IC221 Lab: Shell Scripting AY22 Spring
100 points total

See solutions at https://github.com/umbertofontana/systems-programming

Learning Objectives

- Setting up a shell script
- Basic shell scripting with variables and control flows
- Storing the output of execution with sub shells
- Loops and iteration
- Mastering your bash environment

Submission
Submit all shell script files to submit site

- `allusers.sh`
- `getname.sh`
- `getsize.sh`
- `getallsizes.sh`
- `isbiggerthan.sh`
- `isbiggerthanall.sh` (optional extra credit)

Test Program
To help you complete the lab, there is a test program that will run some basic checks against your scripts. It is not designed to be comprehensive, but may help you find errors. Run it as follows:
`./test`

Task 1 (15 points)

The file `/etc/passwd` contains all the *login information* (not passwords) for users on the system. Each line looks a little like this:

```
username     groupid             home dir
 |             |                     |
 v             v                     v
door:x:35001:10120:W.T. Door {}:/home/scs/door:/bin/bash
        ^               ^                          ^
        |               |                          |
       uid             name                  default shell
```

Write a script, `allusers.sh`, that will parse the `/etc/passwd` file and print a list of all the common names (not usernames). For example, "`MIDN W T Door`," rather than `mXXXXXX`. (Hint: `man cut`)

Notes: The username should appear in the 5th field. The fields are delimited by colons (:). Some usernames may be blank. This will work on a lab machine or Linux VM, but may not work in WSL.

Task 2 (20 points)

Write a script, `getname.sh` that takes a username as an argument and prints the full name of that user. The full name should be extracted from the `/etc/passwd` file. Here is a sample output:

```
$ ./getname.sh door
Door, W. T. USNA Annapolis
```

Because you want to match precise usernames, you can use something like the following `grep` regular expression in your script: `grep "^USERNAME:"`

Where `USERNAME` is replaced by the username you are searching for as specified in the command line arguments, i.e., `$1`. If the user is not found, your script should print nothing.

This will work on a lab machine or Linux VM, but may not work in WSL.

Task 3 (20 points)

Write a script, `getsize.sh`, which takes a path as an argument and prints out the size of the file/dir at that path. Your script must do error checking and it must print error messages to STDERR. Here is some sample output:

```
$ ./getsize.sh file.txt
4000

$ ./getsize.sh file_does_not_exist
ERROR: File file_does_not_exist does not exist

$ ./getsize.sh file_does_not_exist > /dev/null
ERROR: File file_does_not_exist does not exist

$ ./getsize.sh file_does_not_exist 2> /dev/null
```

You should be able to use `cut`, `ls`, and `wc` to get the information you need. All errors should be written to `stderr` such that the error normally appears in the terminal output, but does *not* appear if `stderr` is redirected to `/dev/null`:

```
$ ./getsize.sh file_does_not_exist > /dev/null
ERROR: File file_does_not_exist does not exist

$ ./getsize.sh file_does_not_exist 2> /dev/null
```

Hint: You may find it useful to use the `tr` command. The option `-s`, in particular, could be useful to get rid of extra whitespace so that your cut fields are more consistent. For example:

```
CMD1 | tr -s ' ' | cut -d ' ' -f X
```

Will reduce two spaces into a single space before sending the data to `cut`, which makes it easier to parse.

Task 4 (20 points)

Create a script called `getallsizes.sh` which takes in any number of files on the command line and prints their sizes. Here's some sample usage:

```
$ ls -l
total 12
-rw-r----- 1 door scs    0 Dec 29 14:56 empty.txt
-rwxr-x--- 1 door scs  277 Dec 29 14:56 getallsizes.sh
-rw-r----- 1 door scs 4000 Dec 29 14:56 larger.txt
-rw-r----- 1 door scs 1847 Dec 29 14:56 medium.txt

$ ./getallsizes.sh empty.txt
empty.txt 0

$ ./getallsizes.sh *.txt
empty.txt 0
larger.txt 4000
medium.txt 1847

$ ./getallsizes.sh empty.txt doesnotexist.txt larger.txt
empty.txt 0
ERROR: File doesnotexist.txt does not exist
larger.txt 4000

$ ./getallsizes.sh empty.txt doesnotexist.txt larger.txt 2> /dev/null
empty.txt 0
larger.txt 4000

$ ./getallsizes.sh empty.txt doesnotexist.txt larger.txt > /dev/null
ERROR: File doesnotexist.txt does not exist
```

