User Contributed Software (UCS)

4.3 Berkeley Software Distribution
490149 Rev. B

December 1987

Integrated Solutions
1140 Ringwood Court
San Jose, CA 95131
(408) 943-1902

This manual reflects system enhancements made at Berkeley and sponsored in part by the Defense Advanced Research Projects Agency (DoD), Arpa Order No. 4871 monitored by the Naval Electronics Systems Command under contract No. N00039-84-C-0089. The views and conclusions contained in these documents are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defense Research Projects Agency or of the US Government.

User Contributed Software

This release includes the following user contributed software:

Name	Description	Contributor(s)	Manual
rcs	revision control system	Walter Tichy, Purdue	PS1:13
kermit	file transfer protocol	Columbia University	UCS
mh	MH mail system	Rand Corporation	USD:8
news	"readnews" bulletin board system	Matt Glickman, Berkeley	UCS
patch	apply diffs to originals	Larry Wall, SDC	UCS
umodem	file transfer protocol	Lauren Weinstein	UCS
rn	readnews front end	Larry Wall	UCS

Besides listing the name of each program, its function, and its origin, this table tells you which manual contains the program's documentation. The acroynyms at the right of the table are abbreviations of titles of UNIX manuals. The key below identifies the manuals by their acronyms.

PS1 UNIX Programmer's Supplementary Documents, Volume 1
UCS User Contributed Software (this manual)
USD UNIX User's Supplementary Documents

Numbers following acronyms identify the article in the manual. For example, USD:8 refers to the eighth article in the UNIX User's Supplementary Documents manual.

Revision Control System

Refer to Section 13 in the UNIX Programmer's Supplementary Documents manual, Volume 1 (PS1:13) for information on the Revision Control System.

NAME

kermit – kermit file transfer

SYNOPSIS

kermit [*option* ...] [*file* ...]

DESCRIPTION

Kermit is a file transfer program that allows files to be moved between machines of many different operating systems and architectures. This man page describes version 4C of the program.

Arguments are optional. If **Kermit** is executed without arguments, it will enter command mode. Otherwise, **kermit** will read the arguments off the command line and interpret them.

The following notation is used in command descriptions:

fn A Unix file specification, possibly containing either of the "wildcard" characters '*' or '?' ('*' matches all character strings, '?' matches any single character).

fnl A Unix file specification which may not contain '*' or '?'.

rfn A remote file specification in the remote system's own syntax, which may denote a single file or a group of files.

rfnl A remote file specification which should denote only a single file.

n A decimal number between 0 and 94.

c A decimal number between 0 and 127 representing the value of an ASCII character.

cc A decimal number between 0 and 31, or else exactly 127, representing the value of an ASCII control character.

[ ] Any field in square braces is optional.

{*x,y,z*} Alternatives are listed in curly braces.

Kermit command line options may specify either actions or settings. If **Kermit** is invoked with a command line that specifies no actions, then it will issue a prompt and begin interactive dialog. Action options specify either protocol transactions or terminal connection.

COMMAND LINE OPTIONS

–s *fn* Send the specified file or files. If *fn* contains wildcard (meta) characters, the Unix shell expands it into a list. If *fn* is '-' then **Kermit** sends from standard input, which must come from a file:

```
kermit -s - < foo.bar
```

or a parallel process:

```
ls -l | kermit -s -
```

You cannot use this mechanism to send terminal typein. If you want to send a file whose name is "-" you can precede it with a path name, as in

```
kermit -s ./-
```

–r Receive a file or files. Wait passively for files to arrive.

–k Receive (passively) a file or files, sending them to standard output. This option can be used in several ways:

```
kermit -k
```

Displays the incoming files on your screen; to be used only in "local mode" (see below).

```
kermit -k > fn1
```

Sends the incoming file or files to the named file, *fnl*. If more than one file arrives, all are

concatenated together into the single file *fn1*.

kermit -k | command

Pipes the incoming data (single or multiple files) to the indicated command, as in

kermit -k | sort > sorted.stuff

–a *fn1* If you have specified a file transfer option, you may specify an alternate name for a single file with the **-a** option. For example,

kermit -s foo -a bar

sends the file foo telling the receiver that its name is bar. If more than one file arrives or is sent, only the first file is affected by the **–a** option:

kermit -ra baz

stores the first incoming file under the name baz.

–x Begin server operation. May be used in either local or remote mode.

Before proceeding, a few words about remote and local operation are necessary. **Kermit** is "local" if it is running on a PC or workstation that you are using directly, or if it is running on a multiuser system and transferring files over an external communication line — not your job's controlling terminal or console. **Kermit** is remote if it is running on a multiuser system and transferring files over its own controlling terminal's communication line, connected to your PC or workstation.

If you are running **Kermit** on a PC, it is in local mode by default, with the "back port" designated for file transfer and terminal connection. If you are running **Kermit** on a multiuser (timesharing) system, it is in remote mode unless you explicitly point it at an external line for file transfer or terminal connection. The following command sets *Kermit*'s "mode":

–l *dev* Line — Specify a terminal line to use for file transfer and terminal connection, as in

kermit -l /dev/ttyi5

When an external line is being used, you might also need some additional options for successful communication with the remote system:

–b *n* Baud — Specify the baud rate for the line given in the **–l** option, as in

kermit -l /dev/ttyi5 -b 9600

This option should always be included with the **–l** option, since the speed of an external line is not necessarily what you expect.

–p *x* Parity — **e, o, m, s, n** (even, odd, mark, space, or none). If parity is other than none, then the 8th-bit prefixing mechanism will be used for transferring 8-bit binary data, provided the opposite **Kermit** agrees. The default parity is none.

–t Specifies half duplex, line turnaround with XON as the handshake character.

The following commands may be used only with a **Kermit** which is local — either by default or else because the **–l** option has been specified.

–g *rfn* Actively request a remote server to send the named file or files; *rfn* is a file specification in the remote host's own syntax. If *fn* happens to contain any special shell characters, like '*', these must be quoted, as in

kermit -g x*.\?

–f Send a 'finish' command to a remote server.

–c Establish a terminal connection over the specified or default communication line, before any protocol transaction takes place. Get back to the local system by typing the escape character

(normally Control-Backslash) followed by the letter 'c'.

–n Like **–c**, but after a protocol transaction takes place; **–c** and **–n** may both be used in the same command. The use of **–n** and **–c** is illustrated below.

On a timesharing system, the **–l** and **–b** options will also have to be included with the **–r**, **–k**, or **–s** options if the other **Kermit** is on a remote system.

If **kermit** is in local mode, the screen (stdout) is continously updated to show the progress of the file transer. A dot is printed for every four data packets, other packets are shown by type (e.g. 'S' for Send-Init), 'T' is printed when there's a timeout, and '%' for each retransmission. In addition, you may type (to stdin) certain "interrupt" commands during file transfer:

Control-F: Interrupt the current File, and go on to the next (if any).

Control-B: Interrupt the entire Batch of files, terminate the transaction.

Control-R: Resend the current packet

Control-A: Display a status report for the current transaction.

These interrupt characters differ from the ones used in other **Kermit** implementations to avoid conflict with Unix shell interrupt characters. With System III and System V implementations of Unix, interrupt commands must be preceeded by the escape character (e.g. control-\).

Several other command-line options are provided:

–i Specifies that files should be sent or received exactly "as is" with no conversions. This option is necessary for transmitting binary files. It may also be used to slightly boost efficiency in Unix-to-Unix transfers of text files by eliminating CRLF/newline conversion.

–w Write-Protect — Avoid filename collisions for incoming files.

–q Quiet — Suppress screen update during file transfer, for instance to allow a file transfer to proceed in the background.

–d Debug — Record debugging information in the file debug.log in the current directory. Use this option if you believe the program is misbehaving, and show the resulting log to your local **Kermit** maintainer.

–h Help — Display a brief synopsis of the command line options.

The command line may contain no more than one protocol action option.

INTERACTIVE OPERATION

Kermit's interactive command prompt is "C-Kermit>". In response to this prompt, you may type any valid command. **Kermit** executes the command and then prompts you for another command. The process continues until you instruct the program to terminate.

Commands begin with a keyword, normally an English verb, such as "send". You may omit trailing characters from any keyword, so long as you specify sufficient characters to distinguish it from any other keyword valid in that field. Certain commonly-used keywords (such as "send", "receive", "connect") have special non-unique abbreviations ("s" for "send", "r" for "receive", "c" for "connect").

Certain characters have special functions in interactive commands:

? Question mark, typed at any point in a command, will produce a message explaining what is possible or expected at that point. Depending on the context, the message may be a brief phrase, a menu of keywords, or a list of files.

ESC (The Escape or Altmode key) — Request completion of the current keyword or filename, or insertion of a default value. The result will be a beep if the requested operation fails.

DEL (The Delete or Rubout key) — Delete the previous character from the command. You may also

use BS (Backspace, Control-H) for this function.

^W (Control-W) — Erase the rightmost word from the command line.

^U (Control-U) — Erase the entire command.

^R (Control-R) — Redisplay the current command.

SP (Space) — Delimits fields (keywords, filenames, numbers) within a command. HT (Horizontal Tab) may also be used for this purpose.

CR (Carriage Return) — Enters the command for execution. **LF** (Linefeed) or **FF** (formfeed) may also be used for this purpose.

\ (Backslash) — Enter any of the above characters into the command, literally. To enter a backslash, type two backslashes in a row (\\). A single backslash immediately preceding a carriage return allows you to continue the command on the next line.

You may type the editing characters (DEL, ^W, etc) repeatedly, to delete all the way back to the prompt. No action will be performed until the command is entered by typing carriage return, linefeed, or formfeed. If you make any mistakes, you will receive an informative error message and a new prompt — make liberal use of '?' and ESC to feel your way through the commands. One important command is "help" — you should use it the first time you run **Kermit.**

Interactive **Kermit** accepts commands from files as well as from the keyboard. When you enter interactive mode, **Kermit** looks for the file .kermrc in your home or current directory (first it looks in the home directory, then in the current one) and executes any commands it finds there. These commands must be in interactive format, not Unix command-line format. A "take" command is also provided for use at any time during an interactive session. Command files may be nested to any reasonable depth.

Here is a brief list of **Kermit** interactive commands:

! Execute a Unix shell command.

bye Terminate and log out a remote **Kermit** server.

close Close a log file.

connect Establish a terminal connection to a remote system.

cwd Change Working Directory.

dial Dial a telephone number.

directory Display a directory listing.

echo Display arguments literally.

exit Exit from the program, closing any open logs.

finish Instruct a remote **Kermit** server to exit, but not log out.

get Get files from a remote **Kermit** server.

help Display a help message for a given command.

log Open a log file — debugging, packet, session, transaction.

quit Same as 'exit'.

receive Passively wait for files to arrive.

remote Issue file management commands to a remote **Kermit** server.

script	Execute a login script with a remote system.
send	Send files.
server	Begin server operation.
set	Set various parameters.
show	Display values of 'set' parameters.
space	Display current disk space usage.
statistics	Display statistics about most recent transaction.
take	Execute commands from a file.

The 'set' parameters are:

block-check	Level of packet error detection.
delay	How long to wait before sending first packet.
duplex	Specify which side echoes during 'connect'.
escape-character	Character to prefix "escape commands" during 'connect'.
file	Set various file parameters.
flow-control	Communication line full-duplex flow control.
handshake	Communication line half-duplex turnaround character.
line	Communication line device name.
modem-dialer	Type of modem-dialer on communication line.
parity	Communication line character parity.
prompt	Change the **Kermit** program's prompt.
receive	Set various parameters for inbound packets.
send	Set various parameters for outbound packets.
speed	Communication line speed.

The 'remote' commands are:

cwd	Change remote working directory.
delete	Delete remote files.
directory	Display a listing of remote file names.
help	Request help from a remote server.
host	Issue a command to the remote host in its own command language.
space	Display current disk space usage on remote system.
type	Display a remote file on your screen.
who	Display who's logged in, or get information about a user.

FILES

$HOME/.kermrc	*Kermit* initialization commands
./.kermrc	more *Kermit* initialization commands

SEE ALSO

cu(1C), uucp(1C)
Frank da Cruz and Bill Catchings, *Kermit User's Guide*, Columbia University, 6th Edition

DIAGNOSTICS

The diagnostics produced by **Kermit** itself are intended to be self-explanatory.

BUGS

See recent issues of the Info-Kermit digest (on ARPANET or Usenet), or the file ckuker.bwr, for a list of bugs.

MH Mail System

Refer to Section 8 in the UNIX User's Supplementary Documents (USD:8) manual for information on the MH Mail System.

NAME

checknews – check to see if user has news

SYNOPSIS

checknews [**ynqevvN**] [*newsgroup list*] [*readnews options*]

DESCRIPTION

Checknews reports to the user whether or not he has news.

–y reports "There is news" if the user has news to read. If the **–N** flag is given, then the newsgroups requested are also printed.

–n reports "No news" if there isn't any news to read.

–q causes **checknews** to be quiet. Instead of printing a message, the exit status indicates news. A status of 0 means no news, 1 means there is news.

–v alters the **–y** message to show the name of the first newsgroup containing unread news. Doubling v (e.g. **–vv**) will cause an explanation of *any* claim of new news, and is useful if **checknews** and **readnews**(1) disagree on whether there is news.

–e executes **readnews** if there is news.

–N causes the next argument to be read and interpreted as a comma-separated list of newsgroups to be checked.

If there are no options, **–y** is the default.

FILES

~/.newsrc	Active newsgroups
/usr/lib/news/active	Options and list of previously read articles

SEE ALSO

inews(1), **postnews**(1), **readnews**(1), **vnews**(1), **news**(5), **newsrc**(5), **expire**(8), **recnews**(8), **sendnews**(8), **uurec**(8)

BUGS

The **–N** flag should really be named **–n** to be consistent with other news programs, but **–n** was already used. If the **–v** flag is used with the **–N** flag, the first newsgroup in the list where there is news should be printed instead of the entire list. If the **–N** flag is used and *readnews* is invoked (with **–e**) it does not restrict news reading to those groups checked, but reads all newsgroups where there is new news.

NAME

expire – remove outdated news articles

SYNOPSIS

/usr/lib/news/expire [**–n** *newsgroups*] [**–i**] [**–I**] [**–a**] [**–v** *level*] [**–p**]
[**–h**] [**–r**] [**–e** *days*] [**–E** *days*]
/usr/lib/news/expire –f *user@site.DOMAIN*
/usr/lib/news/expire –u

DESCRIPTION

Expire is the program that removes out-of-date news articles from your system. You need to use a special program to do this, instead of just using **find**(1) or **rm**(1), because of the history file. If you just delete messages, then the history file will become incorrect because it will think that they are still there.

