Doing More with Less, Part Two

Jerry Peek

Over the years, "Power Tools" has shown a number of uses for the handy pager program *less*. This month, let's see even more.

Changing Options Interactively

Last month, you saw how to set default options in the `LESS` environment variable and in a *lesskey* file. (Recall that the *lesskey* file can centrally store your preferences, and you can even create a global preferences file for your system. See `man lesskey`.) Many options can be reset as you're running *less*. Simply type a dash, the option name string, and any argument to the option.

If the option is a toggle, its value is toggled. For example, if you're using *less* to read a manual page via *man*, the option `-i` (`--ignore-case`) is set by default so searches using the `/` command are case-insensitive. While you're reading a manual page, typing `-i` or `--ignore-case` makes searches case-sensitive. *less* explains the option that's been changed with a message `Case is significant in searches (press Return)`. (You generally don't need to press Return; just type the next command.)

If you type an option name that requires an argument, *less* immediately prompts for the argument. If you type `-j`, which sets the line number on the screen where the next search target will be displayed, you'll see the prompt `Target line:` at the bottom of the screen. Enter the line number and press Return.

To see the description and current setting of an option, type the option name, using an underscore wherever a dash normally appears. If you type `_i`, for instance, *less* either prints `Ignore case in searches` or `Case is significant in searches`.

(The built-in help has a brief summary of the options. Type `h` and scroll down to the "Options" section.)

Searching

The `/` command, followed by a string or an *ed*-like regular expression (not a shell wildcard pattern), searches forward in the file for the next occurrence of the pattern. The display jumps so that line falls on the first line of the screen or to the target line set by the `-j` option. (Setting `-j5` shows you four lines of context above the matching line.)

Use `?` instead of `/` to search backward (toward the start of the file). The `n` command repeats the search in the same direction (forward or backward), and `N` repeats the search in the opposite direction.

All of the matches on the current screen are highlighted unless you've set the option `-g` (which highlights only the first match) or `-G` (which prevents highlighting). You can also type `Escape-u` to turn highlighting off (or back on) for the current screen only; the next search command turns highlighting on.

If there are lots of matches for your search pattern, here's a way to save time. When search highlighting is set (if you haven't used `-g` or `-G`), *less* highlights all matches on the current screen so you can check all of them without scrolling. If you've also set the `-a` option, when you press `n` to repeat the search, *less* skips all the matches you've just seen and jumps to the first match following that screen.

less has several special search modes that change the way searches are performed. You set one of these modes by typing a special character immediately after the `/` or `?` command that starts a search. *Table One* lists these characters.

Using `^K` (Control-K) to inspect a file can be really handy when you're trying to find small things in a big window. It keeps your current position in the file (not moving forward or backward) and highlights all occurrences of the search pattern.

For instance, if want to view all of the decrement (`--`) and increment (`++`) operators in a screen full of C code, type `/^K` (slash, then Control-K) and *less* prompts with `Keep-pos /`. Then type the search pattern, `--`, and press Return; *less* highlights all of the decrement operators without moving. To see the increment operators on the same page, use the Control-R prefix character, as well as Control-K. Control-R searches for exactly what you type. (Typing `/^K++` gives an error because `+` is a metacharacter.) *less* will prompt with:

```
Keep-pos Regex-off /
```

Type `++`, then Return.

Starting a search with `/!` (or `?!`) jumps to the next (or previous) line that does not contain the pattern. (If highlighting is enabled, the occurrences of the pattern are still highlighted, but the search jumps over them.) In many cases, you may find that a regular expression containing negation

TABLE ONE: Characters that set search mode

CHARACTER	EFFECT
`^K`	Keep current position
`^R`	Don't interpret metacharacters
`!`	Find non-matching lines
`*`	If no match, search next file
`@`	Search from the first file

— such as `/[^;]$` to find the next non-empty line that doesn't end with a semicolon — is better than the special `!` "non-matching" operator.

Let's look at `*` and `@` in the next section.

Browsing and Searching Multiple Files

Like most command-line utilities, *less* accepts multiple filename arguments on its command line. (Because the shell expands wildcard characters like `*` into a list of matching filenames, wildcards work too, of course.) *less* starts by displaying the first file. Typing `:n` jumps to the next file; `2:n` jumps to the second-next file (skipping the next file), and so on. Use `:p` to jump to the previous file, or `3:p` for the third-previous file. If you don't have *less* set to prompt with the current filename, type `=` (or Control-G or `:f`) to reveal the current filename.

To view to the first file in the series, use `:x`. To view the *n*th file, use *n*:x— for instance, `4:x` to see the fourth file. If you've opened a file by mistake, or you want to remove it from the file list for any reason, type `:d` while you're viewing the file.

less can also search multiple files. This can be handy when, say, you're searching for all uses of a particular function name in a series of program files. Let's see a few ways to do searches that span files. You might start by typing `less */*.c` on the command line to read all files in first-level subdirectories with names ending in *.c*.

If you type `/doit` (to look for the a function named `doit`) while reading the first file and use `n` to repeat the search, *less* looks for that pattern only in the current file. It eventually prints `Pattern not found`.

