Exercises week 8: Multi-threading II - Revision

Klaas Isaac Bijlsma s2394480
David Vroom s2309939

February 12, 2018

Exercise 58

Become familiar with ***packaged_task***

We used the following code,

main.cc

```
1  #include <iostream>
2  #include <future>
3  #include <thread>
4  #include <iomanip>
5  #include <numeric>
6
7  using namespace std;
8
9  enum
10 {
11     ROWS = 4,
12     COLS = 6,
13     COMMON = 5,
14 };
15
16 future<double> fut[ROWS][COLS];
17
18 double innerProductWrapper(double *rowPtr, double *colPtr)
19 {
20     return inner_product(rowPtr, rowPtr + COMMON, colPtr, 0);
```

```
21 }
22
23 void computeElement(double *rowPtr, double *colPtr, size_t row, size_t col)
24 {
25     packaged_task<double (double *, double *)> task(innerProductWrapper);
26     fut[row][col] = task.get_future();
27     thread(move(task), rowPtr, colPtr).detach();
28 }
29
30 int main()
31 {
32     double lhs[ROWS][COMMON] =
33     {
34         {1, 2, 3, 4, 1},
35         {3, 4, 5, 7, 4},
36         {2, 4, 5, 9, 3},
37         {21, 8, 9, 42, 4}
38     };
39
40     double rhsT[COLS][COMMON] =
41     {
42         {1, 2, 3, 4, 2},
43         {3, 4, 5, 7, 2},
44         {2, 4, 5, 90, 3},
45         {21, 8, 9, 42, 4},
46         {1, 2, 3, 4, 8},
47         {3, 4, 5, 7, 4}
48     };
49
50     for (size_t row = 0; row != ROWS; ++row)
51         for (size_t col = 0; col != COLS; ++col)
52             computeElement(lhs[row], rhsT[col], row, col);
53
54     for (size_t row = 0; row != ROWS; ++row)
55     {
56         for (size_t col = 0; col != COLS; ++col)
57         {
58             try
59             {
60                 cout << setw(5) << fut[row][col].get();
61             }
```

```
62              catch (exception &msg)
63              {
64                  cout << "Exception: " << msg.what() << '\n';
65                  return 1;
66              }
67          }
68          cout << '\n';
69      }
70  }
```

Exercise 60

Learn to implement a multi-threaded algorithm (2)

We used the following code,

main.cc

```
#include <iostream>
#include <algorithm>
#include <future>

using namespace std;

void quickSort(int *beg, int *end)
{
    if (end - beg <= 1)
        return;

    int lhs = *beg;
    int *mid = partition(beg + 1, end,
        [&](int arg)
        {
            return arg < lhs;
        }
    );

    swap(*beg, *(mid - 1));

    future<void> fut1 = async(launch::async, quickSort, beg, mid);
    future<void> fut2 = async(launch::async, quickSort, mid, end);

    fut1.get(); // block current thread until nested threads are ready
    fut2.get();
}

int main()
{
    int ia[] = {16, 2, 77, 40, 12071, 12, 3134, 42,
                5, 2453, 45, 3456, 35, 6, 56, 546, 2};

```

```
34      size_t iaSize = 17;
35
36      quickSort(ia, ia + iaSize);
37
38      for (int el: ia)
39          cout << el << '\n';
40  }
```