Connect to remote hosts with secure shell

In our last installment we discovered the terminal process and how to run commands using Unix or Linux. We listed several commands that are important to commit to memory.

Man(ual) pages

No one is expected to memorize the entire Unix manual (although there are those who've read it in its entirety). We're going to demonstrate how you can refer to the Unix manual to build out your knowledge with commands that we introduce, or that you may discover on your own as part of your learning journey. You will learn how to query the manual using its interface for the specific information you need for your script or command line statement.

For example, the shell manual-browser is accessed using the shorthand "$ man <enter>" followed by a utility or other query. If you type in only the access phrase you will receive the prompt "What manual page did you want?" as in the following:

```
~ $ man
What manual page do you want?
~ $ man grep
```

The Unix manual page for the grep utility will open in your terminal with details about how to use the program and specifications about what options are available to you. You can find the associated manual page for every program that comes with Unix (or Linux, depending on which system you're currently using).

When you issue commands, you can specify one or more options

using letters preceded by a hyphen or two. Most programs have the option to show some form of help text. For example, when browsing man pages you can type ‘h’ at any time to get its help text. This is important because it tells you how to exit! (Quit anytime by typing ‘q’.)

Note that you will be navigating mostly by keystrokes rather than moving a cursor around the screen with your mouse. We’ll explore the concept further with the Vim text editor. For now, just notice that arrow keys work to go up or down by line, and the spacebar ‘pages’ you down in screens. Type ‘q’ and quit should bring you back to your terminal prompt. Note that ‘j’ and ‘k’ keys, like up and down arrow keys, move you up or down the same way by line.

The man page for grep tells us how to print a brief help message:

~ $ grep –help

Use $ man grep and look under description for the option: –help. You’ll find most options only use a single hyphen but in grep’s case, the grep utility man page tells us that for the help text, two hyphens are required. The single hyphen ‘-h’ option is reserved for something else. This is worth noting because in most programs the help text requires only one hyphen: ‘-h.’ Whenever your intuition fails you, seek out the man page for the program you’re using for the correct syntax. Browsing the array of options for programs can also be a source of inspiration for scripting/constructing command line statements.

****

At this point you may want to refer to the “Commands to Know” section from the previous installment of this series. Another basic command you can commit to memory is $ echo ‘this text’ <enter>. The command simply ‘echoes’ the text encapsulated by quotes to the terminal. It is an excellent program to

demonstrate piping output to create a new file containing some text. Also, remember that anything after the hashtag is a comment that the program will ignore, used to convey information to other developers or simply as notes to oneself. In our examples here the comments are meant to convey information to you, the reader.

****

Plumb your output with pipes

Using pipes in the Unix programming environment is pretty commonplace for complex tasks. When you issue a command you can ‘pipe’ to another program in a single statement. The output will be used as input for the next program in a chain that can be however long you want. This is how short statements compound to become very long ones.

For example, if you print a file to the screen with $ cat, then you can pipe the output to word count ($ wc -w) and retrieve the number. Note that $ wc will return word, character, line, and byte count statistics, so using the phrase $ wc -w gives us the word count statistic, only.

Now, if you want to, you can use that output for another task, the output of which can be used for yet another task, and so on. Imagine the possibilities. There are also special pipes to replace or concatenate files with output from other programs. All these features can help you build content programmatically.

So let’s begin, starting very simply. Basic pipe syntax:

cmd1 | cmd2 (pipes output from cmd1 to cmd2)
cmd > file.txt (pipes output to replace contents of file.txt)
cmd >> file.txt (pipes output to concatenate on to contents of a file.txt)

$ echo ‘follow the white rabbit’ > testwc.txt # creates a file

```
named testwc.txt
$ cat testwc.txt | wc -w
4
$ rm testwc.txt # clean up
```

Opening a shell at GoDaddy

We're going to open secure shell processes on remote machines over the internet using SSH (Secure Shell). With GoDaddy, you need to switch the SSH service on and use the automatically generated cPanel username and password. Navigate to your web-hosting account (or create one) and find "Manage Hosting." From there, click the domain name you are hosting to find "Settings." Under "Account" you should see your cPanel login and a text link to change the password. Click "Server" under "Settings" to find the SSH switch and turn it on. You'll use your account's cPanel login string and a new password (change it) with SSH.

