Bern University
of Applied Sciences

414078-HS2018-0 - C++ Programming II

EXERCISE-04

TABLE OF CONTENTS

1 Introduction

This exercise gives an introduction to threads and data races. In particular, you will learn the following topics when completing this exercise:

- Starting and Stopping threads
- Passing parameters to threads by value and reference, hence creating shared memory between threads
- Implement a (thread safe) file logger class
- Using `mutex`, `lock_guard` and `call_once` to synchronise threads

2 Submission

Submit your source code (as a zip-file) to Ilias **before the deadline** specified in Ilias.

3 Thread basics

In order to demonstrate the effects of data races a log file class is implemented and multiple threads are logging to it.

- Implement a class `ThreadLogFile` which opens a file stream, e.g. "threadLog.txt" when constructed and closes the file when destructed.
- Implement a member function `print` which takes the thread ID and an `int` value as parameters.
- For simplicity, implement a global function `logToFile`, which internally calls `print`, such that the following code:

```
LogFile log;
logToFile(log, 1);
```

  produces a file log similar to:

```
Log from thread:  139939765286656 with value:  1
```

- Next, create a thread which writes the value 2 to the same log file.
- Create a `vector` of threads and in a for loop, fill it with `N` threads logging to the file with the parameter `value` being the loop counter. Choose `N` to be a multiple number of supported threads by your system, *i.e.* $10 \cdot$`N`.
- Verify that the output file is messed up, *i.e.* not thread safe. It will look similar to:

```
Log from thread:  139939765286656 with value:  1
Log from thread:  Log from thread:  139939748501248139939740108543
Log from thread:  139939765286656 with value:  1
Log from thread:  Log from thread:  139939748501248139939740108543
with value:  4
```

- Make sure to join all the threads within an additional loop.

4 Data Races

In the first section of this exercise you've created a situation where multiple threads are racing for the same resource, namely the log file. Fix the issue, i.e. extend class `ThreadLogFile` to be thread safe using RAII methods provided by STL.

- Verify that the output will look similar to:

```
From 139629015045888:  2
From 139629015045888:  0
From 139628917028608:  3
From 139628998260480:  2
From 139628891850496:  6
.
.
.
```

- Within the `print()` function of class `ThreadLogFile`, add a header line to your log file. Make sure the header is written only once with minimal overhead.