Piping and redirection

Piping and redirection

It's a pain to have to order for your kids at the drive-thru; sometimes you'd like them to order directly and have the food go directly to them instead of through you. In a linux shell, this is called redirection. It uses a familiar metaphor: "pipes".

The linux operating system expects some "standard input pipe" and gives output back through a "standard output pipe". These are called "stdin" and "stdout" in linux. There's also a special "stderr" for errors; we'll ignore that for now.

Usually, your shell is filling the operating system's stdin with stuff you type - the commands with options. The shell passes responses back from those commands to stdout, which the shell usually dumps to your screen.

The ability to switch stdin and stdout around is one of the key reasons linux has existed for decades and beat out many other operating systems.

The syntax for doing this switching around can be confusing because it uses codes.

Exercise

Redirect stdout of the `ls -1` command to the file `whatsHere`

Redirecting STDOUT

```
ls -1 > whatsHere
cat whatsHere
```

Redirect stdout of `ls -1` to the `head -2` command, then to the file `whatsHere`

Piping one command's output to another, and then redirecting STDOUT to a file

```
ls -1 | head -2 > whatsHere
cat whatsHere
```

So: the redirect output (stdout) character is >, and the "pass output along as input" is the pipe character `|`.

Redirect stdout of `ls -1` to the `head -2` command, then to the file `whatsHere`, then use that file to list the sizes of those two files

Piping, redirection, and command substitution

```
ls -1 | head -2 > whatsHere
cat whatsHere
ls -l `cat whatsHere`
```

(You may have to copy and paste that last command - it uses the backtick character ` to tell the shell to execute the command `cat whatsHere` and then use the result as an option to the `ls -l` command).

Perfect - now my kids (i.e. other commands) can order (stdin) through the same drive-thru window and get their kid-pack (stdout) directly!

Not all shells are equal - the bash shell lets you redirect stdout with either > or `1>`; stderr can be redirected with `2>`; you can redirect both stdout and stderr using `&>`. If these don't work, use google to try to figure it out. The web site stackoverflow is a usually trustworthy and well annotated site for OS and shell help.

Confused about commands vs. programs?

A command is something entered to stdin that the OS understands how to run. That command might be a list of other commands, a command built-in to the linux operating system, or an executable program (sometimes called a binary).

Command is the general term, but, for example, when we run `samtools` we're entering the command `samtools` to run the executable binary program `samtools`.

The operating system must have previously been told where the actual executable file called `samtools` exists in the filesystem, so when a user enters the command `samtools` it knows where to find the executable `samtools` and run it.

Saving even more typing

You can use the `alias` command to set up new names for commands or run commands with a standard set of options. Put these in your `./profile_user` file if you want them available every time that you log in to Lonestar.

Changing default ls behavior

```
alias ls='ls -al'
```

Forgetting an alias

```
unalias ls
```

Conclusion

We've based this course on linux for a reason - it's a very powerful environment for doing all manner of bioinformatics. Linux built-in commands are powerful allies in work with NGS data.

Now let's return to the course outline for the next section.