CSE-381: Systems 2

Homework #2: Part B

Due: Tuesday September 18 2018 before 11:59 PM (Midnight)

Email-based help Cutoff: 5:00 PM on Sun, Sept 16 2018

Maximum Points for This Part: 30

Objective

The objective of this part of the homework is to develop 1 C++ program to:

- Understand the use of `uid` and `gid` in Linux (input data files are in the same format as `/etc/passwd` and `/etc/group`)
- Gain familiarity with development and testing of C++ programs
- Understand concepts of stream and file processing.
- Review basics of problem solving
- Understand the use of `std::unordered_map`

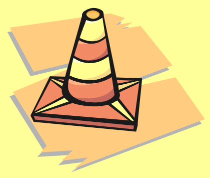
Submission Instructions

This part of the homework assignment must be turned-in electronically via Canvas using the CODE plugin. See video for using the plug-in at: https://youtu.be/P2bWUt5KqbU. Ensure your program compiles without any warnings or style violations. Ensure you have tested operations of your program as indicated. Once you have tested your implementation, upload the following onto Canvas:

- Just the one C++ source file with the naming convention `MUID_hw2.cpp`, where `MUID` is your Miami University unique ID.

General Note: Upload each file associated with homework (or lab exercises) individually to Canvas. Do not upload archive file formats such as zip/tar/gz/7zip/rar etc.

Grading Rubric:



The programs submitted for this homework **must pass necessary base case test(s) in order to qualify for earning any score at all.** Programs that do not meet base case requirements will be assigned zero score!
Program that do not compile, have a method longer than 25 lines, or just some skeleton code will be assigned zero score.

- **Base case points**: 15 points
- **Additional tests**: 10 points
- **Overall code quality, design, code reuse, documentation etc.**: 5 points. These points are typically the hardest to earn in more advanced courses.
- **-1 Points**: for each warning generated by the compiler (warnings are most likely sources of errors in C++ programs)

- **-1 Points**: for each style violation in the programs reported by CSE department's C++ style checker. Ensure you use correct `Miami University C++ Project` setting in `NetBeans`.

Develop program to print group membership

Objective

The objective of this program is to print login-IDs of the users belonging to a given set of `gids` specified as command-line arguments. The necessary data is read from 2 given text files.

Background

In Linux, users are internally represented using a unique number called user ID of `uid`. Moreover, a set of users can be logically organized into a group. Such groups are represented by a group ID or `gid`. Typically, these numbers are seldom used and instead a name is associated with these numbers and the names are often used. This program will serve as an excellent tool to quickly identify membership in a given group.

Data file formats

Prior to solving any problem is important to study the supplied data. So ensure you view the data files (yes, of course you can do this in `NetBeans`). The supplied data files used are nearly in the same format as they are in a real Linux OS as described below. Needless to add, you will need to copy these files to your `NetBeans` project in order to read/use them.

- **User data (`passwd`)**: The supplied `passwd` file contains user information in the following colon (`:`) delimited format:

  ```
  loginID:passkey:uid:…
  ```

  For example the following line from `passwd` "`raodm:xyz:1000:…`" contains the login ID `raodm` as the first entry, `xyz` is some passkey (not used) followed by the `uid` (`int`). Rest of the information in the line is not used.

- **Group information (`groups`)**: The supplied `groups` file contains group information in the following colon (`:`) delimited format:

  ```
  groupID:passkey:gid:members…
  ```

  For example the following line from `groups` "`staff:x123:3:1002,1000`" contains the group ID `staff` as the first entry, `x123` is some passkey (not used) followed by the `gid` (`int`), followed by a comma separated list of `uids`. This results in group corresponding to output "`3 = staff:  lewisjp3(1002) raodm(1000)`", where the `uid` for each member is shown in parentheses.

Sample outputs

One you have completed your program, you can test its operation using the command shows below and compare your output to the output shown below. Note that group IDs are

Base case #1:

```
$ ./raodm_hw2 0
0 = root: root(0)
```

Base case #2:

```
$ ./raodm_hw2 1
1 = bin:
```

Test case #3 [Must pass to earn full points]:

```
$ ./raodm_hw2 100
100 = Group not found.
```

Test case #4 [Must pass to earn full points]:

```
$ ./raodm_hw2 0 1 2 6 100 6 2
0 = root: root(0)
1 = bin:
2 = faculty: raodm(1000) campbest(1001) kiperjd(1003) raychov(1004)
bachmaer(2000) inclezd(1500) davisk4(2001) femianjc(2002) crossv(2010)
castroa(2011) ahmede(2012)
6 = theory: davisk4(2001) inclezd(1500) raychov(1004) femianjc(2002)
100 = Group not found.
6 = theory: davisk4(2001) inclezd(1500) raychov(1004) femianjc(2002)
2 = faculty: raodm(1000) campbest(1001) kiperjd(1003) raychov(1004)
bachmaer(2000) inclezd(1500) davisk4(2001) femianjc(2002) crossv(2010)
castroa(2011) ahmede(2012)
```

Notes/Tips:

- Use `std::istringstream` to process each line. See example in slides on processing Comma Separated Values (in this example, instead of comma you are using colons)
- Use a `unordered_map` to store `uid⇔loginID` information to ease look-up from processing group membership.
- It would be easier to compute and store the line of output to print for each `gid` in another `unordered_map`.

Submit to Canvas

This homework assignment must be turned-in electronically via Canvas via the `CODE Plugin`. Ensure your program compiles without any warnings or style violations and operates correctly, at least for the base case. Once you have tested your implementation, upload just one C++ source file via the `CODE plugin`.