Linux

Streams, Pipes and Redirects

Watch the following videos BEFORE you begin this exercise:

https://www.youtube.com/watch?v=r-KlaB-c68U

https://www.youtube.com/watch?v=nCHjYP7kqYU

https://www.youtube.com/watch?v=faTXIBdZt-0

https://www.youtube.com/watch?v=o-permmaDYs

Redirection and Piping

As you study this section, answer the following questions:

- What is the difference between *redirection* and *piping*?
  - Redirection is when you are directing output toward or form a file. Piping is when you are directing output toward or form a command. ?
- When might you choose to redirect the input of a command?
  - You might redirect the input of a command (that is, use the contents of a file as the input to a command) when you are working with a command that doesn't read the contents of a file when the filename is used as an argument. ?
- What are the three default file descriptors that Linux uses to classify information for a command?
  - 0 refers to standard input, 1 refers to standard output, 2 refers to standard error.
- How can you overcome the 128 KB shell command size restriction?
  - You may be able to overcome the 128 KB shell command size restriction by using xargs.

After finishing this section, you should be able to complete the following tasks:

- Redirect the standard output from the screen to a file.
- Redirect and append new content to an existing file.
- Redirect a standard error from a command to a file.
- Redirect the standard input to a command.
- Pipe the output of a command to the input of another command.
- Use the **pipe** command to search a file for specified text.
- Use the **pipe** command to create a text stream.

Redirection and Piping Facts

Administrators often use the following methods to create, send, or gather information on a Linux system:

Method	Description
Redirection	*Redirection* directs standard input, output, and error streams from and to locations other than the default. Be aware of the following redirection details: • By default, the Linux system classifies information with the following file descriptors: ○ Standard input (stdin) comes from the keyboard. In redirection, 0 represents stdin. ○ Standard output (stdout) displays on the monitor. In redirection, 1 represents stdout. ○ Standard errors (stderr) display on the monitor. In redirection, 2 represents stderr. • Linux commands use the greater-than symbol (>) to show redirection of output, the less-than symbol arrow (<) to indicate redirection of input, and the double greater-than symbol (>>) to append the output to another file or command. • The **tee** command reads from standard input and writes to standard output *and* files.
Piping	*Piping* directs the output of one command into the input of another command. Pipes: • Use the pipe symbol (|). • Can combine several commands to make a stream.

The following table shows the results of several redirection and piping commands:

Example	Result
ls /usr > /tmp/deleteme	**ls /usr > /tmp/deleteme** places the list of files in the **/usr** directory into a file named **/tmp/deleteme**.
ls /nonesuch > /tmp/deleteme	**ls /nonesuch > /tmp/deleteme** does not write anything to a file and sends an error message to the monitor '/nonesuch not found'.
ls /nonesuch 2 > /tmp/deleteme	**ls /nonesuch 2 > /tmp/deleteme** writes the standard error message to a file named **/tmp/deleteme**.
ls /bin /nonesuch > /tmp/deleteme	**ls /bin /nonesuch > /tmp/deleteme** writes the contents of the **/bin** directory to the **/tmp/deleteme** file, but sends the error message '/nonesuch not found' to the screen.

ls /bin /nonesuch > /tmp/deleteme 2>&1	**ls /bin /nonesuch > /tmp/deleteme 2>&1** directs the standard output to the **/tmp/deleteme** file, then directs that the standard error messages be sent to the same place as the standard output. Both the list of files in the **/bin** directory and the error message are written to the file.
ls /bin /nonesuch 2>&1 > /tmp/deleteme	**ls /bin /nonesuch 2>&1 > /tmp/ deleteme** writes the contents of the /bin directory to the **/tmp/deleteme** file, but sends the error message '/nonesuch not found' to the screen. This is because standard error messages are directed to the same place that standard output goes, but this is before standard output has been directed to the file.
ls /bin >> /tmp/deleteme	**ls /bin >> /tmp/deleteme** appends the list of files from the **/usr** directory on to the end of the **/tmp/deleteme** file.
sort < unordered_file.txt > ordered_file.txt	**sort < unordered_file.txt > ordered_file.txt** takes input from the unordered_file.txt file sends it to the sort command, and then writes a new file named **ordered_file.txt**.
cat /usr/wordlist1 /usr/wordlist2 | sort	Sends the output of the cat command, the contents of wordlist1 and wordlist2, to the input of the sort command. The result is a sorted list of the combined contents of **wordlist1** and **wordlist2**.
cat /usr/wordlist1 /usr/wordlist2 | mail jdoe	Mails the combined list of words in **wordlist1** and **wordlist2** to the user jdoe.
ls /bin | sort | mail jdoe	Lists the contents of /bin then sorts the combined contents and mails them to jdoe.
cat /usr/wordlist1 /usr/wordlist2 | sort | tee sortedwordlist	Lists the contents of **wordlist1** and **wordlist2** then sorts the combined contents, then sends the results to the monitor and a file named **sortedwordlist**.
cat /usr/wordlist1 | tee log.txt	Writes to the standard output and the log.txt file.

