Loops and Conditionals

C. Ryan Campbell

Duke University

c.ryan.campbell@duke.edu

05 Oct 2017

Overview

Today's Goals

- Fix group git permissions
- Learn loops
- Download multiple datafiles

Cluster repo

- git repo is on the cluster
- There are some permissions issues
- Whenever anyone adds a file, the default is that no one else can edit it
- Problem - updating the git repo fails!
- Solution - `chmod`

Permissions

- What are permissions?
- The users who are given the ability to edit your files
- When I run `ls -l`
- `drwxr-xr-x.  2 cc216 root 58 Sep 28 15:43 software`
- `drwxrwxrwx.  21 cc216 root 561 Sep 29 11:49 490S`
- The 9 letters show:

Permissions

- To edit the permissions you use `chmod`:

```
chmod -R 777 /work/cc216/490s/duke-bio490s
```

- What does this do?
- If you're not sure how would you find out?

Cron Jobs

- It would be hard (and annoying) to rememember to run that command as often as you create files
- So we'll set up a "cron job"
- This is a command that the computer (slogin) will run repeatedly at a set time
- To do so run:

```
export VISUAL=nano; crontab -e
```

- This should open a blank file with nano
- Inside add the following text (paying attention to spaces):

```
01 * * * * chmod -R 777 /work/cc216/490S/duke-bio490s >
/work/cc216/490S/<your netid>/cron.out 2>&1
```

- Finally save (ctrl + O) and exit (ctrl + X)

Variables

- Variables store data
- No data types
- Can be a number, character, or string of characters
- Keep names simple, letters only

```
STR="Hello World"
echo $STR
```

Use the $ to call a variable

So the computer reads this as: `echo Hello World`

Loops

- Variables are integral to loops
- A loop performs a process interatively as a variable changes
- `for` - for all the items in a list, execute the script
- `while` - while a condition is met, execute the script
- `until` - until a condition is met, execute the script
- while and until are opposites

for Loop

```
for i in $( ls ); do
echo item:  $i
done
```

```
for i in $( ls ); do echo item:  $i; done
```

OR

```
for i in ‘seq 1 10’;
do
echo $i
done
```

```
for i in ‘seq 1 10’; do echo $i; done
```

(note that this example isn’t a ', it is an angled quote which shares the tilde key)

while Loop

```
COUNTER=0
while [ $COUNTER -lt 10 ]; do
echo The counter is $COUNTER
let COUNTER=COUNTER+1
done
```

```
COUNTER=0; while [ $COUNTER -lt 10 ]; do echo The counter is
$COUNTER; let COUNTER=COUNTER+1; done
```

"let" allows us to do math with variables, as does $((a + b))

until Loop

- How would we write an “until” loop that does the same thing?

```
COUNTER=0
while [ $COUNTER -lt 10 ]; do
echo The counter is $COUNTER
let COUNTER=COUNTER+1
done
```

Conditionals

- Logical parameters
- Are powerful when automating processes
- `if` - if a condition is met (evaluates T/F)
- `then` - then execute the script
- `else` - until if the condition is not met, execute this script

while Loop

```
if [ "$(ls | wc -l)"-gt 5 ]; then
echo youve got a lot of files in here
else
echo do more work
fi
```

```
if [ "$(ls | wc -l)"-gt 5 ]; then echo youve got a lot of files
in here; else echo do more work; fi
```

Spacing is, as always, important

When in doubt check your variables by echo'ing them

But... How does this help?

- So, how does this help solve the problem of downloading data?
- How can you use these tools to download a set of files?
- What other ways could this make data management easier?

Dotstorming Question

- I've been impressed with everyone's ability to digest class material, complete difficult assignments, and work together with your groups
- I'd like to make sure that the course stays relevant and helpful
- So with that in mind:
- What task/part of your project are you most concerned about your ability to complete?
- Dotstorming Board:
- https://dotstorming.com/b/59d4d3ed03432ec1203b6c4b

The End