent page (* is multiple +'s) |
| - | backward one page |
| = | backward all the way (= is multiple -'s) |

### 2.5.2 Choosing Notes & Responses.

While on the index page, choose a note to read by typing its number followed by a carriage return. (This is the only command that requires a carriage return after it.) Usually the space bar is used to scan text. To skip to a particular note or response, use the features below.

While reading a note, ‘‘;’’ or ‘‘+’’ advances to the first response of the note. The next note is displayed if there are no responses. The number keys (‘‘1’’, ‘‘2’’, ... , ‘‘9’’) advance that many responses. If there are fewer responses, the last response is displayed. The return key skips the responses and goes to the next note. Press ‘‘-’’ or backspace to see the previous page of the current note; if the page currently displayed is the first, the notesfile program displays the first page of the previous note.

While a response is on the screen, the ‘‘;’’ and ‘‘+’’ keys display the next response. As with reading a note, if there are no further responses these keys advance to the next note. The number keys (‘‘1’’, ... , ‘‘9’’) will advance the appropriate number of responses. If there are fewer responses, the last response is displayed. The ‘‘-’’ or backspace keys display the previous page of the current response. If the current page is the first page of the response, these keys display the first page of the previous response. Enter ‘‘=’’ to see the base note of the current note string. Press the return key to proceed to the next note.

## 2.6 Notes & Responses.

### 2.6.1 Reading Notes.

After selecting a note from the index page (or entering the notesfile with your ‘‘sequencer’’ on), the note is displayed. A sample display is shown below:

```
Note 15            Workstation Discussion            2 responses
horton               WICAT 150          4:03 pm  Dec 11, 1981

Wicat System 150

8 MHz 68000, Mem. mgmt, Multibus architecture, 256k to 1.5 Mb RAM,16/32/64Kbyte EPROM,
10 ms interval timer, 2 RS232 (19.6k async, 56k sync), 16 bit parallel intelligent disk controller,
10 Mbyte winchester (5.25", 3600 rpm, access: 3 ms trk-trk, 70 avg, 150 max),
960Kb floppy (5.25", 300 rpm, access 10 ms trk-trk, 267 avg, 583 max)
Options: battery backed clock, graphics with touch panel, video disk control,
High Speed Serial Network Interface
Unix/V7 avail, Pascal, C, APL, ADA, Cobol, Fortran, Lisp, Basic, Asm
```

This is note number 15 in the ‘‘Workstation Discussion’’ file. User ‘‘horton’’ wrote this note at 4:03 pm on December 11th, 1981. Two responses have been written. The note’s title is ‘‘WICAT 150’’. If a director had written the note, the ‘‘director message’’ might have been displayed beneath the note’s title. Director’s notes sometimes contain important information or new policies.

Since notes and responses can each be up to 3 Mbytes long, the display routine breaks text into pages automatically. For all but the last page of a long note or response, the lower right corner of the display shows the percentage of the note that has been shown. For all but the first page of long text, the message ‘‘[Continued]’’ appears in the upper left portion of the display. Use the space bar to see the next page of a long note or response. When the last page is displayed, the space key functions as the ‘‘;’’ key: it proceeds to the next response. The ‘‘-’’ and backspace keys back up the display to the previous page. Only the first 50 pages of text are managed this way; typing ‘‘-’’ from the fifty-second page will return to the fiftieth page. The ‘‘=’’ key returns to the first page of the note.

While reading a note, it is possible to:

- Display the next, previous, or first page of the note.
- Write a response to the displayed note.
- Read next note or previous note.
- Read next unread response or note.
- Return to the index page.
- Skip to a given response.
- Delete the note (if you are its author or a file director).
- Edit the note’s title (if it is yours).
- Edit the note (if it is yours and there are no responses).
- Copy the note to another notesfile.
- Save the note in your file space.
- Mail the note to someone.
- Talk (‘‘write’’) to the author of the note.
- Search for keywords in note titles.
- Search for notes/responses by a particular author.
- Toggle the director message (if privileged).
- Fork a shell.
- Go to another notesfile.
- Make a comment or suggestion about notesfiles.
- Exit the notesfile program.

### 2.6.2 Reading Responses.

Response displays are similar to those of main notes with the exception that ‘‘Response x of y’’ replaces the note’s title. The first response to note 15 is shown below:

```
Note 15              Workstation Discussion
koehler                Response 1 of 2        11:53 pm  Dec 11, 1981

Does anyone have any insight about the relative speeds of the Winchester disks available
on these systems?  The previous disk seems to have track to track response times commensurate
with reasonably fast 8" floppies.  I wonder if some of the manufacturers are using disks that
will not meet reasonable specifications for response time for these kinds of applications.

On the other hand, with intelligent layout of file sectors, the I/O system
could romp and stomp on often used files...
          ====================================
```

The commands for manipulating the text of a long response are the same as those for looking at long notes.

Typing space will move to the next page. Typing ‘‘-’’ or backspace will display the previous page, within the same limitations as for reading notes (only 50 pages are kept). Press ‘‘=’’ to go back to the first page of the text.

The options available while reading responses include:

- Display the next, previous, or first page of the response.
- Go to a different response (usually the next one).
- Go to the next unread note/response.
- Reread the base note.
- Reread the previous note.
- Return to the index page.
- Copy the response to another notesfile.
- Mail the response to someone.
- Save the response in your file space.
- Talk to the response’s author.
- Write another response to the note.
- Search for keywords in note titles.
- Search for notes/responses by particular authors.
- Delete the response (if you are its author or a file director).
- Edit the response (if it is yours and there are no later responses).
- Fork a shell
- Go to another notesfile.
- Register a suggestion or complaint about the notesfile program.
- Exit the notesfile program.

### 2.6.3 Writing Notes & Responses.

Write new base notes by hitting ‘‘w’’ while reading the index page. The notesfile system will then invoke an editor ( ‘‘ed’’ by default; use either of the shell variables NFED or EDITOR to change it). After the prompt, compose the text you wish to enter, then write the text to the disk and leave the editor. The system will prompt you for various options if they are available: anonymity, director message status, and the note’s title.

To write a response to a note type ‘‘w’’ while that note or any of its responses is displayed. The same steps used to write a base note should then be followed.

### 2.6.4 Mailing Notesfile Text.

Both notes and responses can be mailed to other users (with optional appended text). The capital ‘‘M’’ (‘‘mail’’) command gives you the opportunity to edit the text then send it to anyone. Its inferior counterpart, ‘‘m’’, allows you to mail a message to anyone. To mail to the author of the text, use capital ‘‘P’’ (‘‘Personal comment’’) to send the text and your comments; use ‘‘p’’ for a simple letter.

To use a specific mail program, set the environment variable MAILER. If this is not set, a standard mail program is used.

### 2.6.5 Forwarding Text To Other Notesfiles.

There are several methods for forwarding text from one notesfile to another. Single notes or responses can be copied with the ‘‘c’’ or ‘‘C’’ command while entire note strings can be forwarded with the ‘‘f’’ and ‘‘F’’ commands.

The ‘‘f’’ (‘‘forward’’) command is given when a base note is displayed on the screen. When given, the ‘‘f’’ command causes the base note and all of its responses to be copied to another notesfile. The user is prompted for the destination notesfile. The copied note and all of the copied responses contain header information detailing their origin. Where ‘‘f’’ copies the note string without change, the ‘‘F’’ command allows the user to edit the text of the note and each response before inserting it into the target notesfile.

The ‘‘c’’ (‘‘copy’’) command prompts for a destination notesfile then copies the currently displayed note or response to the target notesfile. The user is allowed to choose between forwarding the note as a response or as a new base note. The ‘‘c’’ command does not give the user a chance to edit the text before inserting it in the new notesfile. The extended copying command ‘‘C’’ allows editing of the note text before it is copied to the other notesfile.

Both the ‘‘c’’ and ‘‘C’’ commands provide for the forwarded text to be entered as either a new note or as a response to an existing note. In the latter case, an index page is given to the user for choosing the appropriate note to which to respond.

#### 2.6.6 Saving Text in Local Files.

The ‘‘s’’ (‘‘save’’) command appends the current displayed text to a file of your choice (which is created if not present). Notesfiles prompts for the file name; typing only a carriage return aborts the command -- no text is saved. Capital ‘‘S’’ appends the base note and all its responses. The number of lines saved and the name of the file written are printed when the command completes.

#### 2.6.7 Deletion.

Capital ‘‘D’’ (‘‘delete’’) deletes a note or response if it is yours and has no subsequent responses. Notes already sent to the network can not be deleted by non-directors. Directors can delete any note or response with the ‘‘Z’’ (‘‘zap’’) command.

#### 2.6.8 Online Communication.

Typing ‘‘t’’ (‘‘talk’’) attempts to page the author of the current displayed text. The Unix ‘‘write’’ command to him/her is issued if the author is local and non-anonymous. If the environment variable WRITE is defined, the program it specifies is used to write to the author.

#### 2.6.9 Editing Note Titles.

While reading a base note, type ‘‘e’’ (‘‘edit’’) to change the note’s title (provided you are the author of the note or a notesfile director).

#### 2.6.10 Editing Notes/Responses.

‘‘E’’ allows editing of the text of a note or response. It is not permitted to edit an article if it has subsequent responses or if it has been sent to the network. If the ‘‘later responses’’ are deleted, it is possible to edit the original text.

### 2.7 Other Commands.

#### 2.7.1 Returning to the Index Page.

Type ‘‘i’’ (‘‘index’’) while reading notes or responses to return to the index page.

#### 2.7.2 Searching Titles for Keywords.

While reading, you can search backwards for keywords appearing in note titles. Typing ‘‘x’’ (‘‘x is the unknown title’’) prompts for the substring to be found. Searching begins at the current note (or from the last note shown on the index page) and proceeds towards note 1. The search is insensitive to upper/lowercase distinctions. Use upper case ‘‘X’’ to continue the search. The search can be aborted by hitting the RUBOUT (or DELETE) key.

### 2.7.3 Searching for Authors.

The ‘‘a’’ command searches backwards for notes or responses written by a specific author. Notesfiles prompts for the author’s name. The ‘‘A’’ command continues the search backwards. The author name may be preceded by an optional ‘system!’. Abort the search by hitting the RUBOUT (or DELETE) key.

The entire name need not be specified when searching for articles by a particular author. Author searching uses substring searching. Searching for the author ‘‘john’’ will yield articles written by a local user ‘‘john’’, a remote user ‘‘somewhere!johnston’’, and any articles from the ‘‘uiucjohnny’’ machine. Author searching is case sensitive.

### 2.7.4 Stacking Notesfiles.

Sometimes it is useful to be able to glance at another notesfile while reading notes. Using ‘‘n’’, the user can save (stack) his current place and peruse another notesfile.

When on the index page or while reading notes/responses, type ‘‘n’’ (‘‘nest’’) to read another notesfile. Notesfiles prompts for the notesfile to read. If the notesfile exists, the place is marked in the old notesfile and the new one’s index is displayed.

Type any of the standard keys to leave the nested notesfile. Both ‘‘q’’ and ‘‘Q’’ leave the nested notesfile and return to the previously stacked notesfile. Control-d (‘‘signoff’’) causes the notesfile program to exit regardless of the depth of nesting.

Sequencing is turned off in the new notesfile regardless of its state in the old notesfile. The depth of the stack of notesfiles is limited only by the amount of memory available to the user.

### 2.7.5 Accessing Archives.

As notesfiles grow, it becomes impractical to keep every discussion. In some cases, the old discussions are deleted; other cases require these old discussions to be saved somewhere. Each active notesfile can have an archive notesfile. An archive notesfile contains the old discussions from the active notesfile.

The archive of an active notesfile is accessed by explicitly naming the notesfile (/usr/spool/oldnotes/micronotes for example) or through the ‘‘N’’ command from the active notesfile.

### 2.7.6 Policy Note.

A notesfile director can write an optional policy note to describe the purpose of a notesfile. Read the policy note by typing ‘‘p’’ (‘‘policy’’) from the index page.

