Next Up Previous
: Command Arguments and Parameters : Shell Programming : Metacharacters

Creating New Commands

Given a sequence of commands that is to be repeated more than just a few times, it would be convenient to make the sequence into a single new command. To be concrete, suppose that intent to count users frequently with the pipeline

```
$ who | wc -l
```

The command `who` prints the names of logged in users (and some other details) one per line. This is piped to `wc -l` which counts the number of lines printed. (Under X Windows this is a poor way to count users because each xterm window is counted as a login. We will fix this later.)

The first step in creating a new command is to create a file which contains ``who | wc -l''. This can either be done with a text editor, or more creatively

```
$ echo 'who | wc -l' > nu
```

(Without the quotes, what would appear in `nu`?)

The shell is just a program; its name is `sh`. It can be invoked the same way as any command in the system. In particular, we can make the shell read its input from a file by using input redirection.

```
$ sh < nu
       4
```

Here the shell has read the command from the file `nu` and executed it.

Like other programs, the shell can take arguments. It interprets its first argument as a file to use as input. Thus

```
$ sh nu
```

will produce the same result. (Note that the command `sh < nu` is not the same as `sh nu`. The standard input to commands in the first command is taken from the file `nu` whereas the input to those in the second command is taken from the terminal.)

While `sh nu` is a shorthand way of determining the number of users, it still does not look like a regular command. In Unix if a file is executable and contains text it is assumed to contain shell commands. When the name of such a file is typed, a new shell is run to interpret the commands in the file. Such a file is called a *shell script*. The file `nu` can be made executable with the command:

```
$ chmod +x nu
```

Thereafter it can be invoked simply by typing its name.

```
$ nu
       4
```

To complete the process of making `nu` appear like a ordinary Unix command, it must be moved to one of the standard places where the shell looks for commands. The usual system places are inaccessible to ordinary users, but each user can create a ``bin'' directory in their home directory for their own commands. This directory can be created and the command `nu` installed in it as follows.

```
$ mkdir bin
$ mv nu bin
```

The command `mkdir` creates a new directory and the command `mv` moves `nu` into it. If you created `nu` somewhere other than your home directory the same effect can be obtained by

```
$ mkdir ~/bin
$ mv nu ~/bin
```

The `~` is shorthand for your home directory. (You can refer to other peoples home directories in a similar fashion; `~fred` is the home directory of the user ``fred''.)

Next Up Previous

: Command Arguments and Parameters : Shell Programming : Metacharacters

Ross Ihaka `��17ǯ11��26��