- **프로그래밍 방법**
  - c++ 프로그래밍 언어에 멀티스레드 라이브러리가 표준으로 존재
  - 2011년도에 새로운 c++언어의 표준으로 c++11이 공표되었음
  - 표준 c++언어를 지원하는 컴파일러라면 하드웨어나 운영체제에 관계 없이 사용

원래 표준이 있기 전에는 윈도우를 써서 프로그래밍한다 하면 CreateThread, 리눅스는 pthreadCreate 사용해서 멀티스레드를 만들었다. 이 방법은 호환성이 없다. 다행스럽게도 11년도에 c++11이 나오면서 컴파일러가 지원을 해주었다. 그 이후로 운영체제에 상관없이 멀티스레드 프로그래밍을 할 수 있다. 여기서 말하는 c++11에서 지원하는 멀티스레드 기능은 무엇인가 배울 것이다.

- **왜 c++11을 써야 하는가 (개인적인 경험)**
  - 1985년 “BASIC(또는 FORTRAN) 쓰지 왜 PASCAL 쓰는가? “
  - 1987년 “PASCAL 잘 쓰고 있는데 왜 C를 써야 하지?”
  - 2002년 “우리 C++, STL 써요.” by 송재경
  - 2016년 Netmarble => std::shared_ptr, 람다 함수, std::atomic,... 이곳저곳에서 사용.

c++20이 내년에 나온다. visual studio2019는 이미 나왔다. 왜 최신 것을 쓰냐? 기존 것을 쓰지? 최신을 쓰는 게 맞다. 왜냐하면 최신 기술들이 최신 컴파일러와 최신 언어 표준에 들어가 있기 때문이다. 그렇다면 왜 최신이 나왔을까? 기존 게 잘 안되니까. 옛날 것을 쓰자던 사람들이 옛날에도 많았다. 1985년 파스칼과 c가 나왔었다. 포트란, 베이직 쓰는 사람들이 왜 c 쓰냐 했지만 지금은 다 망했다. 모두 같은 이유이다.
오래된 게임을 그냥 c++을 쓴다. 바꾸면 시간이 걸려서. 새로 만든 게임은 c++11을 쓴다. 지금 광고하는 달빛조각사 그 게임은 엘릭서(Elixir)라는 언어를 사용하고 있다. 이건 나중에 나오는데 멀티스레드 전용 언어이다. 멀티스레드에 특화된 언어이다. 정확하게는 멀티스레드보다는 멀티 프로세스에 가깝다. c와는 완전히 다르다.

- **과거의 멀티스레드 프로그래밍 방법**
  - Windows
    - Win32 라이브러리에서 지원되는 API 사용.
    - Windows는 멀티스레드에 특화된 OS
  - Linux
    - pthread API를 사용하면 됨
    - "gcc-pthread test.cpp"형식으로 사용한다
    - pthread는 POSIX thread의 약자

윈도우는 처음부터 멀티스레드 운영체제였다. 리눅스는 아니었다. 근데 지금은 2000년대 초반부터 멀티스레드가 지원되기 시작했다. pthread 라이브러리를 받아서 설치하고 돌리면 돌아간다. 그런데 지금은 디폴트로 깔려있다.

- **운영체제에서 thread 지원**
  - Windows
    - 프로세스의 하위 개념
    - 모든 프로세스는 처음 시작 시 한 개의 스레드를 갖고 시작됨.
    - 운영체제가 thread를 직접 스케쥴링
    - 멀티 cpu(또는 core)라면 여러 개의 스레드를 동시에 실행시켜 줌
  - Linux
    - 리눅스는 thread라는 개념이 없다
      - 모든 것은 process
      - pthread라이브러리가 마치 thread가 존재하는 것처럼 보이게 해 준다
    - 리눅스에서 thread를 사용할 수 있다.
      - code, data 자원을 공유하는 process를 생성할 수 있다.
      - 다른 운영체제의 thread와 다른 것이 없다.

- **Windows**

운영체제에서 관리를 해주나? 윈도우는 해준다. 스레드라는 개념 자체가 프로세스 하위 개념으로 있다. 모든 프로그램은 처음 시작할 때 한 개의 스레드로 시작하고 스레드를 만들면 시분할로 돌아가며 만들고, 최대의 성능을 뽑아내게 스케쥴링 윈도우가 해준다. 일일히 우리가 지정해주지 않아도 된다. 해주어도 되지만 굳이 할 필요 없다. 우리가 하면 오히려 더 안 좋을 수가 있다. 프로세스가 1개가 아니라 여러개고, 스레드 스케줄을 제대로 하려면 다른 스레드의 빈 곳이 어디인지 정확하게 찾아서 해야하는데 매우 어렵다.