You can purchase a domain name relatively inexpensively through GoDaddy or if you'd rather not, you should be able to find the IP address in the cPanel, also under Settings. You'll have to use the IP address in lieu of the domain name if you opt to go without a domain.

For security purposes, some host providers close the port associated with the SSH service (22) in order to limit scripted hacking attempts (port 22 gets inundated). In GoDaddy's case, they have it off by default and you have to switch it on. Be sure to choose a strong password in order to be relatively safe. In other cases, the SSH service may be available through a different port number. It is also fairly commonplace that Virtual Private Servers (VPS) provision SFTP and SSH accounts automatically using your account username and password. In that case, that's all you need. Check with your host provider to get details on how to use SSH.

When ready, open a remote shell:

```
~ $ ssh cpanel-user@detlefjohnson.com-(or IP address)
Password:
cpanel-user@godaddy-host [~]$
```

If successful, you should get prompted for your password, and the resulting prompt should have GoDaddy information to the left of your tilde bash prompt (user and host names as strings). This information helps you know that you're operating on another machine, and where you are on that machine. Try your handy new navigational commands from our previous installment to print your working directory, list the contents of the directory where you currently are, and now also try a longer form of the list command:

```
$ ll
```

This command works because the default GoDaddy bash profile configured the 'll' alias to the list command using the most commonplace options ($ ls -l <enter>). You'll find, after time, that the list command on its own displays the folder items in a less than helpful way (alphabetically and across the screen). Using the long-form provides additional information and displays items one per line, which is much easier for visual consumption. You'll probably need to edit your own bash profile to get this to work on your local machine, which we'll do together in a future installment.

To see a list of pre-configured aliases simply type: $ alias <enter>. At GoDaddy the are several aliases set up for you by default. Take a look and check out the man page for any program that interests you. Keep in mind that there are aliases here which are meant for users coming from other operating systems. The commands which will (not always) result in the proper man page for you are those which appear after the '=' and are encapsulated by single quotes:

```
alias attrib='chmod'
alias chdir='cd'
```

```
alias cls='clear'
alias copy='cp'
alias d='dir'
alias del='rm'
alias deltree='rm -r'
alias dfh='df -h'
alias dir='/bin/ls $LS_OPTIONS –format=vertical'
alias dira='dir -a'
alias dird='dira | grep ^d'
alias drt='dir -rt'
alias edit='pico'
alias
ps1='PS1="\n[\[\033[1;33m\]\u\[\033[0m\]@\[\033[0;31m\]\h\[\03
3[0m\]][\[\033[1;36m\]\t\[\033[0m\]][\[\033[1;32m\]\w\[\033[0m
\]]\n#"'
alias ff='whereis'
alias l.='ls -d .* –color=auto'
alias ll='ls -l –color=auto'
alias ls='/bin/ls $LS_OPTIONS'
alias mem='top'
alias move='mv'
alias pico='pico -w -z'
alias psg='ps -aef | grep '
alias search='grep'
alias v='vdir'
alias vdir='/bin/ls $LS_OPTIONS –format=long'
alias vi='vim'
alias which='type -path'
```

Public HTML

The '/public_html' directory is set to be web-enabled, which means that it is exposed to the outside world and mapped with requests for your GoDaddy hosting account domain name (its IP address is synonymous by virtue of DNS). Web server settings are what dictate a particular setup. We will discuss these in greater detail and perhaps we will craft our own in a future

installment covering VPS deployment.

We’re going to create files in /public_html that you can request to display rendered HTML. The server will also collect logs written to: ~/access-logs/domain.tld, and are compressed monthly to ~/logs. Let’s create a couple of files, write a redirect, and grep the logs to see our hits.

