Exercises week 7 - Multi-threading I

Klaas Isaac Bijlsma
s2394480

David Vroom
s2309939

January 17, 2018

Exercise 49

Learn to apply basic multi-threading

We used the following code.

`main.ih`

```
#include <string>
#include <thread>
#include <chrono>
#include <vector>
#include <algorithm>
#include <iostream>
#include <iomanip>

using namespace std;
using namespace chrono;

void waiting(bool &ready);
```

`waiting.cc`

```
#include "main.ih"

void waiting(bool &ready)
{
    while (!ready)
    {
        cerr << '.';
        this_thread::sleep_for(seconds(1));
    }
    cerr << '\n';
}
```

main.cc

```
#include "main.ih"

int main(int argc, char **argv)
{
    size_t nPrimes = stoull(argv[1]);
    bool ready = false;

    thread wait(waiting, ref(ready));
    auto startTime = system_clock::to_time_t(system_clock::now());

    vector<size_t> vec{2};
    size_t next = 3;

    while (vec.size() != nPrimes)
    {
        auto iter =
            find_if(vec.begin(), vec.end(),
                [=](size_t prime)
                {
                    return next % prime == 0;
                }
            ); // Eratosthenes sieve
        if (iter == vec.end())
            vec.push_back(next); // next is prime number
        ++next;
    }

    auto endTime = system_clock::to_time_t(system_clock::now());
    ready = true; // Notify waiting thread that computation finished
    wait.join();

    for (size_t elem: vec)
        cout << elem << ' ';
    cout << '\n';

    cout << put_time(localtime(&startTime), "Starting time: %c") << '\n'
         << put_time(localtime(&endTime), "Ending time:   %c") << '\n'
         << "Computation of " << nPrimes << " primes took "
         << endTime - startTime << " seconds\n";
}
```

Exercise 50

Learn to perform time conversions

We used the following code.

main.cc

```
#include <iostream>
#include <chrono>

using namespace std;
using namespace chrono;

int main()
{
    cout << "Hours: ";
    int nHours;
    cin >> nHours;

    cout << "is equal to "
         << minutes(hours(nHours)).count()
         << " minutes\n";

    cout << "Seconds: ";
    int nSec;
    cin >> nSec;

    cout << "is equal to "
         << seconds(nSec).count() / seconds(minutes(1)).count()
         << " minutes\n";
}
```

Exercise 51

Learn to use the chrono/clock facilities

We used the following code.

main.cc

```
#include <iostream>
#include <chrono>
#include <iomanip>
#include <string.h>

using namespace std;
using namespace chrono;

int main(int argc, char **argv)
{
                            // get the current time
    time_point<system_clock> timePoint{system_clock::now()};

                            // convert it to a std::time_t:
    time_t time = system_clock::to_time_t(timePoint);

                            // display the time:
    cout << left << setw(14) << "Current time:"
         << put_time(localtime(&time), "%c") << '\n';

                            // display the gmtime
    cout << left << setw(14) << "Gmtime:"
         << put_time(gmtime(&time), "%c") << '\n';

    string arg = argv[1];
    char suffix = arg.back();
    int count = stoi(arg);

                            // add or subtract specified time to now
    if (suffix == 's')
        timePoint += seconds(count);
    else if (suffix == 'm')
        timePoint += minutes(count);
    else if (suffix == 'h')
        timePoint += hours(count);

                                // convert it to a std::time_t:
    time_t newTime = system_clock::to_time_t(timePoint);

                                // display the time:
    cout << left << setw(14) << "New time:"
         << put_time(localtime(&newTime), "%c") << '\n';
}
```

Exercise 52

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

We used the following code.

handler/handler.ih

```
#include "handler.h"
#include <iostream>

using namespace std;
```

handler/handler.h

```
#ifndef INCLUDED_HANDLER_H
#define INCLUDED_HANDLER_H

#include <ostream>
#include <string>
#include <mutex>

class Handler
{
    public:
        void shift(std::ostream &out, std::string const &text,
                   std::mutex &mut) const;
};

#endif
```

handler/shift.cc

```
#include "handler.ih"

void Handler::shift(ostream &out, string const &text, mutex &mut) const
{
    lock_guard<mutex> lg(mut);

    string str(text);
    out << str << '\n';

    for (size_t idx = 1; idx != str.size(); ++idx)
    {
        char first = str[0];
        str.erase(0,1);
        str.push_back(first);
        out << str << '\n';
    }
}
```

main.ih

```
#include <iostream>
#include <fstream>
#include <thread>
#include <mutex>
#include "handler/handler.h"

using namespace std;

void callShift(Handler const &handlerObj, ostream &out,
               string const &text, mutex &mut);
```

callshift.cc

```
#include "main.ih"

void callShift(Handler const &handlerObj, ostream &out,
               string const &text, mutex &mut)
{
    handlerObj.shift(out, text, mut);
}
```

main.cc

```
#include "main.ih"
```

```
int main(int argc, char **argv)
{
    ofstream out(argv[1]);
    cout << "Give text: \n";
    string txt;
    getline(cin, txt);

    mutex shiftMutex;

    Handler object;
    thread th(callShift, ref(object), ref(out), ref(txt), ref(shiftMutex));

    object.shift(out, txt, shiftMutex);
    th.join();
}
```

Exercise 53

Learn to design a simple producer/consumer program