CS597: CONCURRENCY AND ALGORITHMS

Synchronization

Francis Joseph Serina

CONDITION VARIABLES

Notifies another thread if a condition can be checked

Consumer works with a Unique Lock to wait if a certain condition is satisfied

- If the condition is satisfied; continue holding the lock and proceed
- If the condition is not satisfied; unlock the mutex, block the thread, and try again later

Producer notifies Condition Variable

See Study06

Condition Variables can be reused and can have multiple Consumers waiting

FUTURE

Get data returned by thread

Create thread, via std::async, with method that returns a value

std::async returns a **future**

Future becomes ready (holds data or exception) when thread is done

One-off event, get() can only be called once

Like a unique pointer, futures are <u>movable but not copyable</u>

See Study07

PACKAGED TASKS

Passing tasks between threads

Worker thread waits for additional tasks to execute

Calling thread

- Creates tasks
- Gets the associated future for each task
- Passes task to worker

Future is notified when associated task is complete

See Study08

PROMISE

Pass data between threads

Create promise and associated future

Move promise to Producer thread

Producer sets value into promise

Associated future gets the same value

See Study09

SHARED FUTURE

Multiple threads can receive the same data

Convert future to shared_future via share()

Copy shared future to other threads

All threads with copy of shared future gets notified

See Study10

CHRONO

CLOCK

Source of Time Information

- Time now (now)
- Type of the Value to represent time (time_point)
- Tick period (period)
- If clock ticks at a uniform rate or not (is_steady)

std::chrono::system_clock

std::chrono::steady_clock

std::chrono::high_resolution_clock (may be an alias of system_clock or steady_clock)

DURATION

std::chrono::duration<type, fraction>

- type could be any numerical data type
- fraction is how many seconds each unit of the duration represents

Example

- Each unit of std::chrono::duration<int, std::ratio<1,1000>> represents a millisecond as an integer
- Each unit of std::chrono::duration<float, std::ratio<10,1>> represents 10 seconds as float

Built-in

- Nanoseconds, microseconds, milliseconds, seconds, minutes, hours as some integral type

TIME POINT

Representation of time

- std::chrono::time_point<some clock, some duration>

Value of a time point is the length of time (duration) since an *epoch*

- Epoch is implementation dependent, commonly Jan 1, 1970 00:00:00

TIMEOUT

Delay a thread to give way for other threads

Delay for a *duration*

Delay until *time_point*

Things that can time out

- this_thread::sleep
- condition_variable and variants
- timed_mutex and variants
- unique_lock
- future and variants

See Study11