more advanced Linux (loops, conditionals, shell scripting, other commands, tips)

Can encapsulate any shell commands into a script...

- The script will be interpreted by the shell...Don't forget to check the commands on the command line directly before writing your script

diff, and useful looping examples

- go to your ~/unix_tutorials and create a directory input_set to put the data files in

```
$ mkdir input_set
$ cp /home/adb/unix_tutorial/input_set/* .
```

randomly generated sets of words from /usr/share/dict/words

```
$diff input01 input02
$diff -Y input01 input02
```

- Try the following command which will list all input* files and echo them back.

```
$for i in `ls input*`; do echo $i; done
```

- very useful for various commands, try replacing `echo $i`, e.g:

```
for i in `ls input*`; do echo ---; echo $i; cat $i | grep treasonableness; done
```

(sometime easier/clearer to split this up into a bash script, see forloop_example2 and replace_example)

- Another approach, good for summary data:

```
cat input* | tr ' ' '\n' | grep -v ^$ | sort | uniq -c
```

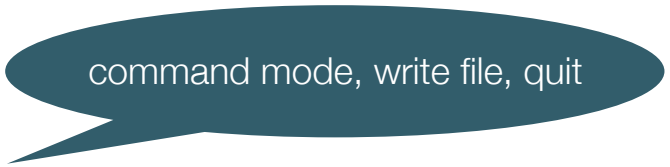
Hello World!

- Create a new file in your favorite text editor:

```
vi test1.sh
```


```
<i>

#!/bin/bash
echo I'm hungry for Cheerios
STR="Hello World!"
echo $STR
pwd
#try out any shell command
```




```
<esc>:wq
```

Note on Quotes:

- You have to be careful with the use of different styles of quotes in your commands or scripts...They have different functions:
  - Double quotes inhibit wildcard replacement (use \ to escape). Double quotes around a string turn the string in to a single command line parameter

```
$ echo "This is a quote \" "
```

  - Single quotes inhibit wildcard replacement, variable substitution and command substitution (special characters don't need to be escaped)

```
$ echo 'This is a quote \" '
```

  - Back quotes cause command substitution

```
$ echo "It is now `date`"
```

Hello World! Execute Permissions

```
$ vi test1.sh
adb@isp02 ~/unix_tutorial 🦄  $ ls -lat
total 64
drwxr-xr-x 5 adb G-25503  4096 Sep 14 10:07 .
drwxr-xr-x 7 adb G-25503  4096 Sep 14 10:07 ..
-rw-r--r-- 1 adb G-25503    30 Sep 14 10:07 test1.sh
adb@isp02 ~/unix_tutorial 🦄  $ ./test1.sh
-bash: ./test1.sh: Permission denied
adb@isp02 ~/unix_tutorial 🦄  $ chmod u+x test1.sh
adb@isp02 ~/unix_tutorial 🦄  $ ./test1.sh
```

Hands on example: A back-up script

```
#!/bin/bash
rsync -avx /Users/adb/Documents/Collaborations/STCFall2017/*.pdf
adb@isp.tacc.utexas.edu:backups
#etc, could set up more backups
```

- (will prompt for password unless you have ssh keys set up)
- -a encapsulates many desirable features (equivalent to -rlptgoD recursive, keep permissions, etc)
- -x extra security to prevent accidental moving of data (e.g. a mounted filesystem, usb stick etc)
- man rsync for all options...

Hello World (with variables)

- Create a new file in your favorite text editor:

```
vi test1.sh
#!/bin/bash
echo I'm hungry for Cheerios
pwd
echo Deliver some Cheerio bits here: pwd
echo Deliver some Cheerio bits here: $(pwd) please
#or
echo hello `pwd` world
```

Conditionals

```
#!/bin/sh
a="pipe"
b="pipette"
if [ $a == $b ]
then
   echo "sometimes a pipe is just a pipe"
else
   echo "the pipe is not a pipe"
fi
```

Loops (examples)

```
#!/bin/sh
for i in $( ls ); do
 echo item: $i
done
```

```
#!/bin/sh
for n in {1..10}
do
    echo $n
done
```

```
#!/bin/sh
for ((n=1; n<=10; n++))
do
 echo $n
done
```

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

- (more to come!)

- Any questions?