Assignment 3: awk

CS3424 - Systems Programming — Sam Silvestro

Document version v1.0

Introduction

For this assignment you will use **awk** to create a program for summarizing and printing information based on the directory listing data of files and information.

You are not to use any other programs, utilities, or scripting languages not covered in class, unless otherwise specifically and explicitly stated in this document.

Your program should take the output from the modified `ls` command line seen below, and process the data in order to output the aggregate information:

```
ls -la --time-style='+%Y/%m/%d %H:%M:%S'
```

In fact, to avoid human error and ensure you are always using the correct command line, I suggest creating and adding a new alias to your `bash` resource configuration file:

```
alias lsa="\ls -la --time-style='+%Y/%m/%d %H:%M:%S'"
```

Note that the inclusion of the leading backslash ensures no other previously-defined/existing `ls` aliases are used; certain other options such as `-h` could cause your script to fail, for example.

Aggregated information requirements

The aggregated information processed from the directory listing data should consist of each of the following, when applicable, in the order specified below (see the input/output example further below for an example of proper output formatting):

- Per-user grouping of file-related counts found in specified directories
  - Username of the entity owning these files
  - Total number of directories found that are owned by this user
  - Total number of files found owned by this user, printing three values:
    - All files
    - Hidden files
    - "Other" files found that are owned by this user
      (these items include, but are not limited to, symbolic links, FIFO's, character or block devices, etc. Basically, anything that is not a regular file nor a directory will fall under this category)
  - Total file storage (in bytes) occupied by this user's *regular* files.

- Itemization of the oldest and newest **regular files** found *(if no regular files exist in the listing, simply report "None" for these items. If only one regular file exists, it is reasonable to report this file as both the oldest and newest.)*
  Also note, if multiple files share the same oldest or newest time stamps, you can break the tie however you wish; there are no guidelines you must adhere to while doing so.
- Total file-related counts found in the specified directories
  - Total users owning files within these paths
  - Total number of files found, printing two values: all files versus hidden files
  - Total number of directories found
  - Total number of "other" files found
    (these items include, but are not limited to, symbolic links, FIFO's, character or block devices, etc. Basically, anything that is not a regular file nor a directory will fall under this category)
  - Total file storage (in bytes) occupied by all regular files listed.

Note: again, **do not** use `sed` , Python, or any other languages or utilities not explicitly allowed by this assignment.

Note 2: ensure to test the processing of `ls` listings for multiple directories, rather than just one. Such listings can be generated by passing more than one directory to `ls` and/or by the simple addition of the `-R` recursive option to the custom `ls` command shown previously. Two examples of such command lines can be seen here:

```
ls -la --time-style='+%Y/%m/%d %H:%M:%S' dir1 dir2 dir3
ls -laR --time-style='+%Y/%m/%d %H:%M:%S' dir1
```

or if you have defined the aforementioned alias, equivalently:

```
lsa dir1 dir2 dir3 file1 dir4
lsa -R dir1 file1 dir2
```

Note that these commands can also include filenames alongside the directory names on the command line as well; this is perfectly permissible and should be accounted for, hence why it was shown in the example above.

Example

The example execution provided below is an excerpt from the following command, executed using the provided example input file:

```
ssilvestro@fox05: /courses/cs/3424/Fall20/assign3$ ./assign3.bash < data/input.txt
```

Alternatively, the script could be executed as follows on any arbitrary directory using the specified `ls` command:

```
ls -la -time-style='+%Y/%m/%d %H:%M:%S'  ~ | ./assign3.bash
```