The normal use of **expire** is to run it at regular intervals with no options. It will remove all articles whose expiration date has passed. If you have a lot of disk space, you can run it once a week. If disk space is tight, you might want to run it every night. The length of time that it takes to run depends, of course, on many factors; on a VAX 750 with a 15-day expiration period and the volume of news that is typical in 1986 (about 5000 articles per week), **expire** will take roughly an hour to run.

Expire has the following options:

–n Specify certain newsgroups whose articles will be expired. The other newsgroups will be left alone. The notation that you use with the –n option is quite similar to that used in the sys file. To expire only the articles in net.origins, leaving everything else alone, type this:

```
/usr/lib/news/expire –n net.origins
```

To expire only the articles in net.micro, but leave net.micro.pc and net.micro.mac alone, type this:

```
/usr/lib/news/expire –n net.micro !net.micro.mac !net.micro.pc
```

For compatibility with the syntax of the sys file, you can also type the command this way, with commas instead of spaces between the fields.

```
/usr/lib/news/expire –n net.micro,!net.micro.mac,!net.micro.pc
```

If you have certain groups that you use as archives, which should never have their articles expired, you must construct an *expire* command that mentions all groups except your archive groups. When doing this, be sure not to forget the groups "junk", "control", and "general". A likely command would be:

```
/usr/lib/news/expire –n all,!local.source,!mod.sources
```

–e Specify an expiration period. Normally **expire** removes articles that are older than 15 days. If you would like it to remove articles that are older than 5 days, you can type

```
/usr/lib/news/expire –e 5
```

If you would like it to remove articles from net.religion and net.politics that are older than 23 days, and leave everything else alone, you can type

```
/usr/lib/news/expire –e 23 –n net.religion net.politics
```

You can specify the –e option as –e15 instead of as –e 15 if you want; this is for compatibility with old versions and old habits.

–E Normally **expire** removes the record of an article from the history file at the same time it removes the article. One of the purposes of the history file is to prevent articles from being duplicated if a second copy arrives a while later, perhaps over some other path. If your site is extremely short on disk space, forcing you to specify a short expiration period in the –e option, you can use the –E option to ask that the information in the history file be kept round a bit longer, until the danger of duplicate arrival has passed. The command

```
/usr/lib/news/expire –e 7 –E 21
```

Causes articles that are 7 or more days old to be removed, and history information that is 21 or more days old to be removed. If you use the –E option, make sure that the value it specifies is always larger than the –e option value, else you will end up with articles that are not in the history file; this can cause problems.

–a Asks that articles be archived (usually in /usr/spool/oldnews) instead of being deleted. An example of its use would be

/usr/lib/news/expire –e 30 –a net.sources,mod.sources,!net.sources.bugs

–a may be used with **–n** . If no pattern is given for **–a** , all newsgroups specified by **–n** will be archived.

–I instructs *expire* to ignore expiration dates stored in articles, and to look at the number of days that have passed since the article was received. Not very many articles have expiration dates in them.

–i is like –I, but it will look at the number of days that have passed and also at the explicit expiration date, and it will remove the article if either of those has passed.

–v sets the verbosity mode. If you have specified a complex collection of options and they are not having the effect that you would like, then set –v2 or –v3 to find out what is going on. Values from 0 to 6 are meaningful, and –v1 is the default. –v0 will turn off messages, and –v6 will cause **expire** to print every possible message.

–p causes **expire** to use the date the article was posted, rather than the date it arrived at your machine, as the basis for expiration. Every now and then there is a ''time warp'' that causes a batch of very very old news to be dumped onto the network; judicious use of the –p option can eradicate it.

–f asks **expire** to remove messages sent by a particular user, regardless of the newsgroup that they are in, and regardless of how old they are. This option is intended not so much to selectively censor voluminous posters (though it has certainly been used for that) but to recover when a *notesfiles* site (running different news software) accidentally releases a duplicate batch of old news. An example of its use is

/usr/lib/news/expire –f rlr@pyuxd.UUCP

Any article whose From: field exactly matches the argument to the –f option will be removed.

–h causes **expire** to ignore the history file, and do its expiration by looking at every article file in the spool directory. This is phenomenally slow—it can take 5 or 6 hours on an otherwise idle Vax 750—but if your history file is damaged and you cannot use **find**(1) because you are relying on expiration dates stored inside articles, then you have no other choice.

–r causes **expire** to rebuild the history file in addition to doing expiration. The –r option implies the –h option; it scans every article in the spool directory and builds a new set of history and **dbm**(3X) files. It also performs expiration, so if you want to rebuild the history file while preserving all articles (as you might want to do on an archival file computer), you must specify

/usr/lib/news/expire –r –I –e 999999

to prevent expiration from taking place. If you do not rely on expiration dates stored inside articles, it is a good tonic to run the following sequence of commands once every now and then:

find /usr/spool/news –size 0 –o –mtime +90 –exec rm –f {} ;
/usr/lib/news/expire –r

This will remove junk files that have somehow managed to find their way into the spooling directory, and then it will rebuild the history file.

–u causes the minimum article-number field in the active file to be updated. This is used when converting from 2.10.1 news to later versions.

SEE ALSO

inews(1), **postnews**(1), **getdate**(3), **news**(5), **recnews**(8), **sendnews**(8), **uurec**(8)

BUGS

If **inews**(1) is run while **expire** is running, it can cause the article that inews is trying to insert to be absent from your history file. There is no automatic interlock between **inews**(1) and **expire,** so you must take care to turn off **inews**(1) while **expire** is running. This bug will likely be fixed soon, but for the moment be careful of it.

The newsgroup pattern argument to the –n option is limited to 1024 characters, which is about 8 lines of text.

NAME

getdate – convert time and date from ASCII

SYNOPSIS

```
#include <sys/types.h>
#include <sys/timeb.h>

time_t getdate(buf, now)
char *buf;
struct timeb *now;
```

DESCRIPTION

Getdate is a routine that converts most common time specifications to standard UNIX format. The first argument is the character string containing the time and date; the second is the assumed current time (used for relative specifications); if **NULL** is passed, **ftime**(3C) is used to obtain the current time and timezone.

The character string consists of 0 or more specifications of the following form:

tod — A *tod* is a time of day, which is of the form *hh*[:*mm*[:*ss*]] (or *hhmm*) [*meridian*] [*zone*]. If no meridian – **am** or **pm** – is specified, a 24-hour clock is used. A *tod* may be specified as just *hh* followed by a *meridian*.

date — A *date* is a specific month and day, and possibly a year. Acceptable formats are *mm/dd*[/*yy*] and *monthname dd*[, *yy*] If omitted, the year defaults to the current year; if a year is specified as a number less than 100, 1900 is added. If a number not followed by a day or relative time unit occurs, it will be interpreted as a year if a *tod*, *monthname*, and *dd* have already been specified; otherwise, it will be treated as a *tod*. This rule allows the output from **date**(1) or **ctime**(3) to be passed as input to *getdate*.

day — A *day* of the week may be specified; the current day will be used if appropriate. A *day* may be preceded by a *number*, indicating which instance of that day is desired; the default is **1**. Negative *numbers* indicate times past. Some symbolic *numbers* are accepted: **last**, **next**, and the ordinals **first** through **twelfth** (**second** is ambiguous, and is not accepted as an ordinal number). The symbolic number **next** is equivalent to **2**; thus, **next monday** refers not to the immediately coming Monday, but to the one a week later.

relative time — Specifications relative to the current time are also accepted. The format is [*number*] *unit*; acceptable units are **year**, **month**, **fortnight**, **week**, **day**, **hour**, **minute**, and **second**.

The actual date is formed as follows: first, any absolute date and/or time is processed and converted. Using that time as the base, day-of-week specifications are added; last, relative specifications are used. If a date or day is specified, and no absolute or relative time is given, midnight is used. Finally, a correction is applied so that the correct hour of the day is produced after allowing for daylight savings time differences.

Getdate accepts most common abbreviations for days, months, etc.; in particular, it will recognize them with upper or lower case first letter, and will recognize three-letter abbreviations for any of them, with or without a trailing period. Units, such as **weeks**, may be specified in the singular or plural. Timezone and meridian values may be in upper or lower case, and with or without periods.

FILES

/usr/lib/libu.a

SEE ALSO

ctime(3), **time**(2)

AUTHOR

Steven M. Bellovin (unc!smb)
Dept. of Computer Science
University of North Carolina at Chapel Hill

BUGS

Because **yacc**(1) is used to parse the date, **getdate** cannot be used a subroutine to any program that also needs **yacc**.

The grammar and scanner are rather primitive; certain desirable and unambiguous constructions are not accepted. Worse yet, the meaning of some legal phrases is not what is expected; **next week** is identical to **2 weeks**.

The daylight savings time correction is not perfect, and can get confused if handed times between midnight and 2:00 am on the days that the reckoning changes.

Because **localtime**(2) accepts an old-style time format without zone information, attempting to pass **getdate** a current time containing a different zone will probably fail.

NAME

inews – submit news articles

SYNOPSIS

inews [**–h**] **–t** *title* **–n** *newsgroups* [**–e** *expiration date*] [**–f** *sender name*] [**–d** *distribution*] [**–F** *references*] [**–o** *organization*] [**–M**] [**–a** *approvedby*]

inews –p *filename*

inews –C *newsgroup*

DESCRIPTION

Inews submits news articles to the USENET news network. It is a raw interface called by news-posting programs. You should not use **inews** directly. Most people use **postnews**(1) to post news articles. Ultimately, of course, **postnews**(1) and other news-posting programs call inews to do the actual submission.

The first form (no **–p** or **–C** options) is for submitting ordinary articles. The body of the article will be read from the standard input. A *title* ("Subject:" field) must be specified (there is no default). Each article is posted to one or more newsgroups. **–n** flag is omitted, the list will default to something like **general**. If you wish to submit an article to multiple newsgroups, the *newsgroups* must be separated by commas and/or spaces.

The **–e** flag is used to override the default expiration date. This is seldom used.

The **–f** flag specifies the article's sender. Without this flag, the sender defaults to the user's name. If **–f** is specified, the real sender's name will be included as a "Sender:" line to prevent forged articles.

The **–d** flag allows you to specify the maximum geographic distribution of your article; for example, a distribution of "aus" limits distribution to Australia, and a distribution of "nj" limits distribution to New Jersey. There is no way to send a message from California for distribution only in New Jersey—your machine must be in the distribution that you ask for.

The **–F** flags is used to attach a list of related articles that this message references; it creates the "In-reply-to:" field of the posted article.

The **–o** is used to override the default organization name.

The **–M** and **–a** flags are to be used only by the moderator of a moderated newsgroup. The **–M** flag causes the "From:" and "Path:" fields of the article to be set to correct values for a moderated newsgroup. The **–a** flag is used to add an "Approved:" line to the header. Note that if the **–M** flag is used in conjuction with the **–h** flag (see below), the article headers must not have a "Path:" field in them already.

The **–h** flag specifies that headers are present at the beginning of the article, and these headers should be included with the article header instead of as text. Everything before the first blank line in the article is taken as a header field, and everything after that blank line is taken to be part of the body of the message. (This mechanism can be used to edit headers and supply additional nondefault headers, but not to specify certain information, such as the sender and article ID, that **inews** itself generates.) **Inews** will ignore nonstandard and misspelled header fields entered with the –h option.

When posting an article **inews** checks the environment for certain information about the sender. If an environment variable **NAME** is defined, **inews** uses its value as the full name of the poster. If **NAME** is not defined, **inews** checks *$HOME/.name* is checked and if it exists, its contents are used as the full name. Otherwise, the system value (often in /etc/passwd) is used. This is useful if the system value cannot be set, or when more than one person uses the same login. If the environment variable **ORGANIZATION** is defined, then **inews** uses its value instead of the system default organization name. If its value begins with a "/", then it is taken to be a file name, and **inews** takes the name of the organization from the contents of the file. This is useful when a person uses a guest login and is not primarily associated with the organization that owns the machine.

The second form (**inews –p**) is used for receiving articles from other machines. If *filename* is given, the article will be read from the file of that name; otherwise the article will be read from the standard input. An expiration date need not be present and a reception date, if present, will be ignored.

When **inews** receives an article this way, it will check the history file to make sure that the article is not already present, and it will make certain consistency checks to make sure that the newsgroup names are legal and that the sys file permits the article to be installed on the local machine. Once the article passes those checks, it is installed in the appropriate directory on the local machine. If the article fails those checks, it is installed in newsgroup "junk" on the local machine. In any event, **inews** will then transmit the article to all systems that match in the sys file and are not mentioned in the "Path:" field of the just-posted message. The details of this transmission are determined by the contents of the sys file.

The third form (*inews –C*) is for creating new newsgroups. The use of this feature is limited to certain users such as the super-user or news administrator. Please note that *inews –C* creates a newsgroup *on all machines that the message reaches*, and not just the local machine. If you accidentally create a newsgroup with *inews –C*, without specifying a distribution, it will be created worldwide. If you want to create a newsgroup locally on your machine, it is safer to edit the active file by hand.

If the file /usr/lib/news/recording is present, it is taken as a list of "recordings" to be shown to users posting news. (This is named after the recording you hear when you dial "information" in some parts of the U.S., asking you to stop and think if you really want do do this, but not actually preventing you.) The recording file contains lines of the form:

newsgroup-specifier TAB filename

for example:

net.all net.recording
local.all,!local.test local.recording

Any user posting an article to a newsgroup matching the pattern on the left will be shown the contents of the file on the right. The file is found in the **LIB** directory (often /usr/lib/news). The user is then told to hit DEL to abort or RETURN to proceed. The intent of this feature is to help companies keep proprietary information from accidently leaking out.