## 2.8 The Sequencer.

Most users prefer to scan notesfiles and see only those notes written since their last reading. The notesfile ‘‘sequencer’’ provides this capability. It is activated by the ‘‘-s’’ option (‘‘sequencer’’) on the command line. When the sequencer is activated, the notesfile system automatically remembers the last time the user read notes in each notesfile. Subsequent entries to the notesfile can use the ‘‘last time’’ information to show only new notes and responses. If there is nothing new in a notesfile, the sequencer proceeds to the next notesfile specified in the command line.

The normal sequencer does not give the user a chance to read the notesfile if there are no new notes or responses; sometimes it is desirable to be able to do so. Use the ‘‘-x’’ option to enable the sequencer and enter the notesfile even if there are no new notes.

No keys need be pressed if there are no new notes in the entire list and the normal (‘‘-s’’) sequencer mode is selected. With the extended (‘‘-x’’) sequencer, the user must type ‘‘q’’, ‘‘Q’’, or control-d for each notesfile regardless of whether there are new notes.

The ‘‘-i’’ mode of sequencing is similar to the ‘‘-s’’ mode. Using the ‘‘-i’’ mode, notesfiles without new entries are passed over. The user starts reading on the index page of notesfiles which contain new notes.

### 2.8.1 Seeing New Notes and Responses.

The sequencer always shows the base note of a modified note string, whether or not is has been shown before, in order to establish the context of the new response(s). The ‘‘j’’ command skips to the next modified text (note or response).

If the rest of a particular note string seems uninteresting, skip to the next modified note string with the ‘‘J’’ (‘‘big Jump’’) command. This skips any new responses on the current note string. It is common to follow closely only a few note strings, skipping others using the ‘‘J’’ command.

The ‘‘last time’’ information is kept in a special file for each user. When the sequencer is enabled, the time for the notesfile is loaded into a variable and used to specify which notes and responses are new. If the sequencer is not enabled, this variable is initialized to January 1, 1970. The ‘‘j’’ and ‘‘J’’ keys use this variable to determine which notes and responses are ‘‘new’’.

If the sequencer is enabled, after exiting a notesfile the ‘‘last time’’ information is updated to the time that the user entered this notesfile. The entry time is used rather than the exit time to ensure that all notes are seen, including ones written during the just completed session. If the sequencer is disabled, the ‘‘last time’’ information is not modified. The ‘‘last time’’ information for a particular notesfile is updated as that notesfile is exited; using ‘‘Q’’ or control-D later will have no effect on the sequencer information for notesfiles already read.

The ‘‘o’’ and ‘‘O’’ commands allow the user to modify the variable used to determine whether notes and responses are ‘‘new’’. The ‘‘o’’ command allows the user to set this variable to any date he wishes. Use the ‘‘O’’ command to set this variable to show only notes and responses written that day. The ‘‘last time’’ file kept for each user is never modified by the ‘‘o’’ and ‘‘O’’ commands.

When no more new notes or responses exist, both the ‘‘j’’ and ‘‘J’’ commands will take the user to the index page. To exit the notesfile, use the ‘‘q’’ command. Exiting with ‘‘q’’ will update the user’s ‘‘last entry’’ time. Exiting with capital ‘‘Q’’ will NOT modify the ‘‘last entry’’ time for that notesfile (neither will control-D).

The ‘‘l’’ and ‘‘L’’ command behave similarly to ‘‘j’’ and ‘‘J’’. The difference is that while ‘‘j’’ and ’’J’ take the user to the last index page when no more new notes or responses exist, the ‘‘l’’ and ‘‘L’’ commands will leave the notesfile as if a ‘‘q’’ had been typed. Thus when no more new notes exist, the ‘‘l’’ command is like typing ‘‘jq’’.

### 2.8.2 Alternate Sequencers.

If several people share a login account, it is convenient for each to have a set of sequencing timestamps. This is accomplished through the use of the subsequencer option of notesfiles.

Specifying the -a option and a subsequencer name causes notes to use a different sequencing timestamp file. Many different subsequencer names can be used with each login account.

The main sequencer file for a given account is distinct from each of its subsequencer files. Each of the subsequencer files is normally distinct. If the subsequencer names are not unique in their first 6 characters, subsequencer files may collide.

### 2.8.3 Automatic Sequencing.

An alternate entry to the notes program allows the user to invoke notes with the sequencer enabled and a list of notesfiles to be scanned with a single, simple command. The ‘‘autoseq’’ command is invoked by typing

autoseq

and reads the environment variable ‘‘NFSEQ’’ to find the names of all notesfiles to be scanned. On some systems, the ‘‘autoseq’’ command may be known as ‘‘readnotes’’, ‘‘autonotes’’ or some similar variant; substitute the appropriate name in the following paragraphs. The ‘‘NFSEQ’’ variable should be defined in .profile for Bourne shell users as follows:

NFSEQ=‘‘pbnotes,micronotes,helpnotes,works’’
export NFSEQ

For users of the C shell, the following line should be added to the .login file:

setenv NFSEQ ‘‘pbnotes,micronotes,helpnotes,works’’

With NFSEQ assigned this value, a call to autoseq will process the notesfiles ‘‘pbnotes’’, ‘‘micronotes’’, ‘‘helpnotes’’, and ‘‘works’’ with the sequencer turned on.

The full naming conventions, pattern matching capabilities, and ‘!’ exclusion described in section 2.2 (‘‘Notesfile Names and Wildcards’’) are available in autoseq. To read all notesfiles with ‘‘unix’’ in their names, and the four test notesfiles (‘‘test1’’ though ‘‘test4’’), the NFSEQ variable might be defined as:

NFSEQ=‘‘*unix*,test[1234]’’

If the first character of an entry in the NFSEQ list is ‘‘:’’, the notesfile system reads the file name following for a list of notesfiles. To have the automatic sequencer read the file ‘‘/usr/essick/.nfseq’’ for a list of notesfiles to scan, define NFSEQ as:

NFSEQ=‘‘:/usr/essick/.nfseq’’

For this feature to work, the file must have group read privileges. The notesfile program runs ‘‘set-uid’’ and can not read files which are readable only by the owner.

The following definitions are also valid. The first one reads the notesfiles specified in the file ‘‘/usr/essick/.nfseq’’ and then reads the notesfiles pbnotes and micronotes. The second definition will read the notesfile pbnotes, those specified in ‘‘/usr/essick/.nfseq’’, micronotes and the ones specified in ‘‘/usr/essick/.other’’. If the notesfile program is unable to read the file specified, it skips to the next entry. For a description of the format of these files, see the section 2.3, ‘‘The -f Option’’.

NFSEQ=‘‘:/usr/essick/.nfseq,pbnotes,micronotes’’

NFSEQ=‘‘pbnotes,:/usr/essick/.nfseq,micronotes,:/usr/essick/.other’’

The automatic sequencer uses the ‘‘-s’’ mode of sequencing. The user does not enter notesfiles which have no new text. By specifying ‘‘-x’’ or ‘‘-i’’ on the command line, the user can use the appropriate sequencer mode.

The subsequencer option of notes is available from the autoseq program by specifying ‘‘-a name’’ on the command line, and has identical semantics with use of this option when invoking notes.

### 2.9 Environment Variables.

The notesfile program reads several environment variables to tailor the system to the user's preferences. Below is a list of the variables, their purpose, and their default values. These defaults are for UNIX 4.xBSD and may be slightly different for other versions of UNIX.

- ''NFED'' specifies which editor will be invoked when the user writes a note or response. If this variable is not specified, the notesfile system looks for the environment variable ''EDITOR'' (which many other programs use). If neither ''NFED'' nor ''EDITOR'' are defined, a default editor is used (/bin/ed).
- ''NFSEQ'' is a list of notesfiles that the user wishes to scan using the automatic sequencing entry to notesfiles. The use of this variable is described in the section on sequencing. If unspecified, the system uses a standard set which usually includes ''general'' and ''net.general''.
- ''PAGER'' is the paging program (''more'', ''pg'') which is used for scrolling the help files. The default paging program is /usr/ucb/more.
- ''MAILER'' determines the mail program to use. This defaults to /usr/ucb/mail.
- ''WRITE'' is used to specify the program for communication between users. If undefined, the Unix program ''write'' is used.
- ''TERM'' determines the type of terminal in use. This must be set for notes to know what screen handling conventions to use. In most cases the value will be correctly initialized by the system at login time.
- ''SHELL'' specifies which shell the user is running. This will almost always be set by the operating system.

## 3 Managing Notesfiles.

The notesfile system is installed by a user who is known as the ''owner'' of the notesfiles (UIUCDCS uses user ''notes''). This user can create, delete, rename, and initiate networking of notesfiles. Each notesfile is assigned a set of ''directors'' (who may or may not be associated with owner of notesfiles). The directors have special privileges for managing the notesfile (see below). The ''owner'' rarely manages the day to day aspects of a notesfile, although he has director, read, write and response privileges to all notesfiles for handling emergencies and failures.

### 3.1 Director Options.

The director can:

- Change the access permissions.
- Write the policy note.
- Change the notesfile title and director message.
- Open or close the notesfile.
- Allow the notesfile to be networked.
- Permit or restrict anonymous notes.
- Compress the notesfile.
- Change the notesfile's archival parameters.
- Delete notes and responses.
- Toggle director message on any note or response.

The director can delete notes or toggle the director message above them while reading the notes. Access other options by typing ''d'' on the index page. A display like this results:

```
                     Workstation Discussion
                 *** Your Director Message Here ***

(a) Anonymous:   ON                        Policy Note Exists: YES
(o) Notesfile:   OPEN                      Next note in slot: 1
(n) Networked:   YES                       Deleted Notes (holes): 0
(A) Is Archive:  NO                        Deleted Responses (holes): 0
(e) Expiration Threshold: Default
(E) Expiration Action:    ARCHIVE
(D) Archive with Dirmsg:  NOCARE
(W) Working Set Size:     Default
(l) Maximum Text/Article: 65000 bytes




Option:
```

= = = = = = = = = = = = = = = = = =

Options available on this page include: access lists, policy note writing, title and director messages, open/close notesfile, network enabling, anonymous notes, notesfile compress, and delete a list of notes.

**3.1.1 Access Lists.**

The notesfile system allows directors to allow or restrict access to each notesfile. The access list can allow or deny read, write, respond, and director options to any user, group, or system. Type ‘‘p’’ (‘‘permissions’’) on the director options page to enter the access list editor. The system prompts for an option: ‘‘m’’ to modify an extant entry, ‘‘d’’ to delete an entry, ‘‘i’’ to insert a new entry, ‘‘r’’ to replot the list, ‘‘q’’ to quit editing the access list, and capital ‘‘Q’’ to quit editing the access list and IGNORE ANY CHANGES MADE. Delete or modify entries by entry number. Scroll the entries using ‘‘+’’ and ‘‘-’’.

After typing ‘‘i’’ to insert a new entry, the system prompts for a user type (‘‘u’’ for user, ‘‘g’’ for group, ‘‘s’’ for system). The system then prompts for the name of the user, group, or system. (Users and groups must be valid names) The default access options are then displayed: read, write, answer (for responses). Use the keys ‘‘r’’, ‘‘w’’, ‘‘a’’, and ‘‘d’’ to toggle the read, write, answer, and director privileges respectively. Some options automatically enable others (e.g., ‘‘w’’ for writing automatically enables ‘‘a’’ for answering). It is not possible to remove answer access while write access is enabled. The ‘‘n’’ key will remove all privileges (‘‘no access’’). Type return (or ‘‘q’’) when the correct options have been entered. The system prompts for another user. Press return at the prompt to exercise other access list options.

The access machinery checks user names before checking group names. If user ‘‘john’’ explicitly has no access but his group does, he will nevertheless be denied access to the file. If there is no explicit entry for user ‘‘john’’, a check is made for permissions granted to his group(s). (n.b.: an entry for user ‘‘Other’’ will match all users, circumventing group permissions. The behavior typically desired can be achieved with a group ‘‘Other’’ just as well.)

