ComS 252 Homework 8: Bash/sed/awk scripts

Group assignment (check syllabus for group penalty details)

Due November 7, 2023

1 Objectives

For this assignment, you will write Bash scripts. You may invoke `sed`, `awk`, or other utilities as needed in your scripts. Note that the scripts will need to be hand-graded by a TA.

2 Build a virtual machine

1. Download the ISO file to initialize the virtual machine for homework.
2. In VirtualBox, create a new virtual machine for this assignment, with default settings.
3. Set the ISO file as the optical disk, and boot up the VM.
4. Select "Build Hw08 virtual machine" from the boot menu.
5. After installation completes, remove the homework initialization ISO from the Optical Drive. You are encouraged to take a snapshot of the VM at this point, in case you need to roll back to a fresh install.
6. At first boot, the VM initializes itself by fetching and running a script from the server. This requires Internet access, and VPN access if you are off campus. The script will, among other things, create a user account with your ISU username. All user accounts will initially have passwords that are the account name, followed by "`pw`".
7. When the VM shuts down after initialization, you are again encouraged to take a snapshot of the VM. That way, if you make a mistake and accidentally trash the user files, you can easily roll back to a freshly initialized VM.

3 Script guidelines

All scripts should be placed in the `bin/` directory, in the user account's home directory. As a convenience, this directory is included in the user's `PATH`. Scripts should have exactly the specified filename, and note that only those files will be submitted, so your scripts will need to be "self contained". Students are encouraged to *thoroughly test each script* before submitting.

4 `units.sh`

Write a Bash script named `units.sh`, that takes a single command-line argument, which is a number of bytes. The script should then display the (roughly) equivalent number of bytes using both the base ten, and the binary, SI prefixes. If the number of bytes is less than 1,000, then the script should simply display the number of bytes. If the number of bytes is at least 1,000, then both the base ten and binary outputs should have the format

```
www.fff units
```

where `www` is at most 3 digits and represents the whole portion, `fff` is exactly 3 digits and represents the fractional portion, and `units` are the appropriate units.

4.1 Error handling

An appropriate error message should be displayed if the script is invoked without exactly one command-line argument. If one argument is given, the script may assume that it is a number between 0 and $10^{15} - 1$.

4.2 Examples

The following examples serve to illustrate the expected behavior of the script for various inputs.

```
user$ units.sh
Usage: units.sh #bytes
user$ units.sh 42
 42 bytes
 42 bytes
user$ units.sh 1003
  1.003 kbytes
  0.979 Kibytes
user$ units.sh 65536
 65.536 kbytes
 64.000 Kibytes
user$ units.sh 123456
123.456 kbytes
120.562 Kibytes
user$ units.sh 1234567
  1.234 Mbytes
  1.176 Mibytes
user$ units.sh 1020304050
  1.020 Gbytes
973.037 Mibytes
user$ units.sh 10203040506070
 10.203 Tbytes
  9.279 Tibytes
user$ units.sh 908070605040302
908.070 Tbytes
825.884 Tibytes
user$
```

5 `myls.sh`

Write a Bash script named `myls.sh`, that displays the files and directories passed as arugments, with file sizes in bytes. Specifically, for each argument, the script should do the following.

- If the argument is a directory, display a size of "`/`", followed by the directory name.
- If the argument is a file, display its size in number of bytes, followed by the file name. You may use `ls` to obtain the file size.
- If the argument is neither a file nor a directory, skip it.

If at least one file is displayed, then after processing all arguments, the script should print a separator line, followed by a line with the total number of bytes.

5.1 Output format

Output should be formatted into two columns. The first column is for file sizes (or "`/`"), is right justified, and is exactly 15 spaces wide. The second column is for file or directory names, is left justified, and fills the

remaining terminal width. Your script may assume that the second column will be wide enough to display any directory or file names. The columns are separated by two spaces.

5.2 Examples

```
user$ myls.sh
user$ myls.sh bin
              /  bin
user$ myls.sh foo bar submit.log bin
            831  submit.log
              /  bin
==============  ========================================
            831  Total bytes
user$ myls.sh /boot/*
         243942  /boot/config-5.17.5-300.fc36.x86_64
              /  /boot/efi
              /  /boot/grub2
       61153387  /boot/initramfs-0-rescue-1b32fd1f4d344e5bda0adc263d16b46.img
       18733348  /boot/initramfs-5.17.5-300.fc36.x86_64.img
              /  /boot/loader
             46  /boot/symvers-5.17.5-300.fc38.x86_64.gz
        6235060  /boot/System.map-5.17.5-300.fc36.x86_64
       11802352  /boot/vmlinuz-0-rescue-1b32fd1f4d344e5bda0adc263d16b46
       11802352  /boot/vmlinuz-5.17.5-300.fc36.x86_64
==============  ========================================
      109970487  Total bytes
user$
```

6 addcol.sh

Write a Bash script named `addcol.sh` that reads a spreadsheet from standard input, adds a column, and writes the result to standard output. The input and output format is a single CSV (comma separated value) file. Each line of the file is a row in the spreadsheet, and columns are separated by commas. The first line of the file contains column headers.

The script should take a single command line argument, which is an item name. This will be the name of the column to add to the spreadsheet, and the name of the new column header. The remaining column data should be added as follows. The first column in each row contains a filename, called the *key file*. If the key file exists, and contains a line of the form

```
item=new column value
```

where `item` is the item name given on the command line, then "`new column value`" should be the new column value. Otherwise, the new column value should be the empty string.

6.1 Assumptions

You may assume the following, which may simplify your script.

- The key files will contain only lines of the form

  ```
  identifier=data
  ```

- Identifiers will not be repeated within a key file.

- Data within a key file will not contain commas or equals signs, but spaces and other symbols are allowed.
- Input CSV files will contain commas only to separate the columns; there will not be any quoted strings containing commas.

6.2 Example

```
user$ cat aa.txt
DATE=01/06/2010
Info=Just some other stuff to be ignored
user$ cat bb.txt
cat: bb.txt: No such file or directory
user$ cat cc.txt
Name=cc
what=ancient C compiler
DATE=05/12/1971
author=Dennis Ritchie
user$ cat dd.txt
MANDATE=this is a tricky one!
user$ cat in.csv
Key,Row number,Description,Count
aa.txt,1,,-1
bb.txt,2,Ball bearing,55
cc.txt,3,,1
dd.txt,4,unknown,
user$ addcol.sh DATE < in.csv
Key,Row number,Description,Count,DATE
aa.txt,1,,-1,01/06/2010
bb.txt,2,Ball bearing,55,
cc.txt,3,,1,05/12/1971
dd.txt,4,unknown,,
user$
```

7 Submitting your work

From your user account, run "`sudo Turnin`" to submit your work. Again, this requires Internet access (and VPN access, from off campus), as this will collect and upload your work to the homework server. With a few execptions, your scripts will be graded by a human, through testing and examination.