CSS 390C - Autumn 2020 Midterm Exam

Introduction

This test is worth 20% of your final grade, one point = 1%. There are 30 points available, with any excess to be applied to points lost in the assignments and/or the final exam. It's usually most effective to tackle the easiest questions first, so you might want to skim through the exam before getting started. Weightings may not correspond to difficulty.

Each question describes an *interesting* problem. You may solve it using any combination of tools in your kit, including Bash with Unix utilities, sed, awk, Perl, or Python (if you already know it). Where there are multiple ways to solve a given problem, pick the single approach that you are most comfortable with.

A script (solution) might not need to be longer than a single line, but you do *not* to have to find the shortest, cleverest, elegantest (yeah, I just made up that word) solution to get full marks. Your solution may use multiple script files if necessary. The goal is to solve a (somewhat realistic) problem. Be reasonable. Be clear.

You won't be penalized for missing semicolons and such, so don't worry if you don't remember precise syntax--as long as you're reasonable and clear. For the purposes of the exam, don't bother cleaning up temporary files if your scripts create any.

You may bring in a single sheet of crib notes. Hand in all writings made in class and your crib sheet; do not remove anything. Do not erase anything: neatly cross out anything you don't want graded. Computers are not permitted. Calculators are not required. To determine if the instructions have been read, an extra point will be awarded for underlining your name.

Take a deep breath and don't panic. You have (nominally) 2 hours starting now. Good luck!

1. [6 points]: Find all directories descendant from the current directory and rename them to add a **`.d`** extension. The directory names may contain spaces, but will not contain any funny punctuation characters (like quote or newline). For a one-point penalty, you may assume the directories are no more than 4-deep.

Hint 1: You probably need to rename child directories before their parents.

Hint 2: There are lots of ways to do this, including trivial one-liners if you happen to know the right commands and flags. You don't need to find the simplest solution; find one that is easy to implement. One approach is to generate a list of directory names and turn it into a string processing problem. The output would be the text of a script that would do the actual renaming (using the **`mv`** command) which you pass to the shell.

Hint 3: If one string is the prefix of another, **`sort`** normally puts the shorter string first. You might want to find a way to reverse that.

2. [5 points + 1 bonus]: Identify all files in the current directory that begin with a `#!` line and verify that its “interpreter” exists. You may assume the file names do not contain any newline, quote or colon characters. Bonus point if your solution correctly handles file-names with spaces.

a) [2 points] Report only the names of the missing interpreters.

b) [3 points] Report only the names of the scripts that are missing their interpreter.

Alternatively, you may combine the two subproblems into a single script (whichever is easier for you).

Hint 1: Without flags, the `grep` command simply prints the matching line(s). The `-Hn` flags display the file name and line number of any matches, e.g.

```
$ grep '^#!' proc.pl
#! /usr/bin/perl -w
$ grep -Hn '^#!' proc.pl
proc.pl:1:#! /usr/bin/perl -w
$
```

Hint 2: You can assign the output of a shell command to a shell variable using `var="$(cmd)"` and you can feed the value of a shell variable as standard input to a command using `echo $var | cmd`. You can pipe to a loop or conditional and the built-in `read` command reads a line from standard input, assigning it to a shell variable (or clears the variable and exits failure on end-of-file).

3 [8 points]: A particular analytic program generates error, warning, and info log messages, useful for trouble-shooting. In particular, info messages of the form shown below describe *cookies* (128-bit numbers printed in hexadecimal) associated with zero or more *segments* (uppercase letter, number, underscore, number), where a segment is a set of cookies. Although it is related to online advertising, for solving this problem, you do not have to know anything about cookies or segments other than the definitions given above.

The log may also contain other messages which must be discarded (ignored), but the string **`evaluated:`** uniquely identifies the *interesting* lines.

a) [3 points] Write a script to find all the cookies that belong to both segment **`D08734_72525`** *and* **`D08734_74065`**.

b) [5 points] Using your results from part a (or otherwise), determine if there are any other segments common to that set of cookies.