FILES

/usr/spool/news/.sys.nnn	temporary articles
/usr/spool/news/*newsgroups*/*article_no*.	Articles
/usr/lib/news/active	List of known newsgroups and highest local article numbers in each.
/usr/lib/news/seq	Sequence number of last article
/usr/lib/news/history	List of all articles currently stored on this machine.
/usr/lib/news/sys	System subscription list
/usr/lib/news/distributions	Suggested distribution code names

SEE ALSO

Mail(1), **binmail**(1), **mailx**(1), **checknews**(1), **msgs**(1), **postnews**(1), **readnews**(1), **vnews**(1), **getdate**(3), **news**(5), **newsrc**(5), **expire**(8), **recnews**(8), **sendnews**(8), **uurec**(8)

AUTHORS

Matt Glickman
Mark Horton
Stephen Daniel
Tom Truscott
Rick Adams

NAME

news – USENET network news article, utility files

DESCRIPTION

There are two formats of news articles: **A** and **B**. **A** format is the only format that version 1 netnews systems can read or write. Systems running the version 2 netnews can read either format and there are provisions for the version 2 netnews to write in **A** format. **A** format looks like this:

A*article-ID*
newsgroups
path
date
title
Body of article

Only version 2 netnews systems can read and write **B** format. **B** format contains two extra pieces of information: receipt date and expiration date. The basic structure of a **B** format file consists of a series of headers and then the body. A header field is defined as a line with a capital letter in the first column and a colon somewhere on the line. Unrecognized header fields are ignored. News is stored in the same format transmitted, see *Standard for the Interchange of USENET Messages* for a full description. The following fields are among those recognized:

From: *user@host.domain[.domain ...] (Full Name)*
Newsgroups: *Newsgroups*
Message-ID: *<Unique Identifier>*
Subject: *descriptive title*
Date: *Date Posted*
Expires: *Expiration Date*
Reply-To: *Address for mail replies*
References: *Article ID of article this is a follow-up to.*
Control: *Text of a control message*

Here is an example of an article:

```
Path: cbosgd!mhuxj!mhuxt!eagle!jerry
From: jerry@eagle.uucp (Jerry Schwarz)
Newsgroups: net.general
Subject: Usenet Etiquette -- Please Read
Message-ID: <642@eagle.UUCP>
Date: Friday, 19 Nov 82 16:14:55 EST
Followup-To: net.news
Expires: Saturday, 1 Jan 83 00:00:00 EST
Organization: Bell Labs, Murray Hill

The body of the article comes here, after a blank line.
```

The *sys* file line has four fields, each separated by colons:

system-name:*subscriptions*:*flags*:*transmission command*

Of these fields, only the *system-name* and *subscriptions* need to be present.

The *system name* is the name of the system being sent to. The *subscriptions* is the list of newsgroups to be transmitted to the system. The *flags* are a set of letters describing how the article should be transmitted. The default is **B**. Valid flags include **A** (send in **A** format), **B** (send in **B** format), **N** (use "ihave/sendme" protocol), **U** (use "uux –c" and the name of the stored article in a "%s" string).

The *transmission command* is executed by the shell with the article to be transmitted as the standard input. The default is "uux – –z –r *sysname*!rnews". Some examples:

xyz:net
oldsys:net,mod,to.oldsys:A
berksys:net,ucb::/usr/lib/news/sendnews –b berksys:rnews
arpasys:net,arpa::/usr/lib/news/sendnews –a rnews@arpasys
old2:net,mod:A:/usr/lib/sendnews –o old2:rnews
user:net.sf-lovers::mail user

Somewhere in the *sys* file, there must be a line for the host system. This line has no *flags* or *commands*. A "#" as the first character in a line denotes a comment.

The *history*, *active*, and *ngfile* files have one line per item.

SEE ALSO

checknews(1), **inews**(1), **postnews**(1), **readnews**(1), **vnews**(1), **getdate**(3), **expire**(8), **recnews**(8), **sendnews**(8), **uurec**(8)

NAME

newsrc – information file for **readnews**(1) and **checknews**(1)

DESCRIPTION

The .newsrc file contains the list of previously read articles and an optional options line for **readnews**(1) and **checknews**(1). Each newsgroup that articles have been read from has a line of the form:

newsgroup: *range*

The *range* is a list of the articles read. It is basically a list of numbers separated by commas with sequential numbers collapsed with hyphens. For instance:

general: 1-78,80,85-90
mod.computers.laser-printers: 1-7
net.news: 1
mod.ai! 1-5

If the ":" is replaced with an "!" (as in **mod.ai** above) the newsgroup is not subscribed to and will not be shown to the user.

An options line starts with the word **options** (left-justified). Then there are the list of options just as they would be on the command line. For instance:

options –n all !net.sf-lovers !mod.human-nets –r
options –c –r

A string of lines beginning with a space or tab after the initial options line will be considered continuation lines.

FILES

~/.newsrc options and list of previously read articles

SEE ALSO

checknews(1), **readnews**(1), **vnews**(1)

NAME

postnews – submit news articles

SYNOPSIS

postnews [*article*]

DESCRIPTION

Postnews is a program that calls **inews**(1) to submit news articles to USENET. The commands should be self-explanatory, however you may type "?" to most prompts to get a list of the possible options (obviously, except for the "Subject" of the article, "Keywords", etc). It will prompt the user for the title of the article (which should be a phrase suggesting the subject, so that persons reading the news can tell if they are interested in the article), for the newsgroup, and for the distribution.

The distribution is typically a geographic region or corporate region. Typing "?" will get you a list of the possible distributions. You should use the minimum distribution that will serve your purpose for posting the article. For example, if you are selling your car in New Jersey, it is doubtful that someone in California (or Europe) would be willing to buy it. If you don't restrict the distribution to your local area, you will cause this article to be transmitted unnecessarily around the world. Currently, with a distribution of **world**, the article will be seen in the United States, Canada, Europe, Japan, Korea and other places. A distribution header will, if given, be included in the headers of the article, affecting where the article is distributed to.

After entering the title, newsgroup, and distribution, the user will be placed in an editor. If **EDITOR** is set in the environment, that editor will be used. Otherwise, **postnews** defaults to **vi**(1).

An initial set of headers containing the subject and newsgroups will be placed in the editor, followed by a blank line. The article should be appended to the buffer, after the blank line. The initial headers can be changed, or additional headers added, while in the editor, if desired.

After you have finished typing in your article, you have the option of sending it, listing it, quitting without sending it or saving it in a file so you can finish editing it and post it later.

For posting news from a program, see *inews*(1).

SEE ALSO

Mail(1), **binmail**(1), **checknews**(1), **inews**(1), **mailx**(1), **readnews**(1), **vi**(1), **news**(5), **newsrc**(5), **expire**(8), **recnews**(8), **sendnews**(8), **uurec**(8)

NAME

readnews – read news articles

SYNOPSIS

readnews [**–a** *date*] [**–n** *newsgroups*] [**–t** *titles*] [**–leprxhfuM**] [**–c** [*mailer*]]

readnews –s

DESCRIPTION

Readnews without argument prints unread articles. There are several interfaces available other than the default:

Flag	Interface
–M	An interface to **mailx**(1).
–c	A **binmail**(1)–like interface.
–c *''mailer''*	All selected articles written to a temporary file. Then the mailer is invoked. The name of the temporary file is referenced with a ''%''. Thus, ''mail –f %'' will invoke mail on a temporary file consisting of all selected messages.
–p	All selected articles are sent to the standard output. No questions asked.
–l	Only the titles output. The *.newsrc* file will not be updated.
–e	Like **–l** but also updates the *.newsrc* file.

The **–r** flag causes the articles to be printed in reverse order. The **–f** flag prevents any followup articles from being printed. The **–h** flag causes articles to be printed in a less verbose format, and is intended for terminals running at 300 baud. The **–u** flag causes the *.newsrc* file to be updated every 5 minutes, in case of an unreliable system. (Note that if the .newsrc file is updated, the **x** command will not restore it to its original contents.)

The following flags determine the selection of articles.

–n *newsgroups*	Select all articles that belong to *newsgroups*.
–t *titles*	Select all articles whose titles contain one of the strings specified by *titles*.
–a [*date*]	Select all articles that were posted past the given *date* (in *getdate*(3) format).
–x	Ignore .newsrc file. That is, select articles that have already been read as well as new ones.

Readnews maintains a .newsrc file in the user's home directory that specifies all news articles already read. It is updated at the end of each reading session in which the **–x** or **–l** options weren't specified. If the environment variable **NEWSRC** is present, it should be the path name of a file to be used in place of *.newsrc*.

If the user wishes, an options line may be placed in the .newsrc file. This line starts with the word **options** (left justified) followed by the list of standard options just as they would be typed on the command line. Such a list may include: the **–n** flag along with a newsgroup list; a favorite interface; and/or the **–r** or **–t** flag. Continuation lines are specified by following lines beginning with a space or tab character. Similarly, options can be specified in the **NEWSOPTS** environment parameter. Where conflicts exist, option on the command line take precedence, followed by the .newsrc **options** line, and lastly the **NEWSOPTS** parameter.

You can use the **–s** flag to print the newsgroup subscription list.

When the user uses the reply command of the default or **binmail**(1) interfaces, the environment parameter **MAILER** will be used to determine which mailer to use. The default is **mail**(1).

If the user so desires, he may specify a specific paging progam for articles. The environment parameter **PAGER** should be set to the paging program. The name of the article is referenced with a "%", as in the **–c** option. If no "%" is present, the article will be piped to the program. Paging may be disabled by setting **PAGER** to a null value. By default, the pager is *cat*(1).

COMMANDS

This section lists the commands you can type to the default and *binmail* interface prompts. The default interface will suggest some common commands in brackets. Just hitting return is the same as typing the first command. For example, "[ynq]" means that the commands "y" (yes), "n" (no), and "q" (quit) are common responses, and that "y" is the default.

Command	Meaning
–	Go back to last article. This is a toggle, typing it twice returns you to the original article.
#	Report the name and size of the newsgroup.
!	Shell escape.
<message ID>	Look for a particular article. (See *Standard for Interchange of Usenet Messages* for a description of message ID's).
b	Back. Back up one article.
c	Cancel the article. Only the author or the super user can do this.
d	Read a digest. Breaks up a digest into separate articles and permits you to read and reply to each piece.
D	Decrypt. Invokes a Caesar decoding program on the body of the message. This is used to decrypt rotated jokes posted to **net.jokes**. Such jokes are usually obscene or otherwise offensive to some groups of people, and so are rotated to avoid accidental decryption by people who would be offended. The title of the joke should indicate the nature of the problem, enabling people to decide whether to decrypt it or not. An explicit *number* rotation (usually 13) may be given to force a particular shift.
e	Erase. Forget that this article was read.
f [*title*]	Submit a follow up article. Normally you should leave off the title, since the system will generate one for you. You will be placed in your **EDITOR** to compose the text of the followup.
fd	Followup directly, without edited headers. This is like **f**, but the headers of the article are not included in the editor buffer.
h	Header. Print a more verbose header.
H	Print a very verbose header, containing all known information about the article.
K	Kill. Mark all remaining articles in this newsgroup as read and skip to the next newsgroup.
n	No. Goes on to next article without printing current one. In the **binmail** interface, this means "go on to the next article", which will have the same effect as **y** or just hitting return.
N [*newsgroup*]	Next Newsgroup. Go to the next newsgroup or named newsgroup.
p	Print. Reprint previous article.
P	Previous Newsgroup. Go back to previous newsgroup.
q	Quit. The *.newsrc* file will be updated if **–l** or **–x** were not on the command line.
r	Reply. Reply to article's author via **mail**. You are placed in your **EDITOR** (by default vi(1)) with a header specifying "To", "Subject", and "References" lines taken from the message.

You may change or add headers, as appropriate. You add the text of the reply after the blank line, and then exit the editor. The resulting message is mailed to the author of the article.

rd Reply directly. You are placed in **MAILER** (**mail** by default) in reply to the author. Type the text of the reply and then control-D.

s [*file*] Save. The article is appended to the named file. The default is *Articles*. If the first character of the file name is "|", the rest of the file name is taken as the name of a program, which is executed with the text of the article as standard input. If the first character of the file name is "/", it is taken as a full path name of a file. If **NEWSBOX** (in the environment) is set to a full path name, and the file contains no "/", the file is saved in **NEWSBOX**. Otherwise, it is saved relative to **HOME**.

U Unsubscribe from this newsgroup. Also goes on to the next newsgroup.

v Print the current version of the news software.

w Same as **s**.

x Exit. Like quit except that .newsrc is not updated.

X *system*
Transmit article to the named system.

y Yes. Prints current article and goes on to next.

number Go to *number*.

+[*n*] Skip *n* articles. The articles skipped are recorded as "unread" and will be offered to you again the next time you read news.

The commands **c, f, fd, r, rd, e, h, H,** and **s** can be followed by –'s to refer to the previous article. Thus, when replying to an article using the default interface, you should normally type **r–** (or **re-**) since by the time you enter a command, you are being offered the next article.

EXAMPLES

readnews Read all unread articles using the default interface. The .newsrc file is updated at the end of the session.

readnews –c "ed %" –l
Invoke the **ed**(1) text editor on a file containing the titles of all unread articles. The **not** updated at the end of the session.

readnews –n all !fa.all –M –r
Read all unread articles except articles whose newsgroups begin with **fa.** via *mailx* in reverse order. The .newsrc file is updated at the end of the session.

readnews –p –n all –a last thursday
Print every unread article since last Thursday. The .newsrc file is updated at the end of the session.

readnews –p > /dev/null &
Discard all unread news. This is useful after returning from a long trip.

ENVIRONMENT VARIABLES

EDITOR
Editor invoked by **f** command. (Default is */usr/ucb/vi*.)

MAILER
Mailing program invoked by the **r** command. (Default is */bin/mail*.)

NAME Your full name used in header of articles posted by you. (Default is the comments field of your id in */etc/passwd*.)

NEWSBOX

File or directory where articles saved with the s command are stored. (Default is same as **HOME**.)

NEWSOPTS
Options for *readnews*.

ORGANIZATION
Full name of this site used header of articles posted by you.

PAGER
Paging program invoked by articles with more than 16 lines. (Default is */usr/ucb/more*.)

SHELL
The shell invoked by the **!** command. (Default is */bin/sh*.)

FILES

/usr/spool/news/*newsgroup/number*	News articles
/usr/lib/news/active	Active newsgroups and numbers of articles
/usr/lib/news/help	Help file for default interface
~/.newsrc	Options and list of previously read articles

SEE ALSO
binmail(1), **checknews**(1), **inews**(1), **mail**(1), **mailx**(1), **news**(5), **newsrc**(5), **postnews**(1), **vnews**(1), **getdate**(3), **news**(5), **newsrc**(5), **expire**(8), **recnews**(8), **sendnews**(8), **uurec**(8)
How to Read the Network News by Mark Horton.
Standard for Interchange of Usenet Messages by Mark Horton.

