

## Week 7: II

### Exercise 57

../57/main.ih

```
1 #define ERR(msg) printf("%s : %d", (msg), __LINE__)
2
3 using namespace std;
4
5 #include <string>    // Input arguments
6 #include <iostream>  // Output
7 #include <chrono>    // chrono:: facilities
8 #include <iomanip>    // put_time
```

../57/main.cc

```
1 #include "main.ih"
2
3 int main(int argc, char const **argv)
4 {
5     if (argc != 2)                                // Conditional exit
6     {
7         cerr << "Please pass an argument";
8         return 1;
9     }
10
11     string argvString = argv[1];  // Offset string (for s, m, h)
12     int offset = stoi(argv[1]);   // Offset integer, int for possible negatives
13
14     auto adjClock = chrono::system_clock::now();           // Curr. time
15     time_t adjClockT = chrono::system_clock::to_time_t(adjClock); // Same, time_t
16
17     cout << "Current time      : " << put_time(localtime(&adjClockT), "%c") << '\n'
18          << "Current time (GMT): " << put_time(gmtime(&adjClockT), "%c") << "\n"
19          << "Adjusted time      : ";
20     // Basic output of current time in local timezone and GMT
21
22     switch (argvString.back())
23     {
24     case 's':
25         adjClockT =
26             chrono::system_clock::to_time_t(adjClock + chrono::seconds{offset});
27         break;
28     case 'm':
29         adjClockT =
30             chrono::system_clock::to_time_t(adjClock + chrono::minutes{offset});
31         break;
32     case 'h':
33         adjClockT =
34             chrono::system_clock::to_time_t(adjClock + chrono::hours{offset});
35         break;
36     default:
37         cout << "Invalid time offset."; // Invalid input
38         return 1;
39     }
40     // Switch based on last letter of input string (s, m, h): determines offset
41     // for adjusted time
42     cout << put_time(localtime(&adjClockT), "%c");
43 }
```

## Exercise 60

../60-4/main.ih

```
1 #define ERR(msg) printf("%s : %d", (msg), __LINE__)
2
3 #include "client/client.ih"
4 #include "warehouse/warehouse.ih"
5
6 #include <iostream>
7 #include <string>
8 #include <thread>
9
10
11
12 using namespace std;
```

../60-4/main.cc

```
1 #include "main.ih"
2
3 int main(int argc, char const **argv)
4 {
5     Warehouse warehouse;
6
7     vector<Client> clients;
8
9     for (size_t idx = 1; idx < argc; ++idx)
10         clients.emplace_back( warehouse, argv[idx]);
11 //adding clients to the clients vector
12
13     vector<thread> threads;
14
15     for (auto &client: clients)
16         threads.emplace_back(&Client::printProduct, ref(client));
17
18     thread addThread(&Warehouse::addlines, ref(warehouse));
19 //thread that adds lines to the queue
20
21 //adding a thread that takes lines from the queue in the warehouse and prints
22 //it in the file corresponding to the client.
23
24     addThread.join(); //joining threads
25
26     for (auto &it: threads)
27         it.join();
28
29     for (auto client: clients) //printing nr of lines per file
30         cout << client.size() << '\n';
31
32 }
```

../60-4/client/client.h

```
1 #ifndef INCLUDED_CLIENT_
2 #define INCLUDED_CLIENT_
3
4 #include "../warehouse/warehouse.ih"
5
6 class Client
7 {
8     Warehouse &d_warehouse;
9     std::string d_outputFile;
10     size_t d_nrlines = 0;
11 }
```

```
12 public:
13     Client(Warehouse &warehouse, std::string const &outputFile);
14
15     size_t size();           //returns nr of lines printed
16     void printProduct();     //prints strings to file
17 private:
18 };
19
20 #endif
```

../60-4/client/client.ih

```
1 #include "client.h"
2 #include <fstream>
3 #include <iostream>
4 #include <chrono>
5 #include <thread>
6
7
8 using namespace std;
```

../60-4/client/c\_client.cc

```
1 #include "client.ih"
2
3 Client::Client(Warehouse &warehouse, string const &outputFile)
4 : d_warehouse(warehouse),
5   d_outputFile(outputFile)
6 {
7
8 }
```

