**운영체제 HW3(1)**

2017111683 남아현

1. 개요

이번 과제는 pthread 라이브리로 하나의 전역변수에 동시에 접근해서 충돌이 발생하는 문제를 재현하고 mutex 또는 synchronized를 사용하여 문제를 해결하는 것이다.

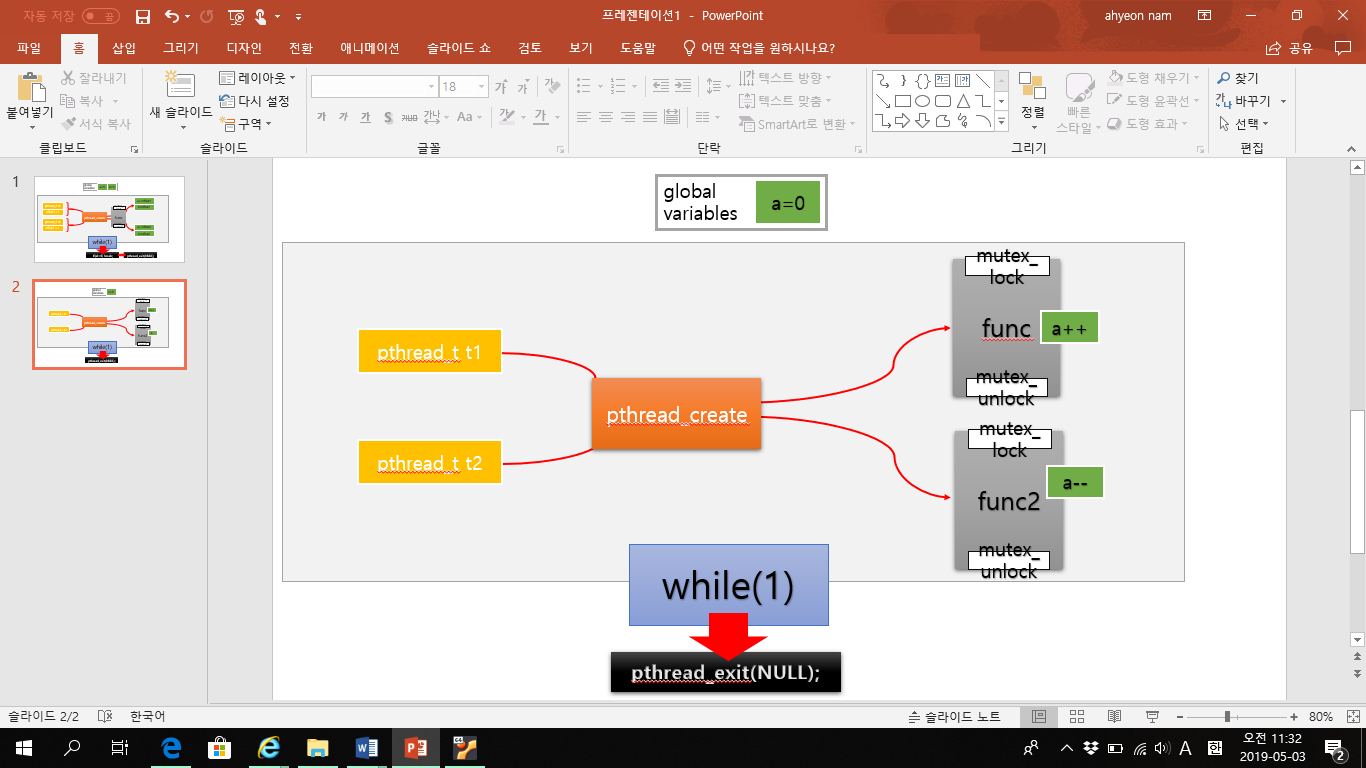
2. 프로그램 구조 설명

2.1 함수에 대한 설명

전역변수가 충돌하는 문제를 재현하기 위해 필요한 사용자지정함수는 비교적 간단하다. 따라서main.c 파일에 사용자지정함수 func, func2와 main함수를 포함시켰다. 우선 func, func2함수에서 사용할 전역변수 a를 volatile로 선언하고 그 값을 0으로 두었다. func와 func2함수 둘 다 for반복문을 사용했고, 한 번 loop할 때마다 a++, printf하여 a값을 확인하도록 하였다. main함수에서 pthread t1, t2를 선언하였다. For문 내부에 pthread\_create를 이용했다. for문이 한 번 돌 때마다 pthread\_create(&t1, NULL, func, (void \*)NULL), pthread\_create(&t2, NULL, func2, (void \*)NULL)를 반복하게 했다. for문을 10번 반복한 이유는 여러 번 pthread\_create를 행함으로써 충돌을 확인하기 위해서이다. printf문을 확인하여 각 func, func2으로 넘어갈 때 값이 이상해지거나 func와 func2순서가 섞이는 현상을 확인할 수 있다.

