Programming Assignment #4 **Due: 4/19**

1. [100 points] Write a program *tattle* that examines the records of system usage and prints reports describing who has been using the computer and when they were logged on.

   See the attached man page for detailed specifications.

   Implementation Notes:

   - Information is kept in the files `/var/run/utmp` and `/var/log/wtmp`, but `/var/run/utmp` is redundant and you only need to read `/var/log/wtmp`. Consult *utmp(5)* and your textbook for additional information.
   - While the *getutent(3)* routines can be used to read `/var/log/wtmp`, you are not allowed to use them (although you should `#include <utmp.h>` to use the `struct utmp` definition). You must read `/var/log/wtmp` directly. Remember that it is a "binary" file.
   - The results should be displayed in order of increasing login time.
   - Apply the following rules in traversing the log records:
     - A log on is noted with a `USER_PROCESS` record.
     - A corresponding log off is noted with a `DEAD_PROCESS` record with the same `ut_line` as the log on. (Some systems have a matching `ut_user` on the `DEAD_PROCESS` record, but don't count on it.)
     - Alternatively, a `BOOT_TIME` record effectively logs off all pending log ons.
     - Rarely, there will be a `USER_PROCESS` record on the the same `ut_line` with no intervening `DEAD_PROCESS` or `BOOT_TIME` record. If this should happen (and you'll have to do some searching to detect it), count it as a log off for the previous user on that line as well as a log on (as usual) for the new user.
     - Ignore records that don't pertain to actual users like `reboot` or `LOGIN` or `runlevel`.
   - It's best to do time comparisons with `time_t` (i.e. seconds), since that's what the `*tmp` files use.
   - When the date is specified but the time is not, you'll need to do an overlapping interval test to see if a logon/logoff record overlaps the date's 24 hour period. Here's a trick that makes this easy: Let (t_0, t_1) be the logon/logoff times and let (t_2, t_3) be the times of midnight on the day in question and 24 hours later. Iff $t_1 < t_2$ or $t_3 < t_0$ there is *no* overlap, so if this is not the case there *is* overlap.
   - Use *strptime(3)* and *strftime(3)* to convert strings to and from `struct tm` times. (We've already talked about how to convert `struct tm` times to and from `time_t` times.)
   - For a typical set of login records, run *tattle* on `elec` or, if you can log in to it, `beta`.

- Be sure that the "`-f`" option works with your code. It will be essential for grading.
- The logs "roll over" (get restarted) at the end of the month, so you might want to save a copy of `/var/log/wtmp` from `elec.tricity.wsu.edu` or `beta.tricity.wsu.edu` close but prior to the end of the month and re-use it with the "`-f`" flag for testing. This would also allow you to test your code on your own machine. (Even assuming you're running Linux, the `/var/log/wtmp` on it is probably pretty boring!)
- If you feel a need to demonstrate to the grader that you remember how to implement a sorting algorithm, fine, but you might want to check out *qsort(3)*, which you are free to use.
- Use *last(1)* to confirm your results.

NAME

tattle - list login records for a particular user or at a particular time

SYNOPSIS

`tattle` [`-d` *date*] [`-f` *filename*] [`-t` *time*] [`-u` *login*[`,`*login*]∗]

DESCRIPTION

By default, *tattle* lists all available login records for all users on all dates and times. The `-u` option restricts the report to only those logins (e.g., "`-u archie,veronica,jughead`"). The `-d` option restricts the report to logins active on that date (format: `mm/dd/yy`). The `-t` option restricts the report to logins active at that time (format: `HH:MM`, 24-hour).

The `-f` option takes data from *filename* (default: `/var/log/wtmp`). (This option is mainly for testing purposes.)

If the date (`-d`) is specified but the time (`-t`) is not, all logins active on that date are reported (possibly restricted by `-u`). If the time (`-t`) is specified but the date (`-d`) is not, the date defaults to the current date.

An output login record consists of this information, with a header containing these columns with these names and in this order:

name	contents
`login`	the user's login
`tty`	the line (control tty) the user logged in on
`log on`	the time they logged on
`log off`	one of the following: · the time they logged off · the time the system rebooted · the string "`(still logged in)`" (*exactly* that)
`from host`	the host they logged in from

All output times are formatted "`mm/dd/yy HH:MM`" with a 24-hour time.

Output is sorted chronologically by login time in increasing order.

ERRORS

These errors, all of which are fatal, are noted on standard error:

- bad date or time
- too many command line arguments
- unknown login (use *getpwnam(3)* to check)