

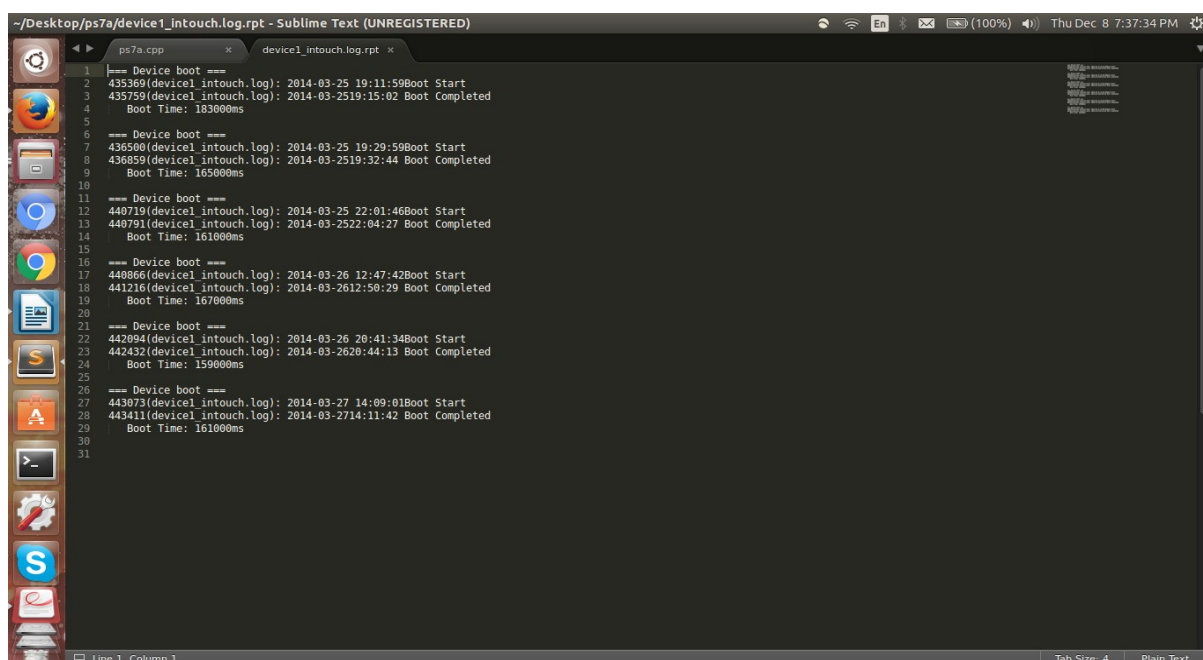
## PS7: Kronos Intouch Parsing

### PS7a:

The Kronos Intouch was a interesting assignment to work on. The objective of this assignment was to implement a scan that can go through the inTouch log files finding specific lines and outputting them into a output file. To accomplish this, we needed to learn about regex commands. The assignment wanted us to find a specific line that started with *(log.c.166) server started* and to find the completion of the boot. We needed to find the elapsed time in between the start time and end time.

One library that was needed to accomplish this assignment was the boost regex software. This software was to have the ability to use functional regex commands and use functions that could recognize them. To start, I to use the *using* function to shorten the deceleration type it make it easier. I created regex commands for the start line and end line. I went through each line and checked each time. If the start exits but the end doesn't, it will give a incomplete error. After it completes the loop, I would subtract the end by the start to get the total elapsed time.