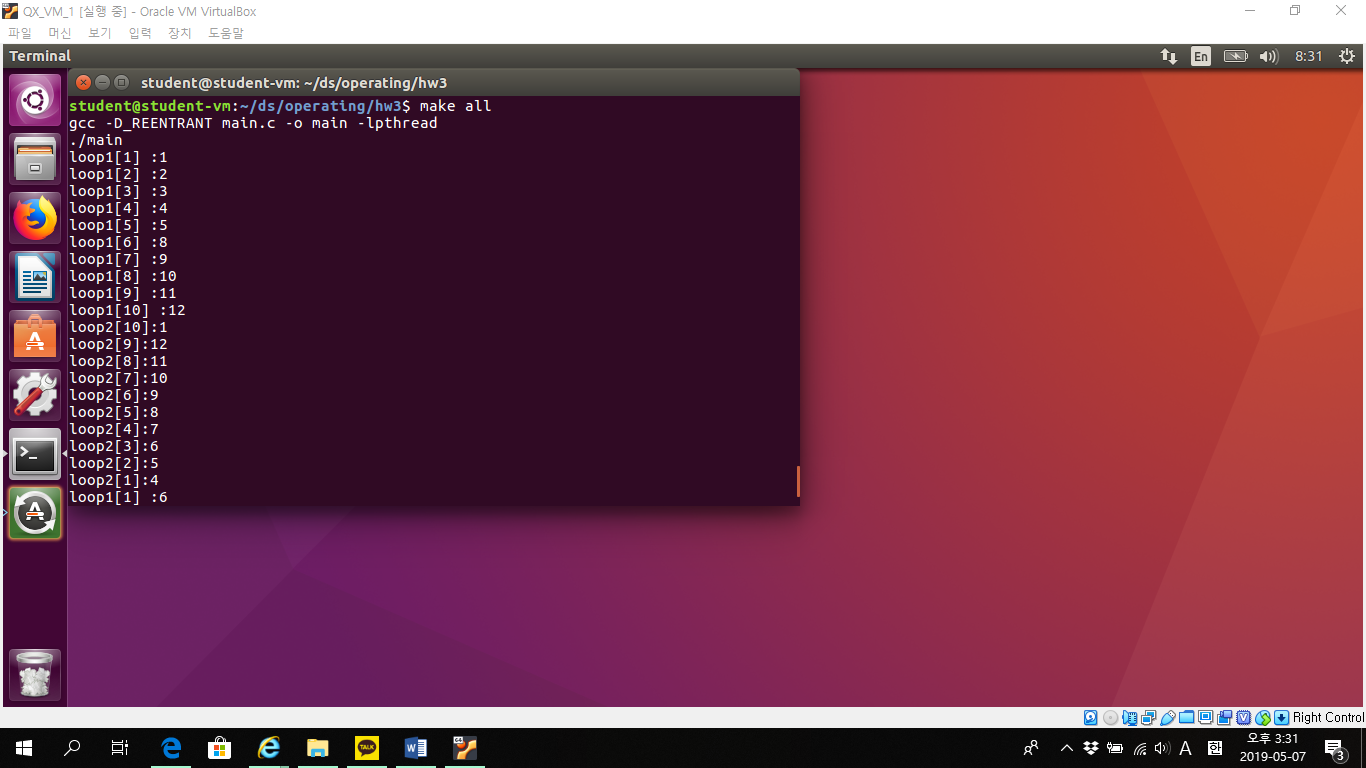
충돌을 확인한 이후에는, mutex를 이용했다. mutex를 선언하고, func와 func2 각각 함수 내부 시작부분에 pthread\_mutex\_lock를, 마지막부분에 pthread\_mutex\_unlock를 넣어 충돌을 방지하였다.

2.2 다이어그램



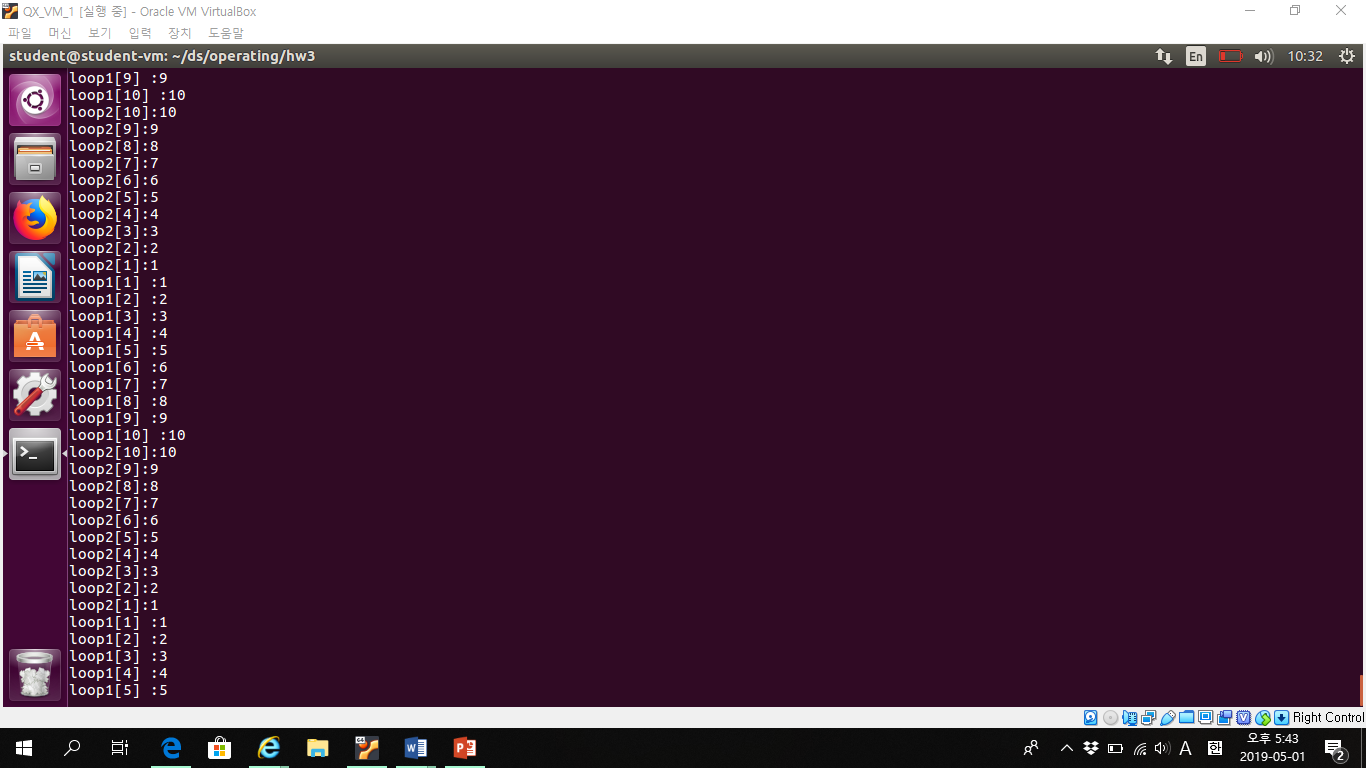
3. 실행 결과

<mutex가 없었을 때의 결과>



\*\*값이 이상해진다.