Lab 1 Redirection

Most processes initiated by Linux 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

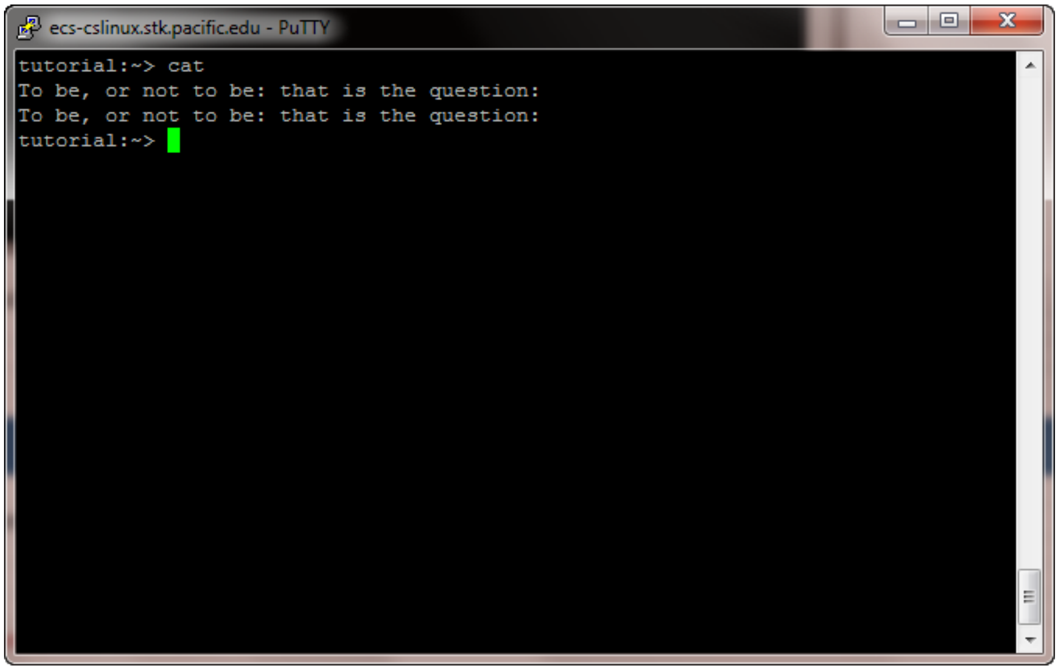
% cat

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 OR ^C -depends on your shell)** 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 OR ^C), copies it to the standard output (the screen).



```
tutorial:~> cat
To be, or not to be: that is the question:
To be, or not to be: that is the question:
tutorial:~> 
```

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

Lab 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, at the prompt type

cat > list1

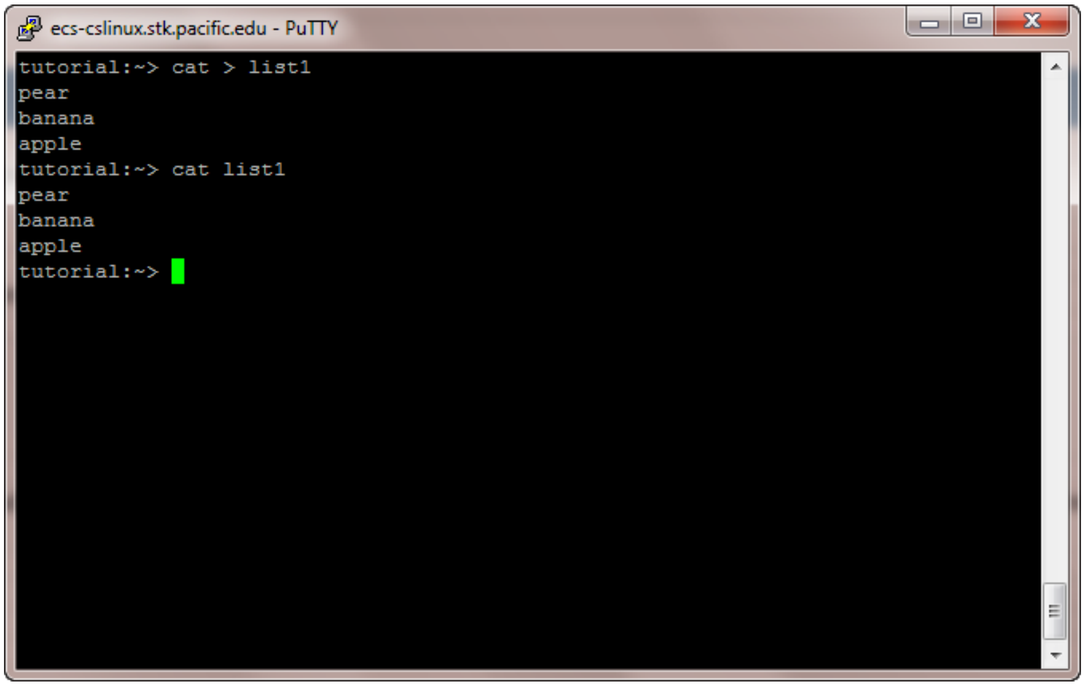
Then type in the names of some fruit. Press `[Return]` after each one.