If no explicit user entry exists, a search for group permissions is initiated. Users can belong to more than one group (although kernels such as Version 7 only allow one at a time) and the notesfile code checks each of the user’s groups. If permissions for several of these groups exist, the user is given the inclusive OR of the several permissions. If none of the user’s groups are given permission, a default permission specified by group ‘‘Other’’ is usually assigned. The ‘‘Other’’ entry matches when none of the other group entries have matched. This entry can be deleted, in which case no access is granted.

The current implementation of system access enforcement is naive. The network software will send to a system only if it has read permission. Reception allows intermediaries to pass on notes even if they are not allowed write access to the notesfile; the access permission is determined from the originating system of each note or response instead of the site actually delivering the article. The name ‘‘Other’’ (capital ‘‘O’’) matches any system name not mentioned explicitly.

Many notesfiles allow several users and groups to have read/write access, a single user to have director access (in addition to the notesfile ‘‘owner’’), and all other users no access.

When a notesfile is first created, a default access list is created. The notesfile ‘‘owner’’ is made director, group ‘‘Other’’ and system ‘‘Other’’ are both given read/write access to the notesfile. If the file ‘‘/usr/spool/notes/.utilities/access-template’’ exists, it contains a list of access-rights to add to the new notesfile’s access list. The file contains lines of access-rights in the format used by the nfaccess program. Access-rights look like ‘‘user:essick=drwa’’; for more information on the format of these entries, see the man(I) page for nfaccess.

#### 3.1.2 Policy Note.

Type ‘‘w’’ (‘‘write policy’’) on the director option page to write a policy note (just like writing any other note).

#### 3.1.3 Title & Director Messages.

Change the notesfile title with ‘‘t’’, the director message with ‘‘m’’. The system prompts for a new message. Typing only a carriage return will not change the old message.

#### 3.1.4 Open/Close.

Type ‘‘o’’ (‘‘open’’) to toggle the availability of the notesfile (subject to the access list). Closed notesfiles are unavailable to non-directors.

#### 3.1.5 Network Options.

Type ‘‘n’’ (‘‘network’’) to toggle the availability of the notesfile for networking. Arrangements must be made with the notesfile system ‘‘owner’’ to do the network transfers.

#### 3.1.6 Anonymous Notes.

Type ‘‘a’’ (‘‘anonymous’’) to toggle the availability of anonymous notes in the notesfile. The availability of the anonymous option may provoke slanderous attacks from users (whose anonymity is completely protected).

#### 3.1.7 Compression.

Type ‘‘c’’ (‘‘compress’’) to compress the notesfile. As notes are deleted, their text and index space is not reclaimed. This command reclaims the space. The notesfile must be closed. On a VAX 11/780, 20 seconds of real time (on a slightly loaded system) is required to reclaim the space of a notesfile with 50 remaining notes (compression time is dependent on remaining notes). Notesfiles should be compressed whenever many of their notes have been deleted. The notesfile archiver ‘‘nfarchive’’ and cron(8) can be used to automate this process.

The director’s option page displays a count of ‘‘holes’’ left by deleted notes and responses. This can be used as an indication of how much wasted space is within the notesfile.

**3.1.8 Expiration Threshold.**

The ‘e’ command allows a notesfile director to modify the notesfile’s expiration threshold. Possible values include specific numbers of days, ‘default’ and ‘never’. The value can be left unchanged by not specifying a new value. The ‘default’ value is assigned to new notesfiles; directors can change it as needed.

The notesfile archiving program (nfarchive) examines the expiration threshold of each notesfile it processes. This threshold determines how long a note string must be inactive before it is eligible for archival. The ‘Default’ expiration threshold uses the expiration time specified on the ‘nfarchive’ command line; this is usually 2 weeks. Specific ages can be specified. The age specified in the notesfile overrides the value on the ‘nfarchive’ command line. The ‘Never’ threshold tells ‘nfarchive’ that this notesfile is not to be archived.

**3.1.9 Working Set Size.**

Each notesfile contains a working set of notes. The working set is the number of notes left in the notesfile by the nfarchive program. When nfarchive runs, it determines a maximum number of notes to delete. This number is the number of notes written in the notesfile minus the number of ‘‘holes’’ caused by deletions minus the working set size. Nfarchive will leave a ‘‘working set size’’ of notes in the notesfile; if fewer notes existed in the notesfile, no notes are archived.

The working set size can be changed by the ‘s’ command from the director page. Possible values include ‘‘default’’ and specific numbers. ‘‘Default’’ specifies that the value supplied during the nfarchive run is to be used; explicit values in the notesfile always override values specified on the nfarchive command line.

**3.1.10 Expiration Action.**

Each notesfile can decide on the destination of expired notestrings. The expiration action field takes one of the values ‘‘Default’’, ‘‘Archive’’ or ‘‘Delete’’. Archive and delete specify that expired notes are to be archived and deleted respectively. The default entry specifies that the expiration action should follow that specified on the nfarchive command line.

**3.1.11 Expire With Director Message.**

Notesfiles can decide how to expire based on director message status individually. This option can assume four values: ‘‘Default’’, ‘‘Nocare’’, ‘‘On’’, and ‘‘Off’’. The on and off values specify that only notes with the director message on or off respectively are eligible for expiration. The nocare value specifies that the director message status is not checked; both director and non-director marked notes are eligible for expiration. Select the default entry to use the value of this parameter as specified on the nfarchive command line.

**3.1.12 Maximum Text per Article.**

The notesfile system imposes limits on the size of each article. Earlier versions restricted articles to 64 kbytes; the current version provides for articles up to 4 Gigabytes. A constant is used to determine the actual maximum allowed per article.

Each notesfile can select a maximum text length per article. This limit is not allowed to exceed the hard-coded limit (currently 3 Mbytes). Articles exceeding this limit are truncated and a message detailing the count of excess bytes and the system responsible for truncating the text is appended.

Initially the maximum text length is set to the highest permissible value. One reason for lowering the limit is to meet restrictions on the size of network transfers.

#### 3.1.13 Deleting and Un-Deleting Many Articles.

Type ‘‘z’’ (‘‘zap’’) to delete many notes (and their responses) quickly. Enter a list of note numbers or note ranges (aa-bb) separated by spaces. Confirm the command with ‘‘y’’; other responses will abort the command. It is economical and prudent to compress the notesfile shortly thereafter. Note that deleting notes in a networked notesfile makes those notes unavailable to those who poll your system for new notes and responses.

The ‘‘u’’ (undelete) command performs the opposite function of the ‘‘z’’ command. This command allows you to specify a list of note strings to be un-deleted. When prompted, the director shoul supply the note numbers he wishes to re-activate. The specified notes are re-activated and can be viewed as before. This command is only effective until a compression of the notesfile; after that time the notes are no longer present in the notesfile.

#### 3.1.14 Director Options for Notes.

Directors may put a ‘‘director message’’ above any note they write. There is one single line director message for each notesfile. Typical director messages are: ‘‘New Policy’’, ‘‘*** This problem fixed or ignored ***’’, or ‘‘-- Eat Flaming Death Fascist Pigs --’’. Directors can also toggle the director message on a note being read (‘‘d’’ for ‘‘director message’’). A director can delete a note (and all its responses) or any response while reading the text of the note or response by typing ‘‘Z’’ (‘‘zap this one’’) and confirming with ‘‘y’’.

#### 3.1.15 Default Sequencer Lists.

Some users never set up an ‘‘NFSEQ’’ environment variable specifying the notesfile they wish to see. The file ‘‘/usr/spool/notes/.utilities/Dflt-Seq’’ contains a default list of notesfiles. Users without an ‘‘NFSEQ’’ variable receive the notesfiles listed in this file. The file can be changed at anytime and will take effect with the next ‘‘autoseq’’ by a user.

### 3.2 Creation & Deletion of Notesfiles.

Only the ‘‘owner’’ of the notesfile system can create notesfiles. Create notesfiles with the mknf command:

mknf [ -aon ] topic1 [ ... ]

The created notesfiles have default status of closed, non-networked, and no anonymous notes permitted. Specify -a to permit anonymous notes in the new notesfiles. Use -o to have the notesfiles marked open for general use and the -n option to enable the notesfiles’ network availability. These status flags can all be modified from the directors page at later times.

Delete notesfiles with rmnf:

rmnf [ -f ] topic1 [ ... ]

Each notesfile to be removed must be verified with ‘‘y’’ after a prompt -- anything else will leave that notesfile intact. Use the -f option to blindly remove notesfiles; the verification step is bypassed when ‘‘rmnf’’ is invoked with the -f option.

The file /usr/spool/notes/.utilities/avail.notes contains a list of the public notesfiles. The notesfile owner should update this file when he creates new notesfiles; this file is not automatically updated by ‘‘mknf’’ and ‘‘rmnf’’. The contents and format of the file are at the discretion of the notesfile system owner.

### 3.3 Intersystem Notesfiles.

The notesfile system provides for intersystem notesfiles in an arbitrary connected network. Copies of a shared notesfile must exist on each of the systems wishing to read notes for that notesfile. The contents are kept in synchrony through occasional exchanges over a network (UIUCDCS uses both uucp and TCP/IP). Notesfiles to be shared must have their ‘‘network status’’ enabled (see director options).

Duplication of notes and responses is prevented by the use of unique identifiers. Each note and response in a notesfile is assigned a unique number. The networking software checks each note as it arrives to see if a copy already exists. In the event of duplication, the extra copy is discarded and the fact is logged in the statistics and the network log.

In the (hopefully rare) event that a response arrives at a system before the base note does, the network reception program will generate a ‘‘foster parent’’ for the orphaned response. When the true parent arrives, the foster parent will be overwritten. A count of orphaned responses received is kept and available through use of the nfstats program (see section 4.4).

#### 3.3.1 Transmitting Notesfile Updates.

The nfxmit program gathers the new notes and responses in specified notesfiles and sends them to a specified site. The notesfile ‘‘owner’’ must occasionally enter the following command (or have it entered for him by cron) to update remote notesfiles with new notes and receive new remote notes:

nfxmit -dsitename [-t datespec] [-r] [-a] [-f file] topic1 [...]

The ‘‘sitename’’ is the name of the remote site to receive the new notes. The remote site should have notesfiles matching those specified by the topic1 parameter. For remote notesfiles with different names, see the section below on Name Mapping.

The optional -t specifies that all notes and responses since that date should be sent (normally -t is omitted and the notesfile system sends only new notes and responses).

The -r option specifies that the remote notes system should not only receive the current changes but also reply in kind. This is useful if the remote system does not automatically run the nfxmit program.

The -a option specifies that articles inserted from the news(I) system are to be sent also. Normally these articles are not sent because the receiver probably has them; the primary use of this switch is for sites that do not run news(I).

Using the -f switch tells nfxmit to read the file specified for a list of notesfiles to transmit; multiple -f parameters are permitted and can be freely intermixed with ‘topic’ parameters. Notesfile name pattern matching is performed on both ‘topic’ parameters and the entries in a file specified by the -f option.

Nfxmit uses uux(1) as a transport and remote execution mechanism. Connections using different protocols and mechanisms can be selected in the file ‘‘/usr/spool/notes/.utilities/net.how’’; its format is described in the section ‘‘Non-Standard Links’’. Uux typically permits a limited set of commands to be executed remotely. The file /usr/lib/uucp/L.cmds contains a list of acceptable commands; this file should be edited to include the ‘‘nfrcv’’ program.

#### 3.3.2 Network Transaction Log.

The network software maintains a log of all transactions, including time, date, number of notes and responses transferred, direction of transfer, and number of notes replicated by transfer. This log is placed in a file called ‘net.log’ and resides in the notesfile utility directory (by default: /usr/spool/notes/.utilities).

This file will grow without bounds. Occasional pruning is a good idea. There is no vital information stored in this file; its purpose is to provide a network audit trail.