AUTHORS
Matt Glickman
Mark Horton
Stephen Daniel
Tom R. Truscott

NAME

recnews – receive unprocessed articles via mail

SYNOPSIS

/usr/lib/news/recnews [*newsgroup* [*sender*]]

DESCRIPTION

Recnews reads a letter from the standard input; determines the article title, sender, and newsgroup; and gives the body to inews with the right arguments for insertion.

If *newsgroup* is omitted, the "To:" line of the letter will be used. If *sender* is omitted, the sender will be determined from the "From" line of the letter. The title is determined from the "Subject" line.

SEE ALSO

checknews(1), **inews**(1), **postnews**(1), **readnews**(1), **vnews**(1), **news**(5), **sendnews**(8), **uurec**(8)

NAME

sendbatch – send news articles in batches

SYNOPSIS

sendbatch [**–s** *size*] [**–c**] [**–c7**] [*–obBC*] *site*

DESCRIPTION

Sendbatch prepares and transmits a batch of news for unbatching by another machine.

If **sendbatch** is called with no flags, it will submit the batched news (from */usr/spool/news/site*) queued up for **site** to *uux* for transmission and exectution on the remote machine.

Several flags are available to modify sendbatch's behavior.

The **–c** flag says to use compression to reduce the site of the transmitted file.

The **–c7** flag is normally used for sending articles over X.25 links. It will cause the batch of news to be compressed, then piped through *encode* to reduce the 8-bit data into 7-bit characters.

The **–s** flag is used to modify the size of the batches. By default, the batch size is 50,000 bytes.

The **–o** flag is used with the **–c** flag to produce and old style compress batch suitable for systems with version 2.10.2 news.

The **–bBC** flags are passed directly to compress and may be used to change its compression algorithms if necessary.

EXAMPLES

To send to a pdp11/70 running 2.10.2 compressed batching:

```
sendbatch –c –C –b12 -o sitename
```

To send to a Vax running 2.10.3 batching:

```
sendbatch –c sitename
```

To send to a Vax over an X.25 network using batches of 30,000 bytes:

```
sendbatch –c7 –s30000 sitename
```

SEE ALSO

compress(1)

NAME

sendnews – send news articles via mail

SYNOPSIS

sendnews [**–o**] [**–a**] [**–b**] [**–n** *newsgroups*] *destination*

DESCRIPTION

Sendnews reads an article from its standard input, performs a set of changes to it, and gives it to the mail program to mail it to *destination*.

An "N" is prepended to each line for decoding by **uurec**(1).

The **–o** flag handles old format articles.

The **–a** flag is used for sending articles via the ARPANET. It maps the article's path from *uucphost*!*xxx* to *xxx*@*arpahost*.

The **–b** flag is used for sending articles via the Berknet. It maps the article's path from *uucphost*!*xxx* to *berkhost*:*xxx*.

The **–n** flag changes the article's newsgroup to the specified *newsgroup*.

SEE ALSO

checknews(1), **inews**(1), **postnews**(1), **readnews**(1), **vnews**(1), **news**(5), **recnews**(8), **uurec**(8)

NAME

uurec – receive processed news articles via mail

SYNOPSIS

uurec

DESCRIPTION

Uurec reads news articles on the standard input sent by **sendnews**(8), decodes them, and gives them to **inews**(1) for insertion.

SEE ALSO

inews(1), **postnews**(1), **readnews**(1), **vnews**(1), **news**(5), **recnews**(8), **sendnews**(8)

NAME

vnews – read news articles

SYNOPSIS

vnews [**–a** *date*] [**–n** *newsgroups*] [**–t** *titles*] [**–rxuc**]

vnews –s

DESCRIPTION

Vnews is a program for reading USENET news. It is based on **readnews**(1) but has a CRT oriented interface. The list of available commands is quite similar, although since **vnews** is a "visual" interface, most **vnews** commands do not have to be terminated by a newline.

Vnews uses all but the last two lines of the screen to display the current article. The next-to-last line is the secondary prompt line, and is used to input string arguments to commands. The last line contains several fields. The first field is the prompt field. If **vnews** is at the end of an article, the prompt is "next?"; otherwise the prompt is "more?". The second field is the newsgroup field, which displays the current newsgroup, the number of the current article, and the number of the last article in the newsgroup. The third field contains the current time, and the last field contains the word "mail" if you have mail. When you receive new mail, the bell on the terminal is rung and the word "MAIL" appears in capital letters for 30 seconds.

Vnews without any arguments prints unread articles.

The following flags determine the selection of articles.

–a [*date*] Select articles posted after the given *date* (in **getdate**(3) format).

–n *newsgroups*
Select articles belonging to *newsgroups*.

–t *titles* Select articles whose titles contain one of the strings specified by *titles*.

–r Print the articles in reverse order.

–x Ignore .newsrc file. That is, select articles that have already been read as well as new ones.

–u Update the .newsrc file every 5 minutes, as in the case of an unreliable system. (Note that if the .newsrc file is updated, the **x** command will not restore it to its original contents.)

If the **–c** flag is specified, **vnews** will print the first page of the article, instead of just the header.

You can use the **–s** flag to print the newsgroup subscription list.

Vnews maintains a .newsrc file in the your home directory that specifies all news articles already read. It is updated at the end of each reading session unless the **–x** option was specified. If the environment variable **NEWSRC** is present, it should be the path name of a file to be used in place of .newsrc.

If you wish, an options line may be placed in your .newsrc file. This line starts with the word **options** (left justified) followed by the list of standard options just as they would be typed on the command line. Such a list may include: the **–n** flag along with a newsgroup list and/or the **–r** or **–t** flag. Continuation lines begin with a space or tab character.

ENVIRONMENT

Options can be specified in the **NEWSOPTS** environment parameter. Where conflicts exist, options on the command line take precedence, followed by the .newsrc **options** line, and lastly the **NEWSOPTS** parameter.

When the user uses the reply command, the environment parameter **MAILER** will be used to determine which mailer to use. The default is usually **/bin/mail**.

If the user so desires, he may specify a specific paging program for articles. The environment parameter **PAGER** should be set to the paging program. The name of the article is referenced with a "%", as in the **–c** option. If no "%" is present, the article will be piped to the program. Paging may be disabled by setting **PAGER** to a null value.

If **EDITOR** is set, it will be used in place of the default editor on your system to edit replies and follow-ups.

If **NAME** is set, it will be used as your full name when posting news or submitting a follow-up. If it is not set, the name will be taken from the file *.name* in your home directory. If this file is not present, the name will be taken from /etc/passwd.

If **NEWSARCHIVE** is set, a copy of any articles you post or follow-up to, will be saved in the specified file. If it is the null string, they will be copied in *author_copy* in your home directory.

If **NEWSBOX** is set, the filename you specify when you save or write a file will be prepended with **NEWSBOX** unless the filename is an absolute pathname.

If **NEWSRC** is set, it will be used in place of the .newsrc file in your home directory.

If **ORGANIZATION** is set, it will be used as the name of your organization whenever you post an article. The default is compiled in and is usually correct. Typically, you would only use this if you were reading news at a site other than normal. (Or if you are trying to be cute.)

COMMANDS

Each **vnews** command may be preceded by a count. Some commands use the count; others ignore it. If count is omitted, it defaults to one. Some commands prompt for an argument on the second line from the bottom of the screen. Standard UNIX erase and kill processing is done on this argument. The argument is terminated by a return. An interrupt (DEL or BREAK) gets you out of any partially entered command.

In the following table, ^B is used as a shorthand for Control-B.

Command	Meaning
CR	A carriage return prints more of the current article, or goes on to the next article if you are at the end of the current article. A SPACE is equivalent to CR.
^B	Go backwards *count* pages.
^F	Go forward *count* pages.
^D	Go forwards half a page.
^U	Go backwards half a page.
^Z	Go forwards *count* lines.
^E	Go backwards *count* lines.
^L	Redraw the screen. **^L** may be typed at any time.
b	Back up one article in the current group.
c	Cancel the article. Only the author of the article or the super user can do this.
d	Read a digest. Breaks up a digest into separate articles and permits you to read and reply to each piece.
e	Erase. Forget that this article was read.
f	Submit a follow-up article. You will be placed in your **EDITOR** to compose the text of the follow-up.
h	Go back to the top of the article and display only the header.
l	Redisplay the article after you have sent a follow-up or reply.
m	Move on to the next item in a digest.
n	No. Go on to the next article without printing current one. . is equivalent to **n**. This is convenient if your terminal has a keypad.

p Show the parent article (the article that the current article is a follow-up to). This doesn't work if the current article was posted by A-news or notesfiles. To switch between the current and parent articles, use the – command. Unfortunately, if you use several **p** commands to trace the discussion back further, there is no command to return to the original level.

q Quit. The *.newsrc* file will be updated unless **–x** was on the command line.

r Reply. Reply to article's author via mail. You are placed in your **EDITOR** with a header specifying "To", "Subject", and "References" lines taken from the message. You may change or add headers, as appropriate. Add the text of the reply after the blank line, and then exit the editor. The resulting message is mailed to the author of the article.

R This is the same as **r** except the body of the article is included in your mail message for you.

ESC-r Reply directly. You are placed in your **MAILER** as if you had run it specifying the author of the article as the recipient of a letter.

s [*file*] Save. The article is appended to the named file. The default is *Articles*. If the first character of the file name is "|", the rest of the file name is taken as the name of a program, which is executed with the text of the article as standard input. If the first character of the file name is "/", it is taken as the full pathname of a file. If **NEWSBOX** (in the environment) is set to a full pathname, and the file contains no "/", the file is saved in **NEWSBOX**. Otherwise, it is saved relative to **HOME**.

ug Unsubscribe to the current group. This is a two character command to ensure that it is not typed accidentally and to leave room for other types of unsubscribes (e.g. unsubscribe to discussion).

v Print the current version of the news software.

w Write. Like save **s**, except that the headers are not written out.

x Exit. Like quit except that .newsrc is not updated.

y Yes. Print the current article and go to the next.

[*n*]**A** Go to article number *n* in the current newsgroup.

D Decrypts a joke. It only handles rot 13 jokes. The **D** command is a toggle; typing another **D** re-encrypts the joke.

H Print a very verbose header, containing all known information about the article.

K Kill (mark as read) the rest of the articles in the current group. This is useful if you can't keep up with the volume in the newsgroup, but don't want to unsubscribe.

N [*newsgroup*]
Go to the next newsgroup or named newsgroup.

[*n*]**+** Skip *n* articles. The articles skipped are recorded as "unread" and will be offered to you again the next time you read news.

– Go back to last article. This is a toggle; typing it twice returns you to the original article.

< Prompt for an article ID or the rest of a message ID. It will display the article if it exists.

Report the name and size of the newsgroup.

? Print an short help message.

! Passes the rest of the command line to the shell. The environment variable **A** is set to the name of the file containing the current article. If the last character of the command is a **&**, then the **&** is deleted and the command is run in the background with stdin, stdout and stderr redirected to /dev/null. If the command is missing, the shell is invoked. Use the **l** command (or essentially any other command) to turn on the display after the program terminates.

EXAMPLES

vnews Read all unread articles using the *visual* interface. The .newsrc file is updated at the end of the session.

vnews –n all !mod.all –r
Read all unread articles except articles whose newsgroups begin with **mod.** in reverse order. The .newsrc file is updated at the end of the session.

vnews –n all –a last thursday
Print every unread article since last Thursday. The .newsrc file is updated at the end of the session.

vnews –p > /dev/null &
Discard all unread news. This is useful after returning from a long trip.

FILES

/usr/spool/news/*newsgroup/number*	News articles
/usr/lib/news/active	Active newsgroups
/usr/lib/news/vnews.help	Help file for *visual* interface
~/.newsrc	Options and list of previously read articles

SEE ALSO

checknews(1), **inews**(1), **postnews**(1), **readnews**(1), **vnews**(1), **getdate**(3), **news**(5), **newsrc**(5), **expire**(8), **recnews**(8), **sendnews**(8), **uurec**(8)

NAME

patch – a program for applying a diff file to an original

SYNOPSIS

patch [*options*] *orig diff* [+ [*options*] *orig*]

DESCRIPTION

Patch will take a patch file containing any of the three forms of difference listing produced by the **diff** program and apply those differences to an original file, producing a patched version. By default, the patched version is put in place of the original, with the original file backed up to the same name with the extension ".orig", or as specified by the **-b** switch. You may also specify where you want the output to go with a **-o** switch. If **diff** is omitted, or is a hyphen, the patch will be read from standard input.

Upon startup, patch will attempt to determine the type of the diff file, unless over-ruled by a **-c**, **-e**, or **-n** switch. Context diffs and normal diffs are applied by the **patch** program itself, while ed diffs are simply fed to the **ed** editor via a pipe.

Patch will try to skip any leading garbage, apply the diff, and then skip any trailing garbage. Thus you could feed an article or message containing a context or normal diff to **patch**, and it should work. If the entire diff is indented by a consistent amount, this will be taken into account.

With context diffs, and to a lesser extent with normal diffs, **patch** can detect when the line numbers mentioned in the patch are incorrect, and will attempt to find the correct place to apply each hunk of the patch. As a first guess, it takes the line number mentioned for the hunk, plus or minus any offset used in applying the previous hunk. If that is not the correct place, **patch** will scan both forwards and backwards for a set of lines matching the context given in the hunk. All lines of the context must match. If **patch** cannot find a place to install that hunk of the patch, it will put the hunk out to a reject file, which normally is the name of the output file plus ".rej". (Note that the rejected hunk will come out in context diff form whether the input patch was a context diff or a normal diff. If the input was a normal diff, many of the contexts will simply be null.)

If no original file is specified on the command line, **patch** will try to figure out from the leading garbage what the name of the file to edit is. In the header of a context diff, the filename is found from lines beginning with "***" or "---", with the shortest name of an existing file winning. Only context diffs have lines like that, but if there is an "Index:" line in the leading garbage, **patch** will try to use the filename from that line. The context diff header takes precedence over an Index line. If no filename can be intuited from the leading garbage, you will be asked for the name of the file to patch.

(If the original file cannot be found, but a suitable SCCS or RCS file is handy, **patch** will attempt to get or check out the file.)

Additionally, if the leading garbage contains a "Prereq: " line, **patch** will take the first word from the prerequisites line (normally a version number) and check the input file to see if that word can be found. If not, **patch** will ask for confirmation before proceeding.

The upshot of all this is that you should be able to say, while in a news interface, the following:

```
| patch -d /usr/src/local/blurfl
```

and patch a file in the blurfl directory directly from the article containing the patch.