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, at the prompt type

cat list1



```
tutorial:~> cat > list1
pear
banana
apple
tutorial:~> cat list1
pear
banana
apple
tutorial:~>
```

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**, at the prompt 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 file, at the prompt type

cat list1

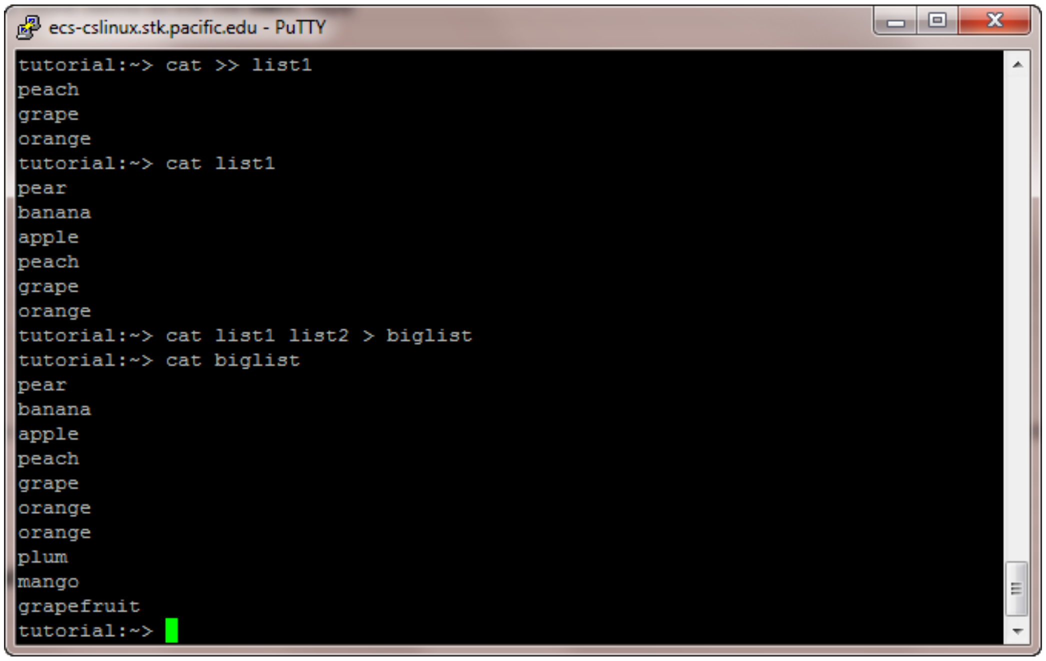
You should now have two files. One contains six fruit, the other contains four fruit. We will now use the `cat` command to join (concatenate) **list1** and **list2** into a new file called **biglist**. At the prompt type:

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, at the prompt type

cat biglist



```
tutorial:~> cat >> list1
peach
grape
orange
tutorial:~> cat list1
pear
banana
apple
peach
grape
orange
tutorial:~> cat list1 list2 > biglist
tutorial:~> cat biglist
pear
banana
apple
peach
grape
orange
orange
plum
mango
grapefruit
tutorial:~>
```