<mutex 추가한 결과>



\*\*값이 그대로 유지되며 function끼리 섞이는 일이 없다.

4. 고찰

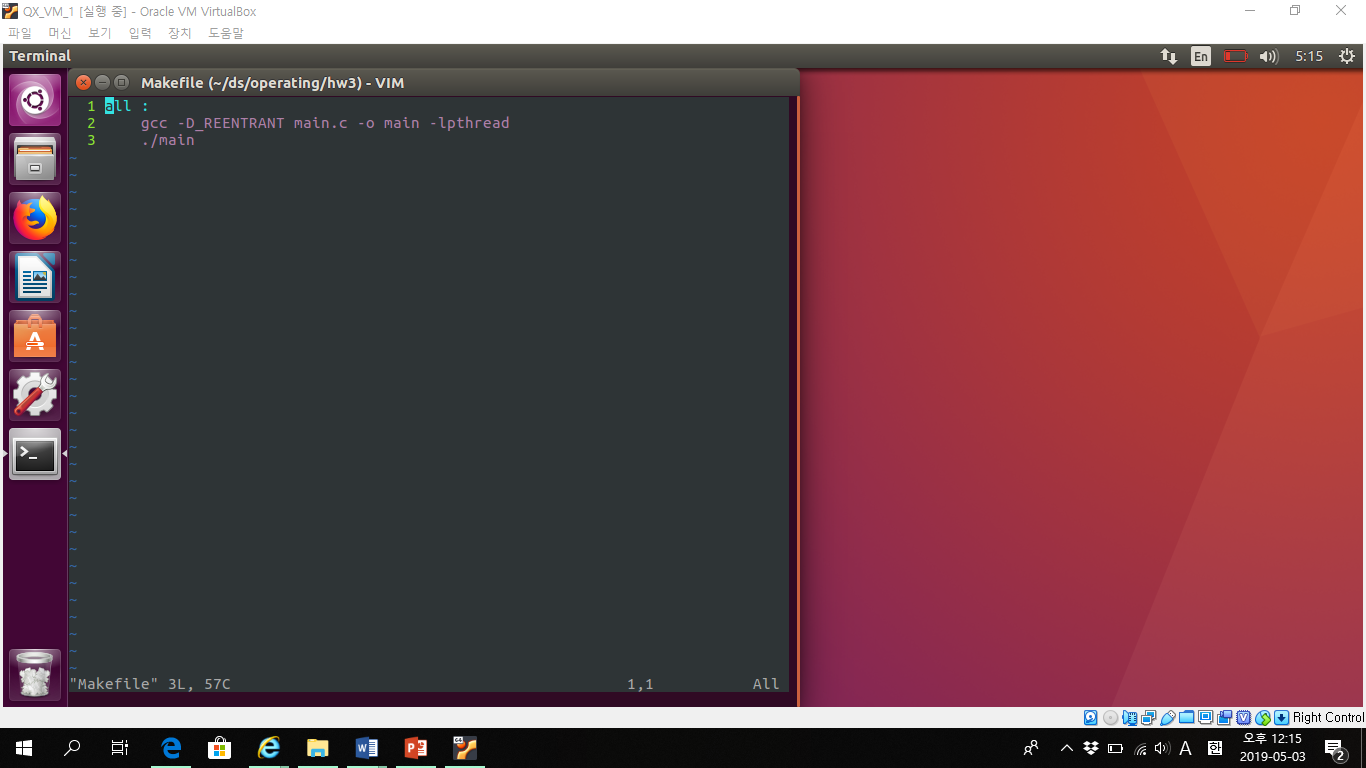
사실 처음에 과제 자체는 큰 어려움이 없으리라 생각했다. 기껏해야 전역변수를 주고, pthread\_create를 만들고 이후 함수를 동시에 실행시키면 되겠다는 자만이 있었다. 하지만 실제로 각각 전혀 다른 함수func1, func2를 만들어 진행했을 때, 값이 제대로 돌아가는지 확인하는 것이 어려웠으며 나조차 헷갈리기 시작했다. 또한 pthread\_create의 보이드포인터 인자값을 처음에 주었는데 과정이 복잡해지는 것 같아 전격 수정했다. 우선 보이드포인터 인자를 NULL로 주었고, 따라서 func1, func2에는 오직 전역변수만을 이용했으며, for문을 통해 반복을 진행했다.

처음에는 아무 생각없이 while문 내부에 pthread\_join 을 사용했다. Pthread\_join을 사용할 경우 오히려 mutex를 사용한 효과를 낸다는 것을 깨달았다. Pthread\_join의 특성상 thread가 종료될때까지 블록상태에서 대기하는 것이므로, 어쩌면 당연한 결과였다. 이에 pthread\_join을 생략하기로 했다. 대신 mutex를 이용해 명확하게 컴파일 순서를 조정하게 하였다. While(1)문 안에 pthread\_create를 넣어 직접 종료하지 않는 한 무한 반복되도록 했는데 이를 통해 전역변수 값이 오류를 확인할 확률을 높이기 위함이다. 이후 mutex lock/unlock의 위치를 고민했다. main함수에 있는 Pthread\_create함수사이에 mutex를 넣는다면 pthread\_create내의 함수진행과정과 전혀 상관없이 mutex가 진행되므로 사실상 무의미한 것이란 사실을 깨달았다. 따라서 mutex를 func, func2에 넣었고, pthread\_create에서 실행되는 func, func2가 모두 마칠때까지 넣음으로써 간단히 해결했다.

