```
 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:                  << lineNum << "(" << 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 buy 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 << lineNum << "(" << 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:      lineNum++;
  108:   }
  109:   // closing the files
  110:   logFile.close();
  111:   outputFile.close();
  112:   }
  113:   return 0;
  114: }
```