Doing More with Less, Part One

Jerry Peek

Developer Mark Nudelman has been working on *less* since version *less-1*, released in 1985. This month, let's cover the *Linux* version *less-394* (yes, it's the 394th version), but you can also get *less* for other operating systems, including *Mac OS X* and *Microsoft Windows* — which makes it a great cross-platform choice for exploring files and viewing data from pipes.

Assuming you know the basics of *less*, let's look instead at overall setup and configuration. Next month we'll dig into details of using this powerful pager.

Default Options

Set the `LESS` environment variable in your shell's setup file (like *.profile*) with the command-line options *less* should use by default. For instance, if you like more-verbose prompting (`-m`), the target of a search to be shown on line 5 of the screen (`-j5`), search targets not to be highlighted (`-G`) and marked in the status column instead (`-J`), set `LESS=-mj5GJ`.

> Recent versions of less have extensive support for international character sets, including double-width.

Some options can have quite a bit of text after the option name, like `-P`, which sets the *less* prompt. To explicitly end an option and its text, use a dollar sign (`$`). For instance, you could set `LESS='-mj5$GJ'` to mark the end of the option `-j` and its argument `5`. (The single quotes around the value prevent the shell from treating `$` as the start of a shell variable.)

To override these default settings, type option names preceded by dash-plus (`-+`) on the command line. For instance, to override the `-J` option in the `LESS` string, type `less -+J filename.`

You can also set and override most options while *less* is running, as you'll see next month.

Controlling Character Display

Recent versions of *less* have extensive support for international character sets, including double-width and nonprintable characters, even on systems that don't support locales. You can also customize how binary and control characters are shown.

If your version of *less* doesn't do the right thing by default, or if you'd like to change the settings, see the *man* page section "National Character Sets".

Setting the Prompt

By default, when you start *less*, it prompts with the filename on the first screen, and then prompts with a colon (`:`).

The `-m` option prompts with the percentage of the file displayed so far, or the number of bytes when showing standard input.

To see the percentage while reading standard input, first jump to the end of the text with the *vi*-like command `G` or the Emacs-like command `ESC->`. *less* then knows how long the text is. Jump back to your previous place with `''` (two single quotes) or `Control-x Control-x'` and the prompt will display the number of lines and the percentage. One of the great features of *less* is that it automatically buffers standard input so you can jump back and forth through text read from a pipe.

The `-M` option adds the filename and the line numbers on the current screen.

You can also customize the prompt by setting the `-P` option. The string after `-P` can include:

- Plain text that's put in the prompt as-is.
- A percent sign (`%`) followed by one or two characters that's expanded into information like the current line or page number, the column number, the name of the next input file, and more.
- Question mark (`?`) and colon (`:`) are the operators for an "if-then-else" test that can change the prompt format on-the-fly. For instance, `?n` is true if this is the first prompt in a new input file, and `?e` is true at end-of-file. A period (`.`) ends the if-else. (Precede any of those special characters with a backslash (`\`) to include them in the prompt literally.)

A complete list of prompt options would fill most of this month's column, so read the manual page (which has a complete list of operators, and quite a few examples) for the details.

Let's see a simple example. When reading a file, *less* prompts with `Reading filename...;` otherwise it prompts `Reading stdin...`. Here's the prompt string:

```
Reading ?f"%f":stdin.\.\.\.
```

The word `Reading` and the space after it are output literally. The `?f` starts an if-else test that's true if a file is being read. If true, the text up to the colon, `"%f"`, is interpreted and output; it's the current filename inside a pair of double quotes. If `?f` is false, the text after the colon, `stdin`, is interpreted and output. The first period ends the if-else. The three other periods (which have backslashes before each to make them literal) are output at the end of every prompt.

If you're also storing other options in your `LESS` environment variable, it's probably most clear to put the `-P` option last. For example, to set the options `-j5` and `-a` plus that prompt, use:

```
LESS='-j5aPReading ?f"%f":stdin.\.\.\.'
```

Customizing the Editor

If you type the `v` command while *less* is running, it starts your default text editor (specified by your `VISUAL` or `EDITOR` environment variable). After you quit the editor, *less* displays the just-edited file.

By default, the first argument *less* gives the editor is *+nnn*, where *nnn* is the current line number in the middle of your screen. If your editor doesn't support that syntax (*vi*, Emacs, *nano*, and *Pico* all do) or if you'd like to customize the way the editor is started, you can change the editor invocation by setting the `LESSEDIT` environment variable, which has the same syntax as the prompt string. The default is:

```
%E ?lm+%lm. %f
```

It starts with the editor name (`%E`). Next is a test to see if the current line number is known, `?lm`. If it's known, a plus sign and the line number are output. Finally, `%f` gives the current filename.

Here's a brief example of changing the editor invocation string. You use the Emacs editor, and you're running *less* from within an *xterm*-like window. When you type the `v` command within *less*, Emacs opens a separate X window by default. This can be a bit disconcerting, so you want to add the `-nw` option to tell Emacs to edit within the same window as *less*. Since Emacs requires that the `-nw` option come first, you'd insert that option before the if-else test, like this command in a *.bash_profile* file:

```
export LESSEDIT='%E -nw ?lm+%lm. %f'
```

Preprocessing less Input

The `LESSOPEN` environment variable can name a program or script to preprocess the text that *less* displays. One popular use of this is to let you read compressed files directly. For instance, typing `less somefile.gz` would display the uncompressed *somefile.gz*. If you often read the directory of a *tar* file by typing `tar - tzvf tarball.tar.gz|less`, your preprocessor script could do the same thing when you type `less tarball.tar.gz`. One preprocessor script can handle many different situations.