Lab 4 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

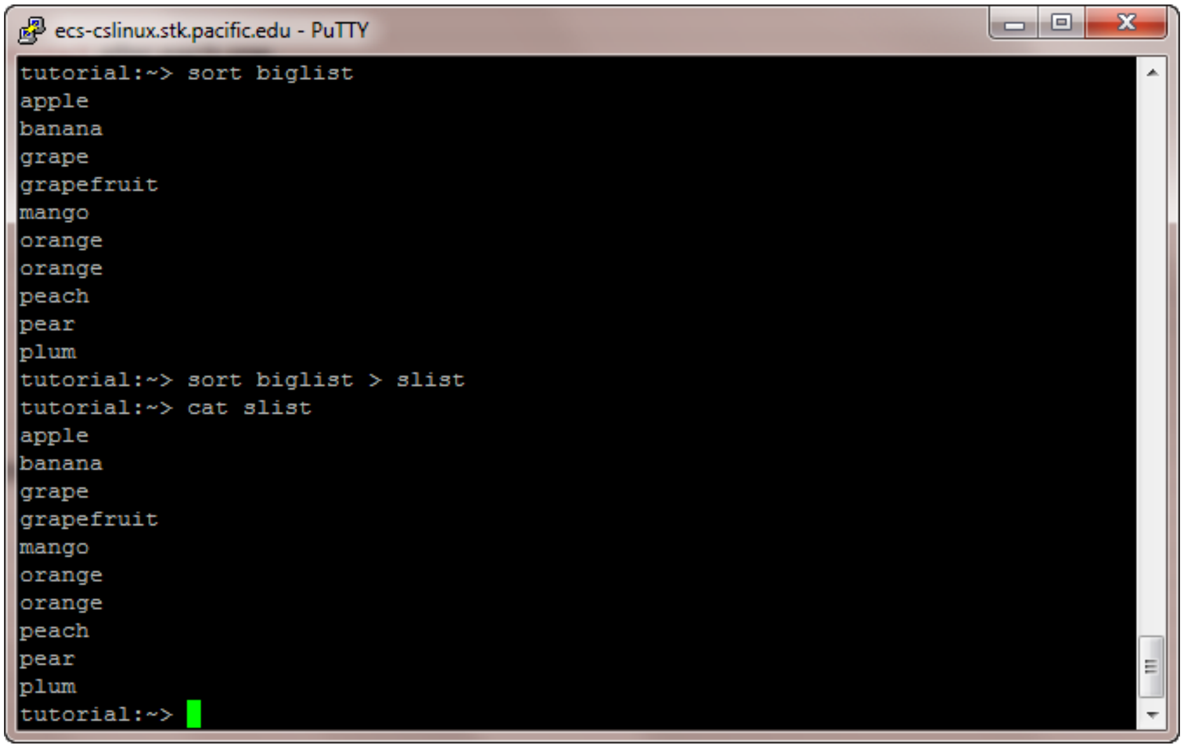
sort < biglist

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**

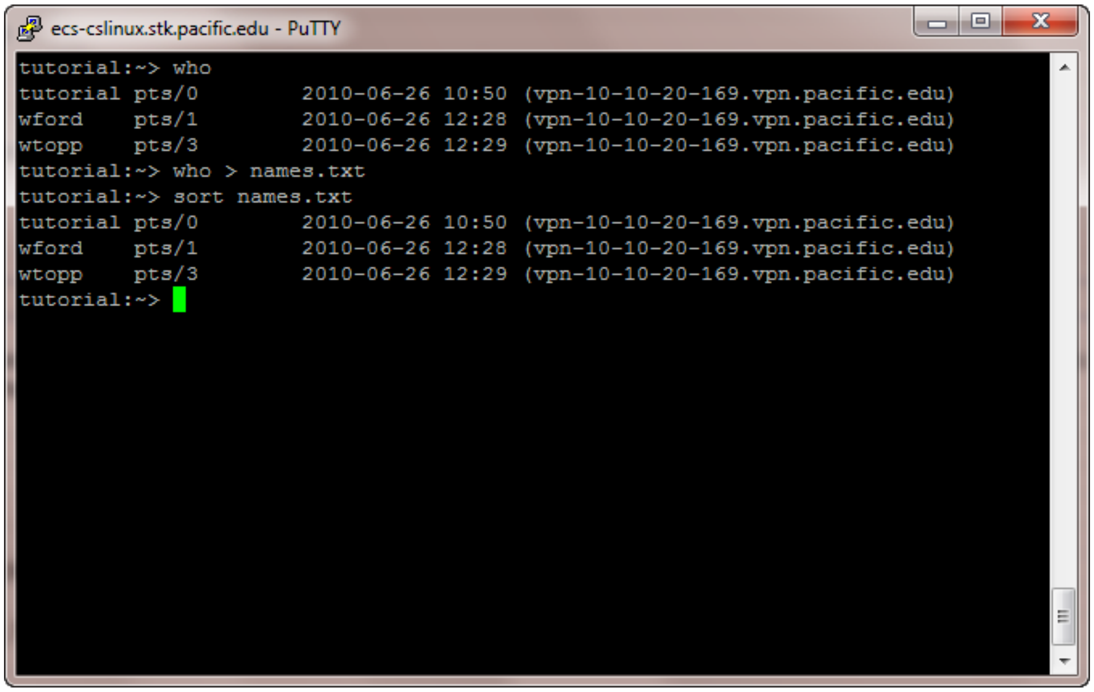


```
tutorial:~> sort biglist
apple
banana
grape
grapefruit
mango
orange
orange
peach
pear
plum
tutorial:~> sort biglist > slist
tutorial:~> cat slist
apple
banana
grape
grapefruit
mango
orange
orange
peach
pear
plum
tutorial:~>
```

