CPS 5401 Fall 2015
August 27 Class

Shell Programming

Writing a shell script is easy - start up your editor with a file called try.sh. Then type a few commands into this file, one per line. For example

```
#!/bin/bash
hostname
date
ls
```

Then save it as text. You want to make this file executable, so in the terminal window type 'chmod u+x try.sh', adding eXecute permission for the User. Run this script by typing its name (on some systems you may need to type './try.sh' - the '.' stands for the current directory'). You should see the machine's name, the date and a list of files.

Wildcard characters

The * and ? characters have a special meaning to the shell when used where filenames are expected. If you have files called bashful, sneezy and grumpy in your directory and you type

```
ls *y
```

you'll list sneezy and grumpy because * can represent any number of characters (except an initial '.'). ? represents a single character, so

```
ls *u?
```

will print filenames whose penultimate letter is u.

Arguments from the command line

It is easy to write a script that takes arguments (options) from the command line. Type this script into a file called args.sh.

```
#!/bin/bash
echo this $0 command has $# arguments.
echo They are $*
```

The echo command echoes the rest of the line to the screen by default. Within a shell script, $0 denotes the script's name, $1 denotes the first argument given on the command line and so on. Make the script executable then try it out with various arguments. For example, if you type

./args.sh first second third

then

$# will be replaced by 3, the number of arguments,
$0 will be substituted by args.sh,
$* will be substituted by first second third (* having a wildcard meaning)

Constructs

The shell has loop and choice constructs. These are described in the shell's manual page (type man bash on Linux or MacOS). Here are some examples. Type them into a file to see what they do.

```
# Example 1 : While loop. Keep looping while i is less than 10
# The first line creates a variable. Note that to read a
# variable you need to put a '$' before its name
  i=0
  while [ $i -lt 10 ]
    do
      echo i is $i
      let i=$i+1
    done

# Example 2 : While loop
# This script keeps printing the date. 'true' is a command
# that does nothing except return a true value.
# Use ^C (Ctrl-C) to stop it.
  while true
    do
     echo "date is"
     date
    done

# Example 3: For Loop
# Does a letter, word and line count of all the files in
# the current directory.
# The * is expanded to a list of files. The
# variable ‘file’ successively takes the value of
# these filenames. Preceding a variable name by ‘$’
# gives its value.
  for file in *
    do
      echo "wc $file gives"
      wc $file
    done
```

```
# Example 4: If
# like the above, but doesn't try to run wc on directories
  for file in *
    do
      if [ ! -d $file ]  #if $file isn't a directory
      then
        echo "wc $file gives"
        wc $file
      else
        echo "$file is a directory"
      fi
    done
```

Exercises:

1. Suppose you need to change all the filenames in a directory that have suffix .f77 to have the suffix .f90 instead. Write a script called suffix.sh to do this. (Hint: Use the basename command).

2. Write a shell script called lsdirs.sh that lists just the directories in the current directory.