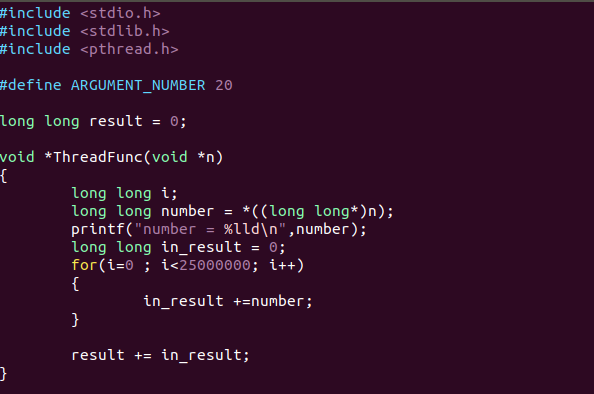
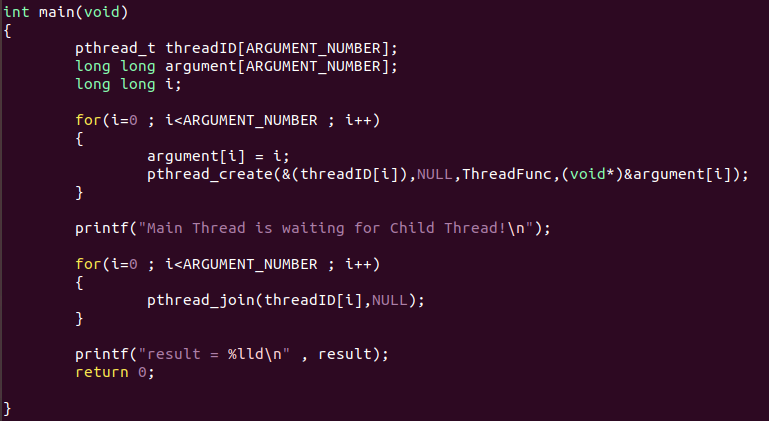
**2016025469 컴퓨터공학과 서건식**

**운영 체제 HW#4**

**제출 일자 : 2020/04/17**

1. **과제 A**
2. **프로그램 설명**

****

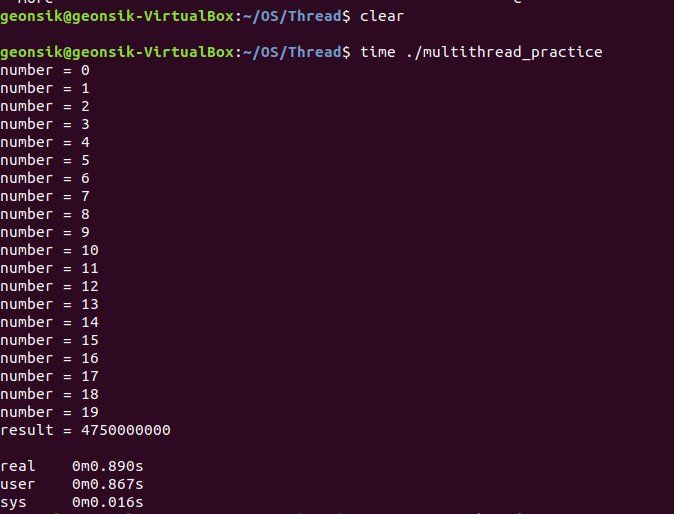
****

**기존에 제공된 코드와의 차이점은 pthread 헤더파일을 include 하여, 총 20개의 쓰레드를 만들고 그 쓰레드마다 동시에 작업을 실행하도록 하였습니다.**

**ThreadFunc에 지역변수 in\_result를 만들어, 각 쓰레드의 독립적인 스택공간을 활용할 수 있도록 하였습니다.**

**Main 함수에서는 pthread\_t threadID 배열을 선언하여 for문을 통해 thread를 만들었고, pthread\_join함수를 통해 끝날 때 까지 기다려 줍니다.**

1. **Time 결과값 설명**
2. **기존 코드**

****

1. **Solution 코드**

****

**수정한 코드의 real 타임이 훨씬 줄어든 것을 확인할 수 있습니다.**

1. **시간 차이가 나는 이유 설명**

**시간차이가 나는 이유는 멀티쓰레드 프로세싱의 차이입니다. 기존 코드는 쓰레드를 형성하지 않습니다. 그저 argument라는 배열의 값에 해당하는 인덱스에 도달할 때 마다 ThreadFunc 라는 함수를 호출하여 더하기 때문입니다.**

**하지만 솔루션 코드는, 20개의 쓰레드를 형성하여 동시에 더하는 작업을 하기 때문에 훨씬 빠릅니다. 다만 전역변수로 지정한 result 에 그대로 접근하게 되면 동기화 문제가 발생하므로, 각 쓰레드의 스택 영역에 지역변수를 설정한 후에 그 지역변수의 모든 값을 더한 후 전역변수 result에 반영해주면 동기화 문제를 lock없이 해결 가능합니다.**

1. **과제 B**
2. **Mutex 예비레포트**
3. **Mutual Exclusion**

**상호배제를 의미하며 한 프로세스가 공유 자원을 접근하는 코드를 수행 중이라면, 다른 프로세스들은 공유 자원을 접근하는 코드를 수행할 수 없다는 조건입니다.**

**이를 해주는 것은 동기화를 위함이며, 각 쓰레드가 Critical Section에 동시에 접근하게 되면 데이터의 값이 비결정적 입니다. 머신 인스트럭션은 High-Level Language의 sentence가 한줄 임에도 여러 개의 인스트럭션으로 컴파일 되기 때문에 중간에 스케쥴링이 일어나게 되면 잘못된 값을 저장할 수 있기 때문입니다.**

1. **Mutex**

**Mutual Exclusion 의 기법 중 하나인 Mutex입니다.**

**MUTual Exclusion 에서 알파벳을 따온 것입니다.**

**Critical section을 가진 스레드들의 실행 시간을 서로 겹치지 않게 단독으로 실행하게 하는 기술 이며, 이러한 공유 리소스에 대한 접근을 조율하기 위해서 Locking과 Unlocking을 사용합니다.**

1. **함수**

**뮤텍스의 생성을 위해선 pthread\_mutex\_t 타입의 변수를 선언 해주고 초기화 해줘야 합니다.**

1. **잠금**

**Pthread\_mutex\_lock()**

1. **잠금 해제**

**Pthread\_mutex\_unlock()**