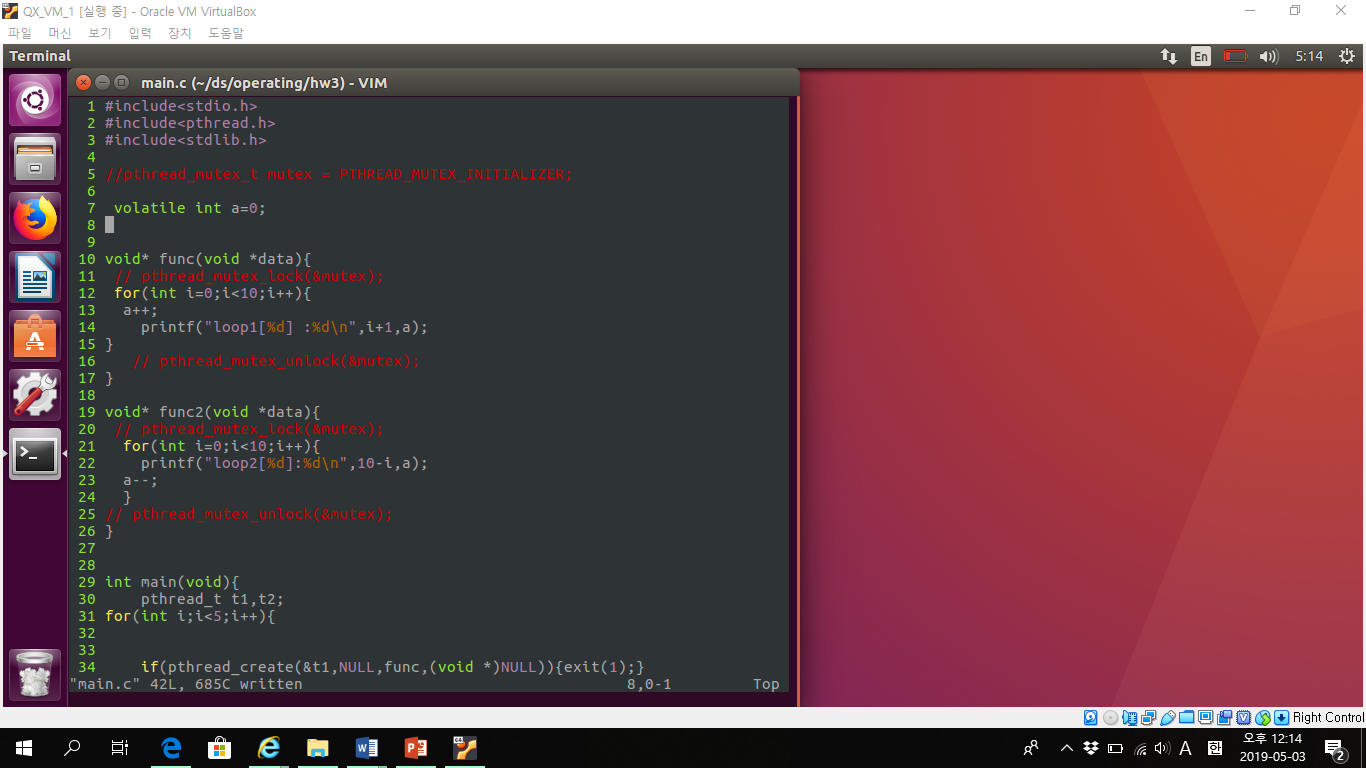
5. 프로그램 소스 파일

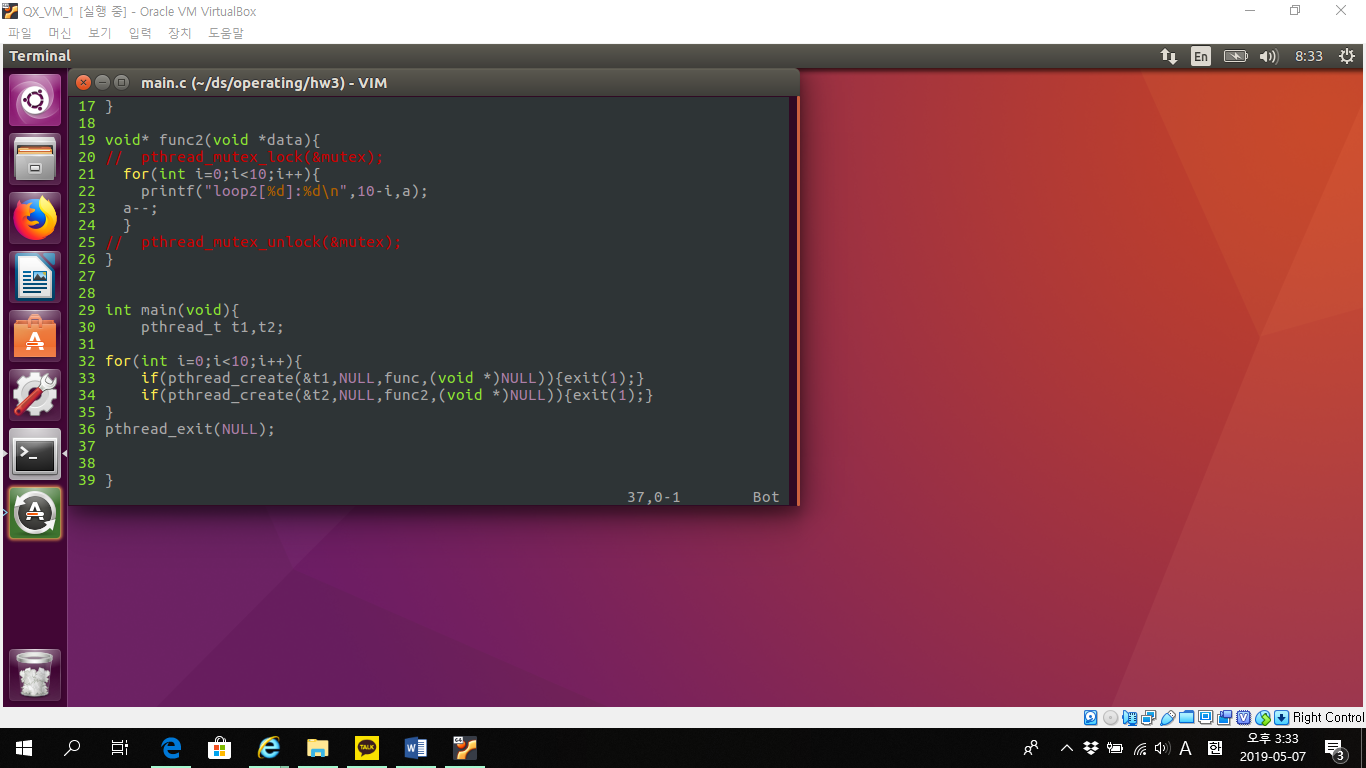
프로그램 소스 파일은 총 2개로 구성되어 있다. 사용자 정의 함수인 func, func2와 main함수를 담고있는 main.c, 그리고 Makefile이다.

<Makefile>



<main.c>





\*\*이후 // 주석처리한 부분을 더하면 mutex가 동작함

6. 자료 출처

https://iplusu.tistory.com/entry/Pthread2-쓰레드에서의-데이터-일치-문제

**운영체제 HW3(2)**

2017111683 남아현

1. 개요

행렬 곱셈을 수행하는 단일 스레드와 멀티 스레드 프로그램을 작성하여 그 수행 시간을 비교한다. 본 과제 수행에서는 행렬 값을 직접 부여했다.

