```
 1: // Copyright 2019 Adam Baptista
 2:
 3: #include <boost/regex.hpp>
 4: #include <iostream>
 5: #include <string>
 6: #include <cstdlib>
 7: #include <fstream>
 8: #include "boost/date_time/gregorian/gregorian.hpp"
 9: #include "boost/date_time/posix_time/posix_time.hpp"
10:
11: using std::cout;
12: using std::endl;
13: using std::string;
14: using std::ifstream;
15: using std::ofstream;
16:
17: using boost::regex;
18: using boost::smatch;
19: using boost::gregorian::date;
20: using boost::gregorian::from_simple_string;
21: using boost::posix_time::ptime;
22: using boost::posix_time::time_duration;
23:
24: template <typename T>
25: int to_int(const T& sm) {
26:     return atoi(sm.str().c_str());
27: }
28:
29: int main(int argc, char* argv[]) {
30:     smatch match;
31:     string line, str_boot, str_time, str_date, str_done;;
32:     ptime time_1, time_2;
33:     int line_num = 1;
34:     bool boot = false;
35:
36:
37:     if (argc != 2) {
38:         cout << "Invalid # of command lind arguments" << endl;
39:         return 0;
40:     }
41:     ifstream inFile(argv[1], ifstream::in);
42:     if (!inFile.is_open()) {
43:         cout << "Unable to open file \"" << argv[1] << "\"" << endl;
44:         return 0;
45:     }
46:
47:     string outFileName(string(argv[1]) + ".rpt");
48:     ofstream outFile;
49:     outFile.open(outFileName.c_str());
50:
51:     str_boot = "(.*log.c.166.*)";
52:     str_done = "(.*oejs.AbstractConnector:Started SelectChannelConnector.*)"
;
53:     str_time = "([[:digit:]]{2}):([[:digit:]]{2}):([[:digit:]]{2})";
54:     str_date = "([[:digit:]]{4})-([[:digit:]]{1,2})-([[:digit:]]{1,2}) ";
55:
56:     regex re_boot(str_date + str_time + str_boot);
57:     regex re_done(str_date + str_time + str_done);
58:
59:
60:     while (getline(inFile, line)) {
```

```
61:         if (regex_match(line, match, re_boot)) {
62:             if (boot)
63:                 outFile << "**** Incomplete boot **** \n" << endl;
64:         date _date(from_simple_string(match[0]));
65:         ptime temp(_date, time_duration(to_int(match[4]), to_int(match[5]),
66:             to_int(match[6])));
67:         time_1 = temp;
68:
69:         outFile << "=== Device boot ===" << endl;
70:         outFile << line_num << "(" << argv[1] << "): ";
71:         outFile << match[1] << "-" << match[2] << "-" << match[3] << " ";
72:         outFile << match[4] << ":" << match[5] << ":" << match[6] << " ";
73:         outFile << "Boot Start" << endl;
74:         boot = true;
75:
76:         } else if (regex_match(line, match, re_done)) {
77:             if (boot) {
78:                 date _date(from_simple_string(match[0]));
79:                 ptime temp(_date, time_duration(to_int(match[4]),
80:                     to_int(match[5]), to_int(match[6])));
81:                 time_2 = temp;
82:
83:                 time_duration td = time_2 - time_1;
84:
85:                 outFile << line_num << "(" << argv[1] << "): ";
86:                 outFile << match[1] << "-" << match[2] << "-"
87:                     << match[3] << " ";
88:                 outFile << match[4] << ":" << match[5] << ":"
89:                     << match[6] << " ";
90:                 outFile << "Boot Completed" << endl;
91:
92:                 outFile << "\tBoot Time: ";
93:                 outFile << td.total_milliseconds() << "ms \n" << endl;
94:
95:                 boot = false;
96:             } else {
97:                 outFile << "**** Unexpected boot ****\n" << endl;
98:             }
99:         }
100:        line_num++;
101:    }
102:    return 0;
103: }
104:
```