More More Linux Commands

Written by Dave Pawlowski, October 3, 2012

Introduction

There are many, many common commands that you will find on any Linux system. You've been introduced to several, here are a few more that will help you become a more skilled Linux user.

gzip *filename*

gzip is a command that will "zip" up, or compress, a file so that it takes up less space. Typically doing this will make the file take up half as much space.

gunzip *filename*

This will unzip a file that has been zipped.

tar

tar is a command that lets you package up and compress several files into a single "file". There are many flags that you can use with tar, but we will only deal with the most common here. First, copy dpawlows/Public/Phy379/scripts.tar to one of your directories.

Typically, a set of files that have been tarred together is referred to a "tarball". You can extract the files from the tarball using:

```
% tar -xvf scripts.tar
```

The -v flag is common among many Linux commands and stands for verbose, i.e. it will tell you what is going on. Here, you get a list of the files that were extracted. If you

```
% ls
```

you should see that, indeed, those files were extracted. The tar command can be used to extract files from a tarball as well as create a tarball. All you have to do is use the -c flag instead of the -x flag to create:

```
% tar -cvf allscripts.tar *.sh
```

will create a new tarball called allscripts.tar that contains all of those scripts. Note the order of the arguments, the file to be created goes first, and the files to tar up second. Also, note the use of the wildcard.

In addition to putting all the files together, it is also possible to compress the files, you know, like when you download those files that end in .zip.

```
% tar -czvf allscripts.tgz *.sh
```

Will create a zipped tarball. Again, changing the -c to -x will extract the files from an existing tarball.

As with any file, the extensions do not mean anything. .tar and .tgz are typical extensions given to a file that has been tarred and tarred/compressed, but you don't have to use those extensions. The common extensions do help let other people know what to do with them if they are given such files. Also, certain GUI programs need to have the correct extension in order to work. Another reason we don't like GUIs. They generally aren't very smart.

scp

You should know how to login to a remote unix system using ssh, but what if you want to transfer a file? scp allows you to do this *securely*. The syntax is as follows:

scp username@hostname:'path of file to transfer' username@hostname:'path to transfer to'

The first arguement contains information about the file that you want to transfer, and the second arguemnt contains information about where you want to transfer to. You should, in general, include the quote marks on whichever machine is the remote machine. You don't have to include the username or hostname of your local machine. For example, if I wanted to send a file to my yipe account, I would do:

```
% scp scripts.tar dpawlows@yipe.emich.edu:'~/'
```

This will copy the file to my home directory on Yipe. Try this with your own account. To get a file from yipe:

```
% scp dpawlows@yipe.emich.edu:'~/scripts.tar' '.'
```

will bring the file to my local machine and put it in my current working directory.

If you want to transfer an entire directory, simply include the -r flag.

```
% scp -r ~/Documents/ yipe.emich.edu:'~/'
```

will transfer my Documents directory and all of its contents to yipe. Note that I don't actually need to include my username if it is the same on the remote system as it is on my local machine.

awk

awk is an extermely powerful Linux utility that has entire books devoted to it. awk is a tool that is used to work with regular expressions, or a pattern of characters used to match the same characters in a search through text. The pattern can include special characters to refine the search, as well as do a bunch of other stuff. If you end up writing lots of shell scrips which manipulate text, learing regular expressions is well worth the time.

In this context, we are just going to use awk for a couple things.

The first will be obtaining the length of a string:

```
% set string="Processing NGC 2345"
% set nchar=`echo $string |awk '{print length($0)}'`
```

Note, the quote marks that surround everything right of the equals sign are typed by pressing the key with the tilde on it, not the normal single quote key. If you look closely, they are indeed different. Those are special quotes that allow the execution of commands within the string itself. Basically, Linux sees that whole thing as a string, except it checks for commands and executes them if one or more is found.

```
% echo $nchar
```

This should tell you the length of the string that you entered.

It is also possible to determine the position of a substring within another string using awk:

```
% set catalog="NGC "
% set number=2345
% set object="Processing $catalog$number."
% set index1=`echo $object |awk '{print index($0,"ngc")}'`
% set index2=`echo $object |awk '{print index($0,"NGC")}'`
```

Finally, we can use awk to extract a substring fomr a larger string:

```
% set littlestring1=`echo $object |awk '{print substr($0,12,8)}'`
% set littlestring2=`echo $object |awk '{print substr($0,16)}'`
```

Compare $littlestring1, $littlestring2, and $object to see what awk is doing here. The numbers in the substr method of awk refer to the positions of a character in the larger string. It is useful to use the index method combined with the substr method: index can help you find the position of a substring, while substr will actually extract it.

There is much, much more that you can do with awk. For example, you can split up a string into an array, change the case of a string, and substitute one substring for another. The tools that we have dealt with are good enough for now though. They can be quite useful in scripts, especially for dealing with files and directories.

cmp

If you have two files that you want to compare and make sure that they are identical, the cmp command will do that for you. You should have a few shell scripts in your current directory if you tried using tar as described above:

```
% cp for1.sh forcopy.sh
% cmp for1.sh forcopy.sh
```

will return absolutely nothing, since those two files are definitely identical. However,

```
% cmp for1.sh for2.sh
```

will tell you that the two files do indeed differ, and point you to the first differing occurrence.

diff

If you actually want some detail about how two files differ, then use diff:

```
% diff for1.sh for2.sh
```

This will tell you every line that differs between two files, with the less than and greater than symbols referring to the first and second files respectively.

w

Tells you who is logged in and what they're doing.

last -1 *username*

Tells you when the user last logged on and off and from where. If you leave off any arguments, you will get a list of everyone's login info.

write *username*

Allows you to write short messages to another user. This is often disabled by default.

du *directory*

Shows the disk usage of the files and directories in *directory*.

df

Shows the disk usage of all the disks connected to the system.

quota -v

Shows your disk quota (if you have one). If you have access to a public system, i.e., yipe.emich.edu, you will have a limited amount of disk space that you are able to use.