MANCHESTER 1824

EEEN 30052
Concurrent Systems

A1. Assignment Briefing 1

Dr Peter Green
Department of Electrical and Electronic Engineering,
University of Manchester
peter.green@manchester.ac.uk

Introduction

- The CodeBlocks Integrated Development System (IDS)
  - Will be used for the assignment
    - You can develop in whatever environment you like **BUT NOTE**:
      - **We cannot support any environment other than CodeBlocks**
      - Part of the assessment will involve testing your code in CodeBlocks
        - » **So if your submitted program does not work in CodeBlocks**
          - **You will receive a mark of 0 for the assignment**
- Please see the short video
  - about downloading and creating a project in CodeBlocks
    - In the Programming folder on Blackboard
  - Please ask for help in the lab if you get stuck
- NOTE: downloading the pre-prepared CodeBlocks project
  - Does not seem to work in Computer Clusters 4 and 5
  - However, if you create the project from scratch
    - And use the .h and .cpp files from Blackboard (Competitor and ThreadMap)
    - The project will successfully compile

`rand` and `srand`

- Random number generation is needed in the simulation
  - To represent the running time of each athlete
    - Given the world records for 100m sprint
      - This is probably a number in the range 9 s to 15 s
- The standard C/C++ functions `rand` and `srand`
  - should be used in this assignment
  - See the 2 documents in Programming/Assignments on Blackboard

Time Delays

- The random numbers are used to set time delays
  - In the threads representing the runners in the race
- Creating time delays is discussed in Lecture 4
  - See the following 2 slides

std::this_thread

Function Name	Purpose
get_id	Get thread id
yield	Yield to other threads
sleep_until	Sleep until a point in time
sleep_for	Sleep for time interval

http://www.cplusplus.com/reference/thread/this_thread/

- `this_thread` is a namespace that defines a set of functions
  - That enable a thread to perform **some action on itself**
  - It exists within the header of `<thread>`, along with class thread
  - The parameters of the 2`sleep` functions
    - Depend on `std::chrono`

EEEN30052 Lecture 4 30

Using `sleep_for` and `sleep_until`

- When using these functions
  - it is necessary to specify the time units you want to use
- These options are specified by types in the `chrono` library:
  - `std::this_thread::sleep_for(chrono::microseconds(1000));`
    - This will delay execution by 1000 microseconds
  - `std::this_thread::sleep_for(chrono::minutes(5));`
  - This will delay execution by 5 minutes
- Using `sleep_until` requires some knowledge of `chrono` library
  - and is beyond the scope of this unit

```
std::chrono::nanoseconds
std::chrono::microseconds
std::chrono::milliseconds
std::chrono::seconds
std::chrono::minutes
std::chrono::hours
```

EEEN30052 Lecture 4 31

Declarations and Definitions

- The sample code suggests the following

```
int main() {
   thread theThreads[NO_TEAMS][NO_MEMBERS];
   Competitor teamsAndMembers[NO_TEAMS][NO_MEMBERS];
```

- These statements declare these array
  - but DO NOT POPULATE THEM
- To do this you need to write a block of statements
  - Where each element of each array is assigned an object
    - Vis the thread or Competitor constructors