If the patch file contains more than one patch, **patch** will try to apply each of them as if they came from separate patch files. This means, among other things, that it is assumed that separate patches will apply to separate files, and that the garbage before each patch will be examined for interesting things such as filenames and revision level, as mentioned previously. You can give switches (and another original file name) for the second and subsequent patches by separating the corresponding argument lists by a '+'. The argument list for a second or subsequent patch may not specify a new patch file, however.

Patch recognizes the following switches:

–b causes the next argument to be interpreted as the backup extension, to be used in place of ''.orig''.

–c forces **patch** to interpret the patch file as a context diff.

–d causes **patch** to interpret the next argument as a directory, and cd to it before doing anything else.

–D causes **patch** to use the "#ifdef...#endif" construct to mark changes. The argument following will be used as the differentiating symbol. Note that, unlike the C compiler, there must be a space between the **–D** and the argument.

–e forces **patch** to interpret the patch file as an ed script.

–l causes the pattern matching to be done loosely, in case the tabs and spaces have been munged in you input file. Any sequence of whitespace in the pattern line will match any sequence in the input file. Normal characters must still match exactly. Each line of the context must still match a line in the input file.

–n forces **patch** to interpret the patch file as a normal diff.

–N forces patch to not try and reverse the diffs if it thinks that they may have been swapped. See the **–R** option below.

–o causes the next argument to be interpreted as the output file name.

–p causes leading pathnames to be kept. If the diff is of the file ''b/a.c'', patch will look for ''a.c'' in the ''b'' directory, instead of the current directory. This probably won't work if the diff has rooted pathnames.

–r causes the next argument to be interpreted as the reject file name.

–R tells **patch** that this patch was created with the old and new files swapped. (Yes, I'm afraid that does happen occasionally, human nature being what it is.) **Patch** will attempt to swap each hunk around before applying it. Rejects will come out in the swapped format. The **–R** switch will not work with ed diff scripts because there is too little information to reconstruct the reverse operation.

If the first hunk of a patch fails, **patch** will reverse the hunk to see if it can be applied that way unless the **–N** option is supplied. If it can, the **–R** switch will be set automatically. If it can't, the patch will continue to be applied normally. (Note: this method cannot detect a reversed patch if it is a normal diff and if the first command is an append (i.e. it should have been a delete) since appends always succeed. Luckily, most patches add lines rather than delete them, so most reversed normal diffs will begin with a delete, which will fail, triggering the heuristic.)

–s makes **patch** do its work silently, unless an error occurs.

–x<number>
sets internal debugging flags, and is of interest only to **patch** patchers.

ENVIRONMENT

No environment variables are used by **patch.**

FILES

/tmp/patch*

SEE ALSO

diff(1)

DIAGNOSTICS

Too many to list here, but generally indicative that **patch** couldn't parse your patch file.

The message ''Hmm...'' indicates that there is unprocessed text in the patch file and that **patch** is attempting to intuit whether there is a patch in that text and, if so, what kind of patch it is.

CAVEATS

Patch cannot tell if the line numbers are off in an ed script, and can only detect bad line numbers in a normal diff when it finds a "change" command. Until a suitable interactive interface is added, you should probably do a context diff in these cases to see if the changes made sense. Of course, compiling without errors is a pretty good indication that it worked, but not always.

Patch usually produces the correct results, even when it has to do a lot of guessing. However, the results are guaranteed to be correct only when the patch is applied to exactly the same version of the file that the patch was generated from.

BUGS

Could be smarter about partial matches, excessively deviant offsets and swapped code, but that would take an extra pass.

If code has been duplicated (for instance with #ifdef OLDCODE ... #else ... #endif), **patch** is incapable of patching both versions, and, if it works at all, will likely patch the wrong one, and tell you it succeeded to boot.

If you apply a patch you've already applied, **patch** will think it is a reversed patch, and un-apply the patch. This could be construed as a feature.

NAME

umodem – UNIX-Based Remote File Transfer Facility (version 3.1)

SYNOPSIS

Usage:

umodem -[c!rb!rt!sb!st][options] filename

Major Commands --

c <-- Enter Command Mode
rb <-- Receive Binary
rt <-- Receive Text
sb <-- Send Binary
st <-- Send Text

Options --

1 <-- (one) Employ TERM II FTP 1
4 <-- Enable TERM FTP 4
7 <-- Enable 7-bit transfer mask
a <-- Turn ON ARPA Net Flag
d <-- Do not delete umodem.log file before starting
l <-- (ell) Turn OFF LOG File Entries
m <-- Allow file overwiting on receive
p <-- Turn ON Parameter Display
y <-- Display file status (size) information only

DESCRIPTION

Umodem uses the Christensen protocol to transfer files to and from CP/M systems.

Umodem -- Implements the "CP/M User's Group XMODEM" protocol, the TERM II File Transfer Protocol (FTP) Number 1, and the TERM II File Transfer Protocol Number 4 for packetized file up/downloading.

There is currently no batch transfer capability. The program writes logging data to a file in the user's home directory called umodem.log.

The program will do a protocol file transfer with error checking to or from a CP/M system running Ward Christensen's program MODEM or one of its derivatives (MODEM7 or APMOD777 etc.) or any program that uses the same protocols (e.g. ZPRO, TERM II). Note that executable and squeezed files must use the -sb or -rb options.

Umodem supports an interactive mode in which the user may perform a number of Umodem-oriented functions without leaving Umodem. These functions (and their commands) are:

UMODEM COMMAND MODE OPTIONS

Usage: r or s or option

Major Commands --

rb <-- Receive Binary
rt <-- Receive Text
sb <-- Send Binary
st <-- Send Text

Options --

1 <-- (one) Employ TERM II FTP 1

3 <-- Enable TERM FTP 3 (CP/M UG)
7 <-- Toggle 7-bit transfer mask
a <-- Turn ON ARPA Net Flag
l <-- Toggle LOG File Entries
m <-- Allow file overwiting on receive
x <-- Exit
y <-- Display file status (size) information only

UMODEM COMMAND MODE

The following is a sample session illustrating what can be done in the command mode of Umodem.

$ umodem -c

UMODEM Version 3.5 -- UNIX-Based Remote File Transfer Facility

UMODEM: LOG File '/user/rxc/umodem.log' is Open

UMODEM Command Mode -- Type ? for Help
3 L UMODEM> ?

Usage: r or s or option
Major Commands --
rb <-- Receive Binary
rt <-- Receive Text
sb <-- Send Binary
st <-- Send Text
Options --
1 <-- (one) Employ TERM II FTP 1
3 <-- Enable TERM FTP 3 (CP/M UG)
7 <-- Enable 7-bit transfer mask
a <-- Turn ON ARPA Net Flag
l <-- Toggle LOG File Entries
m <-- Allow file overwiting on receive
x <-- Exit
y <-- Display file status (size) information only

3 L UMODEM> 1

TERM FTP 1 Selected
1 L UMODEM> m

File Overwriting Enabled
1 LM UMODEM> m

File Overwriting NOT Enabled
1 L UMODEM> 7

7-Bit Transfer Selected
17 L UMODEM> 7

7-Bit Transfer NOT Selected
1 L UMODEM> y umodem.c

UMODEM File Status Display for umodem.c
Estimated File Size 42K, 331 Records, 42252 Bytes

1 L UMODEM> x

FILES

umodem.log keeps a log of transfers to and from and any problems during transfer.

AUTHOR

-- Lauren Weinstein, 6/81
-- (Version 2.0) Modified for JHU/UNIX by Richard Conn, 8/1/81
-- Version 2.1 Mods by Richard Conn, 8/2/81
-- Version 2.2 Mods by Richard Conn, 8/2/81
-- Version 2.3 Mods by Richard Conn, 8/3/81
-- Version 2.4 Mods by Richard Conn, 8/4/81
-- Version 2.5 Mods by Richard Conn, 8/5/81
-- Version 2.6 Mods by Bennett Marks, 8/21/81 (Bucky @ CCA-UNIX)
-- Version 2.7 Mods by Richard Conn, 8/25/81 (rconn @ BRL)
-- Version 2.8 Mods by Richard Conn, 8/28/81
-- Version 2.9 Mods by Richard Conn, 9/1/81
-- Version 3.0 Mods by Lauren Weinstein, 9/14/81
-- Version 3.1 Mods by Lauren Weinstein, 4/17/82
-- Version 3.2 Mods by Michael M Rubenstein, 5/26/83
-- Version 3.3 Mod by Ben Goldfarb, 07/02/83
-- Version 3.4 Mods by David F. Hinnant, NCECS, 7/15/83
-- Version 3.5 Mods by Richard Conn, 08/27/83

NAME

rn – new read news program

SYNOPSIS

rn [*options*] [*newsgroups*]

DESCRIPTION

Rn is a replacement for the **readnews**(1) program that was written to be as efficient as possible, particularly in human interaction. **Rn** attempts to minimize the amount of "dead" time spent reading news—it tries to get things done while the user is reading or deciding whether to read, and attempts to get useful information onto the screen as soon as possible, highlighting spots that the eye makes frequent reference to, like subjects and previously read lines. Whether or not it's faster, it SEEMS faster.

If no newsgroups are specified, all the newsgroups which have unread news are displayed, and then the user is asked for each one whether he wants to read it, in the order in which the newsgroups occur in the .newsrc file. With a list of newsgroups, **rn** will start up in "add" mode, using the list as a set of patterns to add new newsgroups and restrict which newsgroups are displayed. See the discussion of the 'a' command on the newsgroup selection level.

Rn operates on three levels: the newsgroup selection level, the article selection level, and the paging level. Each level has its own set of commands, and its own help menu. At the paging level (the bottom level), **rn** behaves much like the **more**(1) program. At the article selection level, you may specify which article you want next, or read them in the default order, which is either in order of arrival on your system, or by subject threads. At the newsgroup selection level (the top level), you may specify which newsgroup you want next, or read them in the default order, which is the order that the newsgroups occur in your .newsrc file. (You will therefore want to rearrange your .newsrc file to put the most interesting newsgroups first. This can be done with the 'm' command on the Newsgroup Selection level. WARNING: invoking readnews/vnews (the old user interface) in any way (including as a news checker in your login sequence!) will cause your .newsrc to be disarranged again.)

On any level, at ANY prompt, an 'h' may be typed for a list of available commands. This is probably the most important command to remember, so don't you forget it. Typing space to any question means to do the normal thing. You will know what that is because every prompt has a list of several plausible commands enclosed in square brackets. The first command in the list is the one which will be done if you type a space. (All input is done in cbreak mode, so carriage returns should not be typed to terminate anything except certain multi-character commands. Those commands will be obvious in the discussion below because they take an argument.)

Upon startup, **rn** will do several things:

1. It will look for your .newsrc file, which is your list of subscribed-to newsgroups. If rn doesn't find a .newsrc, it will create one. If it does find one, it will back it up under the name ".oldnewsrc".
2. It will input your .newsrc file, listing out the first several newsgroups with unread news.
3. It will perform certain consistency checks on your .newsrc. If your .newsrc is out of date in any of several ways, **rn** will warn you and patch it up for you, but you may have to wait a little longer for it to start up.
4. **Rn** will next check to see if any new newsgroups have been created, and give you the opportunity to add them to your .newsrc.
5. **Rn** goes into the top prompt level—the newsgroup selection level.

Newsgroup Selection Level

In this section the words "next" and "previous" refer to the ordering of the newsgroups in your .newsrc file. On the newsgroup selection level, the prompt looks like this:

******** 17 unread articles in net.blurfl—read now? [ynq]

and the following commands may be given at this level:

y,SP Do this newsgroup now.

.command
Do this newsgroup now, but execute *command* before displaying anything. The command will be interpreted as if given on the article selection level.

= Do this newsgroup now, but list subjects before displaying articles.

n Go to the next newsgroup with unread news.

N Go to the next newsgroup.

p Go to the previous newsgroup with unread news. If there is none, stay at the current newsgroup.

P Go to the previous newsgroup.

– Go to the previously displayed newsgroup (regardless of whether it is before or after the current one in the list).

1 Go to the first newsgroup.

^ Go to the first newsgroup with unread news.

$ Go to the end of the newsgroups list.

g newsgroup
Go to *newsgroup*. If it isn't currently subscribed to, you will be asked if you want to subscribe.

/pattern Scan forward for a newsgroup matching *pattern*. Patterns do globbing like filenames, i.e., use ? to match a single character, * to match any sequence of characters, and [] to specify a list of characters to match. ("all" may be used as a synonym for "*".) Unlike normal filename globbing, newsgroup searching is not anchored to the front and back of the filename, i.e. "/jok" will find net.jokes. You may use ^ or $ to anchor the front or back of the search: "/^test$" will find newsgroup test and nothing else If you want to include newsgroups with 0 unread articles, append /r. If the newsgroup is not found between the current newsgroup and the last newsgroup, the search will wrap around to the beginning.

?pattern Same as /, but search backwards.

u Unsubscribe from current newsgroup.

l string List newsgroups not subscribed to which contain the string specified.

L Lists the current state of the .newsrc, along with status information.

Status	Meaning
<number>	Count of unread articles in newsgroup.
READ	No unread articles in newsgroup.
UNSUB	Unsubscribed newsgroup.
BOGUS	Bogus newsgroup.
JUNK	Ignored line in .newsrc (e.g. readnews "options" line).

(A bogus newsgroup is one that is not in the list of active newsgroups in the active file, which on most systems is /usr/lib/news/active.)

m name Move the named newsgroup somewhere else in the .newsrc. If no name is given, the current newsgroup is moved. There are a number of ways to specify where you want the newsgroup—type h for help when it asks where you want to put it.

c Catch up— mark all unread articles in this newsgroup as read.

o pattern Only display those newsgroups whose name matches *pattern*. Patterns are the same as for the '/' command. Multiple patterns may be separated by spaces, just as on the command line. The restriction will remain in effect either until there are no articles left in the restricted set of newsgroups, or another restriction command is given. Since *pattern* is optional, 'o' by itself will remove the restriction.

a pattern Add new newsgroups matching *pattern*. Newsgroups which are already in your .newsrc file, whether subscribed to or not, will not be listed. If any new newsgroups are found, you will be asked for each one whether you would like to add it. After any new newsgroups have been added, the 'a' command also restricts the current set of newsgroups just like the 'o' command does.

& Print out the current status of command line switches and any newsgroup restrictions.

&switch {switch}
Set additional command line switches.

&& Print out the current macro definitions.

&&keys commands
Define additional macros.