### 3.3.3 Non-Standard Links.

Some systems will be unable to keep the notesfile network software in a public directory (such as /usr/bin). Other sites will have non-uucp links. The ‘net.how’ file is for these cases. ‘Net.how’ is kept in the notesfile utility directory (/usr/spool/notes/.utilities) and contains information on linking to remote systems. Entries in the file are made for systems with non-standard links and have the following format:

```
system:direction:protocol::command string
```

The system field contains the name of the remote system. The direction field contains either an ‘x’ or ‘r’ (no quotes) and specifies the direction that the line is for. An ‘x’ specifies that the command string is for sending notes to the remote site; an ‘r’ specifies that the command string is used in coercing the remote system to send its new notes and responses back. Lines beginning with a ‘#’ are comment lines and ignored by the notesfile code.

The protocol field is either empty or contains an integer value. An empty field indicates protocol 0. Currently only protocols 0 and 1 are supported. The notesfile receiving programs automatically switch between protocols.

The command string is a printf control string (without quotes) with two ‘%s’ entries. The first is for filling in the name of the notesfile, the second is for the local system name. Many entries in the ‘net.how’ file will be to place different paths on the ‘nfrcv’ and ‘nfxmit’ commands. The default command line is:

```
uux -z - system\!nfrcv %s %s
```

for the ‘x’ entry and for the ‘r’ entry:

```
uux system\!nfxmit %s -d%s
```

In the following sample from our net.how file, the host ‘‘uicsovax’’ is connected via UUCP and the notesfile networking programs live in non-standard directories. The host ‘‘etherhost’’ is reachable over a local network and the Berkeley ‘‘rsh’’ commands can be used to ship data between the local host and ‘‘etherhost’’.

```
uicsovax:x:::uux - uicsovax!/mnt/dcs/essick/.commands/nfrcv %s %s
uicsovax:r:::uux uicsovax!/mnt/dcs/essick/.commands/nfxmit %s -d%s
etherhost:x:::rsh etherhost /usr/bin/nfrcv %s %s
etherhost:r:::rsh etherhost /usr/bin/nfxmit %s -d%s
```

### 3.3.4 Notesfile Name Mapping.

To provide flexibility in the naming of notesfile across systems, the network software looks in the directory /usr/spool/notes/.utilities/net.aliases for mappings of local notesfile names to remote notesfile names. Each file in the directory is named after a system (e.g., pur-ee or uicsovax). Each of these files contains lines which specify the mapping of local notesfiles to the particular systems notesfiles. Lines beginning with ‘#’ in these files are comment lines and are ignored in the matching process. Data lines in the files look like:

```
local_nf:remote_nf
```

If there is no entry for a particular notesfile or the file for that system is missing, the local name is used.

Mapping is performed by the transmission program ‘‘nfxmit’’. The ‘‘nfrcv’’ program does not consult this table.

### 3.4 Initial Installation & Parameters.

Installation of the notesfile system requires the following:

- A working familiarity with version 7 UNIX (or later) (and UUCP for intersystem transfers).
- The notesfile distribution tape. (The distribution is already in /usr/src/new/notes, the user-contributed software area, in 4.3BSD).
- A signon for the notesfile owner. This signon should be in its own group. The notes package runs set-gid; mixing the notes group with other groups is not recommended.
- An ‘‘anonymous’’ signon. This signon should be an unused user-id. The notesfile system prohibits this user-id from reading and writing notes.
- The ability to write into the directories where the notesfile binaries and data base are to live.

Appendix A (‘‘Notesfile Installation Checklist’’) is a helpful guide for installing the notesfile system. The distribution is made from a VAX running 4.2 Bsd. In most cases there are only a few files that need to be modified: ‘‘src/parms.h’’, ‘‘src/Makefile’’, and ‘‘src/newsgate.h’’ (if you are interfacing to USENET).

First: read in the distribution tape. The tape is a TAR format tape. This will read in several directories worth of data. The ‘‘src’’ directory contains all source code for the notesfile system and the internal help files. The ‘‘doc’’ directory contains an nroff source for the user manual and the macros that were used to generate it. The ‘‘man’’ directory contains the nroff sources for the notesfile manual pages. The ‘‘Samples’’ directory is a collection of various system files from our machine; they are included to provide examples in setting the matching files on your system.

After the tape has been read in, select the appropriate parameters.

#### 3.4.1 Selecting Appropriate Constants.

Before compiling the notesfile system, it must be configured to match your system. The location of spooling directories and command directories, the UNIX kernel, and policies on ‘‘non-standard’’ software may require changes in the default configuration.

Configuring the notesfile system is accomplished through modifying two files, both in the ‘‘src’’ directory of the distribution. The ‘‘parms.h’’ file contains information on values for the default archival time, UNIX kernel type, and the behavior of the notesfile system such as whether to keep core images when the notesfile system detects an internal error. The ‘‘Makefile’’ file contains definitions of the spooling and command directories, the notesfile ‘‘owner’’, and the ‘‘anonymous’’ user.

The distribution is configured for a UNIX 4.2 BSD system. The most frequently changed values, after the kernel definition, are the notesfile ‘‘owner’’ and the location of spooling and command directories.

Appendix A (‘‘Notesfile Installation Checklist’’) details each of the options in ‘‘parms.h’’ and ‘‘Makefile’’. It describes the meaning of the option and the effects of changing the value. The recommended method for choosing an appropriate configuration is to sit at a terminal with the checklist and mark each option as it is selected or rejected.

#### 3.4.2 Compiling the Notesfile System.

After all the appropriate parameters are set up, the next step is to generate the notesfile system. First, the requisite directories and files in /usr/bin (or wherever) should be manufactured. This procedure should be run exactly once and will probably have to be run by the superuser. To make these files type:

```
make base
```

The next phase should be run while signed in as the notesfile ‘‘owner’’. Type:

make boot

If all goes well, this should end with a message to the effect that the notesfile system has been installed (approximately 17 minutes on a VAX-11/780) If anything goes wrong, perusal of the terminal dialogue should point out the offending steps. The most likely cause of errors will be a missing directory.

To replace the notesfile software any time after these two steps have been run, type:

make install

If at some time you are must change some of the internal constants of the notesfile package (such as string lengths and blocking factors), all the existing notesfiles on your system will be incompatible with the new instantiation of the program. The ‘‘nfdump’’ and ‘‘nfload’’ programs are useful for converting the data base from one format to another. The ‘‘nfdump’’ program should be compiled with the old configuration and the ‘‘nfload’’ program should be compiled with the new configuration. Documentation on these programs can be found in the appendices.

#### 3.4.3 User Subroutines.

The notesfile package contains several routines available to users in general. The nfabort routine generates a core image, places it in a specified place, logs the fact in a notesfile, and terminates. The nfcomment routine allows users to log text in a notesfile. These routines are useful for debugging information, and communication between program developers and users. Users can load these routines by specifying ‘-lnfcom’ on the ‘cc’ or ‘ld’ command lines.

The normal installation procedure (‘‘make install’’) will compile the nfcomment routine and place it in the library directory. As distributed, the file is placed in the ‘‘/usr/local/lib’’ directory. To change this, modify the definition of ‘‘LIBDIR’’ in ‘Makefile’.

### 3.5 Installing the Man(1) Documentation.

Install the man(1) pages for the notesfile with the following commands. They are best done as super-user.

cd man
make install

The nroff sources for the documentation will be installed in the directory /usr/man in subdirectories man1, man3, and man8.

### 3.6 Interfacing to News(I).

The notesfile system interfaces to the news(I) system. The newsinput program parses articles from the news system and inserts them in the appropriate notesfiles. The newsoutput program moves notes from the notesfile system to the news system. Neither program will move text between the two systems if the notesfile is not enabled for networking (see section 3.1.5). Newsoutput will not retransmit articles inserted by newsinput. Sites sharing notesfiles can all run both newsinput and newsoutput.

Appendix B contains instructions on how to interface a notesfile system to a news system. Configurations for isolated notesfiles sites, notesfile subnetworks, mapping between notesfile and newsgroup names, and a checklist for the gateway installation are included.

Since the notes system stores the entire text of a single notesfile in a single file, a notesfile system uses less space than the same scale news system. The notesfile system is generally more space-efficient than B news.

### 3.7 Housekeeping.

Notesfiles has a number of utilities to manage its data base. A number of shell scripts are included with the distribution which invoke the archival programs and trim logging files which otherwise grow without bounds. These shell scripts are included in the ‘‘Samples’’ subdirectory of the notesfile distribution and can be invoked automatically through cron(8).

#### 3.7.1 Cleaning up Disk Space.

When a note or response is deleted, a hole is left in that notesfile. This makes for quick deletion but tends to leave some disk space wasted. The compress option on the director page is one way that this space is reclaimed. The nfarchive program (see section 3.7.2) also returns unused space. Have cron run the nfarchive program weekly to automate space reclamation.

#### 3.7.2 Automatic Removal of Notes.

The nfarchive program archives notes untouched for more than a specified number of days. Archived notes and their responses are stored in an ‘‘archive’’ notesfile. ‘‘Archive’’ notesfiles are typically found in the directory /usr/spool/oldnotes; this may vary between installations. The archives can be accessed automatically using the ‘‘N’’ command from the index, note and response displays or by explicitly naming the notesfile (notes /usr/spool/oldnotes/mynotes). The notes are deleted after they have been archived. After the archival, a compression of the notesfile is performed to recover the space formerly occupied by the archived notes. Nfarchive assumes that all months have 30 days and all years have 12 months. The format of the nfarchive command line is:

nfarchive [-d] [-m+ or -m-] [-#] [-w#] [-f file] topic

The -d option specifies that no archiving is to take place; the notes eligible are to be deleted.

The -m option specifies whether to check the director message status before archiving notes. Use ‘‘-m+’’ to archive notes which meet the age requirement and have the director message on. The ‘‘-m-’’ option archives notes of sufficient age with the director message turned off. If this option is omitted, notes are archived on the basis of age only.

The -# option specifies the age of eligible notes. The increment is days. The default is determined by the value of ARCHTIME in ‘parms.h’ and is distributed as fourteen days. Each notesfile can specify a (possibly) different expiration threshold. Normally the notesfile will specify ‘default’ and nfarchive uses the time specified on its command line. Ages specified in a notesfile override the times given on the nfarchive command line.

The -w# parameter specifies the working set size to use for notesfiles which do not specify a working set size. This determines a minimum number of notes to remain in a notesfile. If unspecified, the default working set size is 0. This value can be changed through the WORKSETSIZE variable in ‘‘parms.h’’.

Working sets and expiration thresholds work together in the following manner. Nfarchive first determines a maximum number of notes to delete. This is calculated from the number of notes in the notesfile (total notes minus ‘‘holes’’ caused by deletions) and the working set size. The archival program proceeds through the notesfile deleting notes older than the expiration threshold until it runs out of notes or the maximum number of notes has been deleted.

The -f option specifies a file containing a list of notesfiles to be archived. Multiple -f parameters are permitted and can be intermixed with ‘topic’ parameters. Notesfile name pattern matching is performed on both ‘topic’ parameters and the entries in a file specified by the -f option.

Mapping between an active notesfile and its archive is listed in the file ‘‘.utilities/net.aliases/Archive-into’’. This file contains pairs of ABSOLUTE notesfile names (‘‘/usr/spool/notes/mynotes’’ instead of ‘‘mynotes’’). The first entry specifies the active notesfile; the second entry specifies the archive notesfile. Notesfiles not appearing in the archive mapping file will be placed in ‘‘/usr/spool/oldnotes’’ with the same final component as the active notesfile. Conflicts are possible. If unmapped, the notesfiles ‘‘/usr/spool/notes/mynotes’’ and ‘‘/some/other/place/mynotes’’ will be archived into the same archive ‘‘/usr/spool/oldnotes/mynotes’’.

#### 3.7.3 Cleaning Sequencer Files.

A tool to remove sequencer entries for deleted users will be included in future releases of the notesfile package. This tool will traverse the sequencer directory removing files belonging to users whose signons no longer exist. The process can be made automatic by including an entry in the cron table.

