

Exercises week 7: Multi threading I - Revision

Klaas Isaac Bijlsma
s2394480

David Vroom
s2309939

February 12, 2018

Exercise 51

Learn to use the chrono/clock facilities

In the previous attempt we used an if-else ladder, this is now changed to a switch. Furthermore we made a separate function to display the time, to prevent DRY and to reduce the length of the main function.

We used the following code.

main.ih

```
1 | #include <iostream>
2 | #include <chrono>
3 | #include <iomanip>
4 | #include <string>
5 |
6 | using namespace std;
7 | using namespace chrono;
8 |
9 | enum TimeMode
10 | {
11 |     GMTIME,
12 |     LOCALTIME
13 | };
14 |
15 | void printTime(time_point<system_clock> const &timePoint,
16 |               TimeMode mode = LOCALTIME);
```

printtime.cc

```
1 #include "main.ih"
2
3 void printTime(time_point<system_clock> const &timePoint, TimeMode mode)
4 {
5     time_t time = system_clock::to_time_t(timePoint);
6
7     if (mode == GMTIME)
8     {
9         cout << put_time(gmtime(&time), "Gmtime: %c\n");
10        return;
11    }
12
13    cout << put_time(localtime(&time), "Local time: %c\n");
14 }
```

main.cc

```
1 #include "main.ih"
2
3 int main(int argc, char **argv)
4 {
5     time_point<system_clock> timePoint{system_clock::now()};
6
7     printTime(timePoint);
8
9     printTime(timePoint, TimeMode::GMTIME);
10
11    string arg = argv[1];
12    int count = stoi(arg);
13    // add or subtract specified time to now
14    switch (char suffix = arg.back())
15    {
16        case 's':
17            timePoint += seconds(count);
18            break;
19
20        case 'm':
21            timePoint += minutes(count);
```

```

22         break;
23
24         case 'h':
25             timePoint += hours(count);
26             break;
27
28         default:
29             cout << "Invalid suffix\n";
30             return 1;
31     }
32
33     cout << "Below the modified time.\n";
34     printTime(timePoint);
35 }

```

Exercise 52

Learn to define a thread with objects that aren't functors

In the previous attempt, we defined a function that calls the member function `shift` of class `Handler`, to start a thread. This cluttered the global namespace, therefore we now use a (anonymous) lambda function. Also, we didn't start a second thread, now we do.

We used the following code.

handler/handler.ih

```
1 | #include "handler.h"
2 | #include <iostream>
3 |
4 | using namespace std;
```

handler/handler.h

```
1 | #ifndef INCLUDED_HANDLER_H
2 | #define INCLUDED_HANDLER_H
3 |
4 | #include <ostream>
5 | #include <string>
6 | #include <mutex>
7 |
8 | class Handler
9 | {
10 |     public:
11 |         void shift(std::ostream &out, std::string const &text,
12 |                   std::mutex &mut);
13 | };
14 |
15 | #endif
```

handler/shift.cc

```
1 | #include "handler.ih"
```

```

2
3 void Handler::shift(ostream &out, string const &text, mutex &mut)
4 {
5     lock_guard<mutex> lg(mut);
6
7     string str(text);
8     out << str << '\n';
9
10    for (size_t idx = 1; idx != str.size(); ++idx)
11    {
12        char first = str[0];
13        str.erase(0,1);
14        str.push_back(first);
15        out << str << '\n';
16    }
17 }

```

main.cc

```

1 #include <iostream>
2 #include <fstream>
3 #include <thread>
4 #include <mutex>
5 #include "handler/handler.h"
6
7 using namespace std;
8
9 int main(int argc, char **argv)
10 {
11     ofstream out(argv[1]);
12
13     cout << "Give text: \n";
14     string txt;
15     getline(cin, txt);
16
17     mutex shiftMutex;
18     Handler object;
19     thread th(
20         [&]
21         {

```

```

22         object.shift(out, txt, shiftMutex);
23     }
24 );
25
26     thread th2(
27         [&
28         {
29             Handler{}.shift(out, txt, shiftMutex);
30         }
31     );
32
33     th.join();
34     th2.join();
35 }

```