5. Viewing & Creating Files

This lab will show you the different ways you can view what's in a file as well as some ways to actually create files. Everyone has a file called .cshrc in their home directories which is a file created to set your Unix environment variables. This file should not be altered or tampered with unless you know what you are doing. Because this file is full of text and you do not have any other files created yet, we will use his file to show the viewing commands work.

Lab 5.1 - cat

The **cat** command prints the contents of the specified files to the screen.

```
cat .cshrc        // the .cshrc is a file created to set your Unix environment variables
```

Lab 5.2 - more

The **more** command can be used to view a file or files one page at a time. While executing more, press the space bar to display the next screen of text. Pressing return will advance the screen forward one line at a time.

```
more .cshrc
```

Lab 5.3 - pg

The **pg** command can be used to view a file or files one page at a time Pressing return will advance the screen forward one screen at a time. There are other options available for searching, using the pg command, available on the manual pages.

```
 pg .cshrc
```

Lab 5.4 - head and tail

The **head** command reads the first few lines of an input file and send them to your terminal screen.

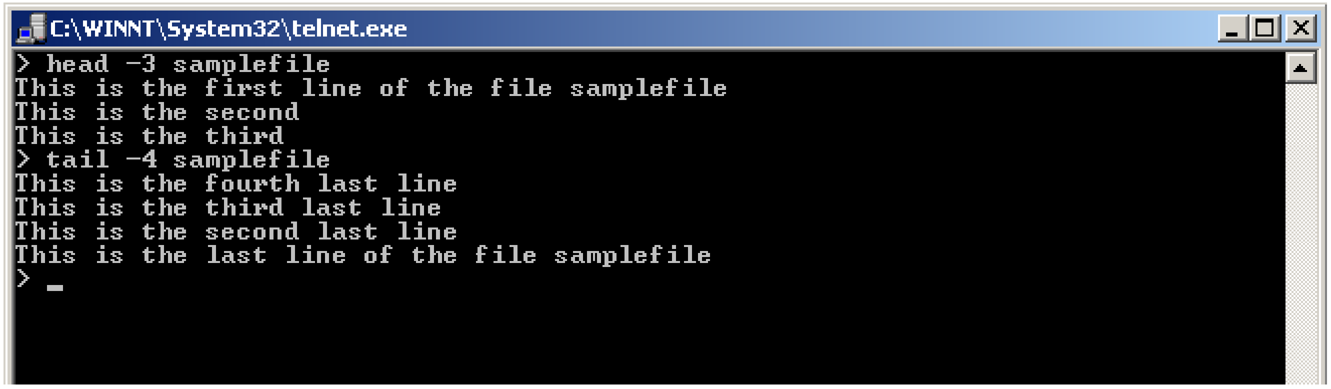
```
head -4 .cshrc
```

In this example, 4 is an optional value specifying the number of lines to be read. If you don't give a number, the default value of 10 is used.

The **tail** command reads the last few lines of an input file and send them to your terminal screen.

tail -4 .cshrc

In this example, 4 is an optional value specifying the number of lines to be read. If you don't give a number, the default value of 10 is used.



```
> head -3 samplefile
This is the first line of the file samplefile
This is the second
This is the third
> tail -4 samplefile
This is the fourth last line
This is the third last line
This is the second last line
This is the last line of the file samplefile
> _
```

Figure 5.1: Sample use of the head and tail command

Lab 5.5 - Touch

There are many ways to create files. The **touch** command is used to create an empty file, which can be edited later, for example the command:

touch zeebrafile

ls zeebrafile

zeebrafile

will create an empty file called **zeebrafile** in your current directory.

Lab 5.6 – Redirection (another way to create files)

Most processes initiated by UNIX commands write to the standard output (that is, they write to the terminal screen), and many take their input from the standard input (that is, they read it from the keyboard). There is also the standard error, where processes write their error messages, by default, to the terminal screen.

In UNIX, we can redirect both the input and the output of commands.
When you issue a Unix command like date, the output of the command goes to the *standard output or terminal screen* so you can see the results of your command.

```
date
Tue Dec 19 21:16:03 CST 2000
```

5.6.1 Redirecting standard output to a file

In Unix, you can redirect standard output to go to a file. This output redirection is very useful if you want to save the output of a command to a file rather than just letting it flash across the screen and eventually scroll away from view.
You can redirect standard output to go to a file by using the > sign after the command and before a file name. For example, you can redirect the output of the date command to a file called apple:

```
date > apple
```

Notice that we did not see the date appear on the screen like before. The output of the date command is in the file apple. We can use the cat command we learned earlier to show the contents of apple.

```
cat apple
Tue Dec 19 21:16:43 CST 2002
```

When you use the > for redirecting standard output to the file, Unix puts what would appear in standard output (the screen) into the file. This erases what would previously have been in the file.

```
cat apple
Tue Dec 19 21:16:43 CST 2002
date > apple
cat apple
Tue Dec 19 21:17:07 CST 2002
```

Notice the output of the date command stored in apple has changed with the updated time. It has overwritten what was in the file previously.

You can redirect the output of any command.

```
ls -l > apple2
cat apple2
```

Suppose you want to join a couple of files

```
cat apple apple2 > apple3
cat apple3
```

This would add the contents of ' apple ' and ' apple2 ' and then write these contents into a new file named ' apple3 ' .

5.6.2 Appending to a file by redirecting standard output

If you want to keep adding content to a file using redirection and not just overwrite what was there already, use the >> symbol (two > right in a row, no spaces). The >> symbol appends what would go to standard output to a file. So, for example, we can do this:

```
cat apple
Tue Dec 19 21:17:07 CST 2000
date >> apple
cat apple
Tue Dec 19 21:17:07 CST 2000
Tue Dec 19 21:17:53 CST 2000
```

Notice that apple in the end now has two lines of output from the date command in it. The file has to exist already obviously if you are appending to it or else the command will not work.

```
cal >> apple      (can append any commands not just same one)
```