!command
Escape to a subshell. One exclamation mark (!) leaves you in your own news directory. A double exclamation mark (!!) leaves you in the spool directory for news, which on most systems is /usr/spool/news. The environment variable SHELL will be used if defined. If *command* is null, an interactive shell is started.

q Quit.

x Quit, restoring .newsrc to its state at startup of **rn**. The .newsrc you would have had if you had exited with 'q' will be called .newnewsrc, in case you didn't really want to type 'x'.

^K Edit the global KILL file. This is a file which contains /pattern/j commands (one per line) to be applied to every newsgroup as it is started up, that is, when it is selected on the newsgroup selection level. The purpose of a KILL file is to mark articles as read on the basis of some set of patterns. This saves considerable wear and tear on your 'n' key. There is also a local KILL file for each newsgroup. Because of the overhead involved in searching for articles to kill, it is better if possible to use a local KILL file. Local KILL files are edited with a '^K' on the article selection level. There are also automatic ways of adding search commands to the local KILL file—see the 'K' command and the K search modifier on the article selection level.

If either of the environment variables VISUAL or EDITOR is set, the specified editor will be invoked; otherwise a default editor (normally vi(1)) is invoked on the KILL file.

Article Selection Level

On the article selection level, **rn** selects (by default) unread articles in numerical order (the order in which articles have arrived at your site). If you do a subject search (^N), the default order is modified to be numerical order within each subject thread. You may switch back and forth between numerical order and subject thread order at will. The –S switch can be used to make subject search mode the default.

On the article selection level you are *not* asked whether you want to read an article before the article is displayed; rather, **rn** simply displays the first page (or portion of a page, at low baud rates) of the article and asks if you want to continue. The normal article selection prompt comes at the END of the article (though article selection commands can be given from within the middle of the article (the pager level) also). The prompt at the end of an article looks like this:

End of article 248 (of 257)—what next? [npq]

The following are the options at this point:

n,SP Scan forward for next unread article. (Note: the 'n' (next) command when typed at the end of an article does not mark the article as read, since an article is automaticaly marked as read after the last line of it is printed. It is therefore possible to type a sequence such as 'mn' and leave the article marked as unread. The fact that an article is marked as read by typing 'n', 'N', '^N', 's', or 'S' within the MIDDLE of the article is in fact a special case.)

N Go to the next article.

^N Scan forward for the next article with the same subject, and make ^N default (subject search mode).

p Scan backward for previous unread article. If there is none, stay at the current article.

P Go to the previous article.

– Go to the previously displayed article (regardless of whether that article is before or after this article in the normal sequence).

^P Scan backward for the previous article with the same subject, and make ^N default (subject search mode).

^R Restart the current article.

v Restart the current article verbosely, displaying the entire header.

^L Refresh the screen.

^X Restart the current article, and decrypt as a rot13 message.

X Refresh the screen, and decrypt as a rot13 message.

b Back up one page.

q Quit this newsgroup and go back to the newsgroup selection level.

^ Go to the first unread article.

$ Go to the last article (actually, one past the last article).

number Go to the numbered article.

range{,range} command{:command}
Apply a set of commands to a set of articles. A range consists of either <article number> or <article number>–<article number>. A dot '.' represents the current article, and a dollar sign '$' represents the last article.

Applicable commands include 'm' (mark as unread), 'M' (delayed mark as unread), 'j' (mark as read), "s dest" (save to a destination), "!command" (shell escape), "=" (print the subject) and "C" (cancel).

j Junk the current article— mark it as read. If this command is used from within an article, you are left at the end of the article, unlike 'n', which looks for the next article.

m Mark the current article as still unread. (If you are in subject search mode you probably want to use M instead of m. Otherwise the current article may be selected as the beginning of the next subject thread.)

M Mark the current article as still unread, but not until the newsgroup is exited. Until then, the current article will be marked as read. This is useful for returning to an article in another session, or in another newsgroup.

/pattern Scan forward for article containing *pattern* in the subject. See the section on Regular Expressions. Together with the escape substitution facility described later, it becomes easy to search for various attributes of the current article, such as subject, article ID, author name, etc. The

previous pattern can be recalled with "<esc>/". If *pattern* is omitted, the previous pattern is assumed.

/pattern/h
Scan forward for article containing *pattern* in the header.

/pattern/a
Scan forward for article containing *pattern* anywhere in article.

/pattern/r Scan read articles also.

/pattern/c
Make search case sensitive. Ordinarily upper and lower case are considered the same.

/pattern/modifiers:command{:command}
Apply the commands listed to articles matching the search command (possibly with h, a, or r modifiers). Applicable commands include 'm' (mark as unread), 'M' (delayed mark as unread), 'j' (mark as read), "s dest" (save to a destination), "!command" (shell escape), "=" (print the subject) and "C" (cancel). If the first command is 'm' or 'M', modifier r is assumed. A K may be included in the modifiers (not the commands) to cause the entire command (sans K) to be saved to the local KILL file, where it will be applied to every article that shows up in the newsgroup.

For example, to save all articles in a given newsgroup to the line printer and mark them read, use "/^/| lpr:j". If you say "/^/K| lpr:j", this will happen every time you enter the newsgroup.

?pattern Scan backward for article containing *pattern* in the subject. May be modified as the forward search is: ?pattern?modifiers[:commands]. It is likely that you will want an r modifier when scanning backward.

k Mark as read all articles with the same subject as the current article. (Note: there is no single character command to temporarily mark as read (M command) articles matching the current subject. That can be done with "/<esc>s/M", however.)

K Do the same as the k command, but also add a line to the local KILL file for this newsgroup to kill this subject every time the newsgroup is started up. For a discussion of KILL files, see the "^K' command below. See also the K modifier on searches above.

^K Edit the local KILL file for this newsgroup. Each line of the KILL file should be a command of the form /pattern/j. (With the exception that **rn** will insert a line at the beginning of the form "THRU <number>", which tells **rn** the maximum article number that the KILL file has been applied to. You may delete the THRU line to force a rescan of current articles.) You may also have reason to use the m, h, or a modifiers. Be careful with the M modifier in a kill file—there are more efficient ways to never read an article. You might have reason to use it if a particular series of articles is posted to multiple newsgroups. In this case, M would force you to view the article in a different newsgroup.

To see only newgroup articles in the control newsgroup, for instance, you might put

/^/j
/newgroup/m

which kills all subjects not containing "newgroup". You can add lines automatically via the K command and K search modifiers, but editing is the only way to remove lines. If either of the environment variables VISUAL or EDITOR is set, the specified editor will be invoked; otherwise a default editor (normally vi) is invoked on the KILL file.

The KILL file may also contain switch setting lines beginning with '&'. Additionally, any line beginning with 'X' is executed on exit from the newsgroup rather than on entrance. This can be used to set switches back to a default value.

r Reply through net mail. The environment variables MAILPOSTER and MAILHEADER may be used to modify the mailing behavior of **rn** (see environment section). If on a nonexistent article such as the "End of newsgroup" pseudo-article (which you can get to with a '$' command), invokes the mailer to nobody in particular.

R Reply, including the current article in the header file generated. (See 'F' command below). The YOUSAID environment variable controls the format of the attribution line.

f Submit a followup article. If on a nonexistent article such as the "End of newsgroup" pseudo-article (which you can get to with a '$' command), posts an original article (basenote).

F Submit a followup article, and include the old article, with lines prefixed either by ">" or by the argument to a **–F** switch. **Rn** will attempt to provide an attribution line in front of the quoted article, generated from the From: line of the article. Unfortunately, the From: line doesn't always contain the right name; you should double check it against the signature and change it if necessary, or you may have to apologize for quoting the wrong person. The environment variables NEWSPOSTER, NEWSHEADER and ATTRIBUTION may be used to modify the posting behavior of **rn** (see environment section).

C Cancel the current article, but only if you are the contributor or superuser.

c Catch up in this newsgroup; i.e., mark all articles as read.

u Unsubscribe to this newsgroup.

s destination

Save to a filename or pipe using sh. If the first character of the destination is a vertical bar, the rest of the command is considered a shell command to which the article is passed through standard input. The command is subject to filename expansion. (See also the environment variable PIPESAVER.) If the destination does not begin with a vertical bar, the rest of the command is assumed to be a filename of some sort. An initial tilde '~' will be translated to the name of the home directory, and an initial environment variable substitution is also allowed. If only a directory name is specified, the environment variable SAVENAME is used to generate the actual name. If only a filename is specified (i.e. no directory), the environment variable SAVEDIR will be used to generate the actual directory. If nothing is specified, then obviously both variables will be used. Since the current directory for rn while doing a save command is your private news directory, saying "s ./filename" will force the file to your news directory. Save commands are also run through % interpretation, so that you can say "s %O/filename" to save to the directory you were in when you ran *rn*, and "s %t" to save to a filename consisting of the Internet address of the sender.

After generating the full pathname of the file to save to, **rn** determines if the file exists already, and if so, appends to it. **Rn** will attempt to determine if an existing file is a mailbox or a normal file, and save the article in the same format. If the output file does not yet exist, **rn** will by default ask you which format you want, or you can make it skip the question with either the –M or –N switch. If the article is to be saved in mailbox format, the command to do so is generated from the environment variable MBOXSAVER. Otherwise, NORMSAVER is used.

S destination

Save to a filename or pipe using a preferred shell, such as csh. Which shell is used depends first on what you have the environment variable SHELL set to, and in the absence of that, on what your news administrator set for the preferred shell when he or she installed *rn*.

| command

Shorthand for "s| command".

w destination
: The same as "s destination", but saves without the header.

W destination
: The same as "S destination", but saves without the header.

& Print out the current status of command line switches.

&switch {switch}
: Set additional command line switches.

&& Print out current macro definitions.

&&keys commands
: Define an additional macro.

!command
: Escape to a subshell. One exclamation mark (!) leaves you in your own news directory. A double exclamation mark (!!) leaves you in the spool directory of the current newsgroup. The environment variable SHELL will be used if defined. If *command* is null, an interactive shell is started.

  You can use escape key substitutions described later to get to many run-time values. The command is also run through % interpretation, in case it is being called from a range or search command.

= List subjects of unread articles.

Print last article number.

Pager Level

At the pager level (within an article), the prompt looks like this:

—MORE—(17%)

and a number of commands may be given:

SP Display next page.

x Display next page and decrypt as a rot13 message.

d,^D Display half a page more.

CR Display one more line.

q Go to the end of the current article (don't mark it either read or unread). Leaves you at the "What next?" prompt.

j Junk the current article. Mark it read and go to the end of the article.

^L Refresh the screen.

X Refresh the screen and decrypt as a rot13 message.

b,^B Back up one page.

gpattern Goto (search forward for) *pattern* within current article. Note that there is no space between the command and the pattern. If the pattern is found, the page containing the pattern will be displayed. Where on the page the line matching the pattern goes depends on the value of the –g switch. By default the matched line goes at the top of the screen.

G Search for g pattern again.

^G This is a special version of the 'g' command that is for skipping articles in a digest. It is equivalent to setting "–g4" and then executing the command "g^Subject:".

TAB This is another special version of the 'g' command that is for skipping inclusions of older articles. It is equivalent to setting "–g4" and then executing the command "g^[^c]", where *c* is the first character of the last line on the screen. It searches for the first line that doesn't begin with the same character as the last line on the screen.

!command
Escape to a subshell.

The following commands skip the rest of the current article, then behave just as if typed to the "What next?" prompt at the end of the article. See the documentation at the article selection level for these commands.

$ & / = ? c C f F k K ^K m M r R ^R u v Y ^
number
range{,range} command{:command}

The following commands also skip to the end of the article, but have the additional effect of marking the current article as read:

n N ^N s S| w W

Miscellaneous facts about commands

An 'n' typed at either the "Last newsgroup" prompt or a "Last article" prompt will cycle back to the top of the newsgroup or article list, whereas a 'q' will quit the level. (Note that 'n' does not mean "no", but rather "next".) A space will of course do whatever is shown as the default, which will vary depending on whether rn thinks you have more articles or newsgroups to read.

The 'b' (backup page) command may be repeated until the beginning of the article is reached. If **rn** is suspended (via a ^Z), then when the job is resumed, a refresh (^L) will automatically be done (Berkeley-type systems only). If you type a command such as '!' or 's' which takes you from the middle of the article to the end, you can always get back into the middle by typing "^L'.

In multi-character commands such as '!', 's', '/', etc, you can interpolate various run-time values by typing escape and a character. To find out what you can interpolate, type escape and 'h', or check out the single character % substitutions for environment variables in the Interpretation and Interpolation section, which are the same. Additionally, typing a double escape will cause any % substitutions in the string already typed in to be expanded.

Options

Rn has a nice set of options to allow you to tailor the interaction to your liking. (You might like to know that the author swears by "–e –m –S –/".) These options may be set on the command line, via the RNINIT environment variable, via a file pointed to by the RNINIT variable, or from within rn via the & command. Options may generally be unset by saying "+switch". Options include:

–c checks for news without reading news. If a list of newsgroups is given on the command line, only those newsgroups will be checked; otherwise all subscribed-to newsgroups are checked. Whenever the –c switch is specified, a non-zero exit status from **rn** means that there is unread news in one of the checked newsgroups. The –c switch does not disable the printing of newsgroups with unread news; this is controlled by the –s switch. (The –c switch is not meaningful when given via the & command.)

–C<number>
tells **rn** how often to checkpoint the *.newsrc*, in articles read. Actually, this number says when to start thinking about doing a checkpoint if the situation is right. If a reasonable checkpointing situation doesn't arise within 10 more articles, the is checkpointed willy-nilly.

–d<directory name>
sets the default save directory to something other than ~/News. The directory name will be globbed

(via csh) if necessary (and if possible). Articles saved by **rn** may be placed in the save directory or in a subdirectory thereof depending on the command that you give and the state of the environment variables SAVEDIR and SAVENAME. Any KILL files (see the K command in the Article Selection section) also reside in this directory and its subdirectories, by default. In addition, shell escapes leave you in this directory.

-D<flags>
enables debugging output. See common.h for flag values. Warning: normally **rn** attempts to restore your when an unexpected signal or internal error occurs. This is disabled when any debugging flags are set.

-e causes each page within an article to be started at the top of the screen, not just the first page. (It is similar to the **-c** switch of **more**(1).) You never have to read scrolling text with this switch. This is helpful especially at certain baud rates because you can start reading the top of the next page without waiting for the whole page to be printed. It works nicely in conjuction with the **-m** switch, especially if you use half-intensity for your highlight mode. See also the **-L** switch.

