Start Recording Your Commands

Run **script -a**to record the commands that are typed in to the prompt. Run the following

% script -a

On-line Manuals

Going forward, it is recommended that you use the **man** command for each command that we have seen and will see. For example,

% man wget

You can also find examples of how to use commands on StackOverflow by search for the command using a search engine.

3.1 Redirection

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.

We have already seen one use of the cat command to write the contents of a file to the screen.

Now type cat without specifying a file to read

% cat

Then type a few words on the keyboard and press the [Return] key.

Finally hold the [Ctrl] key down and press [d] (written as ^D for short) to end the input.

What has happened?

If you run the cat command without specifying a file to read, it reads the standard input (the keyboard), and on receiving the 'end of file' (^D), copies it to the standard output (the screen).

In UNIX, we can redirect both the input and the output of commands.

3.2 Redirecting the Output

We use the > symbol to redirect the output of a command. For example, to create a file called **list1** containing a list of fruit, type

% cat > list1

Then type in the names of some fruit. Press [Return] after each one.

pear  
banana  
apple  
^D (Control D to stop)

What happens is the cat command reads the standard input (the keyboard) and the > redirects the output, which normally goes to the screen, into a file called **list1**

To read the contents of the file, type

% cat list1

Exercise 3a

Using the above method, create another file called **list2** containing the following fruit: orange, plum, mango, grapefruit. Read the contents of **list2**

The form >> appends standard output to a file. So to add more items to the file **list1**, type

% cat >> list1

Then type in the names of more fruit

peach  
grape  
orange  
^D (Control D to stop)

To read the contents of the list1 file, type

% cat list1

To read the contents of the list1 and list2 files, type

% cat list1 list2

You should now have two files. One contains six fruit, the other contains four fruit. We will now use the cat command to join (concatenate) **list1** and **list2** into a new file called **biglist**. Type

% cat list1 list2 > biglist

What this is doing is reading the contents of **list1** and **list2** in turn, then outputting the text to the file **biglist**

To read the contents of the new file, type

% cat biglist

3.3 Redirecting the Input

We use the < symbol to redirect the input of a command.

The command sort alphabetically or numerically sorts a list. Type

% sort

Then type in the names of some vegetables. Press [Return] after each one.

carrot  
beetroot  
artichoke  
^D (control d to stop)

The output will be

artichoke  
beetroot  
carrot

Using < you can redirect the input to come from a file rather than the keyboard. For example, to sort the list of fruit, type

% sort < biglist

and the sorted list will be output to the screen.

To output the sorted list to a file, type,

% sort < biglist > slist

Use cat to read the contents of the file **slist**

3.4 Pipes

To see a long list containing details of files in your current working directory, type

% ls -l

One method to get a sorted output of **ls -l** is to type,

% ls -l > tmp.txt  
% sort < tmp.txt

This is a bit slow and you have to remember to remove the temporary file called 'tmp.txt' when you have finished. What you really want to do is connect the output of the who command directly to the input of the sort command. This is exactly what pipes do. The symbol for a pipe is the vertical bar |

For example, typing

% ls -l | sort

will give the same result as above, but quicker and cleaner.

To find out how many lines are in the output of **ls -l**, type

% ls -l | wc -l

Exercise 3b

**cat jokes.txt** is the command to show the contents of the file named 'jokes.txt'.

Using pipes, print all lines in 'jokes.txt' containing the word 'What', sort the result, and write the resulting output to a file called 'what.txt'.

[PipeAnswer](https://uwrvf.instructure.com/courses/474790/pages/pipeanswer)

3.5 Wildcards

The characters \* and ?

The character \* is called a wildcard, and will match against none or more character(s) in a file (or directory) name. For example, in your **unixstuff** directory, type

% ls list\*

This will list all files in the current directory starting with **list....**

Try typing

% ls \*list

This will list all files in the current directory ending with **....list**

The character ? will match exactly one character.  
So ls ?ouse will match files like **house** and **mouse**, but not **grouse**.  
Try typing

% ls ?list

3.6 Filename conventions

We should note here that a directory is merely a special type of file. So the rules and conventions for naming files apply also to directories.

In naming files, characters with special meanings such as **/ \* & %** , should be avoided. Also, avoid using spaces within names. The safest way to name a file is to use only alphanumeric characters, that is, letters and numbers, together with \_ (underscore) and . (dot).

File names conventionally start with a lower-case letter, and may end with a dot followed by a group of letters indicating the contents of the file. For example, all files consisting of C code may be named with the ending .c, for example, prog1.c . Then in order to list all files containing C code in your home directory, you need only type ls \*.c in that directory.

**Beware**: some applications give the same name to all the output files they generate.  
  
For example, some compilers, unless given the appropriate option, produce compiled files named **a.out**. Should you forget to use that option, you are advised to rename the compiled file immediately, otherwise the next such file will overwrite it and it will be lost.

3.7 Submit Your Work

Stop the script command by typing

% exit

Then type

% check50 uwrf-csis/csis248/main/lab3

Make sure all checks pass. If they do not, then you probably skipped a step in the lab. The checks show which commands are missing. Rerun the **script -a** command and finish the lab and verify that all checks pass. Don't forget to use the **exit** command to stop the script command.

Finally, submit your work by typing the following command and answering the question that follows.

% submit50 uwrf-csis/csis248/main/lab3

Summary

|  |  |
| --- | --- |
| command > file | redirect standard output to a file |
| command >> file | append standard output to a file |
| command < file | redirect standard input from a file |
| command1 | command2 | pipe the output of command1 to the input of command2 |
| cat file1 file2 > file0 | concatenate file1 and file2 to file0 |
| sort | sort data |
| who | list users currently logged in |
| \* | match any number of characters |
| ? | match one character |