If the preprocessor returns no output, *less* opens the input file as normal. The preprocessor is invoked once for each input file.

Although the program specified by `LESSOPEN` can write the preprocessed text into a temporary file (which can be removed by another program specified in the `LESSCLOSE` envariable), there's often a simpler way: Create a program that writes the preprocessed text to its standard output. Begin the `LESSOPEN` value with a vertical bar (`|`), then list the program's name, and a `%s` where the input filename should be substituted.

So, if you write a script named *lesspipe* and put it somewhere in your search path, you'd set:

```
LESSOPEN="|lesspipe %s"
```

Listing One shows a sample *lesspipe* shell script. It outputs a heading before the *tar* listings; because *less* displays the text it reads from the script's output, the heading appears before the listing.

Custom Commands

The *less* commands are mostly compatible with commands understood by the old *more* pager and with *vi* and Emacs edi-

LISTING ONE: Sample *lesspipe* shell script

```
#!/bin/sh
# lesspipe - preprocessor for less(1)
# Invoked via envariable: LESSOPEN='|lesspipe %s'

case "$1" in
*.tar)
   # Handle tar archive:
   echo "==== Directory of tarball '$1':"
   tar -tvf "$1"
   ;;
*.tar.gz|*.tgz)
   # Handle compressed tar archive:
   echo "==== Directory of tarball '$1':"
   tar -tzvf "$1"
   ;;
*.gz|*.[zZ])
   # Handle other compressed files.
   # gunzip can read .gz, .z, and .Z formats:
   gunzip -c "$1"
   ;;
esac
```

tor commands. That's a lot of commands! If your fingers tend to find the wrong keys sometimes, you may wish to disable many of the *less* commands you never use.

If you're using a non-U.S. keyboard, some of the *less* commands may fall on inconvenient or missing keys; you might want to assign them to more convenient keys. Or, for instance, you might think that `v` isn't a good command to start the editor; you'd rather use `e`.

And if your shell setup file (like *.profile*) has several environment variables for *less*, you could prefer to move that configuration information to a single central file.

Set the `LESS` environment variable in your shell's set-up file with the command-line options *less* should use by default.

All of that, and more, can be set up in a *lesskey* file. You write a plain-text input file with a filename you choose and pass that name to *lesskey:*

```
$ lesskey ~/.lesskey.input
```

lesskey generates a binary configuration file for *less*. The default name is `~/.less`. (You can also generate a system-wide file for all users. The *lesskey man* page explains.)

The *lesskey* input file can have up to three sections:

- A section starting with `#command` that defines command keys
- A `#line-edit` section that controls how you can modify the command line at the bottom of the *less* screen
- The `#env` section that defines environment variables like `LESS` and `LESSEDIT`.

Other lines starting with `#` and blank lines are ignored.

Each line in the `#command` section has the following syntax, separated by whitespace:

```
command-key-string action optional-input
```

The *command-key-string* is a sequence of up to 15 keys, including literal names like `e` (for the lowercase `e` key), escapes like `\t` (for the tab key), a caret followed by a letter (such as `^E` to represent control-E), and a backslash followed by one to three octal digits that specify a character by its octal value. If this string is the same as a default *less* command, your definition overrides the default.

When you type a *command-key-string* to *less*, it executes the corresponding *action*. The *action* is one of a long list of mnemonic names (listed in the *lesskey man* page) like `forw-screen` or `repaint`. It can also be the special word `invalid` (which makes *less* beep to signal an error) or `noaction` (which is simply ignored). Use those two words to disable default *less* commands.

The *optional-input* is input text that's fed to *less* as if you'd typed it on the keyboard after the *action* has been performed. For instance, let's say you wanted to define a key to display a particular file. The *action* would be `examine`; this causes *less* to prompt you for the filename to read. The *optional-input* would be the filename which you'd otherwise type from the keyboard. (If you didn't include an *optional-input*, then *less* would prompt you for the filename to examine.)

Each line in the `#env` section starts with the name of an environment variable, then an equal sign surrounded by optional whitespace (which is ignored), followed by the value of the environment variable. Any backslashes in the value should be doubled; that is, use `\\` for each single backslash you want to store in the environment variable.

As an example, let's disable the `v` key (which, by default, opens the file in your editor) and define the `e` key to do the same thing (instead of its default meaning, moving forward one line). Let's also define the `LESS` and `LESSEDIT` environment variables. *Listing Two* shows the input file.

There are many more details in the *lesskey* manual page.

Wrapup

There's even more you can do to configure *less*. To learn more, read the online manual pages and go to the *less* home page, http:// www.greenwoodsoftware.com/less/.

Next month, we'll dig into using *less*.

Jerry Peek is a freelance writer and instructor who has used Unix and Linux for 25 years. He's happy to hear from readers; see http://www.jpeek.com/contact.html. You can find an archive of "Power Tools" columns at http://www.linux-mag.com/channel/power-tools/.

LISTING TWO: Sample *lesskey* input file

```
# $HOME/.lesskey.input - input file for
lesskey(1)

#command
v    invalid
e    visual

#env
LESS = -j5aPReading ?f"%f":stdin.\\.\\.\\.
LESSEDIT = %E -nw ?lm+%lm. %f
```