Example (each log line begins with a timestamp; here the output is line-wrapped by typographical necessity, not a property of the log file):

```
15:53:35.005 [main] INFO com.audiencescience.app.smac.functional.EvaluateProfiles - evaluated:
0000ffaec100ee8e895e141ca0d0aeba ==> []

15:53:35.027 [main] INFO com.audiencescience.app.smac.functional.EvaluateProfiles - evaluated:
01000065d23f2c8e861b1294dff4f0a9 ==> [F09828_0, F09828_15394, F09828_15398]

15:53:35.032 [main] INFO com.audiencescience.app.smac.functional.EvaluateProfiles - evaluated:
010000a7e61b71fc0cf606c40eb11cc1 ==> []

15:53:35.039 [main] INFO com.audiencescience.app.smac.functional.EvaluateProfiles - evaluated:
010000e9b5453aca43030547875adfe9 ==> [F09828_0, F09828_14186, L15029_10004]

15:53:35.044 [main] INFO com.audiencescience.app.smac.functional.EvaluateProfiles - evaluated:
010000fcebc7a03fbc949d4c53433514 ==> [D08734_70047, D08734_70051, D08734_70056, D08734_70077,
D08734_70081, D08734_70085, D08734_70090, D08734_70102, D08734_70657, D08734_70664, D08734_70753,
D08734_71697, D08734_71727, D08734_71733, D08734_71735, D08734_71772, D08734_71944, D08734_72086,
D08734_72107, D08734_72523, D08734_72525, D08734_72603, D08734_72639, D08734_72658, D08734_72683,
D08734_72729, D08734_72734, D08734_72836, D08734_72985, D08734_73269, D08734_73276, D08734_73315,
D08734_73333, D08734_73361, D08734_73548, D08734_73579, D08734_73862, D08734_73865, D08734_73867,
D08734_73868, D08734_73869, D08734_73925, D08734_73947, D08734_74065, D08734_74096, D08734_74097,
D08734_74105, D08734_74112, D08734_74493, D08734_74504, D08734_74640, D08734_74675, D08734_74736,
D08734_74738, D08734_80163, D08734_80164, D08734_80165, D08734_80166, E12334_0, E12334_10036,
E12334_10037, E12334_10039, E12334_10040, E12334_10046, E12334_10047, E12334_10048, E12334_10052,
E12334_10055, E12334_10057, E12334_10064, E12334_10077, E12334_10116, E12334_10164, E12334_10188,
E12334_10249, E12334_10250, E12334_10255, E12334_10258, E12334_10269, E12334_10271, E12334_10273,
E12334_10276, E12334_10280, E12334_10283, E12334_10292, E12334_10298, E12334_10300, E12334_10313,
E12334_10315, E12334_10336, E12334_10361, E12334_10363, E12334_10364, E12334_10365, E12334_10367,
E12334_10371, E12334_10377, E12334_10378, E12334_10379, E12334_10381, E12334_10387, E12334_10397,
E12334_10429, E12334_10430, E12334_10432, E12334_10433, E12334_10434, E12334_10436, E12334_10437,
E12334_10475, E12334_10477, E12334_10478, E12334_10479, E12334_10480, E12334_10483, E12334_10484,
E12334_10494, E12334_10495, E12334_10500, E12334_10502, E12334_10504, E12334_10569, E12334_10578,
E12334_10582, E12334_10586, E12334_10593, E12334_10599, E12334_10669, E12334_10680, E12334_10729,
E12334_10731, E12334_10732, E12334_10738, E12334_10739, E12334_10743, E12334_10747, E12334_10748,
E12334_10757, E12334_10764, E12334_10767, E12334_10768, E12334_10769, E12334_10772, E12334_10799,
E12334_10802, E12334_10815, E12334_10816, E12334_10987]

15:53:35.052 [main] INFO com.audiencescience.app.smac.functional.EvaluateProfiles - evaluated:
0100019fbe69904789124c6f2c559d66 ==> [F09828_0, F09828_14186, F09828_15398, F09828_15399, F09828_15400,
F09828_15401, F09828_15402, F09828_15407, F09828_15408, F09828_15411]
```

4 [10 points]: A friend of yours who is not very computer-savvy tried to compute the sha1sum (a 20-digit hex number) of each of his 500 audio files, and accidentally (we're not sure how) renamed each file to the value of its checksum. Fortunately, the **`exiftool`** utility program displays the embedded metadata of media files.

