Input/Output Redirection

Most of the UNIX utilities we use have these three channels: **standard input (stdin), standard output (stdout) and standard error (stderr)**. These channels have file descriptors (stdin 0, stdout 1, stderr 2) which represent them. By default, the **standard input** of a command is taken from the keyboard. Then the output is printed out to the terminal. Same goes for the **standard error*. But you can redirect these channels to take **standard input** from a file.

```
$ echo 0< file
```

This will redirect the file contents to the echo command. You don't need to write 0. You can also put the command output in a file instead of the terminal.

```
$ echo "hello" 1> file
```

Again, you don't have to write 1, but I included this file descriptor to be more explicit. You can redirect error messages to the standard error file, with the **2>**. The file will always be **clobbered** if you use just one **>** character. By using two of them **>>** you will append to a file, not overwriting any previous contents.

You can also create **command pipelines** which is just a fancy way of saying that you want to send an output of one command to the input of another.

```
$ ls | wc -l
```

This **pipe |** character will redirect the **ls** listing to the **wc** command. **wc** command will then tell us how many lines are there in the listing.

If you want to redirect output to more files, use the **tee** command.

```
$ echo "hello" | tee file
```

This will print hello to the terminal and also to the file. **tee** can have more arguments. It will send the same output to all of the argument files.
In zsh you can just put more redirections, because of the enabled **multios** feature.

```
$ echo "hello" > file > /dev/tty
```

This line will redirect the word hello to the file and also to the **tty** device file, which is the terminal.
cat command will print out the contents of a file to the terminal.