-E<name>=<val>
sets the environment variable <name> to the value specified. Within *rn*, "&-ESAVENAME=%t" is similar to "setenv SAVENAME '%t'" in *csh*, or "SAVENAME='%t'; export SAVENAME" in *sh*. Any environment variables set with **-E** will be inherited by subprocesses of *rn*.

-F<string>
sets the prefix string for the 'F' followup command to use in prefixing each line of the quoted article. For example, "-F<tab>" inserts a tab on the front of each line (which will cause long lines to wrap around, unfortunately), "-F>>>>" inserts ">>>>" on every line, and "-F" by itself causes nothing to be inserted, in case you want to reformat the text, for instance. The initial default prefix is ">".

-g<line>
tells **rn** which line of the screen you want searched-for strings to show up on when you search with the 'g' command within an article. The lines are numbered starting with 1. The initial default is "-g1", meaning the first line of the screen. Setting the line to less than 1 or more than the number of lines on the screen will set it to the last line of the screen.

-h<string>
hides (disables the printing of) all header lines beginning with *string*. For instance, -hexp will disable the printing of the "Expires:" line. Case is insignificant. If <string> is null, all header lines except Subject are hidden, and you may then use **+h** to select those lines you want to see. You may wish to use the baud-rate switch modifier below to hide more lines at lower baud rates.

-H<string>
works just like **-h** except that instead of setting the hiding flag for a header line, it sets the magic flag for that header line. Certain header lines have magic behavior that can be controlled this way. At present, the following actions are caused by the flag for the particular line: the Newsgroups line will only print when there are multiple newsgroups, the Subject line will be underlined, and the Expires line will always be suppressed if there is nothing on it. In fact, all of these actions are the default, and you must use **+H** to undo them.

-i=<number>
specifies how long (in lines) to consider the initial page of an article—normally this is determined automatically depending on baud rate. (Note that an entire article header will always be printed regardless of the specified initial page length. If you are working at low baud rate and wish to reduce the size of the headers, you may hide certain header lines with the **-h** switch.)

-l disables the clearing of the screen at the beginning of each article, in case you have a bizarre terminal.

-L tells **rn** to leave information on the screen as long as possible by not blanking the screen between

pages, and by using clear to end-of-line. (The **more**(1) program does this.) This feature works only if you have the requisite termcap capabilities. The switch has no effect unless the **–e** switch is set.

–m=<mode>
enables the marking of the last line of the previous page printed, to help the user see where to continue reading. This is most helpful when less than a full page is going to be displayed. It may also be used in conjunction with the **–e** switch, in which case the page is erased, and the first line (which is the last line of the previous page) is highlighted. If **–m=s** is specified, the standout mode will be used, but if **–m=u** is specified, underlining will be used. If neither =s or =u is specified, standout is the default. Use **+m** to disable highlighting.

–M forces mailbox format in creating new save files. Ordinarily you are asked which format you want.

–N forces normal (non-mailbox) format in creating new save files. Ordinarily you are asked which format you want.

–r causes **rn** to restart in the last newsgroup read during a previous session with **rn.** It is equivalent to starting up normally and then getting to the newsgroup with a g command.

–s with no argument suppresses the initial listing of newsgroups with unread news, whether **–c** is specified or not. Thus **–c** and **–s** can be used together to test "silently" the status of news from within your *.login* file. If **–s** is followed by a number, the initial listing is suppressed after that many lines have been listed. Presuming that you have your sorted into order of interest, **–s5** will tell you the 5 most interesting newsgroups that have unread news. This is also a nice feature to use in your *.login* file, since it not only tells you whether there is unread news, but also how important the unread news is, without having to wade through the entire list of unread newsgroups. If no **–s** switch is given **–s5** is assumed, so just putting "rn –c" into your .login file is fine.

–S<number>
causes **rn** to enter subject search mode (^N) automatically whenever a newsgroup is started up with <number> unread articles or more. Additionally, it causes any 'n' typed while in subject search mode to be interpreted as '^N' instead. (To get back out of subject search mode, the best command is probably '^'.) If <number> is omitted, 3 is assumed.

–t puts **rn** into terse mode. This is more cryptic but useful for low baud rates. (Note that your system administrator may have compiled **rn** with either verbose or terse messages only to save memory.) You may wish to use the baud-rate switch modifier below to enable terse mode only at lower baud rates.

–T allows you to type ahead of rn. Ordinarily rn will eat typeahead to prevent your autorepeating space bar from doing a very frustrating thing when you accidentally hold it down. If you don't have a repeating space bar, or you are working at low baud rate, you can set this switch to prevent this behavior. You may wish to use the baud-rate switch modifier below to disable typeahead only at lower baud rates.

–v sets verification mode for commands. When set, the command being executed is displayed to give some feedback that the key has actually been typed. Useful when the system is heavily loaded and you give a command that takes a while to start up.

–/ sets SAVEDIR to "%p/%c" and SAVENAME to "%a", which means that by default articles are saved in a subdirectory of your private news directory corresponding to the name of the the current newsgroup, with the filename being the article number. +/ sets SAVEDIR to "%p" and SAVENAME to "%^C", which by default saves articles directly to your private news directory, with the filename being the name of the current newsgroup, first letter capitalized. (Either +/ or –/ may be default on your system, depending on the feelings of your news administrator when he, she or it installed *rn*.) You may, of course, explicitly set SAVEDIR and SAVENAME to other values—see discussion in the environment section.

Any switch may be selectively applied according to the current baud-rate. Simply prefix the switch with +speed to apply the switch at that speed or greater, and –speed to apply the switch at that speed or less. Examples: –1200–hposted suppresses the Posted line at 1200 baud or less; +9600–m enables marking at 9600 baud or more. You can apply the modifier recursively to itself also: +300–1200–t sets terse mode from 300 to 1200 baud.

Similarly, switches may be selected based on terminal type:

```
–=vt100+T               set +T on vt100
–=tvi920–ETERM=mytvi get a special termcap entry
–=tvi920–ERNMACRO=%./.rnmac.tvi
                        set up special keymappings
+=paper–v               set verify mode if not hardcopy
```

Some switch arguments, such as environment variable values, may require spaces in them. Such spaces should be quoted via ", ', or \ in the conventional fashion, even when passed via RNINIT or the & command.

Regular Expressions

The patterns used in article searching are regular expressions such as those used by **ed**(1). In addition, \w matches an alphanumeric character and \W a nonalphanumeric. Word boundaries may be matched by \b, and non-boundaries by \B. The bracketing construct \(... \) may also be used, and \digit matches the digit'th substring, where digit can range from 1 to 9. \0 matches whatever the last bracket match matched. Up to 10 alternatives may given in a pattern, separated by \| , with the caveat that \(... \| ... \) is illegal.

Interpretation and Interpolation

Many of the strings that **rn** handles are subject to interpretations of several types. Under filename expansion, an initial "~/" is translated to the name of your home directory, and "~name" is translated to the login directory for the user specified. Filename expansion will also expand an initial environment variable, and also does the backslash, uparrow and percent expansion mentioned below.

All interpreted strings go through backslash, uparrow and percent interpretation. The backslash escapes are the normal ones (such as \n, \t, \nnn, etc.). The uparrow escapes indicate control codes in the normal fashion. Backslashes or uparrows to be passed through should be escaped with backslash. The special percent escapes are similar to printf percent escapes. These cause the substitution of various run-time values into the string. The following are currently recognized:

%a Current article number.

%A Full name of current article (%P/%c/%a). (On a Eunice system with the LINKART option, %P/%c/%a returns the name of the article in the current newsgroup, while %A returns the real name of the article, which may be different if the current article was posted to multiple newsgroups.)

%b Destination of last save command, often a mailbox.

%B The byte offset to the beginning of the part of the article to be saved, set by the save command. The 's' and 'S' commands set it to 0, and the 'w' and 'W' commands set it to the byte offset of the body of the article.

%c Current newsgroup, directory form.

%C Current newsgroup, dot form.

%d Full name of newsgroup directory (%P/%c).

%D "Distribution:" line from the current article.

%f "From:" line from the current article, or the "Reply-To:" line if there is one. This differs from %t in that comments (such as the full name) are not stripped out with %f.

%F "Newsgroups:" line for a new article, constructed from "Newsgroups:" and "Followup-To:" lines of current article.

%h Name of the header file to pass to the mail or news poster, containing all the information that the poster program needs in the form of a message header. It may also contain a copy of the current article. The format of the header file is controlled by the MAILHEADER and NEWSHEADER environment variables.

%H Host name (your machine's name).

%i "Message-I.D.:" line from the current article, with <> guaranteed.

%I The reference indication mark (see the –F switch.)

%l The news administrator's login name, if any.

%L Login name (yours).

%m The current mode of rn, for use in conditional macros.

i	Initializing.
n	Newsgroup selection level.
a	Article selection level (What next?).
p	Pager level (MORE prompt).
A	Add this newsgroup?
C	Catchup confirmation.
D	Delete bogus newsgroups?
M	Use mailbox format?
R	Resubscribe to this newsgroup?

Note that yes/no questions are all upper-case modes. If, for example, you wanted to disallow defaults on all yes/no questions, you could define the following macro:

\040 %(%m=[A-Z]?h:)

%M The number of articles marked to return via the 'M' command. If the same article is Marked multiple times, "%M" counts it multiple times in the current implementation.

%n "Newsgroups:" line from the current article.

%N Full name (yours).

%o Organization (yours).

%O Original working directory (where you ran rn from).

%p Your private news directory, normally ~/News.

%P Public news spool directory, normally /usr/spool/news.

%r Last reference on references line of current article (parent article id).

%R References list for a new article, constructed from the references and article ID of the current article.

%s Subject, with all Re's and (nf)'s stripped off.

%S Subject, with one "Re:" stripped off.

%t "To:" line derived from the "From:" and "Reply-To:" lines of the current article. This always returns an Internet format address.

%T "To:" line derived from the "Path:" line of the current article to produce a uucp path.

%u The number of unread articles in the current newsgroup.

%U The number of unread articles in the current newsgroup, not counting the current article.

%x The news library directory.

%X The rn library directory.

%z The length of the current article in bytes.

%~ Your home directory.

%. The directory containing your dot files, which is your home directory unless the environment variable DOTDIR is defined when rn is invoked.

%$ Current process number.

%/ Last search string.

%% A percent sign.

%{name} or %{name–default}
The environment variable "name".

%[name] The value of header line "Name:" from the current article. The "Name: " is not included. For example "%D" and "%[distribution]" are equivalent. The name must be spelled out in full.

%`command`
Inserts the output of the command, with any embedded newlines translated to space.

%"prompt"
Prints prompt on the terminal, then inputs one string, and inserts it.

%(test_text=pattern?then_text:else_text)
If *test_text* matches *pattern*, has the value *then_text*, otherwise *else_text*. The ":else_text" is optional, and if absent, interpolates the null string. The = may be replaced with != to negate the test. To quote any of the metacharacters ('=', '?', ':', or ')'), precede with a backslash.

%digit The digits 1 through 9 interpolate the string matched by the nth bracket in the last pattern match that had brackets. If the last pattern had alternatives, you may not know the number of the bracket you want— %0 will give you the last bracket matched.

Modifiers: to capitalize the first letter, insert '^': "%^C" produces something like "Net.jokes". Inserting '_' causes the first letter following the last '/' to be capitalized: "%_c" produces "net/Jokes".

ENVIRONMENT

The following environment variables are paid attention to by *rn*. In general the default values assumed for these variables by **rn** are reasonable, so if you are using **rn** for the first time, you can safely ignore this section. Note that the defaults below may not correspond precisely to the defaults on your system. To find the actual defaults you would need to look in config.h and common.h in the rn source directory, and the file INIT in the rn library.

Those variables marked (%) are subject to % interpolation, and those marked (~) are subject to both % interpolation and ~ interpretation.

ATTRIBUTION (%)
Gives the format of the attribution line in front of the quoted article included by an F command.

Default: In article %i %f writes:

CANCEL (~)
The shell command used to cancel an article.

Default: inews –h < %h

CANCELHEADER (%)
The format of the file to pass to the CANCEL command in order to cancel an article.

Default:
Newsgroups: %n
Subject: cmsg cancel %i
References: %R
Reply-To: %L@%H.UUCP (%N)
Distribution: %D
Organization: %o

%i cancelled from rn.

DOTDIR
Where to find your dot files, if they aren't in your home directory. Can be interpolated using "%.".

Default: $HOME

EDITOR (~)
The name of your editor, if VISUAL is undefined.

Default: whatever your news administrator compiled in, usually vi.

FIRSTLINE (%)
Controls the format of the line displayed at the top of an article. Warning: this may go away.

Default: Article %a %(%U%M!=^00$?(%U more%(%M!=^0$? + %M Marked to return)\))in %C:, more or less.

HIDELINE
If defined, contains a regular expression which matches article lines to be hidden, in order, for instance, to suppress quoted material. A recommended string for this purpose is "^>...", which *doesn't* hide lines with only '>', to give some indication that quoted material is being skipped. If you want to hide more than one pattern, you can use "| " to separate the alternatives. You can view the hidden lines by restarting the article with the 'v' command.

There is some overhead involved in matching each line of the article against a regular expression. You might wish to use a baud-rate modifier to enable this feature only at low baud rates.

Default: undefined

HOME Your home directory. Affects ~ interpretation, and the location of your dot files if DOTDIR is not defined.

Default: $LOGDIR

KILLGLOBAL (~)
Where to find the KILL file to apply to every newsgroup. See the '^K' command at the newsgroup selection level.

Default: %p/KILL

KILLLOCAL (~)
Where to find the KILL file for the current newsgroup. See the commands 'K' and '^K' at the article selection level, and the search modifier 'K'.

Default: %p/%c/KILL

LOGDIR
: Your home directory if HOME is undefined. Affects ~ interpretation, and the location of your dot files if DOTDIR is not defined.

  Default: none.

  Explanation: you must have either $HOME or $LOGDIR.

LOGNAME
: Your login name, if USER is undefined. May be interpolated using "%L".

  Default: value of getlogin().

MAILCALL (~)
: What to say when there is new mail.

  Default: (Mail)

MAILFILE (~)
: Where to check for mail.

  Default: /usr/spool/mail/%L

MAILHEADER (%)
: The format of the header file for replies. See also MAILPOSTER.