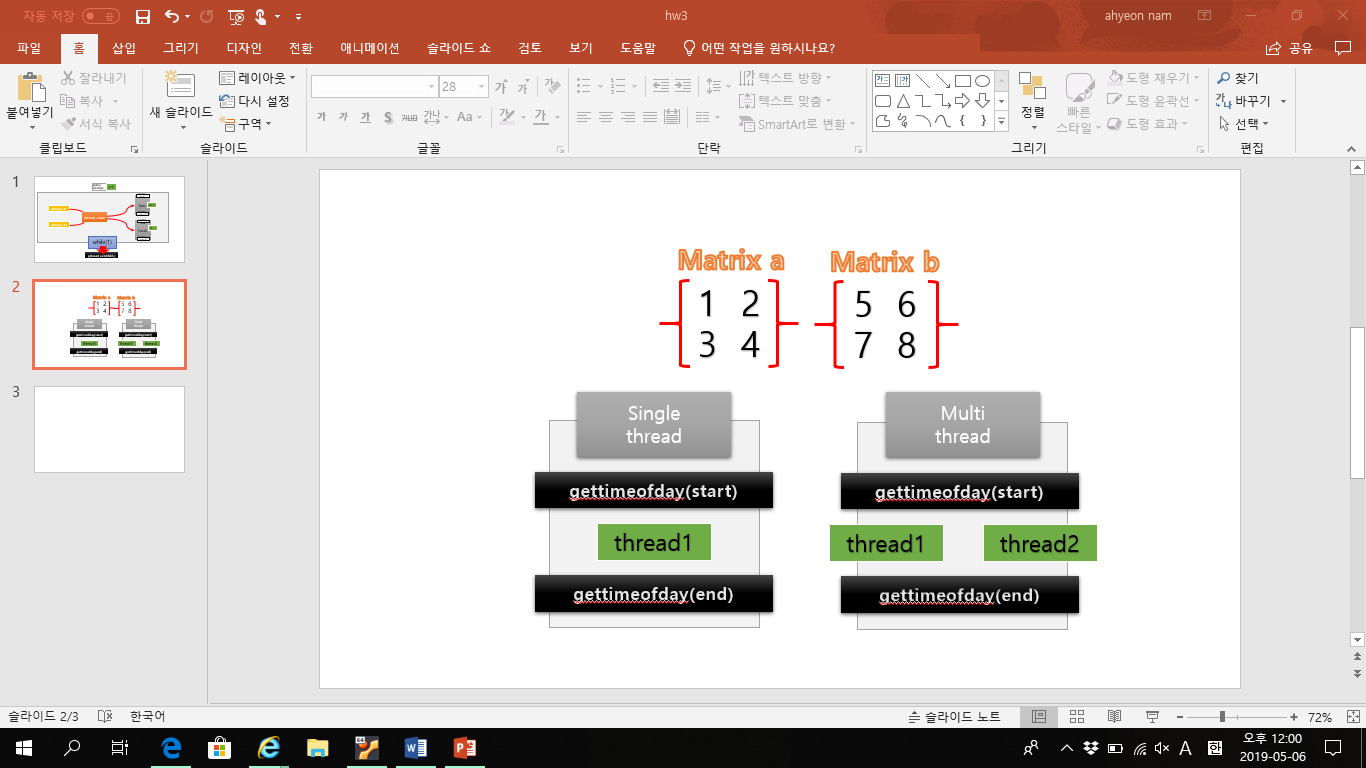
2. 프로그램 구조 설명

2.1 함수에 대한 설명

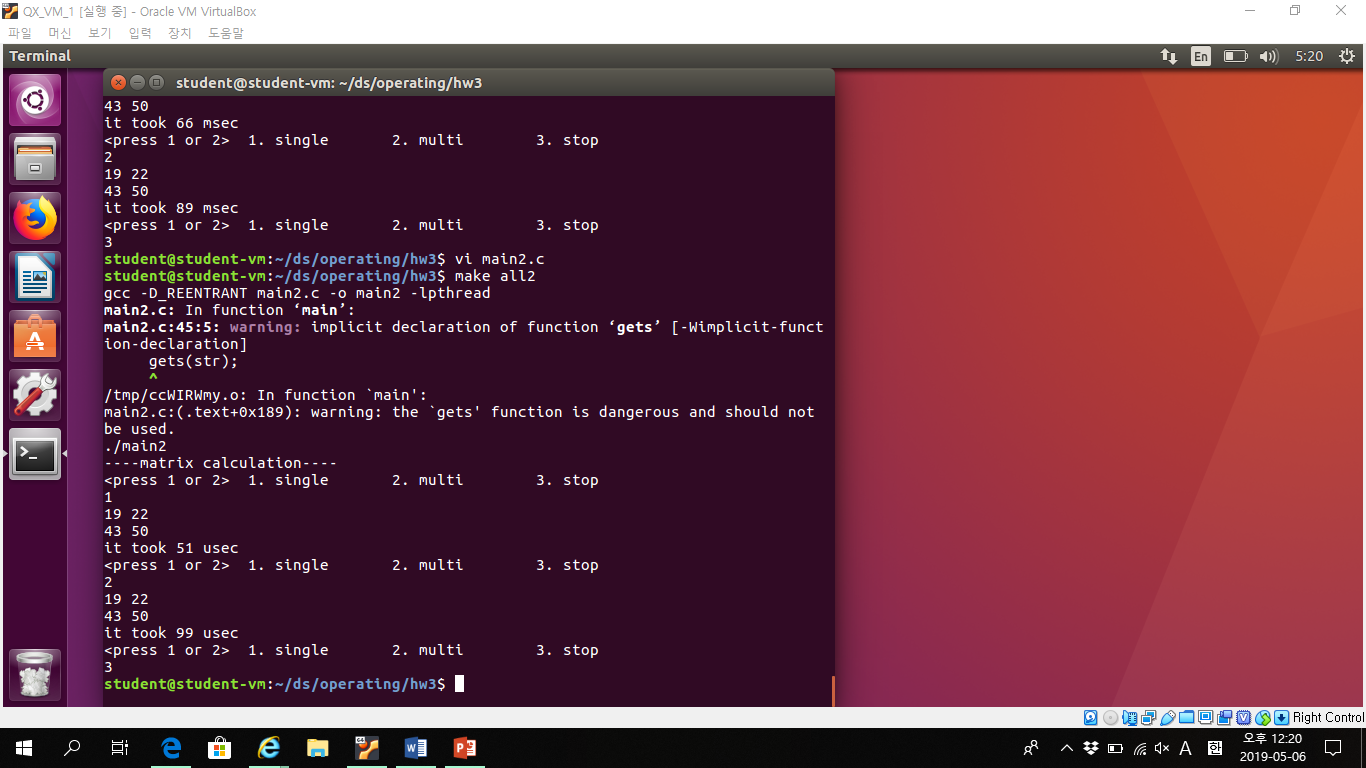
편의상2X2행렬 곱셈으로 진행했다. 이를 위해 단일스레드를 위한 sing와 멀티스레드를 위한 multi1, multi2라는 사용자 지정함수를 만들었다. 이 때 계산 인자를 행렬a와 행렬b라고 생각하고, 전역변수로 배열 a, b에 각각 임의의 값을 주었다. 이 후, 단일 스레드를 위한 함수 sing에는 2X2행렬 곱셈을 전부를 진행했으며 멀티 스레드를 위한 multi1, multi2에는 곱셈 연산을 절반으로 나눠 배분했다. Pthread\_create를 이용해 단일 스레드에는 sing를, 멀티 스레드에는 multi1, multi2를 썼고, 연산이 모두 끝나면 printf를 이용해 결과값을 확인하였다. 또한, gets함수와 strcmp함수를 이용해 1을 클릭하면 단일 스레드, 2를 클릭하면 멀티 스레드, 3을 클릭하면 실행이 끝나도록 했다.

이 과제의 핵심인 함수 시간 측정의 경우, <sys/time.h> 에 정의된 gettimeofday를 이용하였다. 우선 이미 정의된 구조체인 timeval구조체를 선언한다. 이후 gettimeofday 코드를 연산 시작 직전과 끝난 직후 넣는다. 이 후 연산이 끝난 시각을 기록한 값에서 연산 시작 직전 기록한 값을 빼 수행 시간을 구한다.