Example Input Data

```
ssilvestro@fox05: /courses/cs/3424/Fall20/assign3$ head -n 30 data/input.txt
```

```
total 17160
drwxrwxrwt 98 root   root      528384 2020/04/07 13:38:14 .
drwxr-xr-x 26 root   root        4096 2018/09/04 10:50:29 ..
drwx------  2 pmp099 students    4096 2020/03/03 20:57:31 appInsights
-rw-------  1 mce237 students     199 2020/03/01 18:41:59 build4129.log
-rw-------  1 mce237 students     199 2020/03/01 20:18:42 build8335.log
-rw-------  1 mce237 students     199 2020/03/01 20:10:44 build3549.log
-rw-------  1 mce237 students     199 2020/03/01 20:08:55 build4369.log
-rw-------  1 mce237 students     199 2020/03/01 18:18:44 build4943.log
-rw-------  1 mce237 students     199 2020/03/01 20:17:13 build0725.log
-rw-------  1 mce237 students     199 2020/03/01 19:08:39 build5604.log
-rw-------  1 mce237 students     420 2020/03/01 20:08:08 build9771.log
-rw-------  1 mce237 students     199 2020/03/01 20:08:32 build5695.log
-rw-------  1 mce237 students     732 2020/03/01 20:13:35 build6382.log
-rw-------  1 mce237 students     420 2020/03/01 20:07:57 build4429.log
drwxr-xr-x  3 bfn715 students    4096 2020/03/03 23:07:12 dlight_bfn715
drwx------  3 dad980 students    4096 2020/03/05 15:44:15 dlight_dad980
drwx------  3 hrb980 students    4096 2020/04/06 09:54:44 dlight_hrb980
drwx------  3 hrm102 students    4096 2020/04/06 18:43:17 dlight_hrm102
drwx------  3 kaq447 students    4096 2020/02/26 17:58:46 dlight_kaq447
drwx------  3 mce237 students    4096 2020/03/30 00:04:57 dlight_mce237
drwx------  3 mjy610 students    4096 2020/02/27 15:33:54 dlight_mjy610
drwx------  3 pdq039 students    4096 2020/04/06 18:43:48 dlight_pdq039
drwx------  3 xie192 students    4096 2020/03/23 17:47:37 dlight_xie192
drwx------  3 ynb963 students    4096 2020/04/07 13:26:46 dlight_ynb963
-rw-------  1 hrb980 students      95 2020/03/09 16:25:53 exe604592.txt
-rw-------  1 hrb980 students      74 2020/04/03 13:39:09 exe740144.txt
-rw-------  1 hrb980 students    1470 2020/03/09 13:28:36 exe479302.txt
-rw-------  1 hrb980 students    1134 2020/04/06 10:16:23 exe873346.txt
-rw-------  1 mce237 students    1538 2020/03/01 18:17:50 exe431728.txt
...
...
```

Example Output

```
Username: mjy610
        Directories: 3

Username: hrb980
   Files:
               All: 196
            Hidden: 2
        Directories: 3
        Storage (B): 77543 bytes

Username: pdq039
        Directories: 3

Username: zqu051
```

```
   Files:
               All: 452
            Hidden: 0
       Storage (B): 652583 bytes

Username: mce237
   Files:
               All: 52
            Hidden: 1
       Directories: 4
       Storage (B): 2729344 bytes

Username: dad980
   Files:
               All: 4
            Hidden: 1
       Directories: 3
       Storage (B): 6614 bytes

Username: pmp099
       Directories: 2
            Others: 10

Username: ynb963
   Files:
               All: 1
            Hidden: 0
       Directories: 3
       Storage (B): 2894 bytes

Username: xie192
       Directories: 3

Username: kaq447
   Files:
               All: 2
            Hidden: 0
       Directories: 3
       Storage (B): 3092 bytes

Username: bfn715
       Directories: 3
```

```
Username: root
   Files:
                All: 1
             Hidden: 0
        Directories: 5
             Others: 1
        Storage (B): 11 bytes

Username: hrm102
        Directories: 3

Oldest file:
    -r--r--r--  1 root   root          11 2020/02/23 12:11:04 yum.↩
       p1922.lock
Newest file:
    -rw-------  1 hrb980 students    1308 2020/04/07 11:10:08 ↩
       output1682000050179

Total users:           13
Total files:
    (All / Hidden):    ( 708 / 4 )
Total directories:     38
Total others:          11
Storage (B):           3472081 bytes
```