../60-4/client/printProduct.cc

```
1 #include "client.ih"
2
3 void Client::printProduct()
4 {
5     ofstream outputStream(d_outputFile);
6
7     while(!(d_warehouse.empty() && d_warehouse.isitfinished()))
8     {
9         bool printit = true;
10        //printit is used to distinguish between an empty string that should be
11        //printed as it is an empty string in the warehouse and an empty string
12        //that is returned because there is nothing in the warehouse. In the
13        //latter case printit is set to false and it wont be printed.
14        string tmp = d_warehouse.getProduct(printit);
15
16        if(printit)
17        {
18            outputStream << tmp << '\n';
19            ++d_nrlines;
20        }
21    }
22 }
```

../60-4/client/size.cc

```
1 #include "client.ih"
2
3 size_t Client::size()
4 {
5     return d_nrlines;
6 }
```

../60-4/warehouse/warehouse.h

```
1  #ifndef INCLUDED_WAREHOUSE_
2  #define INCLUDED_WAREHOUSE_
3
4  #include <queue>
5  #include <string>
6  #include <mutex>
7  #include <condition_variable>
8
9  class Warehouse
10 {
11     std::queue<std::string> d_queue;
12     std::mutex wMutex;
13     std::condition_variable condition;
14     bool d_finished = false;
15
16     public:
17         Warehouse();
18
19         std::string &front(); //returns string that has been in queue the longest
20
21         bool empty(); //checks whether queue is empty
22
23         std::string getProduct(bool &printit);
24                             //used by clients to retrieve a string from
25                             // the queue
26
27         bool isitfinished(); //checks if there is more input to come
28
29         void addlines(); //processes input to queue and calls finished when there
30                             //is no more input
31
32     private:
33         std::string next(); //removes and returns string from queue
34         void addProduct(std::string const &line); //adds a string to the queue
35         void finished(); //sets d_finished to true and notifies all waiting
36
37 };
38
39 #endif
```

../60-4/warehouse/warehouse.ih

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

../60-4/warehouse/addLines.cc

```
1  #include "warehouse.ih"
2
3  void Warehouse::addlines()
4  {
5      string inputString;
6      while (getline(cin, inputString)) // While there is still user input
7          addProduct(inputString);    // Push that input to the queue
8
9      finished();                      // When input is done, signal that
10 }
```

../60-4/warehouse/addProduct.cc

```
1  #include "warehouse.ih"
```

```
2
3 void Warehouse::addProduct(string const &line)
4 {
5     lock_guard<mutex> lk(wMutex);
6     d_queue.push(line);
7
8     if (d_queue.size() == 1)
9         condition.notify_all(); //notify waiting clients a string is available
10
11 }
```

../60-4/warehouse/c\_warehouse.cc

```
1 #include "warehouse.ih"
2
3 Warehouse::Warehouse()
4 //:
5 {
6 }
```

../60-4/warehouse/empty.cc

```
1 #include "warehouse.ih"
2
3 bool Warehouse::empty()
4 {
5     return d_queue.empty();
6 }
```

../60-4/warehouse/finished.cc

```
1 #include "warehouse.ih"
2
3 void Warehouse::finished()
4 {
5     d_finished = true;
6     condition.notify_all(); //notify waiting processes there will be no more
7                             //products so they should stop waiting for one.
8 }
```

../60-4/warehouse/front.cc

```
1 #include "warehouse.ih"
2
3 string &Warehouse::front()
4 {
5     return d_queue.front();
6 }
```

../60-4/warehouse/getProduct.cc

```
1 #include "warehouse.ih"
2
3 string Warehouse::getProduct(bool &printit)
4 {
5     unique_lock<mutex> ul(wMutex);
6     while (empty() && !d_finished)
7         condition.wait(ul);
8
9     if (!empty())
10         return next();
11
12     printit = false;
13     return {}; //return empty string, which can happen if no more strings
```

```
14         //are going to be available
15     }
```

../60-4/warehouse/isitfinished.cc

```
1  #include "warehouse.ih"
2
3  bool Warehouse::isitfinished()
4  {
5      return d_finished;
6  }
```

../60-4/warehouse/next.cc

```
1  #include "warehouse.ih"
2
3  string Warehouse::next()
4  {
5      string front = d_queue.front();    // Get element from queue
6      d_queue.pop();                    // Remove that element
7
8      return front;                     // Return it
9  }
```