CSE 374 Assessment 1

Linux and Bash tools

Instructions

This assessment is designed to be completed during one 50 minute class period. You may leave early if you complete the assessment early. Turn-in your assessment by handing it to a staff member; you must also show your Husky card for verification.

This assessment is designed to measure your knowledge of the skills taught during the first three weeks of the quarter. You may not use any additional materials: this means that no note sheets, phones, or other sources of information are permitted.

Please write your answers in the answer boxes provided. If you must write an answer outside of that space, please note that in or near the answer box.

This assessment is worth a total of 25 points. Partial credit is available.

Your information

Your Name:	Solution
Your UnetID:	

Linux basics (4 points)

Getting help (2 points): Getting help. You are told that you will need to use the program **`tree`** to implement your homework assignment. List THREE things that you could do to understand **`tree`** and how to use it. If something involves the command line, be specific about exactly what commands you would use.

man tree
tree -- help .
Linux pocket guide
online Linux man pages
google search

File systems (1 point): You execute the command **`pwd`** which produces the following output: **`/data/netid/userid/cse374`**. You then discover that **`tree`** produces a diagram of the files, and when you execute **`tree`** you get the following information:

```
├── hw0.script
├── hw1.script
├── names.txt
├── regex
│   ├── numberslist
│   ├── welcome
│   └── words -> /usr/share/dict/words
└── scripts
    └── shiftdemo
```

Given this information, what would be the absolute path (starting with **`/`**) to the file **`numberslist`**?

/data/netid/userid/cse374/regex/numberslist

Processes (1 point): Name one process that you have run on Calgary.

ps
emacs
grep

Basic shell commands (6 points)

Commands (4 points): For each of the following desired outputs, write the letter corresponding to the correct command in the box. (You may not use all the commands.)

Available commands:

```
a. ps; ls
b. CTRL-X
c. cat FILE
d. mkdir newfolder
e. CTRL-Z
f. rm *.c
g. ls *
h. cd ~
i. chmod g-rw,o-rw FILE
j. ps -u userid
k. cd newfolder
l. ls -l
m. del *.emacs
```

Change your working directory to your home directory	h
Print a file to standard out	c
Move your current process to the background	e
List all files in your current directory with their permissions	l
Change the permission of FILE so that only you can read or write to it	i
List all the processes you are currently running	j
Remove all files with a .c extension from your current directory	f
Create a new subfolder	d

Alias (2 points): You want to create a new `alias` called `where` that prints the `HOSTNAME` to standard out. You also want to add this to your `.bashrc` file. Write one command that *echoes* the alias creation command and then *redirects* that output to append to the `.bashrc` file.

```
echo 'alias where="echo $HOSTNAME"' >>.bashrc
```

Bash Scripting (7 points)

You want to write a shell script that takes as arguments a simple string and a list of file names. The script should write to standard output only the names of the files that contain one or more copies of the given string. Your script should use `grep` to check files to see if they contain the string. For example, if the script is name `listfiles`, then the command `./listfiles stdio foo.c ham.txt hw4.c handout` would produce the output

```
foo.c
hw4.c
handout
```

if these three files are the only ones that contain the string "`stdio`".

The script should work properly if any of the files have names containing embedded spaces. The script should exit with an appropriate error message, sent to stderr, if there is not *at least one argument* (the search string).

Useful grep information: the exit status code returned from grep when it terminates is one of the following. (Remember, you can get the exit code using the variable `$?`)

- 0 – some line in the file was selected
- 1 – no lines were selected
- 2 – some error occurred

You come up with the following script, but some areas still need to be completed and are marked with **TODO**. In the space provided, fill in the correct code for the **TODO**s.

```
1  #!/bin/bash
2
3  if [ TODO ]; then
4    TODO TODO
5    TODO
6  fi
7
8  str=$1
9  TODO
10
11 for file in "$@"; do
12   TODO &> /dev/null
13   if [ TODO ]; then
15     echo "$file"
16   fi
17 done
```

Line 3: a check to see if fewer than 1 arguments was given: `$# -lt 1`

Line 4: an echo of an error message saying you need at least one input argument:
`echo 'error: $0 requires one or more arguments'`
`' $0 [STRING] [FILE....]`

Line 4: a redirect that sends stdout to stderr: `1>&2`
Line 5: a command to exit the script with an error code: `exit 1`
Line 9: a command that shifts all input arguments down one: `shift`
Line 12: a grep command that looks for the string in the file: `grep "$str" "$file"`
Line 13: a check to see if the grep command exited with a 0: `$2 -eq 0`

Regular expressions & tools (5 points)

Grep (2 points): Write a `grep` command that will find the proper nouns in a file. A proper noun is a word that has *exactly one* capital letter followed by *one or more* lower case letters followed by *at least one space*.

```
grep '[A-Z][a-z]\+ '
```

Sed (3 points): Write a sed command that translate a file to pig latin. Specifically, this command will look for words with a *first letter*, followed by *one or more subsequent letters*, followed by *a space*. Your command will output the *subsequent letters* followed by the *first letter* followed by 'ay' followed by *a space*. In other words the phrase '**Sed is so much fun'** becomes **'edSay siay osay uchmay fun'**.

```
sed 's/\([A-Za-z]\)\([A-Za-z]\+\) /
\2\1ay /g'
```

Version control (3 points)

You want to update your remote repository with the code you just completed. Assume this code exists in your current directory in a file called `module.c`. What *three commands* do you execute to complete this task? (Assume you do not encounter any issues that might require additional commands.) Write the commands precisely as you would at the command line.

```
git add module.c
git commit -m 'added module'
git push.
```

Extra (0 points)

Please use this space to provide any notes you have on this assessment – do you have outstanding questions or concerns about this assessment? Do you have any feedback on the course so far?