2.2 다이어그램



3. 실행 결과



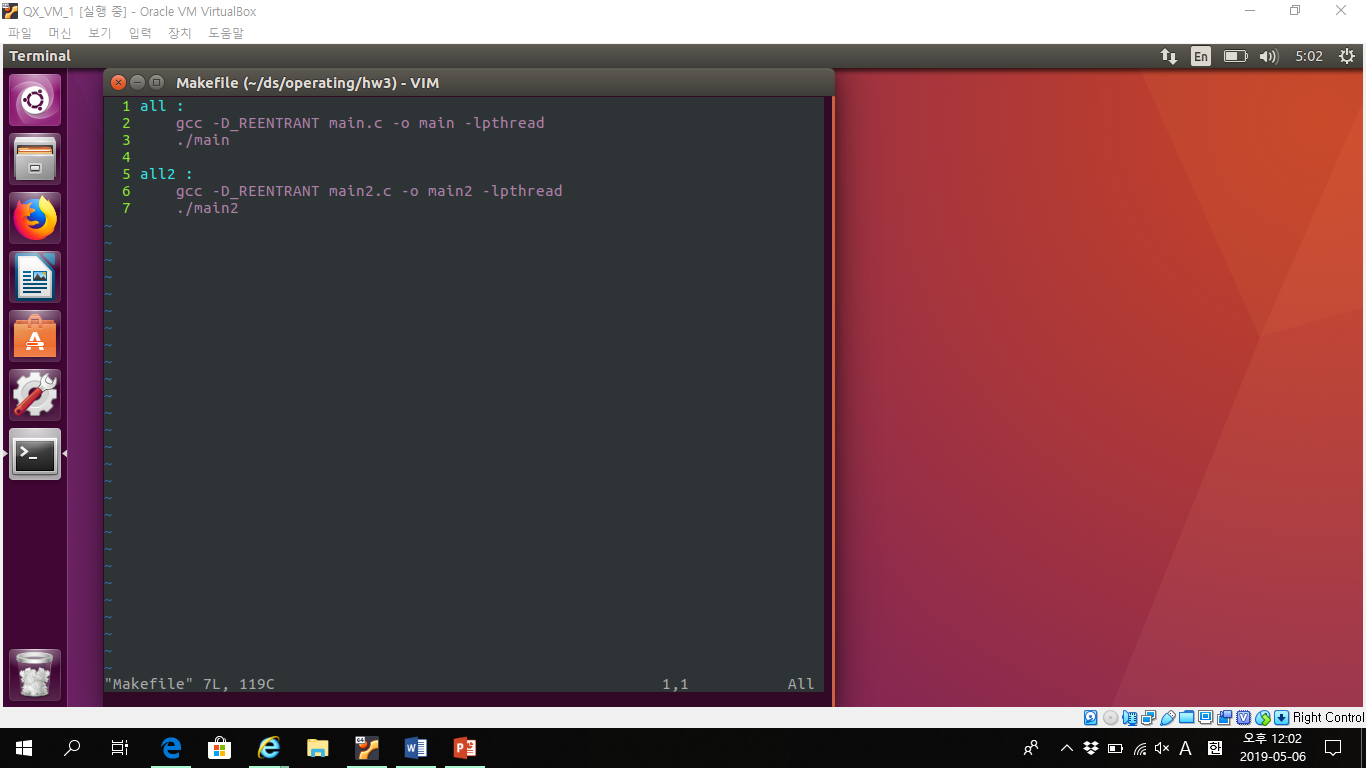
4. 고찰

Hw3 첫번째 과제를 통해 pthread에 대해 어느정도 익숙해진 덕분에 과제2를 하는데 아주 큰 어려움은 없었다. 하지만 gets함수의 경우 implicit declaration of function라는 warning이 계속 떴다. 대부분 헤더파일을 제대로 명시하지 않거나 프로토타입 선언을 하지 않아 발생하는 문제라는 결론을 내렸는데, gets함수의 경우 stdio.h를 기재했음에도 저런 결과가 떴다. 결국 warning을 없앨 해결방안을 찾지 못했다. 함수 생성 방식에서 고민을 많이 했는데 역시 임의로 지정해주는게 가장 낫다는 생각을 했다. 코드는 지저분하게 보이나, 과정을 줄이고 1, 2, 3 만 클릭하면 결과가 나오게끔 깔끔하게 구현한 것 같아 만족스럽다. gettimeofday함수에 관해 특히 고민이 많았는데, 자료 출처에 기재한 사이트를 참고한 덕에 금방 이해할 수 있었다. 처음에 sleep(1)을 생략한 채 make하니 멀티 스레드의 경우 결과값이 제대로 나오지 않았다. 이에 멀티 스레드 연산을 충분히 마치고 printf를 실행하기 위해 delay시키는 sleep(1)을 이용했다.

