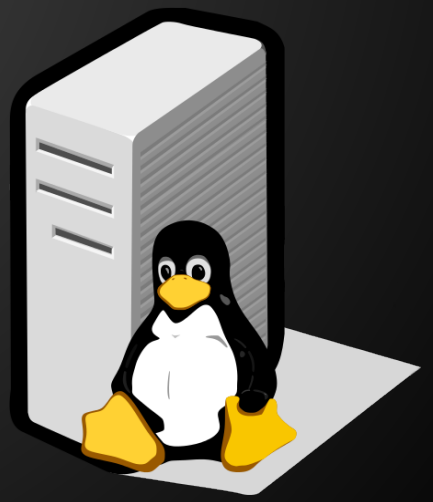
Shell Scripting

BecA Advanced Bioinformatics Workshop

Alan Orth
Nairobi, Kenya
September, 2015



Your first shell script

A shell script is a text file with a list of commands inside. Shell scripts are good for automating tasks you use often, or running “batch” jobs.

Enter the following in a new file, call it `script.sh`:

```
echo "Date and time is:"
date
echo "Your current directory is:"
pwd
```

Your first shell script

Run the script like this:

```
sh script.sh
```

It should output something like this:

```
"Date and time is:"
Mon Aug  18 10:15:00 EAT 2052
"Your current directory is:"
/home/aorth
```

Your second shell script

Create a new script, `script2.sh`:

```
DATE=$(date)
PWD=$(pwd)

echo "Date and time is: $DATE"
echo "Your current directory is: $PWD"
```

Your second shell script

This introduces two new concepts, variables and command substitution.

A variable is a symbolic name for a piece of data, like text, numbers, etc.

Command substitution launches a sub shell to run the named command. It's recommended to use `$(command)` instead of `` `command` ``.

More shell scripts

A more advanced shell script utilizing a loop:

```
for num in 1 2 3
do
   echo "We are on $num…"
done
```

What do you think it does? Can you try to run it? What is a good use case for this?

Sequences

Same thing, but using a “sequence”:

```
for num in {1..3}
do
   echo "We are on $num…"
done
```

This uses functionality built into the command line shell.

More sequences

Same result, but using a command substitution and the `seq` command.

```
for num in $(seq 1 3)
do
   echo "We are on $num…"
done
```

Many ways to achieve the same thing!

“Globbing” (pattern expansion)

Controlled by a list of files from the shell:

```
DATA=/home/aorth/data/sequences

for seq in $DATA/*.fastq.gz
do
   echo "We are on $seq…"
   <do some science!>
done
```

I/O Redirection

By default, command line programs print to *stdout* (“standard out”). I/O redirection manipulates the input/output of Linux programs, allowing you to capture it or send it somewhere else.

Two main kinds of redirection:

>	to a file
|	to another program

I/O Redirection

Redirect the output of your script to a file:

```
sh script.sh > script.out
```

… and to another program, ie `less`:

```
sh script.sh | less
```

Voilà!

Links

Advanced Bash scripting guide:

http://www.tldp.org/LDP/abs/html/

Excellent wiki with common “pitfalls”:

http://mywiki.wooledge.org/BashPitfalls