PA-1: Unix/Linux BASH Programming

Total points: 100 pts

Instructor: Joseph Waclawski

Logistics

1. This assignment must be done individually, not in groups.
2. For this programming assignment (PA-1), you have to download an archive called student-pa1.tar.gz from Blackboard. Upon extraction, you will see a directory named student/, which contains 6 directories named task-1, task-2, and, so on { one of each task.
3. For each task, you have to complete a shell (BASH) script provided in the respective directory. For instance, for Task-1, you have to complete solution-1.sh located in task-1 directory.
4. Unless stated otherwise, you must not modify/edit/rename/move other existing files under the student/ directory.
5. For all the following tasks, you must have a Linux environment set up on your Windows or MAC device. There are instructions on the Blackboard on the Information page.
6. Note that for solving each task you may need several commands, possibly combining them with pipes. Some helpful links are listed in Appendix B.

Obtain the source code for this PA

Obtain student-pa1.tar.gz from blackboard. Unpack the package to find the skeleton code required for this PA. The extracted directory is named student/.

```
$ tar xzf student-pa1.tar.gz
```

Submission Instructions

1. Change directory (cd) to your student/ directory. Now student/ should be your current working directory.
2. Use your netid to create a compressed archive (<netid>-pa1.tar.gz) of your solution directory. For instance, if your netid is johndoe, create a compressed archive as follows:

```
$ cd ..                # moving to the parent directory
$ mv student/ johndoe-pa1/
$  tar czf johndoe-pa1.tar.gz johndoe-pa1/
```

3. Use Blackboard to submit your compressed archive file (e.g., johndoe-pa1.tar.gz).

Grading

You must not rename/move solution-*.sh files under each task-*/ directory.

You must not modify/edit/rename/move other existing files/directories in student/ or in its sub-directories.

You must follow the naming convention for each task as directed, while creating files, directories, archives, and so on.

Your solutions must execute on the Linux machine (lcs-vc-cis486.syr.edu), otherwise your solutions will not be considered for grading.

Task 1 (10 pts)

Complete the BASH script (location: task-1/solution-1.sh) so that it first creates a directory named Assignment-1 in the current folder (i.e., task-1). Inside the Assignment-1 directory, you will then create 10 directories named Query-1, Query-2, : : :, Query-10. Inside each Query-i/ where i is in f1; 2; : : : ; 10g, create a file called response-i.sh.

Example: For instance, inside directory Query-7, your script should create a single file named response-7.sh.

Hint: use mkdir, while/for

Task 2 (10 pts)

Complete the BASH script (location: task-2/solution-2.sh) so that it recursively finds all the files with \.c" extension inside the current directory (i.e., task-2) and archives these files as a tar file named allcfiles.tar. While creating the archive file, your script should exclude all other files and should not maintain the directory structure for these .c files.

Hint: use find, tar, xargs

Task 3 (15 pts)

Complete the BASH script (location: task-3/solution-3.sh) so that it reads the file /proc/cpuinfo, determines the following values (only the numbers), and outputs each of them on a separate line in a file called cpuinfo.txt, while adhering to the same order as shown below.

(First line) The total number of processors
(one line for each processor) Core id of each processor
(one line for each processor) Cache size available to each processor

Example: Consider a machine with 2 processors and \cat /proc/cpuinfo" on this machine produces the following output: (NOTE: \..." represents one or more omitted lines of text; the numbers shown in this example are imaginary)

```
processor       : 0
...
core id         : 0
...
cache size      : 128 KB
...

processor       : 1
...
core id         : 1
...
cache size      : 256 KB
...
```

cpuinfo.txt should contain the values (only the numbers, NOT THE TEXT) as follows when your script is run on a this machine:

```
2
0
1
128
256
```

Explanation of the previous cpuinfo.txt:

```
2     <---- total number of processors
0   <---- core id of processor 0
1   <---- core id of processor 1
128 <---- cache size available to processor 0
256 <---- cache size available to processor 1
```

Hint: use cat, grep, awk, I/O redirection (e.g., >, >>)

Task 4 (15 pts)

Complete the BASH script (location: task-4/solution-4.sh) so that it kills all occurrences of a process called infloop. To test your BASH script, execute the given script named spawner.sh first. It will create 10 copies of the infloop process.