For each ogg vorbis file (use the **`file`** command because obviously you can't depend a **`.ogg`** extension that no longer exists) in the directory tree under **`/data/music/`**, using the metadata fields **`Artist`**, **`Album`**, **`Track Number`**, and **`Title`**, move (rename) the file to

```
    /media/music/<Artist>/<Album>/<Track Number>_<Title>.ogg
```

(replacing the tokens in angle brackets with their actual value, of course). Make sure each target directory exists, creating the artist and album directories if necessary.

You may assume the metadata does not contain quotes or newline characters, but may contain spaces and/or colons.

Hint: one solution follows this logic:

```
for each file under /data/music
      if file is ogg vorbis then
            construct a mkdir command
            construct a command to verify directory exists
            construct a rename command
send the generated commands to bash
```

This approach is a script that generates a script!

Use information gleaned from the following transcript to help generate your solution:

```
morris@yogi$ pwd
/data/music
morris@yogi$ cd e4b457d7d74e51126fba08cb82b14dc1a8958694/9f3c15313aa1a948a5a05bee307271ee6c9a784a
morris@yogi$ pwd
/data/music/e4b457d7d74e51126fba08cb82b14dc1a8958694/9f3c15313aa1a948a5a05bee307271ee6c9a784a
morris@yogi$ ls -aFC
./                                        77987b89f8bc9868327e58bd96d7dacad9375989
../                                       8eab59354988be5c8d647a6f6338686cf2c8dce3
12a2bb2a33143f0e7099928e8b63988976aa8e4e  b760a56d6b33a9a7b245588d8dc213709ae85fe2
279c60dab073ef14aca320787f0ea45663b1ca8a  c5704c3431426dc85dbbf69766160feb38dba93d
5a4abb9c93a20cc34e44b499a212b8dbb467c3ad  caa8d6bd3b110ebd7c3891a20b975786546ae98d
5d3282de3c8a07f4a4bc726ba1d5cd6074ec3558  e7cb4baf52dc82765cc86a8c902988c48705fc13
morris@yogi$ file 12a2bb2a33143f0e7099928e8b63988976aa8e4e
12a2bb2a33143f0e7099928e8b63988976aa8e4e: Ogg data, Vorbis audio, stereo, 44100 Hz, ~160000 bps,
created by: Xiph.Org libVorbis I
morris@yogi$ exiftool 12a2bb2a33143f0e7099928e8b63988976aa8e4e
ExifTool Version Number         : 10.10
File Name                       : 12a2bb2a33143f0e7099928e8b63988976aa8e4e
Directory                       : .
File Size                       : 3.7 MB
File Modification Date/Time     : 2017:10:20 23:04:57-07:00
File Access Date/Time           : 2017:10:20 23:22:32-07:00
File Inode Change Date/Time     : 2017:10:20 23:21:38-07:00
File Permissions                : rw-r--r--
File Type                       : OGG
File Type Extension             : ogg
MIME Type                       : audio/x-ogg
Vorbis Version                  : 0
Audio Channels                  : 2
Sample Rate                     : 44100
Nominal Bitrate                 : 160 kbps
Vendor                          : Xiph.Org libVorbis I 20070622
Title                           : I Wouldnt Want To Be Like You
Artist                          : The Alan Parsons Project
Track Number                    : 2
Tracktotal                      : 10
Album                           : I Robot
Artistsort                      : The Alan Parsons Project
Discid                          : 7609a30a
Musicbrainz Discid              : JPJCNuiBxdLXXSPWRo.6Y2NGpKs-
Duration                        : 0:03:11 (approx)
morris@yogi$ mkdir "/media/music/The Alan Parsons Project"
mkdir: cannot create directory '/media/music/The Alan Parsons Project': File exists
morris@yogi$ mkdir "/media/music/The Alan Parsons Project/I Robot/”
morris@yogi$ mv 12a2bb2a33143f0e7099928e8b63988976aa8e4e "/media/music/The Alan Parsons Project/I
Robot/02_I Woudnt Want To Be Like You.ogg"
```