std::this_thread namespace

- <thread> 헤더
- 스레드 관련 4개의 전역 함수를 제공

함수	설명
this_thread::get_id()	스레드 ID 반환, thread::id 타입
this_thread::sleep_for(duration)	duration 동안 스레드 재움
this_thread::sleep_until(time_point)	time_point 까지 스레드 재움
this_thread::yield()	다른 스레드 스케줄링

std::thread class

- “스레드” 를 나타내는 클래스
- thread 객체 생성시 새로운 스레드가 생성된다.
- 스레드는 반드시 join() 또는 detach() 되어야 한다.
- 주요 멤버 함수

함수	설명
get_id()	스레드 ID 반환, thread::id 타입
join()	스레드가 종료 될 때 까지 대기
detach()	스레드가 독립적으로 실행될 수 있도록 허용

C언어에서의 thread 함수

- pthread_create, CreateThread
- 인자 한 개(void*) 를 가져야 한다.

C++ STL 에서의 thread 함수

- 일반 함수, 멤버 함수, 함수 객체, 람다 표현식등 호출 가능한 모든 요소를 사용할 수 있다.
- 인자 개수에도 제한이 없다.

스레드 함수에 인자 전달하기

- 스레드 클래스의 생성자 인자로 전달
- std::bind 사용
- 참조로 전달할 때는 ref() 사용

스레드 동기화

- mutex, timed_mutex, recursive_mutex, recursive_timed_mutex, shared_mutex, shared_timed_mutex
- conditional_variable
- call_once

std::lock_guard

- RAII 를 사용한 동기화 객체 관리
- 생성자에서 lock을 하고 소멸자에서 unlock 수행

스레드간 데이터 공유

- 전역 변수, 공유 메모리 등을 사용
- promise, future 객체 사용

promise, future

- <future>
- 스레드간 데이터를 공유하기 위한 도구

스레드를 생성하는 2가지 방법

- thread 클래스 사용 – low level api
- async 함수 사용 – high level api

async 함수 템플릿

- <future>
- 함수를 다른 스레드로 실행 하거나 지연된 실행을 할 때 사용

launch policy

launch::async	비동기로 실행(스레드 생성) 스레드 생성에 실패할 경우 예외 발생
launch::deferred	지연된 실행

launch policy

launch::async	비동기로 실행(스레드 생성) 스레드 생성에 실패할 경우 예외 발생
launch::deferred	지연된 실행
launch::async | launch::deferred	스레드 생성이 가능하면 스레드로 그렇지 않으면 지연된 실행