A second sequencer related utility will traverse the sequencer directory and remove entries for deleted notesfiles. A similar user program will be provided to permit users to remove their sequencer entries for notesfiles that they no longer peruse.

#### 3.7.4 Additions to System Boot.

Since the notesfile system maintains several lock files to ensure mutual exclusion of each notesfile, a line similar to the following should be added to the /etc/rc or /etc/rc.local file:

```
rm -f /usr/spool/notes/.locks/*
```

Make sure ‘/usr/spool/notes’ matches the MSTDIR parameter in the Makefile. This line will remove any locks left if the system has crashed.

## 4 Other Notesfile Utilities.

The notesfile distribution includes utility programs to provide hard copy output, additional interfaces to user programs, and statistics. They are described below.

### 4.1 Hard Copy Output.

The program ‘‘nfprint’’ sends to standard output a nicely formatted listing of the notesfile in its command line. Its format is:

```
nfprint [-lnn] [-p] [-t] topic [ note# ] [ note#-note# ] [ ... ]
```

The ‘‘-l’’ option specifies an alternate page size (the default is 66). The optional note number list specifies that only certain notes of the notesfile are to be printed. The list can specify individual notes and ranges. The notes are printed in the order specified.

The -p option specifies that each notestring is to begin on a new page. The -t option signifies that only a table of contents is to be generated.

### 4.2 Piped Insertion of Notes.

The nfpipe program enters text from the standard input into a notesfile:

```
nfpipe topic [-t title] [ -d ] [ -a ]
```

The -t option allows specification of a title. The -d and -a options specify the director and anonymous flags respectively (if available). If no title is specified, one is manufactured from the first line of the note.

### 4.3 User Subroutines.

#### 4.3.1 Nfcomment.

The nfcomment subroutine is callable from a user's C program. It allows any user program to enter text into a notesfile:

```
nfcomment (nfname, text, title, dirflag, anonflag)
```

The parameters are:

```
char *nfname;     /* name of notesfile */
char *text;       /* null terminated text to be entered */
char *title;      /* if non-null, title of note */
int dirflag;      /* != 0 -> director flag on (if allowed) */
int anonflag;     /* != 0 -> anonymous note (if allowed) */
```

If the text pointer is NULL, the text of the note will be read from standard input. If no title is specified the subroutine will manufacture a title from the first line of the note. This routine is useful for error reports, user comments about programs, and automatic logging of statistics or internal states.

This routine can be loaded with a C program by specifying '-lnfcom' on the 'cc' command line.

#### 4.3.2 Nfabort.

Nfabort allows users to generate core images of their process, save the core image in a ''known'' place, and log that fact in a notesfile. This proves useful for intermittent failures; The programmer regularly scans the notesfile and can examine the core dump at leisure. Some of the problems of recreating conditions which cause errors are eliminated by this approach.

Nfabort is callable from the user program. It accepts the following parameters:

```
nfabort (nfname, message, title, cname, exitcode)
```

The parameters are:

```
char *nfname;     /* name of notesfile */
char *message;    /* text string to insert */
char *title;      /* title of the message */
char *cname;      /* prefix for core image destination */
int exitcode;     /* code for exit() */
```

The core image is placed in the file specified by concatenating the ''cname'' argument and a unique integer (the process id of the current process). The notesfile specified by the ''nfname'' parameter receives a note whose body consists of the text pointed to by ''message'' and a line telling the complete pathname of the core image. The title of the note is specified by the ''title'' parameter. After the core image is generated and the note has been written, nfabort terminates with the exit code specified by the ''exitcode'' parameter.

Nfabort generates default values for each of the string parameters if NULL pointers are passed. This routine can be loaded with a C program by specifying '-lnfcom' on the 'cc' command line.

**4.4 Statistics.**

The notesfile system keeps statistics on where notes and responses originate, the number of network accesses, duplications and orphaned responses. Combined with the use of the log maintained by the notesfile networking software, monitoring notesfile traffic is quite easy.

The -s option specifies that only a summary is to be produced, skipping the individual reports. Wildcard constructs with '*', '?', '[', and ']' are recognized by nfstats. Invoke the statistics program with:

nfstats [ -s ] topic1 [ ... ]

Typical output is:

```
    rbenotes on uiucdcs at  6:24 pm  May  7, 1982
                      NOTES     RESPS     TOTALS
Local Reads           359       115       474
Local Written         53        55        108
Networked in          0         0         0
Networked out         0         0         0
Network Dropped       0         0         0
Network Transmissions: 0   Network Receptions: 0
Orphaned Responses Received: 0    Entries into notesfile: 109
Total time in notesfile:   66.57 minutes  Average Time/entry:   0.61 minutes
Created at 10:04 pm  May  5, 1982, Used on 3 days
```

A combined set of statistics is produced at the end of listings of more than one notesfile. The statistics are largely self explanatory.

**4.5 Checking for New Notes.**

The checknotes program checks the notesfiles specified by the NFSEQ environment variable to determine if there are new notes. The exit code is arranged to make the program useful in shell scripts: 0 (TRUE) is there are new notes, 1 (FALSE) otherwise.

Use the ''-q'' option to receive a message

There are new notes

if one or more of the notesfiles have notes/responses written since the user's last entry time into that notesfile.

The ''-n'' option is similar to the ''-q'' option, with the exception that it yields output when there are no new notes. The output of checknotes with the ''-n'' option is:

There are no new notes

Use ''-v'' to print the name of each notesfile with new notes/responses. The ''-s'' option is suitable for use in conditional expressions in shell scripts; no output is generated by this option.

### 4.6 Mail to Notesfiles.

Some mailers (like 4.2 BSD's *sendmail*) allow mail to be directed to specific programs. The nfmail program facilitates sending mail to notesfiles. The subject line of the mailed letter is extracted and used as the title of the note. Nfmail extracts the author's name and system from the body of the letter. Subject lines are examined for ‘‘Re:’’ prefixes and, if they exist, are used in an attempt to link the letter as a reply to an existing base note.

To send mail addressed to ‘‘somenotes’’ to the notesfile ‘‘somenotes’’ with the 4.1 Bsd or later mail delivery mechanism add the following line to the file /usr/lib/aliases. The second line shows the general format for /usr/lib/aliases entries which feed a notesfile. If the Notesfile utilities are stored in a different directory, the path name should be changed appropriately.

```
somenotes: ‘‘|/usr/spool/notes/.utilities/nfmail somenotes’’
mailaddress: ‘‘|/usr/spool/notes/.utilities/nfmail anotesfile’’
```

### 4.7 Modifying Access Rights for Many Notesfiles.

It is convenient to add entries to a large number of access lists simultaneously. The nfaccess program adds an access specification to each of a specified list of notesfiles. Nfaccess functions similarly to chmod(1). Nfaccess is invoked as:

```
nfaccess <access-right> notesfile [notesfile ...]
```

The ‘‘access-right’’ is formatted as: ‘‘type:name=mode’’. Type can be any of ‘‘user’’, ‘‘group’’, or ‘‘system’’; capitalized variants are also valid. The ‘‘type:’’ specification can be omitted. ‘‘User’’ is assumed in these cases. The ‘‘mode’’ field consists of a sequence of the characters ‘‘d’’, ‘‘r’’, ‘‘w’’, ‘‘a’’ and ‘‘n’’. These indicate director, read, write, answer (respond) and null access respectively.

Nfaccess requires user and group entries to be valid by looking for them in /etc/passwd and /etc/group. System entries are not checked for validity. Nfaccess will add the entry to the access list of the specified notesfiles. If an entry for that particular user, group or system exists, the new access right replaces the old access rights. The computed mode is an absolute mode; the previous value in the access list (if any) is replaced with the new mode.

Any user can run the nfaccess program. Nfaccess refuses to modify access lists for any notesfile where the user is not a director. The nfaccess program is stored in the notesfile utility directory, typically ‘‘/usr/spool/notes/.utilities’’.

Nfaccess is often used to remedy missing permissions in a number of notesfiles. One example is when the notesfile administrator is replaced; nfaccess is used to grant director access to the appropriate notesfiles (usually most of them). As new notesfiles are created, the access list can be tuned by placing lists of access-rights in the file ‘‘/usr/spool/notes/.utilities/access-template’’. These access-rights are added to the default access list of newly created notesfiles.

## 5 Summary.

notes [-sxi] [-a subsequencer] [-t type] nf-list

**Everywhere:**

| | |
|---|---|
| ? | help |
| q,k | quit |
| Q,K | Quit w/out save |
| ↑D | Leave **NOW** w/out sequencer |
| ! | Fork Shell |
| B | Register suggestion |

**Index Page:**

| | |
|---|---|
| + | Forward |
| * | Last page |
| - | Backward |
| = | First page |
| 13 | Read note 13 (requires RETURN) |
| a,A | Search for author |
| d | Director options |
| j,J | Jump to first unread note |
| l,L | Like j,J; leave if nothing unread |
| n | Nest notesfiles |
| N | Nest to this notesfile’s archive |
| o | Set oldest date for sequencing |
| O | Set sequencer for today’s notes |
| p | Read policy note |
| r | Replot the screen |
| w | Write a note |
| x,X | Search for title |
| z | like ↑D but update sequencer |

**Reading anything:**

| | |
|---|---|
| spc | Next page of current text |
| - | Previous page of text |
| ; | Next text |
| + | Next text |
| ret | Next note |
| 7 | Forward 7 responses (15 = 8 + 7) |
| = | Goto base note |
| * | Last Response |
| a,A | Search backwards (author) |
| c,C | Copy text to notesfile (C = w/editing) |
| d | Toggle director msg. |
| D | Delete (if yours) |
| e | Edit title |
| E | Edit text |
| f,F | Forward notestring to notesfile (F= w/editing) |
| i | Index page |
| j | Next unread text |
| J | Next unread base note |
| l,L | Like j,J; leave if nothing unread |
| m | Mail to anyone |
| M | Mail w/text |
| n | Nest notesfiles |
| N | Nest to this notesfile’s archive |
| p,P | Personal note to author (P = w/text) |
| r,↑L | Replot current text |
| R | Rotate text (simple decryption) |
| s | Save text |
| S | Save entire note string |
| t | Talk to author |
| w | Write response |
| x,X | Search backwards (title) |
| z | like ↑D but update sequencer |
| Z | Delete (directors only!) |

**Director Options:**

| | |
|---|---|
| a | Toggle anonymous |
| A | Toggle archive status |
| c | Compress notesfile |
| D | Change expire w/dirmsg flag |
| e | Change expiration threshold |
| E | Change expiration action |
| l | Change maximum text/article |
| m | Change director message |
| n | Toggle network availability |
| o | Toggle open status |
| t | Change notesfile title |
| u | Un-delete lots of notes |
| w | Write policy |
| W | Change working set size |
| z | Zap lots of notes |
| p | Access (permission) List: |

| | |
|---|---|
| i | insert |
| d | delete |
| m | modify |
| u | user |
| g | group |
| s | system |
| | |
| r | read access |
| a | answer access |
| d | director options |
| w | write access |
| n | null access |

**APPENDIX A**
**Notesfile Installation Checklist**

### A.1 Installing Notesfile Code.

You can be sure that you have modified all necessary constants in the notesfile system by following this simple checklist.

