PS7: Kronos Time Clock

For this project, we will:

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

1 Overview

The Kronos InTouch dwevice is a Linux-based touch-screen time clock that integrates with different host-based systems. When problems occur with the device, Kronos Service and Engineering needs to gather information on what was occuring 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.

2 Device Startup Diagnostics

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

2.1 Device Boot Up Timing

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

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

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 a 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 and if so, how much time it required
- Whether it failed

Your output should begin with the line number of the startup message "server started", the timestamp of the startup, followed by "Boot Start", followed by either a similar message and the boot time for a successful completion or an incomplete boot message on a 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 by the end of the file.

Your output **must** follow the example in the provided sample reports.

You can find 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.

3 Implementation

We will use the regex library. This originated with Boost, but since C++11 is now part of the standard: https://en.cppreference.com/w/cpp/regex.

- To use the regex library in your source code, use the following header directive

```
#include <regex>
```

- You should have installed the Boost date/time libary during PS0. If you didn't you can do that by using

```
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_BOOTONLY.rpt

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/
- p://www.boost.org/doc/libs/1_57_0/libs/regex/doc/html/boost_regex/syntax/perl_syntax.html

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

- 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 Gregorian features: http://www.boost.org/doc/libs/1_57_0/doc/html/date_time/gregorian.html#date_time.gregorian.date_class

4 Extra Credit

For extra credit, output a summary header, as in the device5_intouch.log_BOOT.rpt file. You can track the start and completion time of each service for additional credit.

5 Python Option

For this assignment, you may optionally choose to write your code in Python rather than C++. In this case, you should submit a Python script named ps7.py (you may optionally include other Python scripts). Your program will still receive the input filename as a command-line argument, which you can access as sys.argv (sys is a module for interacting with the Python interpreter). The official Python tutorials are available at https://docs.python.org/3/tutorial/index.html.

You will use the Python `re` module to perform regular expression matching. The functionality and pattern-matching are similar to that provided by the `<regex>` library, although the functions are a little different. In general, you will compile a regular expression pattern into an object and then invoke methods on that object rather than passing it into a static function like in C++. You can use raw strings to avoid the backslash plague. For example, `r'\n'` is a string with two characters, `'\'` and `'\n'`, rather than a newline character.

You can use the `datetime` module to parse and process dates and timestamps. The syntax is very different from that of the Boost date-time library although the end functionality is similar.

Do not submit a `Makefile`. Gradescope will provide a `Makefile` that will create a simple `ps7` program that runs your `ps7.py` Python script.

6 What to turn in

Your makefile should build a program named `ps7`. It should accept a log file as the argument and create an output file with the same name but with a *suffix* `.rpt`. To run it, you might use at the shell

```
./ps7 device1_intouch.log
```

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

Submit a zip archive to Gradescope containing:

- Your source code `.cpp` file and any other source or header file(s)
- Output from running your code on each of the six InTouch log files. Your output files must be named `device[1-6]_intouch.log.rpt`
- The makefile for your project. The makefile should have targets `all`, `ps7`, `lint` and `clean`. Make sure that all prerequisites are correct.
- Your `Readme-ps7.md` that includes
    1. Your name
    2. Statement of functionality of your program (e.g. fully works, partial functionality, extra credit)
    3. Key features or algorithms used
    4. Any other notes
- Any other source files that you created.

Make sure that all of your files are in a directory named `ps7` before archiving it and that there are no `.o` or other compiled files in it.

7 Grading rubric

Feature	Points	Comment
Autograder	26	Full & Correct Implementation
	4	Contains all code files and builds
	2	Produces output files in the correct locations
	8	Finds the start times
	8	Finds the end times
	4	Finds the durations
Manual Grading	9	
	3	Uses the regex and Boost Date-Time libraries
	6	Data is in the correct format
Readme	5	Complete
	2	Describes something interesting about how the project was implemented.
	3	Describes the regexes used.
Extra Credit	5	
	+2	Has summary header
	+3	Logs the boot process of each service (+1 for process names only)
Penalties		
	-5	Memory leaks or issues
	-5	Linting problems
	-3	Non-private fields
	-10%	Each day late
Total	40	