Lecture 4: Running Commands

CS2042 - UNIX Tools

October 6, 2008

Lecture Outline

1. More Common Commands
   - Reading Commands
   - Writing Commands
   - Recursive Commands
2. Combining Programs
   - Operators
   - Exercises

More or Less

More

more [filename]

- Allows you to scroll through a bunch of text 1 page at a time
- Good for quick viewing of text files or slowing down the output of programs

Less

less [filename]

- Similar to more, but better!
- Lets you scroll up or down, by pages or lines

Head and Tail

Head and Tail

head [-numlines] [filename]
tail [-numlines] [filename]

- Prints the first (**head**) or last (**tail**) lines of a file
- Prints 10 lines by default, or the number specified by *numlines*

Example:

- **head -15 /var/log/Xorg.0.log** - Prints the first 15 lines of **/var/log/Xorg.0.log**

Lecture Outline

1. More Common Commands
   - Reading Commands
   - Writing Commands
   - Recursive Commands
2. Combining Programs
   - Operators
   - Exercises

The Echo Command

echo

echo <text_string>

- Prints the input string to standard output (the terminal)

Example:

- **echo this is a string** - Prints "this is a string" (without quotes)
- **echo "this is a string"** - Prints the exact same thing

- This probably seems stupid and useless now - we'll come back to this near the end of the lecture.

The Cat Command

Concatenate

cat [file1] [file2]

- Concatenates file(s) or standard input and prints them to standard output

Example:

- **cat test1 test2** - Prints the contents of *test1*, then *test2*

- Again, we'll see a practical use for this a little later.

Lecture Outline

1. More Common Commands
   - Reading Commands
   - Writing Commands
   - Recursive Commands
2. Combining Programs
   - Operators
   - Exercises

Recursion

We have learned to copy, delete, and change the permissions of single files. We can even do it with multiple files using wildcards (**rm *.doc**). However, what if we want to act on every file in every subdirectory of our target?

- Use the recursive form of the command.
- Usually means a **-r** or **-R** option; check the manpage for details.
- Doesn't make sense for many commands, such as **mv** - thus, the recursive option doesn't exist

Example:

chmod -R o-w ~/documents/

- Removes write privileges for other users for every file & directory contained in *~/documents/*

Lecture Outline

1. More Common Commands
   - Reading Commands
   - Writing Commands
   - Recursive Commands
2. Combining Programs
   - Operators
   - Exercises

Shell Operators

- && - Run concurrent commands
- | - The "pipe" operator
- > - Output to a file

These special characters (along with a few others which we'll cover later) add a great deal of flexibility to your shell experience.

Running Commands Sequentially

The && Operator

<command1> && <command2>

- Immediately after command1 completes, execute command2

- command1 must complete successfully for command2 to run!

Example:

mkdir photos && chmod o-rw photos

- Creates a directory and sets its permissions

Piping Output to Input

The Pipe Character

<command1> | **<command2>**

- Passes output from command1 to input of command2

- Works for many programs which take input from/provide output to the terminal

Example:

ls -al /bin | **less**

- Allows you to scroll through the long list of programs in */bin*

history | **head -10** | **tail -5**

- Displays the 6th-10th commands from the current session

Outputting to a File

The Greater-Than Operator

<command> > <file>

- Writes output of *command* to the specified file

- Any program that outputs to the terminal can have its output redirected to a file.
- This can be useful for logging output or for creating/modifying files.

Example:

echo "This is a new file." > newfile

- Writes that string to *./newfile*

cat test1 test2 > test3

- Concatenates *test1* and *test2*, storing the result in *test3*

Lecture Outline

1. More Common Commands
   - Reading Commands
   - Writing Commands
   - Recursive Commands
2. Combining Programs
   - Operators
   - Exercises

Exercises:

Create a new directory named *assign1* in your home folder. Use **nano** (or some other editor if you have a preference) to create a file *answers* with the answers to these questions.

1. Does **echo "1234" && echo "5678" > test1.txt** write both strings to *test1.txt*? If not, write a command which does.
   Hint: You may need to use parentheses () to define order of operations!
2. You can pipe input into **cat** - write a command which uses this.
3. Find a plaintext file outside your home directory (try */etc* or */var/log*) which you have permissions to read. Give the absolute filename in *answers* and copy the last 15 lines of the file you found to a new file named *~/assign1/lines*.