____ change to the notesfile source directory
____ tar x [reads the notesfile tape]
____ cd src
____ [edit] parms.h
____ ARCHTIME. Default for how long unmodified note strings hang around.
____ WORKSETSIZE. The default number of notes to leave in a notesfile when archiving.
____ DFLTSH. This should be left as the Bourne shell, /bin/sh -RBE
____ DFLTED. The editor to use if no NFED or EDITOR environment variable exists.
____ SEQFILE. This is the name of a file in the utility directory which contains a list of notesfiles for users without an NFSEQ environment variable. The default is probably just fine.
____ DFLTSEQ. For users without an NFSEQ environment variable and when the file specified by the SEQFILE definition above doesn't exist, we finally fall back to using the notesfiles specified by this string. The nice thing about having things in SEQFILE is that you don't have to recompile the notesfile software everytime you wish to change the default set of notesfiles; instead you edit a file.
____ MAILER. The program which will do mailing. If you are in a networked environment, this mailer should manage to route letters to far away places for you. The notesfile system only retains the name of the destination site; a path to that site is not kept.
____ SUPERMAILER. This should be defined if you have an intelligent mail program. Intelligent here means that you can edit the letter and other fun things.
____ PAGER. A program which shows 1 screenful of information at a time.
____ WRITE. A program which allows online user-user communication (such as /bin/write).
____ FULLDOMAIN. This defines the domain name of your local systems. For many USENET sites, this should be ''UUCP''. Other examples include ''ARPA'' and ''Uiuc.ARPA''. This should not include the system name. In the UIUC Computer Science Department, we have machines named ''uiucdcs'', ''uiucdcsb'', and so on; our value for FULLDOMAIN is ''Uiuc.ARPA''. The notesfile code puts things together to yield ''uiucdcsb.Uiuc.ARPA'' as a full domain name for one of our machines. Note that you do NOT need to place a ''.'' at the beginning of the domain name; the notesfile software does this for you.

- IDDOMAIN. This switch is (for now) undefined. When defined it indicates the the unique id's associated with notes are to have a system component containing the system name and the string defined by FULLDOMAIN. The eventual goal is that this will be defined. Currently, there are problems with using long strings for unique identifiers. This is related to the old notesfile structure which used a 10 character maximum and didn't check for overflow.
  So for now, leave this undefined. I'm not sure when it will be good to enable this option.

____ Select a kernel type for your machine. If no kernel is selected, the code may compile but most likely will not run.
____ PROMPT. If you wish the system to give a command prompt. Otherwise the notesfile system is promptless.
____ USERHOST. If this is defined, the notesfile system uses the convention ''user@host'' to indicate where an article originated. If undefined, the notesfile system uses a ''host!user'' notation. If you are running in an environment which supports Internet style names, you may choose to use this.
____ DYNADIR. When defined, the notesfile system will determine the default spool directory notesfiles from /etc/passwd. The home directory for the user ''notes'' (actually whatever is

specified in the Makefile) is read from /etc/passwd and the parent directory is used as the default spool directory. Thus if notes' home directory is ''/usr/spool/notes/.utilities'', the default directory is ''/usr/spool/notes''. This assumes that notes' home directory is in the .utilities directory.

This is useful for the case where a single binary will be run on several machines with differing disk layouts. On one, the database might live in /usr/spool/notes; another host might have the data base in /mnt/notes. By using DYNADIR and moving the ''notes'' home directory on various machines, only one binary is needed.

If the notes database lives in the same place on all of your machines, a better approach is to use the MSTDIR definition in the Makefile.

____ K_KEY. When defined, the ''k'' and ''K'' keys act similarly to the ''q'' and ''Q' keys respectively. The choice is up to you. Defining them allows reading of notes with one hand. However some people get aggravated when the miss the ''j'' key and hit the ''k'' key. All the documentation considers the ''k'' key to be active.

____ BIGTEXT. Define this is you want to allow notes longer than 65000 bytes. Note that if you have old notesfiles, you will have to dump and reload them with the ''nfdump'' and ''nfload'' programs before the new code will work on them.

____ **The following definitions are pretty much ''stock'' and usually aren't changed.**

____ NFMAINT. This is the name of the notesfile which receives control messages, error reports and other notesfile logging functions. I do not recommend #undef'ing this.

____ AUTOCREATE. When defined, network receptions of previously undefined notesfiles will cause the creation of that notesfile. If undefined, the reception will fail if the notesfile doesn't exist. This is used in environments such as USENET where new notesfiles are often created remotely.

____ STATS. If defined, the statistics keeping code is enabled. If undefined, the notesfile system will not keep statistics. Keeping statistics involves no space overhead and relatively little time overhead; I recommend leaving this defined.

____ FASTSEQ. Enables code which ''fails-quickly'' by determining in an inexpensive operation if there can't be any new articles. When there might be new articles, a more thorough and time consuming algorithm is used.

____ DUMPCORE. This defines a subdirectory of the notesfile utility directory where core images generated by internal consistency checks are placed. If undefined, the errors will be logged but no core image is generated.

____ FASTFORK. This definition enables a quick forking algorithm which exec's the desired program immediately instead of going through the system(III) interface. It avoids an extra fork()/execl() and shell startup costs. However some functionality is lost.

____ [finished editing parms.h]

____ [edit] Makefile

____ select BIN. The directory where user commands are kept. The Makefile will NOT create this directory. At UIUC, we use /usr/bin. Another common choice is /usr/local.

____ MSTDIR. The default directory for notesfiles. The Makefile WILL make this directory for you. This is typically /usr/spool/notes.

____ ARCHDIR. Old notes never die, they go here instead; the Makefile WILL make this directory for you.

____ NET. This is the directory where the notesfile networking programs ''nfxmit'' and ''nfrcv'' will be placed. In most cases, ''/usr/bin'' is a good choice. You may wish to change it if your UUCP or other networking mechanisms use other directories. This directory must already exist; the Makefile will not create it.

____ AUTOSEQ. The invocation name for the automatic sequencer. For multiple names like 'autoseq', 'readnotes' and 'autonotes', make links to the file ''/usr/bin/notes'' with the appropriate names (assuming that BIN = '/usr/bin').

____ NOTES. The username of the user who ''owns'' all the notesfiles.

____ NOTESUID. The numeric userid of the notesfile ''owner''. For example NOTES = notes,

NOTESUID = 10.

____ NOTESGRP. The name of the group to which the ‘‘NOTES’’ signon belongs. **It is strongly recommended that this be a special group just for the notes database and programs.**

____ ANON. The name of the ‘‘anonymous’’ user.

____ ANONUID. The numeric userid of the ‘‘anonymous’’ user; this should be an idle user id since it is not allowed to run the notesfile program.

____ LIBDIR. The directory to contain ‘‘libnfcom.a’’, a user accessible library of routines. This is distributed as /usr/local/lib.

____ CFLAGS. You may wish to arrange for split I/D loading on a PDP-11 (the -i flag). It may also be prudent to optimize the code (-O flag). If code size is an issue, remove the RCSIDENT definition; when defined, version control information is included in the binaries and they are correspondingly larger.

____ [finished editing] Makefile

____ [may need to become super-user at this point]

____ type ‘‘make base’’ and assess its completion. It will tell you if all went well. **If you are merely installing a new version of the notesfile code, you should type ‘‘touch base’’ instead of ‘‘make base’’.**

____ Signon as notesfile ‘‘owner’’. While remaining super-user isn’t a fatal flaw at this point, it does mean that several default notesfiles won’t be generated. These can be created by hand if you forget. Nothing from this point on (including future code updates) requires super-user privileges.

____ cd src

____ **If you are merely installing a new version of the code, type ‘‘touch spool’’ now. This tells the makefile that the spool directories already exist.**

____ make boot. This is the final step, it should complete with a message that the system is installed. An error message when doing the ‘‘mknf -on nfmaint nfgripes’’ probaby means you are still super-user. Don’t sweat it; just become notes and type the ‘‘mknf’’ command line over. Everything is now fine.

____ You may have to be Super-User for the next step depending on the modes of your manual directory, /usr/man.

____ cd ../man. [the man page directory for notesfiles]

____ make install. to install the man(I) pages for the notesfile system. They are placed, by default, in the directories /usr/man/man1, /usr/man/man3, and /usr/man/man8.

____ Examine the ‘‘Samples’’ directory for templates of files normally in the notesfile utility directory. These files include shell scripts to run through cron(8) which queue network transmissions, expire old notes, and map between notesfiles and news.

____ Modify UUCP’s ‘‘uuxqt.c’’ to allow remote execution of ‘‘nfrcv’’. This is unnecessary if no notesfile networking will be done or if another remote execution mechanism will be used. Some versions of UUCP have a file ‘‘/usr/lib/uucp/L.cmds’’ which contains names of permitted commands.

____ Modify /etc/rc to remove notesfile locks at boot time. [4.2 BSD machines might prefer to use /etc/rc.local.] Add the command ‘‘rm -f /usr/spool/notes/.locks/*’’.

### A.2 Upgrading Existing Notesfile Databases.

Revision 1.7 of the notesfile system requires converting existing notesfile databases to a new format. A set of programs to accomplish this task are included in the distribution. The ‘‘nfdump’’ program converts notesfiles into an ASCII representation. The ‘‘nfload’’ program converts this ASCII representation back into a properly formatted notesfile. To convert an existing notesfile database, these steps are what I follow:

____ Compile ‘‘nfdump’’ with the OLD notes distribution. If your version of the software is old enough not to have a copy of nfdump, I suggest you either try to adapt the version in the current distribution or using the networking programs ‘‘nfxmit’’ and ‘‘nfrcv’’ to get the information between the old and new databases.

____ Compile ‘‘nfload’’ with the NEW notes distribution.
____ cd /usr/spool/notes
____ mkdir .OLD
____ mv * .OLD
____ run the following script:

```
#! /bin/csh -f
foreach i (‘ls .OLD‘)
  echo $i start
  nfdump-old /usr/spool/notes/.OLD/$i - | nfload-new $i
  echo $i done
end
echo ALL DONE
```

____ rm -rf .OLD

You will also have to convert the sequencer information. In the ‘‘utility/seq-cvt’’ directory there are a pair of programs ‘‘seqtoascii’’ and ‘‘seqtobinary’’. To convert the sequencer information:

____ make ‘‘seqtoascii’’ using the OLD structs.h and parms.h.
____ make ‘‘seqtobinary’’ using the NEW structs.h and parms.h.
____ cd /usr/spool/notes/.sequencer
____ mkdir .OLD
____ mv * .OLD
____ run this shell script:

```
#!/bin/csh -f
foreach i (‘ls .OLD‘)
  echo $i
  seqtoascii .OLD/$i | seqtobinary $i
end
echo ALL DONE
```

____ rm -rf .OLD
____ If you are going to use the FULLDOMAIN option, you may want to go ahead and perform the following steps:
____ run this shell script, appropriately modified to reflect your domain setup. This one reflects the naming at UIUC.

```
#!/bin/csh -f
foreach i (Sy:*)
  echo $i
  ln $i $i.UUCP
  ln $i $i.Uiuc.ARPA
end
echo ALL DONE
```

____ You have now converted your notesfile database to 1.7 format. Install the new binaries and fire away.

## A.3 Hints on Installing Notesfiles on Multiple Systems.

Notesfile binaries are portable across similar machines. User-id’s and hostnames are determined by examining /etc/passwd and through system calls.

To install the Notesfile system on a network of like machines (a collection of 68000 workstations for example) one machine must go through the procedure detailed above. A shell script ‘‘rinstall’’ is included in the notesfile source directory. This shell script will propagate copies across the network. Rinstall is a simple script that assumes the correct hierarchies exist on the target machines. It was written to use the 4.2 BSD ‘‘rcp’’ and ‘‘rsh’’ programs to copy files and remotely execute commands. Different networking commands will require changes to the shell script.

To generate the proper hierarchies on other systems, copy the Makefile to each of the machines and make both ‘‘base’’ and ‘‘spool’’. This will create the proper files and directories for the notesfile system. Then return to the master machine and run the rinstall script to send binaries to each of the other machines.

The ‘‘Samples’’ directory of the Notesfile distribution contains example cron scripts for sending information between a network of systems running notesfiles. These shell scripts can prove helpful in setting up notesfile transmissions around your local network.

## A.4 Mail to Notesfiles.

To use the nfmail program with the 4.1 BSD /etc/delivermail or the 4.2 BSD /usr/lib/sendmail insert lines of the following form in the file /usr/lib/aliases.

```
somenotes: ‘‘|/usr/spool/notes/.utilities/nfmail somenotes’’
gripes: ‘‘|/usr/spool/notes/.utilities/nfmail problems’’
```

## A.5 The Notesfiles/News Gateway.

The notesfile/news gateway may need a little more tickling to convince it to work properly. For more information on how to set this up properly, see section 3.5 (‘‘Interfacing to News’’) and look at the file ‘Src/newsgate.h’. Appendix B (‘‘Interfacing Notesfiles to News’’) is another source of information for connecting the two systems.

