Bash argument parsing

- Most scripts accept a variety of command line arguments
- Convention is to accept them in any order, e.g. -v -i -u, or -u -i -v, or -u -v -i, ... etc
- Some arguments expect a filename or value to follow immediately after, e.g. g++ -o filename
- It would be helpful to come up with a general scheme for processing command line arguments, rather than doing a custom one from scratch for every script

Iterate through args

- Common approach is to iterate through all the command line arguments
- Each time you see a recognized option, e.g. -v, strip it out and apply the appropriate settings within your script
- Each time you see an option with a required parameter, e.g. -o filename, strip them both out and apply the settings
- Anything else in the argument list is something for the main body of the script to handle, store in an array?

Example: handling args like g++

- Suppose we had a script that processed arguments similar to g++, e.g. “`g++ -Wall foo.cpp blah.cpp -o progx`”
- We see the -Wall, set our script variables for error checking appropriately, and remove -Wall from the args list
- We see foo.cpp and blah.cpp, don’t recognize them as specific options, so put them in an array of things to be processed
- We see the -o, grab the filename afterward (progx), set appropriate settings, and remove them from the args list
- What’s in the array are the args for the script to “really” process: foo.cpp and blah.cpp

Use of shift

- The shift command takes the front command line argument (other than the script name) out of the list, e.g.

```
while [ $# -gt 0 ] ; do
    nextarg=$1          # store whatever is in front
    shift               # remove it from the argument list
    echo “${nextarg}” # do something with it
done
```

Use of case for pattern matching

- We'll use case to match each argument against the possible patterns, note that * matches anything and ;; ends processing of an individual case

```
case $arg in
    -v –verbose)
        echo “we found a verbose flag”
        ;;
    -m)
        echo “we found an m flag”
        ;;
    *)
        echo “we found something else”
        ;;
esac
```

Example:

- We'll create a script that accepts the following arguments:
  - -v or –verbose to turn on verbose mode (optional, off by default)
  - -m followed by some integer value to set a max (optional, 100 by default)
  - The name of a source file (required, must come before destination)
  - The name of a destination file (required)
- E.g. some valid runs could look like

```
scriptname -v file1 file2 -m 23
scriptname somefile -m 66 anotherfile
scriptname firstfile secondfile
```

Setup script/variable/defaults

```
#! /bin/bash
# vars to hold source and dest filenames
srcfile=""
destfile=""
# vars for verbose and max settings
verbose=0
max=100
# array for remaining args, count of how many
fixed=()
numargs=0
```

Iterate through command line args

```
while [ $# -gt 0 ] ; do
  key=$1
  shift
  case $key in
    -v|--verbose)
      verbose=1
      ;;
    -m|--max)
      if [ $# -lt 1 ] ; then
        echo "error: -m needs a maxval"
      else
        max=$1
        shift
      fi
      ;;
    *)
      fixed+=("$key")
      ((numargs++))
      ;;
  esac
done
```

Process positional args

```
if [ $numargs -lt 1 ] ; then
    echo "missing arguments srcfile and destfile"
elif [ $numargs -lt 2 ] ; then
    echo "missing argument destfile"
else
    srcfile=${fixed[0]}
    destfile=${fixed[1]}
    echo "processing ${srcfile} and ${destfile}, max is ${max}, verbose is ${verbose}"
    if [ ${numargs} -gt 2 ] ; then
       echo "warning: ignored extra args"
    fi
fi
```