After accomplishing this assignment, I learned how to use regex commands and how to use they can be useful in the future commands. Formating the output file to look specific a specific output was a great skill when getting a job. Also, knowing the what functions are from the boost regex library was also a plus.



```
--/Desktop/ps7a/device1_intouch.log.rpt - Sublime Text (UNREGISTERED)
1  === Device boot ===
2  435369(device1_intouch.log): 2014-03-25 19:11:59Boot Start
3  435759(device1_intouch.log): 2014-03-2519:15:02 Boot Completed
4  Boot Time: 183000ms
5
6  === Device boot ===
7  436500(device1_intouch.log): 2014-03-25 19:29:59Boot Start
8  436859(device1_intouch.log): 2014-03-2519:32:44 Boot Completed
9  Boot Time: 165000ms
10
11 === Device boot ===
12 440719(device1_intouch.log): 2014-03-25 22:01:46Boot Start
13 440791(device1_intouch.log): 2014-03-2522:04:27 Boot Completed
14 Boot Time: 161000ms
15
16 === Device boot ===
17 440866(device1_intouch.log): 2014-03-26 12:47:42Boot Start
18 441216(device1_intouch.log): 2014-03-2612:50:29 Boot Completed
19 Boot Time: 167000ms
20
21 === Device boot ===
22 442094(device1_intouch.log): 2014-03-26 20:41:34Boot Start
23 442432(device1_intouch.log): 2014-03-2620:44:13 Boot Completed
24 Boot Time: 159000ms
25
26 === Device boot ===
27 443073(device1_intouch.log): 2014-03-27 14:09:01Boot Start
28 443411(device1_intouch.log): 2014-03-2714:11:42 Boot Completed
29 Boot Time: 161000ms
30
31
```

```
1: CC = g++
2: CFLAGS = -Wall -Werror -ansi -pedantic -std=c++0x
3: BOOST = -lboost_regex
4:
5: all: ps7a
6:
7: ps7a: ps7a.o
8:      $(CC) ps7a.o -o ps7a $(BOOST)
9:
10: ps7a.o:
11:      $(CC) -c ps7a.cpp $(CFLAGS)
12:
13: clean:
14:      rm -f *.o *~ ps7a
```

```
1: // <Copyright Owners Albara Mehene & Sean Nishi>
2: // regex_match example
3: #include <boost/regex.hpp>
4: #include <boost/date_time/gregorian/gregorian.hpp>
5: #include <boost/date_time/posix_time/posix_time.hpp>
6: #include <iostream>
7: #include <string>
8: #include <fstream>
9:
10: using boost::gregorian::date;
11: using boost::gregorian::years;
12: using boost::gregorian::months;
13: using boost::gregorian::days;
14: using boost::gregorian::date_duration;
15: using boost::gregorian::date_period;
16: using boost::gregorian::from_simple_string;
17:
18: using boost::posix_time::ptime;
19: using boost::posix_time::hours;
20: using boost::posix_time::minutes;
21: using boost::posix_time::seconds;
22: using boost::posix_time::time_duration;
23:
24:
25: int main(int argc, char* argv[]) {
26:     if (argc != 2) {
27:         std::cout << "ERROR: input only one file" << std::endl;
28:         return -1;
29:     }
30:
31:     // open the input file
32:     std::ifstream logFile;
33:     logFile.open(argv[1]);
34:     // name of file
35:     std::string logName(argv[1]);
36:     std::string outputName = logName + ".rpt";
37:     // create the output file
38:     std::ofstream outputFile;
39:     outputFile.open(outputName.c_str());
40:     // space
41:     std::string line;
42:     date stored_date;
43:     date finished_date;
44:     ptime beginTime;
45:     ptime endTime;
46:     boost::smatch m;