## APPENDIX B
## Interfacing Notesfiles to News

The News system provides functions similar to those provided by the Notesfile system. It is possible to gateway articles between the two systems. In USENET, a common configuration is for several notesfiles sites to form a subnetwork of USENET and gateway articles between the local notesfile network and the rest of USENET. These articles propogate across USENET in the news system. Article originating in the news system are gatewayed into the notesfile network. When several notesfile networks exist as subnetworks of a larger news network (such as USENET), articles written in one notesfile network will be correctly interpreted when arriving at another notesfile network. ‘‘Correctly’’ interpreted includes proper linking of responses to their true parents; this is sometimes not possible with articles written in the news system.

The notesfile gateway code recognizes articles meeting the USENET standards. Additionally, the A-news protocol and older B-news protocols are recognized. Incoming (news → notes) articles are parsed and placed in the appropriate notesfiles. Articles written within a notesfile subnetwork are formatted according to USENET standards and transmitted to the news system.

### B.1 Configurations for Sites without News.

Sites running notesfiles without the news program have several possible configuations. If your site is part of a notesfile subnetwork and you wish to have your articles gatewayed to the news system and the rest of USENET, you must find a site who will act as a gateway for your articles. The gateway code performs its task in such a way as to allow several sites to gateway the same article into the news system; the multiple copies will have identical unique identifiers and one copy will be canceled when they meet on a remote system.

#### B.1.1 Sites with no News Neighbors.

If you have no immediate neighbor running news, there is no particular action you should take other than to reassure yourself that some site in the notesfile subnetwork is gatewaying notes-generated articles to the news system. This can be done in one of several ways. If the site wants to gateway articles specifically from your machine, the following command should be executed on that site occasionally. This is typically done through cron(8).

```
newsoutput -syoursite notesfile-list
```

A more typical arrangement is where the gateway site sends all notes-generated articles that arrive on its system into the news system. In this case, the gateway site will execute a command such as the following:

```
newsoutput -a notesfile-list
```

A site gate using this command line will automatically gateway articles written at your site and eliminates the need of running a command similar to the first command line.

#### B.1.2 Sites with Neighbors running News.

If a neighboring system runs both notes and news, you have the option of gatewaying your own articles or allowing the neighbor to gateway articles for you. If the neighbor does not run the notesfile system, you must provide your own gateway functions.

Gatewaying can be done similarly to the site without a news neighbor. You can let your articles propogate across a notesfile network to a gateway site which will send them to the news system. Another option is to gateway your articles, and possibly any notesfile-generated articles, into the news system at the neighboring site. This gets your articles into USENET as quickly as possible. The two options are not mutually exclusive; you can gateway your own articles and allow another site to gateway them. When copies of the same article meet on a news system, the extra copy will be removed from the network.

To gateway articles into news, you must modify the file /usr/spool/notes/.utilities/net.how to tell the newsoutput program how to get to the news system. More information on this can be found in the section ‘‘Copying Notesfiles to News’’ later in this appendix.

To gateway from the news system to the notesfile system, you must arrange to have the news system at the remote site send articles as standard input to the program /usr/spool/notes/.utilities/newsinput on your system.

Information on mapping functions between notesfiles and news, how to configure a news system to send articles to a notesfile system, and how to have a notesfile system send articles to a news system can be found later in this appendix.

## B.2 Configurations for Sites running News.

A site running both notesfiles and news will typically perform gateway functions in both directions, from the notesfile system to the news system and from the news system to the notesfile system. In these cases all the operations are local.

## B.3 Gatewaying between Notesfiles and News.

The two notesfile programs ‘‘newsoutput’’ and ‘‘newsinput’’ perform gatewaying between notesfile and news systems. Newsoutput takes notesfile-generated articles, formats them, and hands them to the news system. Newsinput takes articles from the news system and inserts them in the notesfile system.

### B.3.1 Copying News to Notesfiles.

The news system maintains ‘‘subscription lists’’ for each neighboring system. The subscription list is stored in the file /usr/lib/news/sys on a B-news system. On an A-news system, the subscription list is in /usr/spool/news/.sys

News feeds articles to neighboring systems as they arrive. In many cases, the article is queued for transmission. In other cases, the article is given to a batching program which collects a number of articles for transfer and sends them to an appropriate un-batching program at the receiving end.

In the case where the notesfile system resides on another machine, the news subscription line should be generated similarly to that for a normal news feed with several exceptions. The first is that the newsinput program does not understand about batching; articles must be sent one at a time. Also, one must specify the method of transmitting the article. To feed the system ‘‘somesite’’ with news, the neighbor will add a line of the following form to his /usr/lib/news/sys:

```
somesite:subscription::uux - -n somesite!/usr/spool/notes/.utilities/newsinput
```

Of course, networks other than UUCP can be used.

To forward to a notesfile system on the same machine as the news system, create a pseudo site which doesn’t exist elsewhere (a sitename such as ‘‘nf_sys’’ works), and add a line like:

nf_sys:subscription::/usr/spool/notes/.utilities/newsinput

Articles arriving at the local system will now be forwarded to the notesfile system. By default, news articles are placed in notesfiles with the same name. To map newsgroups to particular notesfiles, see the later section ‘‘Naming Notesfiles and Newsgroups’’.

**B.3.2 Copying Notesfiles to News.**

The newsoutput program transfers notesfile-generated articles from the notesfile system to a news system. The news system does not have to be on the same machine.

Newsoutput accepts parameters telling it to gateway articles from specific systems or any system. Additional options allow backwards compatible headers for older versions of the notesfile software. A typical newsoutput invocation looks like:

newsoutput -a notesfile-list

The -a parameter indicates that notesfile generated articles from any site are to be sent to the news system. Under no circumstances will newsoutput transfer an article to the news system if it has passed through the news system before. Thus if a notes generated article passes from one notesfile subnetwork to another through the news system, the article will not be sent into the news system by anyone in the second notesfile subnetwork.

The ‘‘notesfile-list’’ can contain a mixture of specific notesfiles, wild-card specifications (net.*), or ‘‘-f file’’ parameters which specifies a file containing a list of notesfiles to send.

Alternatively, articles for only one system can be gatewayed with a command line of the form:

newsoutput -ssitename notesfile-list

This invocation method maintains a sequencer for each system; the -a option maintains a single global sequencer.

A third invocation method of newsoutput uses the ‘‘-c’’ option and specifies a specific set of systems to gateway articles for. The command looks like:

newsoutput -c gateway-site-file notesfile-list

The ‘‘gateway-site-file’’ specifies a file containing a list of sitenames. Articles written at any of these sites are gatewayed to the news system. Thus newsoutput has the ability to send articles to news for a single system, several systems, or any system.

Newsoutput assumes that a news system is installed on the local system. If the news system is in a non-standard location on the local system or the news system is on a different machine, newsoutput can be told where to send articles. The file /usr/spool/notes/.utilities/net.how contains command templates for notesfile networking. To specify a non-standard place for the ‘‘rnews’’ program, add a line of the form:

Usenet:x:::uux - -n myneighbor!/usr/bin/rnews

Non-UUCP connections and the like can be specified.

**B.3.3 Naming Notesfiles and Newsgroups.**

Notesfiles and newsgroups for the same topic can have different names. Notesfiles are currently limited in the last component of their name to the length of a filename; under V7, System III, System V, and 4.1 Bsd this is 14 characters. 4.2 Bsd extends the length of a filename to a maximum of 255 characters per component. Newsgroup names are no longer limited in total length; the only restriction in current news versions is that each component (between .'s) is unique in the first 14 characters.

The file ''/usr/spool/notes/.utilities/newsgroups'' defines the relationships between notesfiles and newsgroups. Lines in the file have the general form:

notesfile:base_newsgroup:respone_newsgroup

Lines beginning with the ''#'' character are considered comment lines. The ''response_newsgroup'' field and the colon separating it from the base_newsgroup field are optional.

Entries in this file need be made for only a few reasons: (1) The newsgroup which matches the notesfile is longer than fourteen characters, (2) the notesfile and the newsgroup are different names (e.g. the notesfile 'Bnews' can be linked to the newsgroup 'net.news.b' with an entry of 'Bnews:net.news.b'), (3) you want several newsgroups linked to a single notesfile, and (4) notes and responses from a notesfile should go to different newsgroups (net.general/net.followup is one example). The file is scanned from the beginning until EOF or a match is found. When no match is found, the code returns the original argument as if it had matched itself. The process is similar to having placed a sentinel line of the form:

myarg:myarg

at the end of the file.

The optional third field in the line is used to send notes and responses from a notesfile to separate newsgroups. The net.general/net.followup convention is an example of how this would be used. Typically the net.general and net.followup newsgroups are mapped into the same notesfile. To preserve peace with news users, responses written in a notesfile 'net.general' should go to the newsgroup 'net.followup'. The following pair of lines will map all news from net.general and net.followup into the notesfile net.general. Base notes from the notesfile net.general will go to the newsgroup net.general; responses in the net.general notesfile will be sent to the net.followup newsgroup.

net.general:net.general:net.followup
net.general:net.followup

The first line maps news in net.general to the notesfile net.general. It also does the mapping from notesfiles to newsgroups. The second line maps news from net.followup into the notesfile net.general. Note that the response field is not used in the mapping from newsgroups to notesfiles; it is used only for mapping notesfile responses into a different newsgroup.

To map several newsgroups into the same notesfile, place lines of the following form in the ''newsgroups'' file:

somenotesfile:newsgroup1
somenotesfile:newsgroup2

If you wish to have articles from the notesfile ''somenotesfile'' go to both of the newsgroups, insert a line before the above lines to map articles going to notesfiles. The result would look like:

somenotesfile:newsgroup1,newsgroup2

somenotesfile:newsgroup1
somenotesfile:newsgroup2

The first line is the one used for articles going from notesfiles to news; the newsgroups specification ‘‘newsgroup1,newsgroup2’’ is used on those articles. Articles coming from the newsgroup ‘‘newsgroup1’’ will fail to match the first line, since the program expects exact matching. They will match the second line and be mapped to the notesfile ‘‘somenotesfile’’.

**B.4 Typical Configurations.**

A typical notesfile subnetwork contains a number of pure notesfile sites and several sites running both news and notesfiles. In these situations, some subset of the sites running both notes and news act as gateway sites. The pure notesfile sites don’t concern themselves with gateway operations. The gateway sites typically gateway all notes-generated articles to news. Duplicate articles, gatewayed at several sites, will propogate across the news network. When two or more of these articles meet at some site, the superflous copy will be removed from the network.

If none of the sites in the notesfile subnetwork run the news program, the gateway site will be one or more of the sites having neighbors that do run news. These gateway sites will run newsoutput and direct the output to the news site by making the appropriate entry in /usr/spool/notes/.utilities/net.how for the pseudo-site ‘‘Usenet’’.

**B.5 News Gateway Installation Checklist.**

The following checklist covers the variables in the ‘‘newsgate.h’’ file which may need to be changed on your system. It also mentions which files to modify to automate the transfer of articles between the two systems.