Hint: Check out the man page for `echo` to print without a trailing new line using the `-n` option.

Task 5 (25 points)

Write a script, `isbiggerthan.sh`, which takes as arguments a *path* and a *size* and determines if the file or directory is bigger (or equal to) the given size. Usage: `./isbiggerthan.sh size path`

Sample output:

```
$ ./isbiggerthan.sh 10 empty.txt
no
$ ./isbiggerthan.sh 0 empty.txt
yes
$ ./isbiggerthan.sh 10 empty.txt
no
$ ./isbiggerthan.sh 10 medium.txt
yes
$ ./isbiggerthan.sh 2000 medium.txt
no
$ ./isbiggerthan.sh 2000 larger.txt
yes
```

You must implement error checking. All error output should be printed to `stderr`. Use the following format:

```
$ ./isbiggerthan.sh
ERROR: Require path and size

$ ./isbiggerthan.sh num empty.txt
ERROR: Require a number for size

$ ./isbiggerthan.sh -1 empty.txt
ERROR: Require a positive number for size

$ ./isbiggerthan.sh 1 notafile.txt
ERROR: File notafile.txt does not exist
```

Hint: Checking whether a variable is a number or not is not straightforward. You can adapt your solution from the example below:

```
if [ "$var" -eq "$var" ] 2> /dev/null  # check if it's a number
then
    echo "it's a number"
else
    echo "it's *not* a number"
fi
```

Task 6 BONUS (+10 points) (Skipped)

Create a new script called `isbiggerthanall.sh`, which outputs a list of all the files, at the specified paths, that are bigger than the specified file size.

The script must exit with different non-zero status codes according to the following error conditions:

- exit 1 : not enough arguments (`ERROR: Require a size and at least one file`)
- exit 2 : did not receive a number for *size* (`ERROR: Require a number for size`)
- exit 3 : negative number for *size* (`ERROR: Require a positive number for size`)

If a specified file does not exist, that should be reported to *standard error* in precisely the following format: `ERROR: File /my/example/filename does not exist`

Once complete, the script isbiggerthanall.sh should function properly with these arguments:
`isbiggerthanall.sh size path [path [...]]`

The output is a summary list of all the files, at the specified paths, that are bigger than the specified file size. Sample output :

```
$ ls -l
total 16
-rw-r----- 1 door scs    0 Dec 29 15:09 empty.txt
-rwxr-x--- 1 door scs  870 Dec 29 15:14 isbiggerthanall.sh
-rwxr-x--- 1 door scs  748 Dec 29 15:09 isbiggerthan.sh
-rw-r----- 1 door scs 4000 Dec 29 15:09 larger.txt
-rw-r----- 1 door scs 1847 Dec 29 15:09 medium.txt

$ ./isbiggerthanall.sh 0 *.txt
larger.txt
medium.txt

$ ./isbiggerthanall.sh 2000 *.txt
larger.txt

$ ./isbiggerthanall.sh 9999 *.txt

$ ./isbiggerthanall.sh num larger.txt
ERROR: Require a number for size

$ ./isbiggerthanall.sh -1 larger.txt
ERROR: Require a positive number for size

$ ./isbiggerthanall.sh
ERROR: Require a size and at least one file

$ ./isbiggerthanall.sh 1 doesnotexist.txt
ERROR: File doesnotexist.txt does not exist
```