Manipulating output of a command

If you need help at any time, put your **red** sticky note on the back of your laptop. When you've finished the steps on the *front* of this page, put your **green** sticky note on the back of your laptop.

Commands in this lesson: `head`, `tail`, `less`, `grep`, `wc`, I/O redirection with > and >>, and using |

See more or less

We'll often want to see more or less of a command that has a lot of output.

To see how this works, let's download a large text file - a public domain book. In a browser, visit http://witestlab.poly.edu/books/, pick a book, get the URL of its text version (e.g.: right click and "Copy Link Address" or similar), and use `wget` to download it in the shell.

Rename it to `book.txt` using the `mv` command.

If you run

```
cat book.txt
```

to see the contents of the file, you won't see much - there's just too much output, and it goes by too quickly.

To see the beginning of the book, use

```
head book.txt
```

To see just the end, use

```
tail book.txt
```

You can also specify the number of lines to see with either command, with e.g.

```
head --lines=5 book.txt
tail --lines=10 book.txt
```

To page through one line of output at a time, use

```
less book.txt
```

You can use Enter to continue scrolling through the file, or q to quit at any time.

It's often useful to count the number of lines of output in a file. We can use `wc`:

```
wc -l book.txt
```

Finally, it's nice to be able to see only lines matching a particular pattern. There's a very powerful utility called "grep" that allows us to filter the output to see only those lines that contain a particular word. For example, to see lines containing the word "information", you can run

```
grep "information" book.txt
```

I/O redirection and pipes

The real value of the shell comes in our ability to "chain" together multiple utilities.

For example, suppose we want to count the number of lines in our book that contain the work "information". We can save those lines to a file using the > operator to redirect the output of the `grep` command:

```
grep "information" book.txt > infolines.txt
```

Use `cat infolines.txt` to verify the contents of the new file we've just created. Now, we can run

```
wc -l infolines.txt
```

to see the number of lines.

Alternatively, we can skip writing to a file and just send the output of the `grep` command directly to the `wc` command that follows it with the *pipe* operator, `|`:

```
grep "information" book.txt | wc -l
```

Finally, we may occasionally want to send the output of a command to a file, but append to an existing file rather than create a new one (as > does). To append to an existing file we will use >>.

For example, to create a file that contains all lines with the word "when", followed by all lines with the word "where", you can run

```
grep "when" book.txt" > morelines.txt
grep "where" book.txt >> morelines.txt
```

The second line won't overwrite the text that is written to `morelines.txt` in the first line; it will append to the file instead.