Lab 5 Pipes

To see who is on the system with you, type: who
One method to get a sorted list of names is to type,

who > names.txt
sort names.txt

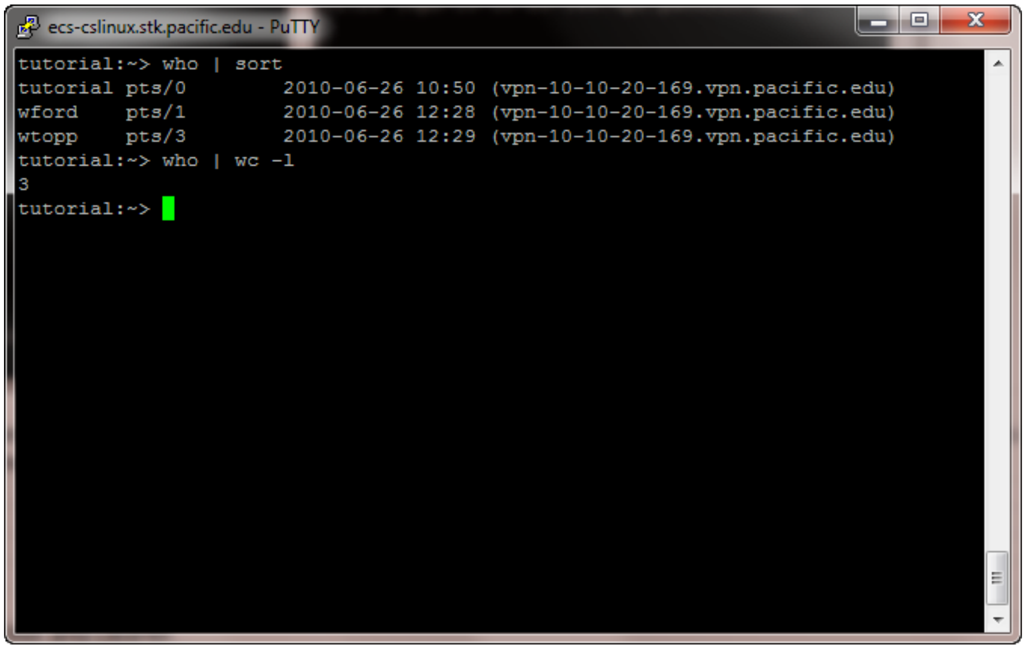


```
tutorial:~> who
tutorial pts/0        2010-06-26 10:50 (vpn-10-10-20-169.vpn.pacific.edu)
wford    pts/1        2010-06-26 12:28 (vpn-10-10-20-169.vpn.pacific.edu)
wtopp    pts/3        2010-06-26 12:29 (vpn-10-10-20-169.vpn.pacific.edu)
tutorial:~> who > names.txt
tutorial:~> sort names.txt
tutorial pts/0        2010-06-26 10:50 (vpn-10-10-20-169.vpn.pacific.edu)
wford    pts/1        2010-06-26 12:28 (vpn-10-10-20-169.vpn.pacific.edu)
wtopp    pts/3        2010-06-26 12:29 (vpn-10-10-20-169.vpn.pacific.edu)
tutorial:~>
```

This is a bit slow and you have to remember to remove the temporary file called names 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: who | sort
This will give the same result as above, but quicker and cleaner.

To find out how many users are logged on, type
who | wc -l



```
tutorial:~> who | sort
tutorial pts/0        2010-06-26 10:50 (vpn-10-10-20-169.vpn.pacific.edu)
wford    pts/1        2010-06-26 12:28 (vpn-10-10-20-169.vpn.pacific.edu)
wtopp    pts/3        2010-06-26 12:29 (vpn-10-10-20-169.vpn.pacific.edu)
tutorial:~> who | wc -l
3
tutorial:~>
```