CPSC 351 - Operating Systems Concepts

Fall 2014

Programming Assignment 3, due November 10

In this assignment, you will use threads to address a common problem in UNIX system administration: watching multiple log files for interesting events.

Interface

Your program should take an interval in seconds and the names of up to five files as command-line arguments. For example:

```
$ ./logmon 5 foo.log bar.log
```

Watch the specified files for changes. When a new line is added to a file, save it and check to see whether it contains the strings `ERROR` or `WARNING`, keeping track of the number of lines in which they appear. Every time the specified time interval has elapsed, report the following:

- The current date and time
- The number of errors and warnings in each file
- The number of errors and warnings across all files
- The last five lines written to any file

For example:

```
foo.log: 1 error, 5 warnings
bar.log: 3 errors, 1 warning
Total: 4 error, 6 warnings

Recent activity:
foo.log: WARNING The volume "Macintosh HD" is almost full.
foo.log: INFO Reloading configuration files.
bar.log: [ERROR] While organizing orchestra, expected cowbell but found drums.
foo.log: WARNING Too many failed logins from user “Mouse”. Possible brute force attack?
bar.log: [NOTICE] Startup items are deprecated. Use a launchd job instead.
```

Implementation

In addition to your `main()` program, use the Pthreads API to create a new thread for each file and a thread to produce progress reports. You will likely need to use Pthreads mutex locks, POSIX SEM semaphores, or another process synchronization construct to avoid race conditions between the threads.

Testing

You can use genlogs.c to produce log messages at various intervals. For example:

```
$ ./genlogs > foo.log &
$ ./genlogs 5 > bar.log &
```

With the two commands given above, a new entry will be written to `foo.log` every second and a new entry will be written to `bar.log` every 5 seconds.

You may use the virtual machine that accompanies the textbook, but it is not a requirement. Unlike the first assignment, this code runs in user mode and should work on any version of Linux (or any other UNIX-like system).

Grading

10 points total; one point for each of the following:

1. Assignment submitted
2. Threads created for each file
3. File changes are detected
4. Thread created for report generator
5. Reports generated at specified interval
6. Counts for each file are correct
7. Total counts are correct
8. Last five lines reported correctly
9. Process synchronization constructs used appropriately
10. No race conditions detected