Thread Coordination Practical Solutions

Program Performing a Download

- Write a threaded program which simulates a download application
  - One thread fetches the data over the network
  - Another thread displays a progress bar
  - A third thread will process the data when the download is complete
- Use only techniques which have been covered in this course to date
  - There is no need to write any networking or GUI interface code
- Make sure that your program is not affected by data races

Hot Loop

- What is meant by a "hot loop"?
  - A hot loop is the result of code like this

```
bool upgrade_process = false;

std::lock_guard data_lck(data_mutex);
while (!update_progress) {}
```

  - The thread will run flat out until update_progress changes to "true"
  - The processor core usage will be at 100%

Hot Loop

- Why are hot loops considered bad?
  - Other threads, which could be doing useful work, cannot run on this core
  - A hot loop uses a lot of electricity, but does little useful work
  - Other threads cannot lock the mutex
  - The thread which is responsible for setting update_progress may not be able to set it
  - Potential infinite loop

Hot Loop Avoidance

- Suggest a way to avoid a "hot loop" in this code

```
bool upgrade_process = false;

std::lock_guard data_lck(data_mutex);
while (!update_progress) {}
```

  - To avoid the hot loop, unlock the mutex after checking the bool

```
std::unique_lock<std::mutex> data_lck(data_mutex);

while (!update_progress) {
   data_lck.unlock();
   std::this_thread::sleep_for(10ms);
   data_lck.lock();
}
```

Hot Loop Avoidance

- Is your proposed solution thread-safe?
  - Yes
  - The bool is only accessed when the mutex is locked
  - When the thread leaves the loop, the mutex is still locked
- What advantages does your solution have?
  - The sleep allows other threads to use the processor code
  - Unlocking the mutex allows other threads to lock it
  - In particular, the thread which sets update_progress

Implementation with Mutex

- Would you describe the example code in the video as elegant?
  - Not particularly
  - Many explicit loops
  - Much explicit locking and unlocking of mutexes
  - It would be better if the thread implementation provided some way for threads to communicate directly