CS 246 Fall 2019 — Tutorial 2

September 18, 2019

Summary

1 Bash Variables

In bash, a variable is assigned a value as follows: `var=42`. You do not need to declare a variable before assigning a value.

Note: There cannot be spaces on either side of the equals symbol.

To access the value of a variable, use one of `$var` or `${var}`.

`${var%<end>}` removes the suffix `<end>` from the string stored in var. If `<end>` is not at the end of `var`, the string is unchanged.

In addition to using variables as arguments, we can also treat the value of a variable as a command and run it:

```
greet="echo hello"
$greet
```

Exercise: What is the output of the following? Try to figure it out on paper before using the shell.

```
foo="Hello World!"
echo "Do you have $1?"
```

Exercise: What is the output of the following? Try to figure it out on paper before using the shell.

```
foo="Hello World!"
[ "Hello World!" == $foo ]
echo $?
```

2 Bash Scripting

A bash script is a series of commands saved in a file so that we can accomplish the same task without having to manually type all the commands. Every script must start with the line `#!/bin/bash`.

Exercise: Write a text file `foo` that can be executed, and such that `./foo` prints the contents of `foo`

2.1 Subroutines in Bash Scripts

Format:

```
mySubroutine() {
    ...
}
```

Subroutines may be called by `mySubroutine arg1 arg2 ... argN`.

Exercise: Write a bash script which takes in two arguments, `ext1` and `ext2`. For each file in the current directory which ends with an `.ext1`, rename the file to end with `.ext2`.

2.2 Debugging

Debugging mode can be activated when running a bash script by placing `-x` at the end of the shebang line, or calling it using `bash -x`.

Exercise: Try running one of the scripts you wrote as an exercise in debug mode.

3 Bash Loops and If Statements

For the condition in both if statements and while loops, the result is checked, and if it's `true`, the program will go into the body of the if statement or while loop.

Form of an if statement in bash:

```
if [ <cond1> ]; then
    ...
elif [ <cond2> ]; then
    ...

......
else
    ...
fi
```

Form of a while loop in bash:

```
while [ <cond> ]; do
    ...
done
```

Form of a for loop in bash:

```
for <var> in <words>; do
    ...
done
```

where `<words>` is a list of whitespace separated strings.

Note: `[ <cond> ]` can be replaced by any command and the exit code will be checked.

Exercise: Write a script which, given a pattern and file, prints `found` if file contains the pattern.

3.1 Test Command

`test` is a bash command. The program is better known by its alias `[`, which is called in the form `[ cond ]`. The exit code of `test` is 0 if `cond` is true and 1 if `cond` is false. It may be useful to review the `man` page for `test` (`man [` brings up the same page).

4 Program Exit Codes

In bash, if a program is successful, the exit code is 0. Otherwise, the exit code is non-zero. The exit code is stored in the variable `$?`.

Exercise: Try running `$?` after commands we have used so far. When does `egrep` return 0? When does `wc` return 0?

5 Script Examples

Example: Create a Bash script `mean` that is invoked as follows:

```
./mean filename
```

The argument `filename` is the name of a file containing a list of whitespace-separated numbers. The script should calculate the mean of these numbers, and print the result to standard output.