CS 261 – C and Assembly Language Programming

Program 5: Login Analyser
File I/O

1 Motivation

One of the major advantages of binary files is that they make it very easy to store and retrieve structures. This is doubly handy when the structures to be stored/retrieved are, for example, log entries from which we could extract useful information. Unfortunately, while binary files are great for software they're less than human readable, so having a utility to parse them can be helpful.

This program, then, is geared toward extracting useful summary data from a binary log file. It also highlights dynamic memory management and simple data structures, and emphasizes independent exploration of the `man` pages, along with other resources for C and Unix features.

2 Overview

The system utility `last` draws on a log maintained by various components of Unix for its raw data. The log, traditionally located at `\var\log\wtmp`, contains records for system logins, logouts, and reboots, among other things, and is stored in binary format to make the wealth of data it contains easier to process.

While a remote user could, conceivably, log in from anywhere, most users will log in from only a handful of IP addresses. For this project, you'll be creating a program to parse the entire `wtmp` log and extract a summary report listing all of the IP addresses used by each user, along with the number of times they have logged in from that address. This data will then be presented via a query interface, allowing your program's user to request data based on a username.

3 Program Behavior

3.1 Input Format

Your program will read its input data from a copy of the `wtmp` log located in the same directory as your executable. (*Not* the system copy of the log, which may rotate during testing.) The log is a binary file composed of a consecutive list of `utmp` structures, which are defined in the system library `utmp.h`. Consult the manual page for `wtmp` for a description of the structure.

All log entries will be correctly formed, and there will be at least one user login recorded in the log. No other guarantees are made as to the number of entries: there may be an arbitrary number of non-login entries, and you are responsible for ensuring you correctly read all login entries.

Note: The precise definition of the `utmp` structure varies depending on the operating system, and the log in question is not available on Windows or macOS. **You will need to employ a Linux machine for developing the file-reading parts of your program.**

Copying the file `\var\log\wtmp` on any of the lab machines will provide you with a sample input file. A sample input file, which matches the test output shown later, will also be provided via Canvas.

3.2 Core Loop

As noted above, a file named `wtmp` will be located in the same place as your executable. Upon execution, your program will:

1. Open the file and read its entire contents.
2. For each user to successfully log into the system, make a note of the IP addresses from which they logged in, and how many times they have logged in from that address.
3. Sort the IP addresses for each user according to login frequency, with highest frequency first.
4. Enter an interactive mode, repeatedly prompting the user for a username and printing the associated IP addresses and login frequencies until the user enters 'exit'.

3.3 Output Format

Sample output is provided later in this assignment. Your output should be substantially similar, and neatly formatted for a human reader. Importantly, you should expect to be asked about users who are not present in the data, and should generate a suitable message for them rather than falling over.

Autolab will use the following conventions to understand your output:

- For users who are present in the input file:
  - Output is split into a header and the IP log
  - Header and log are separated by a single line whose contents begins with a hyphen
  - Include the name of the user, as entered, in the header
  - You may have any other text you in the header (probably including column headers)
  - The IP log begins immediately after the separator line
  - Print one IP address / count pair per line
  - Use neatly aligned columns that leave some whitespace between address and count
  - Present IP addresses in traditional dotted-octet form (e.g. 127.0.0.1)
  - Use a blank line to indicate the end of the log

- If a user is not present in the input file:
  - Print a reasonable message, including the user's name as entered

4 Code Requirements

You must make use of a data structure of some sort to store your summary data. You may not read through `wtmp` more than once, nor may you read it into an array and then repeatedly parse that. I would recommend the use of a (binary) search tree or linked list (or a combination thereof), though you are welcome to use any structure you can come up with.

You will need to dynamically allocate memory for this assignment, and are expected to free any memory you allocate. Your program is also expected to be free of other memory errors. Aside from the above requirement to use some sort of non-array data structure, you may store relevant information in any form you wish. (Note: Your data structure(s) may contain arrays, but they must be primarily something other than an array.)

If you program contains multiple functions, split them up so that `main()` is in one file and the remaining functions are in one or more additional files. Remember to include .h files and appropriate comments, etc.

You must provide a `Makefile` with your submission. Your `Makefile` must provide a default rule which compiles, but does not run, the program, a `clean` rule which delete .o files and the executable, and a `run` rule which will compile, if necessary, and run your program.

As usual, your code must be well-formatted, reasonably commented, and free of global variables. It must also compile without warnings or errors using the `-std=c99`, `-pedantic`, and `-Wall` flags.

You should zip your files (including, minimally, one .c file and one `Makefile`) and submit them using Autolab. DO NOT submit the `wtmp` file used in testing.

5 Hints

The finished version of this program can be quite short (the reference implementation is around 200 lines, not counting comments), but additional research and thought on your part will be required to complete the program. Here are some general tips to get you headed in the right direction. You don't necessarily have to take advantage of all of them.

- The `man` page for `wtmp` includes a definition of the `utmp` structure. The comments contain useful information.
- Only successful logins will end up with a type value equal to `USER_PROCESS`.
- Host byte order for IP addresses on the lab machines is little-endian.
- The function `inet_ntoa()` is handy.

6 Example

Here is some sample output from the reference implementation, given the sample input:

```
User to look up ('exit' to exit): paul.bonamy

Preferred IP Addresses for paul.bonamy

  IP Address    Count
--------------------
69.166.32.188   12
73.11.99.60     3

User to look up ('exit' to exit): wallaces

Cannot find user wallaces

User to look up ('exit' to exit): exit
```