Extra Credit (200% / n)

A 200% bonus will be awarded for those whose script correctly and properly sorts the username-grouped portion of the output based on the total computed storage space for each user (use the "Storage" field for this number), displayed in *ascending* order of their total storage size (i.e. users with the least/no storage consumption will appear first). Break ties alphabetically (e.g. many users will consume zero storage space due to "other" files and directories not contributing to the file count; only regular files contribute).

Once again, n represents the number of students who completed this extra credit portion correctly, in its entirety. No points will be awarded for partial credit; this feature must function properly, as described, in order to be eligible for these extra bonuses.

Extra Credit Output

```
Username: bfn715
        Directories: 3

Username: hrm102
        Directories: 3

Username: mjy610
        Directories: 3

Username: pdq039
        Directories: 3

Username: pmp099
        Directories: 2
             Others: 10

Username: xie192
        Directories: 3

Username: root
   Files:
                All: 1
             Hidden: 0
        Directories: 5
             Others: 1
        Storage (B): 11 bytes

Username: ynb963
   Files:
                All: 1
             Hidden: 0
        Directories: 3
        Storage (B): 2894 bytes

Username: kaq447
   Files:
                All: 2
             Hidden: 0
        Directories: 3
        Storage (B): 3092 bytes
```

```
Username: dad980
   Files:
                All: 4
             Hidden: 1
        Directories: 3
        Storage (B): 6614 bytes

Username: hrb980
   Files:
                All: 196
             Hidden: 2
        Directories: 3
        Storage (B): 77543 bytes

Username: zqu051
   Files:
                All: 452
             Hidden: 0
        Storage (B): 652583 bytes

Username: mce237
   Files:
                All: 52
             Hidden: 1
        Directories: 4
        Storage (B): 2729344 bytes

Oldest file:
    -r--r--r--  1 root   root          11 2020/02/23 12:11:04 yum.↩
       p1922.lock
Newest file:
    -rw-------  1 hrb980 students    1308 2020/04/07 11:10:08 ↩
       output1682000050179

Total users:          13
Total files:
    (All / Hidden):   ( 708 / 4 )
Total directories:    38
Total others:         11
Storage (B):          3472081 bytes
```

Assignment Data

A few sample input files can be found at the following location on the `fox` servers, however it is imperative that you fabricate many of your own examples to ensure that your script functions *according to the specifications outlined above*:
`/usr/local/courses/ssilvestro/cs3424/Fall20/assign3`.

Script Files

Your submission should consist of exactly two files:

- **`assign3.bash`** - a bash script used as the driver program for your `awk` script
- **`assign3.awk`** - the `awk` program used in `assign3.bash`

Script Execution

Your program should each be invoked through a single bash file with input taken from either standard input, or an arbitrary set of filenames specified on the command line, as shown below.
In addition to the above Assignment Data, your program should also work with arbitrary input from the `ls -la -time-style='+%Y/%m/%d %H:%M:%S'` command defined on page 1. This includes both reading from one or more named input files, as well as accepting piped or redirected input directly into standard input, as in these examples:

```
ls -la --time-style='+%Y/%m/%d %H:%M:%S' ~ | ./assign3.bash
```

– or –

```
./assign3.bash listing.txt [listing2.txt [listing3.txt [...]]
```

– or –

```
./assign3.bash < listing.txt
```

Submission

Turn your assignment in via Blackboard. Your zip file, named **`a3-abc123.zip`** with your personal *abc123* should contain only your two `bash` and `awk` files.

If you attempt the extra credit, name your file **`a3-abc123_EC.zip`**. Without the `_EC`, your submission will be graded as normal.