47:     time_duration total_time;
48:     // space
49:     bool s_boot = false;
50:     int lineNum = 1;
51:
52:     // Start of boot: 2014-02-01 14:02:32: (log.c.166) server started
53:     boost::regex Boot_Start(
54:         "([0-9]{4})-([0-9]{2})-([0-9]{2}) "
55:         "([0-9]{2}):([0-9]{2}):([0-9]{2}): "
56:         "\\(log.c.166\\) server started.*");
57:     // If we find the text:
58:     // "2014-01-26 09:58:04.362:INFO:oejs.AbstractConnector:Started
59:     // SelectChannelConnector@0.0.0.0:9080"
60:     boost::regex Boot_End(
61:         "([0-9]{4})-([0-9]{2})-([0-9]{2}) "
```

```
62:      "([0-9]{2}):([0-9]{2}):([0-9]{2}).([0-9]{3}):INFO:"
63:      "oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:9080.*");
64:  // check if input file is open
65:  if (!logFile.is_open()) {
66:      std::cout << "ERROR: no input log file" << std::endl;
67:      return -1;
68:  } else {
69:      // Go through the loop
70:      while (getline(logFile, line)) {
71:          // search the start regex code
72:          if (regex_search(line, m, Boot_Start)) {
73:              // store them into date and time
74:              stored_date = date(stoi(m[1]), stoi(m[2]), stoi(m[3]));
75:              beginTime = ptime(stored_date,
76:                               time_duration(stoi(m[4]), stoi(m[5]), stoi(m[6])));
77:              // condition to see if it will fail to go to the next condition
78:              if (s_boot) {
79:                  s_boot = false;
80:                  outputFile << "**** Incomplete boot ****\n" << std::endl;
81:              }
82:              // draw the start into the output file
83:              outputFile << "=== Device boot ===\n"
84:                          << lineNumber << "(" << logName << "): "
85:                          << m[1] << "-" << m[2] << "-" << m[3]
86:                          << " "
87:                          << m[4] << ":" << m[5] << ":" << m[6]
88:                          << "Boot Start" << std::endl;
89:              s_boot = true;
90:              // Then do the same to the end but checking the regex
91:          } else if (regex_search(line, m, Boot_End)) {
92:              finished_date = date(stoi(m[1]), stoi(m[2]), stoi(m[3]));
93:              endTime = ptime(stored_date,
94:                              time_duration(stoi(m[4]), stoi(m[5]), stoi(m[6])));
95:              // output the rest
96:              outputFile << lineNumber << "(" << logName << "): "
97:                          << m[1] << "-" << m[2] << "-" << m[3]
98:                          << m[4] << ":" << m[5] << ":" << m[6]
99:                          << " Boot Completed" << std::endl;
100:              total_time = endTime - beginTime;
101:              outputFile << " "
102:                          << "Boot Time: " << total_time.total_milliseconds()
103:                          << "ms\n" << std::endl;
104:              s_boot = false;
105:          }
106:          // increment line number to go to next line
107:          lineNumber++;
108:      }
109:      // closing the files
110:      logFile.close();
111:      outputFile.close();
112:  }
113:  return 0;
114: }
```