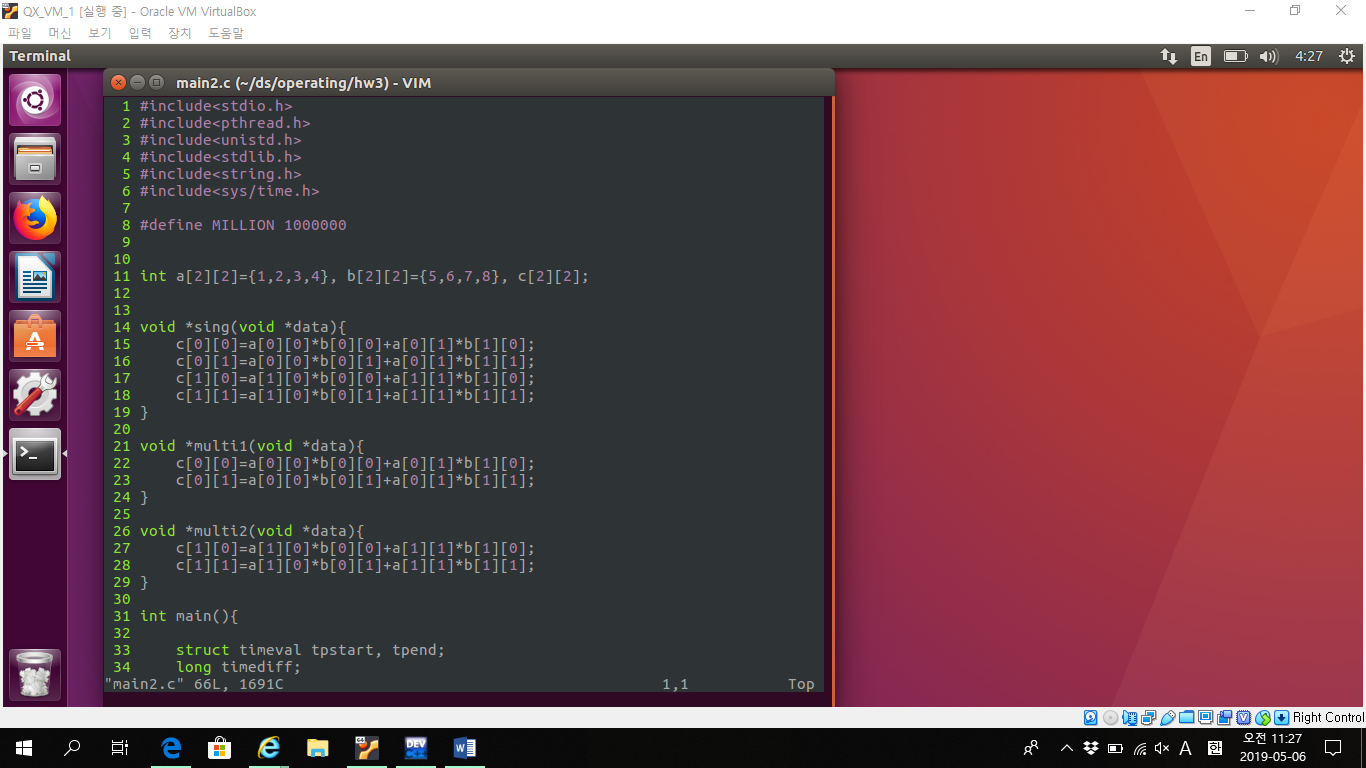
5. 프로그램 소스 파일

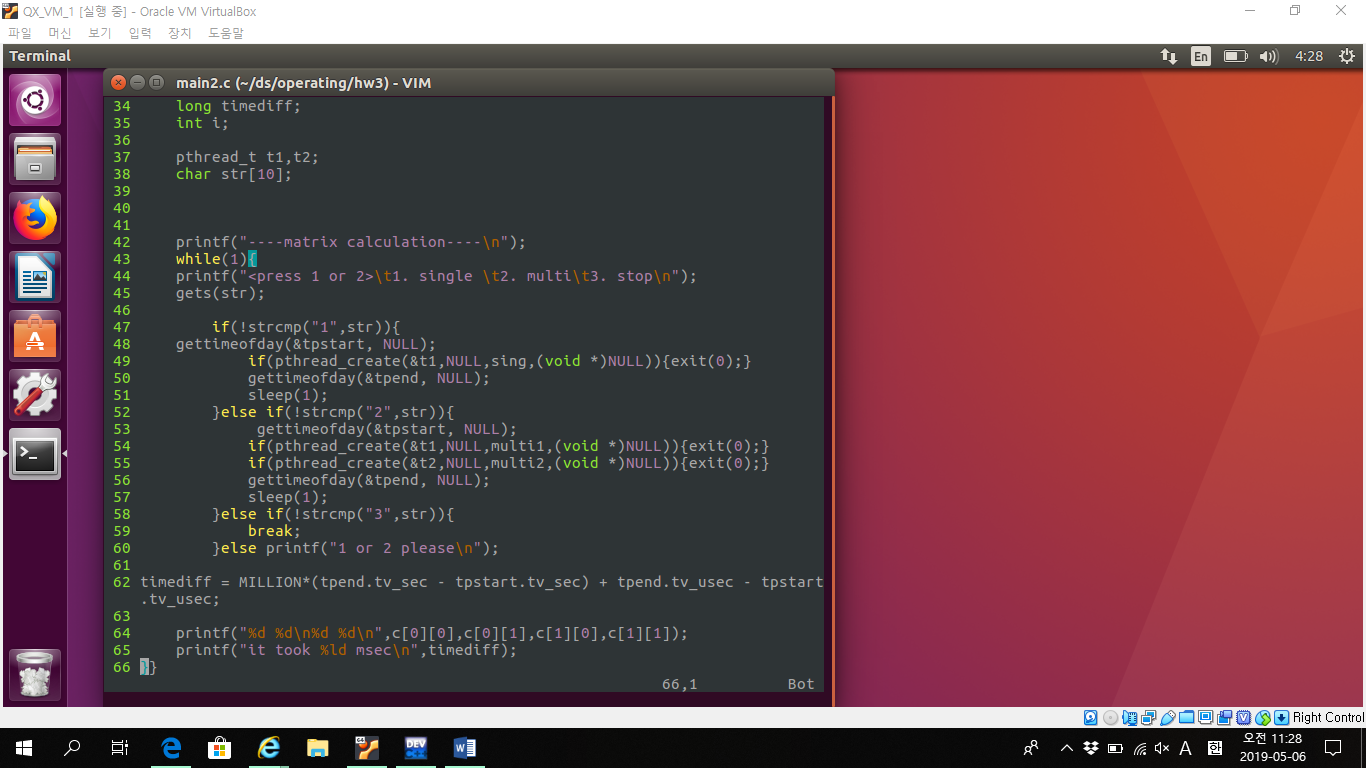
프로그램 소스 파일은 총 2개로 구성되어 있다. 사용자 정의 함수인 sing, multi1, multi2와 main함수를 담고있는 main2c, 그리고 Makefile이다.

<Makefile>



<main2.c 파일>





6. 자료 출처

<gettimeofday에 관한 이해>

https://s2kiess.blog.me/220147384776