ComS 252 Homework 9: Advanced Scripts: Sed and Awk

Group assignment (with 5% penalty per group member)

Due November 2, 2021

1 Objectives

For this assignment, you will write some "advanced" scripts using `bash`, `sed`, and/or `awk`. It is up to you to test your sripts thoroughly, and write them according to the specifications. The examples are provided as illustrations only, to help clarify how the scripts are expected to run.

2 Download

Download the virtual machine `Hw09.ova`. Accounts are `root` and `user` with passwords `rootpw` and `userpw`, as usual. All scripts should be placed in the `scripts/` subdirectory, in `user`'s home directory. Note that this directory is included in `user`'s `PATH`.

3 `extract.awk`: 6 points

Write an `awk` script named `extract.awk` to extract a portion of the input file(s) as follows.

- The first occurrence of a line containing exactly the text "NEXTLINE" should cause the file's next line (if any) to be extracted.
- A line containing exactly the text "RESET" should reset the nextline count, allowing the next "NEXTLINE" line, if any, to also extract text.

If there are no "NEXTLINE" lines, then the script should display nothing. For example:

```
user# cat test0.txt
Ignore this line.
NEXTLINE
This is the one we want
Not this one.
Nor this one.
user# extract.awk test0.txt
This is the one we want
user# cat test1.txt
RESET
The previous line doesn't really do anything.
More lines to ignore
NEXTLINE
This should be displayed
This should not
NEXTLINE
This is too late so do not display it
RESET
Now we need to look for nextline again.
NEXTLINE
Another line to display.
```

```
But not this one,
or this one either.
Now for a tricky case where we
RESET
NEXTLINE
RESET
NEXTLINE
and print more lines including a magic one
user# extract.awk test1.txt
This should be displayed
Another line to display.
RESET
and print more lines including a magic one
user# cat test2.txt
This file has no magic text
anywhere!
user# extract.awk test2.txt
user#
```

3.1 Grading rubric

- 1.0 points: proper `awk` script
- 1.0 points: only prints lines appearing immediately after “`NEXTLINE`”
- 2.0 points: only prints the *first* “`NEXTLINE`” line
- 1.5 points: “`RESET`” causes next “`NEXTLINE`” to work
- 0.5 points: tricky cases

4 `refactor.sed`: 6 points

Write a `sed` script named `refactor.sed` to modify source code as would be required when a method named “oldmethod” is replaced by a method named “newmethod”. Any method call of the form

```
oldmethod(y,z)
```

in the input stream should be replaced by

```
newmethod(z, 0, y)
```

where y and z may be any string containing spaces, letters, digits, and the following symbols:

```
+ - * / % .
```

All other text is copied without change. For example:

```
user# cat code.cc
#include "method.h"

int foo(int a, int b, object C)
{
  int sum = 0;
  for (int i=0; i<oldmethod(C.size, 7); i++) {
    sum += oldmethod(b*b + i - 3, i % a);
  }
  return sum;
}

user# refactor.sed code.cc
#include "method.h"

int foo(int a, int b)
{
  int sum = 0;
  for (int i=0; i<newmethod( 7, 0, C.size)); i++) {
    sum += newmethod( i % a, 0, b*b + i - 3));
  }
  return sum;
}

user#
```

4.1 Grading rubric

- 1.0 points: proper `sed` script
- 1.5 points: "`oldmethod`" text replaced by "`newmethod`"
- 1.0 points: replacement *only* for method calls with 2 arguments
- 1.5 points: arguments are rearranged
- 1.0 points: all other text copied without change

5 `BuildList.sh`: 8 points

Write a `bash` script named `BuildList.sh` that generates lists by selecting the specified column of a spreadsheet stored in CSV format. The script should be invoked as

```
BuildList <column> <file1> <file2> <file3> ...
```

where "column" is the desired column number, and any remaining arguments are input files. The output should be a single line for each input file, consisting of the filename with directory and extension .csv removed, and a comma–separated list built from the specified column of the spreadsheet. The first line of the spreadsheet is assumed to be column headers, and should be ignored. The script may invoke `sed` and `awk` as needed (in fact this is encouraged, because it will make your solution easier). For example:

```
user# cat cs252.csv
Last,First,username
Aronnax,Miles,rassilon
Britt,Barry,bbritt
Lin,Dean,deanlin
Miner,Andrew,asminer
user# cat cs327.csv
Last,First,userid
Huang,Xiaoqiu,xqhuang
Jia,Yan Bin,jia
Lathrop,Jim,jil
Miner,Andrew,asminer
Reiners,Dirk,dreiners
Sheaffer,Jeremy,sheaffer
```

```
Stolee,Derreck,dstolee
user# BuildList 3 cs252.csv ././././cs327.csv
cs252: rassilon, bbritt, deanlin, asminer
cs327: xqhuang, jia, jil, asminer, dreiners, sheaffer, dstolee
user# BuildList 2 cs327.csv cs252.csv
cs327: Xiaoqiu, Yan Bin, Jim, Andrew, Dirk, Jeremy, Derreck
cs252: Miles, Barry, Dean, Andrew
user#
```

5.1 Grading rubric

- 1.0 points: proper `bash` script
- 2.0 points: output line for each input file
- **Each output line**
  - * 1.0 points: starts with base name of input file
  - * 1.5 points: displays a list of elements, pulled from the specified column
  - * 1.5 points: is formatted correctly (list is comma separated, on one line, with no extra commas)
  - * 1.0 points: **does not** include column header from input file

6 Submitting your work

Run the `Turnin` script to submit your work. The script will collect and submit your scripts; be sure they have the correct names. Your scripts will be hand-graded by a human, so you will not receive a score immediately, except for missing scripts.