윈도우가 멀티 cpu 멀티코어 가리지 않고 만들면 모든 코어에서 알아서 할당을 해서 최적의 속도가 나오도록 해준다. 윈도우는 멀티스레드 프로그래밍에 특화되어있다 말해도 과언이 아니다.

멀티스레드를 적극적으로 도입한 최초의 운영체제는 윈도우이다. 그전에 있던 건 연구용이 하나 있었는데 **상용된 최초는 윈도우이다.**

- **Linux**

리눅스는 멀티스레드를 지원하지 않는다. 그것도 맞는 말이다. 모든 것은 process이다. pthread는 리눅스 표준이 아니라 다른 곳에서 만든 것이다. 그래서 리눅스는 지원하지 않는다는 것도 틀린 말이 아닌 것이다. **하지만, 의미상으로는 맞는 말이 아니다. 리눅스에서는 스레드가 없지만, 멀티스레드 프로그래밍은 할 수 있다.**

이게 무슨 말장난이냐? 말장난이 맞다. 프로세스를 만들 때 옵션을 줄 수 있다. 코드 데이터 힙 파일 같은 자원을 공유하는 프로세스를 만들 수 있다는 뜻이다. 스택은 분리를 하고 코드 데이터 힙은 공유하는 프로세스를 만드는데, 그러면 그것은 스레드랑 똑같게 된다. **이름이 스레드가 아니라 프로세스일 뿐. 그런데 차이가 없는 것이다.** 이건 근데 사전적인 의미인데, 지금은 누구도 리눅스에 스레드가 없다고 하지 않는다. 없긴 없지만, 멀티스레드 프로그래밍은 할 수 있다. 프로세스로 만들 때 설정을 바꾸어서 스레드의 기능을 시킬 수 있으니까.

결론적으로, 운영체제는 더 이상 신경 쓰지 않고 프로그래밍할 수 있다.

- **실습 - 싱글 스레드 프로그램**
  - 2를 5000만 번 더하는 프로그램

```
#include <iostream>

int sum;

int main()
{
	for (auto i = 0; i < 50000000; ++i) sum += 2;
	std::cout << "Sum = " << sum << endl;
}
```

처음부터 멀티스레드 프로그램을 만드는 것이 아니라, 싱글을 만들고 바꾸는 것이다. 껍데기는 싱글 스레드 프로그램과 같다.

- **실습 - 멀티 스레드 프로그래밍의 시작**
  - c++11의 thread

```
#include <thread>

void f();

int main()
{
	std::thread t1{f};   // f() executes in separate thread
	t1.join();                // wait for t1
}
```

처음 생성자에서 어디서부터 실행할지 지정해주면 생성되면서 수행도 거기서부터 시작됨. 위 코드에서는 t1이 생성되며 f 함수에서부터 시작된다. 멈췄다가 실행되고 그런 것 없이 만들 때부터 실행되고, 끝내는 건 join이다. close랑 다르다. 스레드는 close, destroy, delete 그런 반환의 개념이 아니다. 끝날 때까지 기다리는 것이다. 스레드를 만들었으면 종료를 시켜야하는데 그때까지 기다리는 것이다. 강제종료 시키는 방법은 없다. 끝날때까지 기다려야 한다. 그리고 끝났다고 하면 할 일이 끝난 것이니 결과를 받아서 저장을 하던지 출력을 하던지 해야 하는 것이다. 이때 join으로 기다려야 한다.

winapi 뒤져서 강제로 종료시킬 수 있지만 표준도 아니고 좋은 방법도 아니다. 왜? 자원관리가 제대로 안되기 때문이다. 스레드가 돌아가면서 자원을 할당받는데 강제 종료를 하면 반환이 안 된다. 메모리 릭, 리소스 릭이 생긴다. 또 강제로 종료한다? 그러면 할 일이 남고 종료할 마음이 없는데 종료시키는 것이다. 그럼 잘못짠 프로그램이다. 같이 마음이 맞아서 끝나야지 강제로 종료한다는 것은 각각 짠 사람이 다르다는 얘기고 절대로 이런식으로 하면 안된다. 강제로 종료해야 하는 일이 생겼다는 것은 프로그램 자체에 버그가 있다는 것이다.

그럼 얘를 누가 종료시키냐? 스스로 종료해야 한다. f는 함수인데 리턴을 하면 종료되는 것이다.