PS6: Kronos Time Clock: Introduction to Regular Expression Parsing

We will:

- Analyze of the Kronos InTouch time clock log by using regular expressions to parse the file
- Verify device boot up timing

Overview

The Kronos InTouch device is a Linux based touch-screen time clock that integrates with different host based systems. When problems occur at with the device, Kronos Service and Engineering needs to gather information on what was occurring on the device at the time of the failure. One of the resources for gathering this information is the InTouch log(s), located on the device itself. Service will retrieve these logs, from the device for review. These log file(s) contain specific information about the operation of the device.

Device Startup Diagnostics

Scan the complete log file and create a text file report chronologically describing the each time the device was restarted.

Device Boot up Timing

(a) When an InTouch device boots up, the first logging message that will be seen is:

```
(log.c.166) server started
```

(b) The completion of the boot up sequence can be seen via the log entry for the device SoftKeys being displayed on the UI:

```
oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:9080
```

Use the timestamps from these two matching log entries to determine the elapsed time for each startup sequence. Take into account that these must be paired, with no

nesting to have proper startup. Any incorrect pairings should be reported as startup failures with line numbers and time stamps.

Your job is to read in an entire InTouch log and report:

- each startup
- whether it completed
  - if so, how much time it required
- whether it failed

Your output should begin with the line number of the startup message "`(log.c.166) server started`", the timestamp of the startup, and either: `success` followed by elapsed time, or `failure`.

- Success is determined by a line containing the string "`oejs.AbstractConnector:Started SelectChannelConnector`" that follows the startup message.
- Failure is determined by another startup message before the success message, or end of file.

Each of these output reports should be on a single line.
You can find the sample code to get you started in stdin_boost.cpp .

The code will read a regex from stdin, and then let you type input strings which are matched against the regex.

Implementation

We will use the Boost regex library:

http://www.boost.org/doc/libs/1_58_0/libs/regex/doc/html/index.html

- To install the Boost regex library on your own machine, run this command:
```
sudo apt-get install libboost-regex-dev
```
- To use the Boost regex library in your source code, use the following:
  - header include directive `#include <boost/regex.hpp>`
- To link to the Boost regex library, use this compiler flag: `-lboost_regex`
- To install the boost date/time library, run the command:
```
sudo apt-get install libboost-date-time-dev
```
- Kronos log files and sample output may be downloaded in kronos.zip
  - Sample output is in the file `device5_intouch.log_BOOT.rpt` in the zip download.

Here is info about the `regex_match` API and simple working sample code:
http://www.cplusplus.com/reference/regex/regex_match/
http://www.cplusplus.com/reference/regex/ECMAScript/

Boost defaults to the Perl syntax for regular expressions. You can find documentations here:
http://www.boost.org/doc/libs/1_57_0/libs/regex/doc/html/boost_regex/syntax/perl_syntax.html

You should use the Boost time and date functions for figuring out elapsed time between server boot and boot completion.

See:

- Main docs: http://www.boost.org/doc/libs/1_57_0/doc/html/date_time.html
- Time examples: http://www.boost.org/doc/libs/1_57_0/doc/html/date_time/examples.html#date_time.examples.time_math
- Using the Gregorian features: http://www.boost.org/doc/libs/1_57_0/doc/html/date_time/gregorian.html#date_time.gregorian.date_class

Submitting

Your executable file must be named ps6. It should accept a log file as the argument and create an output file with the same name but with a *suffix* .rpt. E.g. to run it, you might use at the shell:

```
./ps6 device1_intouch.log
```

and your code should would produce a file named `device1_intouch.log.rpt`.

Submit the following:

- Your source code `.cpp` file and any header file(s).
- A Makefile for building the code.
- Output from running your code on each of the five InTouch log files. Your output files must be named `device[1-5]_intouch.log.rpt`.
- A filled-out of the `ps6-readme.txt` file.

Submit on Blackboard.

Grading rubric

Feature	Value	Comment
core implementation	4	full & correct implementation = 4 pts; nearly complete = 3 pts; part way = 2 pts; started = 1 pt
use of Boost time methods	1	
output files included	2	full & correct = 2 pts; partial = 1 pt; absent = 0 pt
cpplint	1	Your source files pass the style checks implemented in cpplint
Makefile	1	Makefile included
Readme	1	0.5 pts for describing regexs in readme; 0.5 pts for discussion
Total	10	
extra	1	Lambda expression