### PS7b & Psx:

This part of the assignment, we now needed to print out the services in the program to the output. Any time a service, has the line *Starting Service. Logging 1.0* , the program will save that line and print them out into the output file. The program needs to print out the previous problem first then would print out the services after. In addition to accomplishing this assignment, finding upgrades and downgrades to software in logs need to be included.

Key concepts to this assignment is like the previous problem but now to print out additional conditions. Learning how to go back to the start of the boot and finding all lines with services and upgrades/downgrades.

I did not learn so much from this assignment. Sadly I could not finish this program and was not able to takeaway anything. Because of the previous part, I was able to write down the correct regex commands. Though I could not figure out how to go back to the starting line and print out the services.

```
1: CC = g++
2: CFLAGS = -Wall -Werror -ansi -pedantic -std=c++0x
3: BOOST = -lboost_regex
4:
5: all: ps7b
6:
7: ps7b: ps7b.o
8:      $(CC) ps7b.o -o ps7b $(BOOST)
9:
10: ps7b.o:
11:      $(CC) -c ps7b.cpp $(CFLAGS)
12:
13: clean:
14:      rm -f *.o *~ ps7b
```

```
1: // <Copyright Owners Albara Mehene & Sean Nishi>
2: // regex_match example
3: #include <boost/regex.hpp>
4: #include <boost/date_time/gregorian/gregorian.hpp>
5: #include <boost/date_time/posix_time/posix_time.hpp>
6: #include <iostream>
7: #include <string>
8: #include <fstream>
9:
10: using boost::gregorian::date;
11: using boost::gregorian::years;
12: using boost::gregorian::months;
13: using boost::gregorian::days;
14: using boost::gregorian::date_duration;
15: using boost::gregorian::date_period;
16: using boost::gregorian::from_simple_string;
17:
18: using boost::posix_time::ptime;
19: using boost::posix_time::hours;
20: using boost::posix_time::minutes;
21: using boost::posix_time::seconds;
22: using boost::posix_time::time_duration;
23:
24:
25: int main(int argc, char* argv[]) {
26:     if (argc != 2) {
27:         std::cout << "ERROR: input only one file" << std::endl;
28:         return -1;
29:     }
30:
31:     // open the input file
32:     std::ifstream logFile;
33:     logFile.open(argv[1]);
34:     // name of file
35:     std::string logName(argv[1]);
36:     std::string outputName = logName + ".rpt";
37:     // create the output file
38:     std::ofstream outputFile;
39:     outputFile.open(outputName.c_str());
40:     // space
41:     std::string line;
42:     date stored_date;
43:     date finished_date;
44:     ptime beginTime;
45:     ptime endTime;
46:     boost::smatch m;
47:     time_duration total_time;
48:     // space
49:     bool s_boot = false;
50:     int lineNum = 1;
51:
52:     // Start of boot: 2014-02-01 14:02:32: (log.c.166) server started
53:     boost::regex Boot_Start(
54:         "([0-9]{4})-([0-9]{2})-([0-9]{2}) "
55:         "([0-9]{2}):([0-9]{2}):([0-9]{2}): "
56:         "\\(log.c.166\\) server started.*");
57:     // If we find the text:
58:     // "2014-01-26 09:58:04.362:INFO:oejs.AbstractConnector:Started
59:     // SelectChannelConnector@0.0.0.0:9080"
60:     boost::regex Boot_End(
61:         "([0-9]{4})-([0-9]{2})-([0-9]{2}) "
```

```
62:      "([0-9]{2}):([0-9]{2}):([0-9]{2}).([0-9]{3}):INFO:"
63:      "oejs.AbstractConnector:Started SelectChannelConnector@0.0.0.0:9080.*");
64:      // Space
65:      boost::regex start_service(
66:          "Starting\\ Service\\.\\.\\.\\. ([a-z]|[A-Z]+).+");
67:      boost::regex end_service(
68:          "Service\\.\\. started\\.\\. successfully\\.\\.\\.\\. "
69:          "([a-z]|[A-Z]+).+\\.\\.\\.\\.([0-9]+).+");
70:      // check if input file is open
71:      if (!logFile.is_open()) {
72:          std::cout << "ERROR: no input log file" << std::endl;
73:          return -1;
74:      } else {
75:          // Go through the loop
76:          while (getline(logFile, line)) {
77:              // search the start regex code
78:              if (regex_search(line, m, Boot_Start)) {
79:                  // store them into date and time
80:                  stored_date = date(stoi(m[1]), stoi(m[2]), stoi(m[3]));
81:                  beginTime = ptime(stored_date,
82:                      time_duration(stoi(m[4]), stoi(m[5]), stoi(m[6])));
83:                  // condition to see if it will fail to go to the next condition
84:                  if (s_boot) {
85:                      s_boot = false;
86:                      outputFile << "**** Incomplete boot ****\n" << std::endl;
87:                  }
88:                  // draw the start into the output file
89:                  outputFile << "=== Device boot ===\n"
90:                      << lineNumber << "(" << logName << "): "
91:                      << m[1] << "-" << m[2] << "-" << m[3]
92:                      << " "
93:                      << m[4] << ":" << m[5] << ":" << m[6]
94:                      << "Boot Start" << std::endl;
95:                  s_boot = true;
96:                  // Then do the same to the end but checking the regex
97:              } else if (regex_search(line, m, Boot_End)) {
98:                  finished_date = date(stoi(m[1]), stoi(m[2]), stoi(m[3]));
99:                  endTime = ptime(stored_date,
100:                      time_duration(stoi(m[4]), stoi(m[5]), stoi(m[6])));
101:                  // output the rest
102:                  outputFile << lineNumber << "(" << logName << "): "
103:                      << m[1] << "-" << m[2] << "-" << m[3]
104:                      << m[4] << ":" << m[5] << ":" << m[6]
105:                      << " Boot Completed" << std::endl;
106:                  total_time = endTime - beginTime;
107:                  outputFile << " "
108:                      << "Boot Time: " << total_time.total_milliseconds()
109:                      << "ms\n" << std::endl;
110:                  s_boot = false;
111:              }
112:              // increment line number to go to next line
113:              lineNumber++;
114:          }
115:          // closing the files
116:          logFile.close();
117:          outputFile.close();
118:      }
119:      return 0;
120: }
```