Hint: use ps -ef; grep/awk; kill -9; xargs

> Check the usage of -u option of ps. Use -u option to ensure that your script kills only the infloop processes created by you, not by other users.

Task 5 (25 pts)

Complete the BASH script (location: task-5/solution-5.sh) so that it operates the following:

1. Download a tar file named sample data.tar from an URL that will be provided as a command line argument to your script. (Hint: Inside the script, use \$1" to refer to the URL)
2. Extracts the tar file in the current directory to obtain the directory called sample data/
3. Search through the content of the files inside sample data for string \smart". Print how many times this string appear in each file. The output should be written to a file named result.txt.

   > Output format (one line for each file): printf "%s smart %dnn" $filename $count, assum-ing $filename and $count represent the file name and the number of times the desired string appeared in the file, respectively.

4. Search through the content of the files inside sample data for string \operating system". Print how many times this string appear in each file. The output should be written (technically speaking, appended) to the same file named result.txt.

   > Output format (one line for each file): printf "%s operating system %dnn" $filename $count, assuming $filename and $count represent the file name and the number of times the desired string appeared in the file, respectively.

5. This script should also create two directories named smart and OS in the current directory (i.e., task-5). It will then copy all the files that contain the string \smart" (respectively, \operating system") into the directory named smart/ (respectively, OS/).

To test your script on lcs-vc-cis486.syr.edu, you can use "http://127.0.0.1:55005/sample data.tar" as a command line argument.

Hint: use wget; grep; cat; wc

Task 6 (25 pts)

Complete the BASH script (location: task-6/solution-6.sh) so that it operates as follows:

1. Extract input.tar.gz (a compressed archive) provided in the current directory task-6/. (Hint: use tar with specific options for decompressing a gzipped tarball)
2. From the extracted directory (input/), the script should move the files with .txt extension to a directory named TXT. You need to create the directory TXT first.
3. From input/, the script should move the files with .jpg extension to a directory named JPG. You need to create the directory JPG.
4. From input/, the script should move the rest of the files to a directory named ZIP.
5. The ZIP directory should then be compressed and archived. The name of this compressed archive should be rest zipped.tar.gz.

Hint: Use tar, find, xargs, mv

Appendix A: How to execute a bash script?

There are two approaches to execute a script (say, solution-1.sh).

Approach 1

```
$ cd task-1/
$  chmod +x solution-1.sh
$  ./solution-1.sh
```

Approach 2

```
$ cd task-1/
$ bash solution-1.sh
```

Appendix B: Some Useful Links

BASH Programming - Introduction HOW-TO:
http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html

Bash scripting tutorial: https://ryanstutorials.net/bash-scripting-tutorial/

BASH IO redirection: (a) http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO-3.html,
(b) https://catonmat.net/ftp/bash-redirections-cheat-sheet.pdf

BASH pipes http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO-4.html

BASH printf: https://linuxconfig.org/bash-printf-syntax-basics-with-examples

BASH Conditional Expressions:
https://www.gnu.org/software/bash/manual/html_node/Bash-Conditional-Expressions.html

BASH Error Status:
(a) http://tldp.org/LDP/abs/html/exitcodes.html,
(b) https://support.circleci.com/hc/en-us/articles/360002341673-Identifying-Exit-Codes-and-their-meanings, (c) https://shapeshed.com/unix-exit-codes/

Linux Error Codes: https://www.thegeekstuff.com/2010/10/linux-error-codes/

grep command cheatsheet: https://ryanstutorials.net/linuxtutorial/cheatsheetgrep.php

Linux tutorial { Cheatsheets: https://ryanstutorials.net/linuxtutorial/cheatsheet.php

LinuxCommand.org: http://linuxcommand.org/index.php

BASH commands: https://linuxize.com/tags/bash/

Linux commands: https://linuxize.com/tags/terminal/

SSH commands: https://linuxize.com/post/ssh-command-in-linux/

Linux File System https://www.youtube.com/watch?v=HbgzrKJvDRw