CS 136L Lecture 5

Bash scripting

Recap of some syntax

- `$#`, `$@`, `$0`, `$?`
- `read`, `shift`
- `==`, `!=`, `=~` (careful!)
- `&&`, `||`, `!`
- `-d`, `-e`, `-f`, `-r`, `-w`, `-x`

Bash Coding Tips

- Like in C, code in small chunks, compile and test frequently!
- Error messages might not be very useful so debugging small code chunks is important.
- W a t ch a ll w h i t e spa c e s!!!!! (variable names, streams, `if`, `while`, `for`)
- Include the shebang! *#!/bin/bash* - will give you colouring in vi even if file doesn't end in `.sh`
- Remember variables require $ to be accessed. Also $ for embedded commands.
- `x=$((x+1))` to increment variable
- Make script runnable using `chmod a+x ./script_name.sh`
- Debug a script using `bash -x ./script_name.sh`

Debugging Example

This script consumes a single parameter corresponding to a file name checking if it exists and if so it displays words one line at a time but contains several errors. Fix.

```
#!/bin/bash/
if [-e $1] then
  echo "File doesn't exist" > &2
  exit 4
for word in cat $1 do
  echo word
```

Diff

A note about `diff`. When executed, it changes the status code as follows:

- `0` No differences were found.
- `1` Differences were found.
- `>1` An error occurred.

Can gobble output (i.e. don't display difference) by using
`> /dev/null`

Debugging Example 2

This script consumes three parameters and prints exactly `All Same` if all three files are the same and `Not Same` otherwise but contains several errors. Fix.

```
#!/bin/bash
diff $1 $2
is-diff1 = $?
diff $2 $3
ISDIFF = $?
if [ is-diff1 == 0 && ISDIFF == 0]; then
  echo "All Same"
else
  echo Not Same
fi
```

Debugging Example 3

This script is the same as the previous script but consumes an unlimited number of parameters. It contains several errors. Fix.

```
while [ $# -ne 1 ]; do
  diff $1 $2
  if [ $0 -eq 1 ]; then
    echo "Not same"
    exit 1
  fi
  shift
done
echo "All same"
```