Change directory to the web folder and create two files using echo and pipes (echo reprints the first argument, so be sure to encapsulate with quotes when you’re writing more than one word). The ‘&&’ between commands concatenates them to allow us more than one with a single statement:

```
$ cd ~/public_html
$ echo file1 > file1.html && echo file2 > file2.html
```

Now browse http://godaddy-domain-or-ip/file1.html to create a log entry. The page itself should just display the text: file1, which was inserted by echo and a pipe to replace (in this case create) the file contents. Next, we’re going to create a redirect to file2.html by using a ubiquitous command-line text editor called: Vim. Vim (Vi improved) is very confounding at first. You’re going to have to follow along closely just to learn the basics. If you do, a world will open for you where a local text editor is plainly insufficient. You can’t (easily) edit web server settings with notepad or textedit.

Vim

Launching Vim is pretty straightforward, but exiting it is not. To understand Vim in a nutshell, you need to know there are two primary modes. You begin and exit in command mode. The trouble people have exiting Vim is when they try to exit from insert mode. Insert mode is the mode that allows you to type and text is inserted in the document where the cursor is. Command mode is the mode where you issue commands, including

entering insert mode when ready to edit, or exiting the Vim session altogether. We're going to limit our learning here to opening Vim, creating a .htaccess file, entering insert mode, type the redirect directive, exiting insert mode, and then exiting Vim:

```
$ vim .htaccess
```

Vim should open for you with a blank screen, which would mean we're editing the .htaccess file since the command will create the file if it doesn't exist. To enter insert mode, press 'i' and begin by typing the following 302 redirect directive, replacing domain.tld with your domain or IP address:

```
redirect 302 /file1.html http://domain.tld/file2.html
```

Press 'ESC' to exit insert mode. Now that you're back in command mode, you can write the file with the text that's in Vim's buffer (what you typed) and exit back to the command prompt with the command: ':wq' <enter>. That's truly a colon in the command, which actually takes you to 'last line mode,' a mode that allows you to enter the last line command. It's named as such because the prompt appears on the last line of the editor screen.

Next time you visit file1.html you should be redirected to file2.html and the hit should be recorded in our log file. This should illuminate several things for you, dispelling the complexity of some major technical SEO work. The .htaccess file is just a simple text file with instructions. If you have a client that needs a few basic redirects and they happen to be hosted on GoDaddy, then you can use the above instructions to complete the task yourself. In that case, remember to issue one redirect directive per line. Make sure you test that it works as you want. Again, you can "comment a line" in the .htaccess file by using '#' at the start of the line.

Let’s grep some logs

Log files can be configured in a variety of ways but most are typically going to provide IP addresses, a time stamp, and request information including user-agent string. You can now do your own spider spotting! There will be log files entries even for those hits that don’t populate Google Analytics, which requires that users run JavaScript in order to work. It’s a great skill to be adept at using grep with server logs to mine for information. Remember to use the grep man page for help understanding the program and its options. Basic grep search syntax is: $ grep <query> <file-name>. Let’s look for our 302 hits:

```
$ grep 302 ~/access-logs/domain.tld
```

The result will be every line from the file which matches the query, which in our case should list 302 redirect responses, and possibly one or more other lines which have the ‘302’ string somewhere. We can use regular expressions to hone in on precise matches, which we’ll do in a future installment. Stay tuned!

What’s up next?

After this fairly intense piece, which would be pretty difficult if you’ve never done any of this before, we’ll relax a little and just start writing some basic HTML pages to get more experience with Vim commands. We’ll include the most important SEO components of HTML pages so that you can see SEO in action the way a technical SEO professional does, and you’ll know how to write webpage code from scratch before we begin programming pages dynamically using much more powerful code.

About The Author

Detlef Johnson is Editor at Large for Third Door Media. He writes a column for Search Engine Land entitled “Technical SEO for Developers.” Detlef is one of the original group of pioneering webmasters who established the professional SEO field more than 20 years ago. Since then he has worked for major search engine technology providers, managed programming and marketing teams for Chicago Tribune, and consulted for numerous entities including Fortune 500 companies. Detlef has a strong understanding of Technical SEO and a passion for Web programming. As a noted technology moderator at our SMX conference series, Detlef will continue to promote SEO excellence combined with marketing-programmer features and webmaster tips.
http://feeds.searchengineland.com/~r/searchengineland/~3/ghqHmJ2aWbc/connect-to-remote-hosts-with-secure-shell-323311