If, instead, you had typed `/*doit` (using the `*` operator from *Table One*), when *less* runs out of matches in the current file, it searches the following files until it finds another match. A new filename is displayed at the bottom of the screen, as always, when *less* jumps to a new file.

Once you've done a multi-file search, you may want to search all of the files for another pattern. Typing `@` after the `/` operator "rewinds" the file list to the first file on the command line and starts your search from there. If you also want to do a multi-file search from this point, use both `@` and `*`. *less* will prompt:

```
First-file EOF-ignore /
```

After you type the search pattern and press Return, *less* jumps to the first match in the first file.

If you don't think of doing a multi-file search at the time you type the search pattern, that's no problem. After the search finds no more matches in the current file and *less* prints `Pattern not found`, instead of using `n` to repeat the search, use `ESC-n` and the search continues in the next file(s). Similarly, Escape-N jumps to the previous file and searches backward from its end.

New in less-394: History

Version 394 of *less* introduced a new feature: a plain-text history file. (To see what version of *less* you have, run `less -V` or `less --version`.) By default, *less* saves the last 100 search and shell escape commands.

(A shell escape is a Linux command-line preceded by `!`. *less* runs the command. After the commmand exits, *less* resumes showing the file you were viewing. For instance, `!cal` runs *cal* (1) to show this month's calendar.)

To use the stored history, type `/` or `!`, then press the Up-Arrow key on your keyboard. The most recent search or shell escape, respectively, appears next to that prompt. The *less* line editing commands (see" Line Editing" in the *less man* page) let you change the search or shell escape if you need to. Press Return to execute the result.

The history file's default name is *~/.lesshst*. You can override that by storing the file's absolute pathname in the new `LESSHISTFILE` environment variable. Setting `LESSHISTFILE` to `-` (a single dash) disables the history feature.

Scrolling sideways

Next, let's look at the way *less* handles lines that are too long to show on a single line of your window. The default for *less* is to wrap the line onto enough lines of your window so that all of its characters are shown. If the line is quite long, that can take a lot of room in your window and be very confusing to read, as well.

The `-S` option changes *less* to show only enough characters from the beginning of each line to fill the screen. The rest of the line is "off the right-hand edge", and you can see more of it by pressing your Right-Arrow key (or `Escape->`). The Left-Arrow key (or Escape-<;) moves back toward the start of the line. Give `-S` on the command line, or simply type it while *less* is running. Type it again to wrap lines.

The number of new characters revealed each time is half a screen width, by default. (If your terminal is 80 characters wide, you'll see 40 new characters for each press of an arrow key.) Entering a number before pressing an arrow key tells *less* how many characters to scroll on each arrow key press. For example, `80` followed by the Right-Arrow moves 80 characters right for that keypress and the others.

You can also set the shift-width by typing it on the command-line (or in the `LESS` environment variable) with the `-#` option. For instance, `less -#60` *filename* shifts by 60 characters.

You don't have to use `-S` to scroll sideways. When you're viewing one or more long lines with *less*, press your Right-Arrow key; the display will change to show the lines that were formerly wrapped as single lines instead. Moving left to the start of the line resumes the wrapped display.

Marking and returning

less can mark your current location in the file with the command `m` *x*, where *x* is a lowercase letter *a* through *z*. For example, `ma` marks the current location as "a". To return to that location, type a single quote character and the letter—for instance, `'a`— or Control-X Control-X and the letter. This is handy for marking certain spots that you want to remember and visit again as you're exploring a file.

Reading standard input, watching files grow

Twenty years ago or so — when most people used the *more* or *pg* pagers — one of the revolutionary features in *less* was that you could scroll backwards while viewing standard input. *less* does this by automatically buffering *stdin*.

You can also save the buffered standard input into a new file. This almost eliminates the need for the *tee* command in a pipe, such as:

```
$ someprog | tee someprog.out | less
```

To save standard input to a file, use the `s` command. For example, type `s someprog.out`. If *less* hasn't read all of the data from the pipe, it will at this point; there may be a pause as it reads data and writes the log file.

If you're watching a file that may keep growing — like a log file from a running program — the `R` command discards buffered input and re-reads the file to show you the latest.

You can also use the `+F` command-line option, or the `F` command, to follow the end of a file as it grows. This is similar to the Linux `tail -f` command, but it has the advantage that you can suspend watching the file with Control-C, scroll around the buffered input, mark places with the `m` command, then resume watching by typing `F` again. This can be very handy when you're tracking a program through its log file.

Use `++F` on the command-line to watch multiple files. *less* will display the first file and prompt `Waiting for data...(interrupt to abort)`. After you press Control-C (or your other interrupt key), you can use one of the file-changing commands like `:n` that you saw earlier. To quit monitoring all files, use a command like `q` to exit.

To Learn More…

There's more about *less* in the manual pages *less* and *lesskey*, as well as on the *less* home page, http://www.greenwoodsoftware.com/less/.

Jerry Peek is a freelance writer and instructor who has used Unix and Linux for more than 25 years. He's happy to hear from readers; see http://www.jpeek.com/contact.html.