____ [edit] src/newsgate.h
____ MYDOMAIN. This should be set to the domain your site is in. Typical domains are ‘‘UUCP’’ and ‘‘ARPA’’.
____ DFLTRNEWS. The news receiving program. This can normally be left as is; alternate news insertion methods can be specified with more flexibility through the net.how file.
____ EXPANDPATH. If defined, the code looks in the file specified by PATHMAP for a route to an author’s system. The code which does this is in ‘‘src/newspath.c’’ and expects a particular format in the PATHMAP file. You may wish to replace it with code of your own if your data base is in a different format.
____ PATHMAP. This is the full pathname of the routing tables used if EXPANDPATH is defined.
____ [finished editing] src/newsgate.h
____ make newsouput newsinput. This will recompile the notesfile/news gateway programs.
____ Check /usr/lib/news/sys to see that news will be forwarded to the notesfile system.
____ Make entries in /usr/lib/crontab to invoke newsoutput occcasionally. We use every 6 hours.
____ If the news system is on another machine or in a non-standard place, modify /usr/spool/notes/.utilities/net.how. Add a pseudo-site ‘‘Usenet’’ which specifies how to get to the remote machine. One example is:
        Usenet:x:::uux - -z neighbor!/usr/bin/rnews
____ Check /usr/spool/notes/.utilities/newsgroups. A sample of how this file will look is included in the ‘‘Samples’’ directory of the distribution.
____ Everything should be configured now. You will probably want to run several tests on

local or limited distribution newsgroups to satisfy yourself that it works.

**APPENDIX C**
**Distributed Revisions of the Notesfile System**

Several revisions of the Notesfile System are available. This appendix attempts to describe the differences between each revision and the previous one.

## C.1 Previous Revisions.

The Notesfile System was first distributed in June 1982. Since then it has gone through a number of internal revisions and several major revisions. The initial 1.0 revision had numerous bugs in the code and inadequacies for interfacing with the news system. Release 1.3 (the most recently ‘‘announced’’ release) became available in March 1983.

Revisions are maintained with the RCS system. Major releases are number 1.1, 1.2, 1.3 ... 1.x. Internal modifications are numbered off of the base revision. Internal revisions between 1.2 and 1.3 are of the form 1.2.1.x. All files in a distribution will have the same major revision number; files modified since the major release will an internal revision number based off the major revision number.

## C.2 Revision 1.5.

Revision 1.5 is an intermediate revision. Revision 1.4 was stillborn. It’s primary purpose was to integrate a number of useful modifications sent in by notesfile users. A number of recent 1.5+ distributions have almost the same functional differences from previous revisions as the newer revision 1.6 code.

## C.3 Revision 1.6.

Revisions 1.6 of the Notesfile system includes a number of changes. Numerous bugs in the code were repaired. Several functional differences are also evident in this revision of the code. Major changes are listed below in chronological order. To see what has changed since you received your copy of the code, find the first date after you received your distribution and read from there.

Fall 1983:

- Archival techniques are more refined. Previous revisions determined the age at which to expire notesfiles from the nfarchive command line. Each notesfile now contains its own ‘expiration threshold’. This threshold can be set to an arbitrary time (3 days), default to the value specified on the nfarchive command line, or specify never to archive the notesfile. These options allow expiration of the entire ‘‘net.*’’ collection of notesfiles with the single command line ‘nfarchive net.*’. Shorter duration notesfiles (maybe net.jokes) can be explicitly set to a few days; notesfiles like net.bugs can be set to ‘never’. The remaining notesfiles might be set to ‘default’. A program ‘expirechange’ is provided in the utility subdirectory of the distribution to initialize the expiration threshold of existing notesfiles. This is recommended because the previously unused field may contain garbage values.
- A simple program ‘namechange’ is included in the utility directory to change the name within the data base. If you pick up copies of the data base and set them down on other systems this program will change the name of the system the data base thinks it is on for you.
- Alignment within the notesfile descriptor structure caused me to remove 6 bytes of filler when adding a ‘long’ to the structure. The size of the structure must be

constant. The program in utility/structsizes.c prints the sizes of each of the possibly affected structures. It would be prudent to compile and execute this program once with the old structure definitions and once with the new definitions to ensure that the structures are the same size. Someday a notesfile dump/load program will be written that makes this worry disappear.

- Mapping notes out to the news system is more sophisticated. The new scheme allows a notesfile to send base notes to one newsgroup and responses to another newsgroup. This is solely for the net.general/net.followup pair. See the section ‘‘Copying Notesfiles to News’’ for a more detailed explanation of this feature.
- Binaries are portable. With Unix kernels supporting the ‘‘uname’’ or ‘‘gethostname’’ system call the code determines the host at runtime. The code now also looks for the notesfile owner in /etc/passwd to dynamically determine the ‘notesuid’. As an example, a local network of Vaxen all running 4.1a Bsd can run the same binary even if the ‘notes’ user id varies between machines. Eventually it would be nice to have a single binary handle all 4.1a Vaxen, another for all 4.2 Vaxen, a third be adequate for all USG 5.0 3b-20’s. (This does not mean that distributions will be binary only but rather that a local administrator will be able to compile once and ship copies of the binaries around with a simple shell script).
- The ‘‘rinstall’’ shell script updates the notesfile binaries on a remote system. It assumes that the local binaries will work on the remote machine (don’t rinstall from a Vax to a PDP-11). The script uses the 4.1a ‘rcp’ and ‘rsh’ facilities to perform the FTP and set modes on the remote files.

December 1983:

- Notesfiles can be specified as absolute pathnames. Commands such as ‘‘notes /some/place/mynotes’’ are now legal. An anticipated modification will allow search rules for notesfiles similar to those command search rules used by many shells.
- Archives are stored as notesfiles. Now that a notesfile can be specified by an absolute pathname, archives are stored in notesfile format. Access to archives can be either by explicit reference or through the new ‘‘N’’ command which automatically nests to the archive of the current notesfile.
- Nfarchive now understands about ‘‘working sets’’. The working set is the minimum number of notes left in the active notesfile after an archive run.
- Archive destinations are mapped. A file in the notes utility directory (.utilities/net.aliases/Archive-into) maps from active notesfiles to their respective archives. This file contains absolute pathnames. (/usr/spool/notes/somenotes instead of somenotes).
- The beginnings of permission modes for an archive are there. Currently only directors are allowed to write in an archive notesfile. Some more work on copying permission lists and other information particular to the notesfile must be done.
- The director page now contains information about the number of ‘‘holes’’ (deleted notes and responses) in a notesfile. This is useful for determining the need to compress a notesfile.

January 1984:

- Each notesfile can now override the nfarchive command line options for archiving/deleting expired notes and for expiring notes on the basis of the director message status. The director options page offers options to modify these fields. The ‘‘default’’ value specifies using the value supplied on the nfarchive command line.
- The director option page has been rearranged. More information is displayed, more

options are processed. Many of the changes are cosmetic and oriented towards helping the user figure out what is happening.

- The 4.2 Bsd release of Unix now has its own kernel definition. Some of the new features of 4.2 Bsd are thus included. This includes longer filenames and (faster) advisory locking.
- 4.2 Bsd (and 4.1a) allow processes to belong to multiple groups. The notesfile code now uses all of these groups to determine access rights. For example, a user belongs to groups ‘‘alpha’’, ‘‘beta’’ and ‘‘gamma’’. Group ‘‘alpha’’ has read permission, group ‘‘beta’’ has write permission, and group ‘‘gamma’’ has no specific permissions (it’s covered in the ‘‘Other’’ clause). The user is given the inclusive OR of his permissions: in this case he is given read/write privileges. The default ‘‘Other’’ group is used only when none of the user’s groups are explicitly named in the permission list. Explicit permissions for users still takes precedence over group permissions.

February 1984:

- The networking software and the statistics printing package now keep track of how many orphans are adopted by their true parent. This lets us determine how many base notes are actually lost and how many show up behind their children.
- The nfaccess program allows simple and quick editing of access lists for a number of notesfiles. Nfaccess functions similarly to chmod(1), you supply an access right and a list of notesfiles to apply it to. The new access right is placed in the access list of each notesfile specified. In the event of an existing access right, the new right replaces the old one.
- The code now understands about the extra work it must to to function properly under the 4.2 Bsd signal semantics.
- Nfabort provides user programs with a means of leaving core dumps in specified places and logging the fact with an arbitrary message in a notesfile. The notesfile code itself uses this routine when trapping internal errors.
- Finally added the ‘l’, ‘L’ and ‘z’ commands from Lou Salkind and Rick Spickelmier. The ‘l’-‘L’ pair mimic the ‘j’-‘J’ pair with the exception that when no unread notes are left, the l/L commands leave the notesfile. Thus ‘l’ is almost a ‘jq’ command.
- The ! notesfile exclusion feature first implemented by Salkind and Spickelmier is now in this revision. Constructs like:

  notes ‘‘net.*’’ !net.general

  are possible. This example specifies all ‘‘net.’’ notesfiles except net.general.
- Alternate sequencers are now available. An alternate sequencer allows users sharing the same signon to maintain separate sequencer files.

March 1984:

- Revision 1.6.2 created. Needed some distinction since 1.6 was getting rather long lived and we weren’t ready to call it 1.7 yet.
- Each notesfile now enforces its own limit on the size of single notes and responses. This is initialized to a default value when the notesfile is created and can be changed from the director options page. Articles longer than the permitted maximum are truncated and have a message appended detailing how many bytes were ignored and the name of the site where it occurred.
- Customized access lists are generated when each notesfile is created. The file ‘/usr/spool/notes/.utilities/access-template’, if it exists, is used to modify the default access list when a notesfile is created. This file contains ASCII specifications of access rights in the same form as used on the nfaccess

command line. Lines in this file beginning with ‘#’ are considered comments.

- Author searching now understands about substrings. You no longer have to match exactly an author. Thus an author search would find articles written by a user ‘‘mark’’ on any machine, a user ‘‘hallmark’’ on any machine, and any user on the ‘‘market’’ machine.
- The notes/news interface has been rewritten. The news->notes code now understands all of the USENET standards for B-news 2.10. Newsinput understands about the References line. The notes->news code generates articles acceptable to the rest of USENET.

December 1984:

- The nfmail program has been re-written to understand about how to link responses into a notesfile. This means nfmail is now a viable way to have your incoming mail handled. After a little more work is done, it will handle outgoing mail via the ‘‘p’’ command equaly well.
- Notes now runs set-gid. This solves some privilege problems with set-uid programs such as signal delivery. It also makes it easier for users to kill their jobs. Thanks go to Lou Salkind for pointing this out long ago; I just took a long time to realize it.

### C.4 Revision 1.7.

The long ago promised revision 1.7 of the notesfile code is finally a reality. This version incorporates many of the features promised, and a few that weren’t. I thought about merely changing things from revision 1.6.2 to 1.6.3, but there was a change in the database format and I decided a more drastic change in name was called for to match the database format change.

To upgrade to revision 1.7 from a previous revision (even the last 1.6.2 revisions) requires a dump/load sequence with the ‘‘nfdump’’ and ‘‘nfload’’ programs. The man pages for these programs give more information on how to carry out this procedure.

In addition to changing the format of the notesfile database, you must convert the format of the sequencer files. To do this, look at the programs ‘‘seqtoascii’’ and ‘‘seqtobinary’’ in the ‘‘utility/seqcvt’’ directory of the notesfile distribution. Appendix A also contains information on how to convert the sequencer files and database.

These changes took place during December of 1984 and include the following differences from the 1.6.2 revision of the notesfile code.

- The author structure for a note/response now contains the home system name. This provides the ability for gateway machines to assign message-id’s as needed without worrying about corrupting the author’s home system. It also comes in handy within the context of the nfmail program: nfmail can now report a true author for letters and still assign a unique identifier based on the local system.
- Timestamps for articles are now stored in the standard UNIX format: seconds since 00:00 GMT, January 1, 1970. The code recognizes (and stores) both formats and will present either format as needed.
- Notes now supports full domain based addressing. The nfxmit program expects a full doman address (e.g., ‘‘uiucdcs.uiuc.arpa’’), unique id and system information is generated with full domain information, and the notesfile/news gateway now generates complete domain addressing information. **These changes require some care in upgrading from previous releases of the notesfile system.**

### C.5 On the Blackboard.

The primary motivation for more work on the code is to eliminate known bugs. Integrating other's modifications into my code has taken a lower priority.

Sometime down the road, I hope to gather up all the lessons learned from this first version and design a second implementation. Issues to be considered in the second implementation include: shared notesfile data bases between several hosts, different user-interfaces (notes-like, readnews-like), notesfile servers, and interfacing with extant systems.