  Default:

```
To: %T
Subject: %(%i=^$?:Re: %S
Newsgroups: %n
In-Reply-To: %i)
%(%[references]!=^$?References\: %[references]
)Organization: %o
Cc:
Bcc: \n\n
```

MAILPOSTER (~)
: The shell command to be used by the reply commands (r and R) in order to allow you to enter and deliver the response. **Rn** will not itself call upon an editor for replies—this is a function of the program called by *rn*. See also MAILHEADER.

  Default: Rnmail –h %h

MBOXSAVER (~)
: The shell command to save an article in mailbox format.

  Default: %X/mbox.saver %A %P %c %a %B %C "%b" \
  "From: %T %‘date‘"

  Explanation: the first seven arguments are the same as for NORMSAVER. The eighth argument to the shell script is the new From: line for the article, including the posting date, derived either directly from the Posted: line, or not-so-directly from the Date: line. Header munging at its finest.

NAME
: Your full name. May be interpolated using "%N".

  Default: name from /etc/passwd, or ~/.fullname.

NEWSHEADER (%)
: The format of the header file for followups. See also NEWSPOSTER.

  Default:

Newsgroups: %(%F=^$?%C:%F)
Subject: %(%S=^$?%"0ubject: ":Re: %S)
Summary:
Expires:
%(%R=^$?:References: %R
)Sender:
Reply-To: %L@%H.UUCP (%N)
Followup-To:
Distribution: %(%i=^$?%"0istribution: ":%D)
Organization: %o
Keywords: \n\n

NEWSPOSTER (~)

The shell command to be used by the followup commands (f and F) in order to allow you to enter and post a followup news article. **Rn** will not itself call upon an editor for followups—this is a function of the program called by *rn*. See also NEWSHEADER.

Default: Pnews -h %h

NORMSAVER (~)

The shell command to save an article in the normal (non-mailbox) format.

Default: %X/norm.saver %A %P %c %a %B %C "%b"

ORGANIZATION

Either the name of your organization, or the name of a file containing the name of your organization. May be interpolated using "%o".

Default: whatever your news administrator compiled in.

PAGESTOP

If defined, contains a regular expression which matches article lines to be treated as form-feeds. There are at least two things you might want to do with this. To cause page breaks between articles in a digest, you might define it as "^--------". To force a page break before a signature, you could define it as "^-- $". (Then, when you see "--" at the bottom of the page, you can skip the signature if you so desire by typing 'n' instead of space.) To do both, you could use "^--". If you want to break on more than one pattern, you can use "| " to separate the alternatives.

There is some overhead involved in matching each line of the article against a regular expression. You might wish to use a baud-rate modifier to enable this feature only at low baud rates.

Default: undefined

PIPESAVER (%)

The shell command to execute in order to accomplish a save to a pipe ("s| command" or "w| command"). The command typed by the user is substituted in as %b.

Default: %(%B=^0$?<%A:tail +%Bc %A|) %b

Explanation: if %B is 0, the command is "<%A %b", otherwise the command is "tail +%Bc %A| %b".

RNINIT Default values for switches may be passed to **rn** by placing them in RNINIT. Any switch that is set in RNINIT may be overruled on the command line, or via the '&' command from within *rn*. Binary-valued switches that are set with "-switch" may be unset using "+switch".

If RNINIT begins with a '/' it is assumed to be the name of a file containing switches. If you want to set many environment variables but don't want to keep them all in your environment, or if the use of any of these variables conflicts with other programs, you can use this feature along with the **-E** switch to set the environment variables upon startup.

Default: " ".

RNMACRO (~)
The name of the file containing macros and key mappings. See the MACROS section.

Default: %./.rnmac

SAVEDIR (~)
The name of the directory to save to, if the save command does not specify a directory name.

Default:
If –/ is set: %p/%c
If +/ is set: %p

SAVENAME (%)
The name of the file to save to, if the save command contains only a directory name.

Default:
If –/ is set: %a
If +/ is set: %^C

SHELL The name of your preferred shell. It will be used by the '!', 'S' and 'W' commands.

Default: whatever your news administrator compiled in.

SUBJLINE (%)
Controls the format of the lines displayed by the '=' command at the article selection level.

Default: %s

TERM Determines which termcap entry to use, unless TERMCAP contains the entry.

TERMCAP
Holds either the name of your termcap file, or a termcap entry.

Default: /etc/termcap, normally.

USER Your login name. May be interpolated using "%L".

Default: $LOGNAME

VISUAL (~)
The name of your editor.

Default: $EDITOR

YOUSAID (%)
Gives the format of the attribution line in front of the quoted article included by an R command.

Default: In article %i you write:

MACROS
When **rn** starts up, it looks for a file containing macro definitions (see environment variable RNMACRO). Any sequence of commands may be bound to any sequence of keys, so you could remap your entire keyboard if you desire. Blank lines or lines beginning with # in the macro file are considered comments; otherwise **rn** looks for two fields separated by white space. The first field gives the sequence of keystrokes that trigger the macro, and the second field gives the sequence of commands to execute. Both fields are subject to % interpolation, which will also translate backslash and uparrow sequences. (The keystroke field is interpreted at startup time, but the command field is interpreted at macro execution time so that you may refer to % values in a macro.) For example, if you want to reverse the roles of carriage return and space in rn

```
^J      \040
^M      \040
```

\040 ^J

will do just that. By default, all characters in the command field are interpreted as the canonical **rn** characters, i.e. no macro expansion is done. Otherwise the above pair of macros would cause an infinite loop. To force macro expansion in the command field, enclose the macro call with ^(... ^) thusly:

```
@s      | mysavescript
@w      w^(@s^)
```

You can use the %() conditional construct to construct macros that work differently under different circumstances. In particular, the current mode (%m) of **rn** could be used to make a command that only works at a particular level. For example,

^[[O %(%m=p?\040)

will only allow the macro to work at the pager level.

%(%{TERM}=vt100?^[[O) /^J

will do the binding only if the terminal type is vt100, though if you have many of these it would be better to have separate files for each terminal.

If you want to bind a macro to a function key that puts a common garbage character after the sequence (such as the carriage return on the end of Televideo 920 function sequences), DO NOT put the carriage return into all the sequences or you will waste a CONSIDERABLE amount of internal storage. Instead of "^AF^M", put "^AF+1", which indicates to **rn** that it should gobble up one character after the F.

AUTHOR

Larry Wall <lwall@sdcrdcf.UUCP>
Regular expression routines are borrowed from emacs, by James Gosling.

FILES

%/.newsrc	status of your news reading
%/.oldnewsrc	backup copy of your from start of session
%/.rnlock	lock file so you don't screw up your
%/.rnlast	info from last run of rn
%/.rnsoft	soft pointers into /usr/lib/active to speed startup, synchronous with
%/.rnhead	temporary header file to pass to a mailer or news poster
%/.rnmac	macro and keymap definitions
%p	your news save directory, usually ~/News
%x/active	the list of active newsgroups, usually /usr/lib/news/active
%P	the public news spool directory, usually /usr/spool/news
%X/INIT	system-wide default switches

SEE ALSO

newsrc(5), **more**(1), **readnews**(1), **Pnews**(1), **Rnmail**(1)

DIAGNOSTICS

Generally self-documenting, as they say.

BUGS

The **–h** switch can only hide header lines that **rn** knows about.

The '–' command doesn't cross newsgroup boundaries, and only undoes the last article selection.

If you edit your while **rn** is running, **rn** will happily wipe out your changes when it decides to write out the file.

Rn doesn't do certain things (like ordering articles on posting date) that the author feels should be handled by inews.

Marking of duplicate articles as read in cross-referenced newsgroups will not work unless the Xref patch is installed in inews.

If you get carried away with % or escape substitutions, you can overflow buffers.

There should be no fixed limit on the number of newsgroups.

Some of the more esoteric features may be missing on machines with limited address space.

NAME

newsgroups – a program to list unsubscribed newsgroups.

SYNOPSIS

newsgroups *pattern flag*

DESCRIPTION

The **newsgroups** program compares your .newsrc file with the file of active newsgroups, and prints a list of unsubscribed newsgroups matching pattern. If the second argument "flag" is present, only newsgroups not found in your .newsrc are listed, and the display is not paged. If the second argument is missing, the display is paged, and an additional list of unsubscribed newsgroups occurring in your .newsrc is printed.

ENVIRONMENT

DOTDIR

Where to find your .newsrc, if not in your home directory.

Default: $HOME

HOME Your home directory.

Default: $LOGDIR

LOGDIR

Your home directory if HOME is undefined.

FILES

/usr/lib/news/active or a reasonable facsimile
${DOTDIR-{$HOME-$LOGDIR}}/.newsrc

SEE ALSO

rn(1), **newsrc**(5)

DIAGNOSTICS

BUGS

The flag argument is a kludge.

NAME

newsetup – a program to set up a .newsrc file

SYNOPSIS

newsetup

DESCRIPTION

The **newsetup** program creates a new .newsrc file containing all of the currently active newsgroups. It tries to put them in a reasonable order, i.e. local newsgroups earlier, but you'll probably want to change the ordering anyway (if you use **rn**) in order to put interesting newsgroups first. If you already have a .newsrc, it will be backed up with the name ".oldnewsrc".

ENVIRONMENT

DOTDIR

Where to put your .newsrc, if not in your home directory.

Default: $HOME

HOME Your home directory.

Default: $LOGDIR

LOGDIR

Your home directory if HOME is undefined.

FILES

/usr/lib/news/active or a reasonable facsimile
${DOTDIR-{$HOME-$LOGDIR}}/.newsrc

SEE ALSO

rn(1), **newsrc**(5)

DIAGNOSTICS

BUGS

NAME

Pnews – a program for posting news articles

SYNOPSIS

Pnews newsgroup title
Pnews -h headerfile [oldarticle]
Pnews

DESCRIPTION

Pnews is a friendly interface for posting news articles. It will ask several questions, then allow you to enter your article, and then post it using the **inews**(1) program. If you type h and a carriage return at any point, **Pnews** will tell you what it wants to know.

The -h form is used when invoked from **rn**. If your editor can edit multiple files, and you want the article to which you are replying to show up as an alternate file, define the environment variable NEWSPOSTER as "Pnews -h %h %A". You can also modify the the NEWSHEADER environment variable to change the header file that **rn** passes to Pnews.

ENVIRONMENT

AUTHORCOPY
If defined, contains the name of a file to which the finished article will be appended.

Default: article not saved

DOTDIR
Where to find your dot files, if they aren't in your home directory. This is primarily for accounts which are shared by more than one person.

Default: $HOME

EDITOR The editor you want to use, if VISUAL is undefined.

Default: whatever your news administrator installed, usually vi.

HOME Your home directory.

Default: $LOGDIR

LOGDIR
Your home directory if HOME is undefined.

LOGNAME
Your login name, if USER is undefined.

Default: value of "whoami".

NAME Your full name.

Default: name from /etc/passwd, or ~/.fullname.

ORGANIZATION
Either the name of your organization, or the name of a file containing the name of your organization.

Default: whatever your news administrator chose.

USER Your login name.

Default: $LOGNAME

VISUAL The editor you want to use.

Default: $EDITOR

FILES

$DOTDIR/.article
~/dead.article

SEE ALSO

rn(1), **Rnmail**(1), **inews**(1)

DIAGNOSTICS

BUGS

Not the speediest program in the world, but maybe that's a blessing to the net.

NAME

Rnmail – a program for replying via mail

SYNOPSIS

Rnmail*destination_list*
Rnmail -h *headerfile* [*oldarticle*]
Rnmail

DESCRIPTION

Rnmail is a friendly interface for mailing replies to news articles. It will ask several questions, then allow you to enter your letter, and then mail it off. If you type h and a carriage return at any point, **Rnmail** will tell you what it wants to know.

The **–h** form is used when invoked from **rn**. If your editor can edit multiple files, and you want the article to which you are replying to show up as an alternate file, define the environment variable MAILPOSTER as "Rnmail -h %h %A". You can also modify the the MAILHEADER environment variable to change the header file that **rn** passes to **Rnmail**.

ENVIRONMENT

DOTDIR
: If defined, specifies a place other than your home directory where 'dot' files may be stored. This is primarily for accounts which are shared by more than one person.

  Default: $HOME

EDITOR The editor you want to use, if VISUAL is undefined.

  Default: whatever your news administrator installed, usually vi.

HOME Your home directory.

  Default: $LOGDIR

LOGDIR
: Your home directory if HOME is undefined.

LOGNAME
: Your login name, if USER is undefined.

  Default: value of "whoami".

MAILRECORD
: If defined, contains the name of a file to which the finished message will be appended.

  Default: message not saved

ORGANIZATION
: Either the name of your organization, or the name of a file containing the name of your organization.

  Default: whatever your news administrator chose.

USER Your login name.

  Default: $LOGNAME

VISUAL The editor you want to use.

  Default: $EDITOR

FILES

$DOTDIR/.letter
~/dead.letter

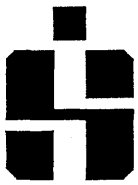
SEE ALSO

rn(1), **Pnews**(1), **mail**(1)

DIAGNOSTICS

BUGS

Uses /bin/mail in the absence of sendmail.



Integrated Solutions

DOCUMENTATION COMMENTS

Please take a minute to comment on the accuracy and completeness of this manual. Your assistance will help us to better identify and respond to specific documentation issues. If necessary, you may attach an additional page with comments. Thank you in advance for your cooperation.

Manual Title: *User Contributed Software (UCS)*	Part Number: *490149 Rev. B*

Name: _____ Title: _____

Company: _____ Phone: ( ) _____

Address: _____

City: _____ State: _____ Zip Code: _____

1. Please rate this manual for the following:

	Poor	Fair	Good	Excellent
Clarity	☐	☐	☐	☐
Completeness	☐	☐	☐	☐
Organization	☐	☐	☐	☐
Technical Content/Accuracy	☐	☐	☐	☐
Readability	☐	☐	☐	☐

Please comment: _____

_____

2. Does this manual contain enough examples and figures?

Yes ☐ No ☐

Please comment: _____

_____

3. Is any information missing from this manual?

Yes ☐ No ☐

Please comment: _____

_____

4. Is this manual adequate for your purposes?

Yes ☐ No ☐

Please comment on how this manual can be improved: _____

_____

_____

_____

_____

_____

_____

Fold Down First

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

First-Class Mail Permit No. 7628 San Jose, California 95131

Postage will be paid by addressee

Integrated Solutions

An NBI Company

ATTN: Technical Publications Manager
1140 Ringwood Court
San Jose, CA 95131

Fold Up Second

Staple Here