**2016025469 컴퓨터공학과 서건식**

**운영 체제 HW#5**

**제출 일자 : 2020/04/23**

1. **과제 A**
2. **자료구조 설명**

**선언한 변수를 기준으로 자료구조를 설명하겠습니다.**

1. **Int critical\_section\_variable**

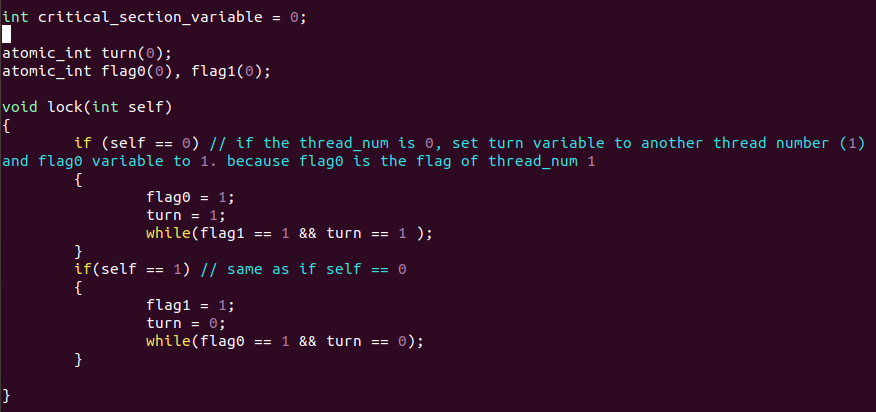
**각 쓰레드가 접근할 전역 변수입니다.**

1. **Atomic\_int turn, flag0, flag1**

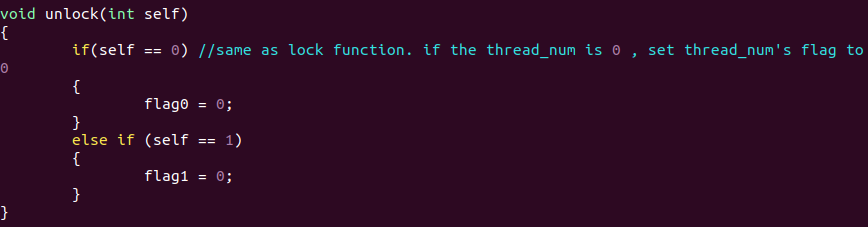
**Atomic\_int 클래스의 변수이므로, 값은 atomic하게 변합니다.**

1. **동기화 방법 설명**

**동기화 방법은, lock과 unlock 함수에서 thread\_num을 인자로 받아 그 인자의 값(쓰레드의 넘버)에 따라 flag과 turn값을 변경하며 동기화 합니다.**

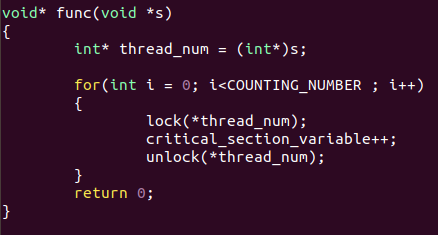
****

**lock함수에서, 들어온 인자값에 따라 경우를 나눠주었으며 flag0과 flag1은 각각 쓰레드1과 쓰레드2의 플래그 값이므로 경우마다 접근하여 값을 변경합니다.**

****

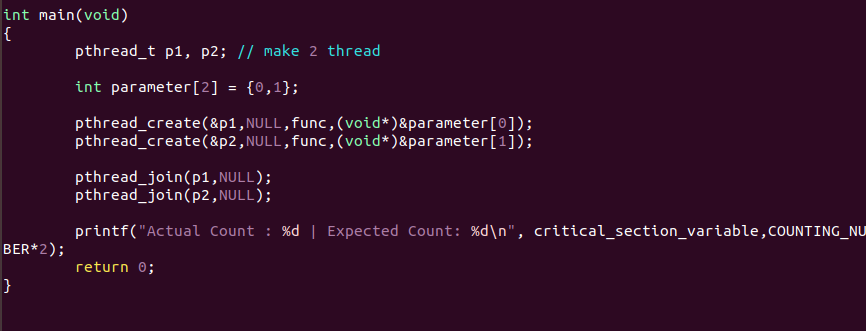
**Unlock 함수 또한 동일합니다. 각 쓰레드넘버에 해당하는 flag값을 false로 바꿔줌으로써 자신의 critical section 작업이 끝났음을 알려줍니다.**

1. **프로그램 구조 설명**

****

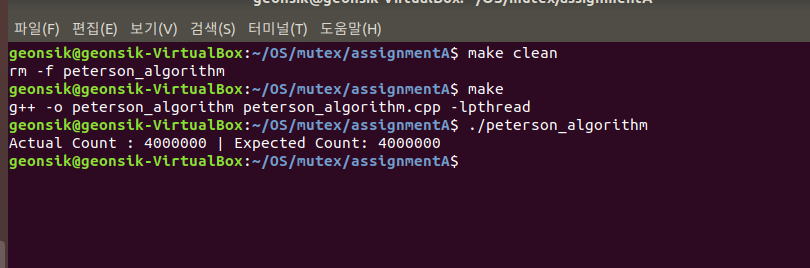
**위에서 설명한 lock과 unlock 함수를 기반으로, critical\_section\_variable이라는 변수(이곳이 critical section 입니다.)에 접근하기 전, 후에 lock 과 unlock함수를 호출하여 동기화를 실행합니다.**

**또한 인자로 받는 s는 thread\_num을 의미하며, main을 보시면**

****

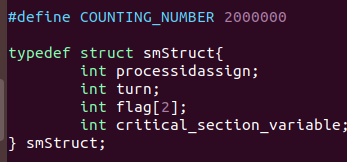
**각각 0과 1로 이루어져 있습니다.**

**main문의 끝에서 실제 계산된 값과 예상된 값을 출력합니다.**

****

**실행결과 예상된 값과 실제 계산된 값이 같으므로 동기화 문제를 해결함을 확인할 수 있습니다.**

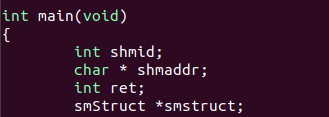
1. **과제 B**
2. **자료구조 설명**
3. **Parent.c**

****

1. **smStruct**

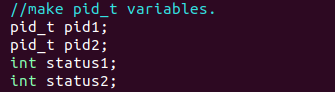
**해당 구조체는 제시해주신 것과 동일합니다.**

1. **Shared memory 변수들**

****

**Shared memory 할당을 위한 변수들입니다.**

1. **자식 process 생성을 위한 변수들**

****

**Process 2개를 생성하기 위한 변수들입니다.**

1. **Child.c**
2. **구조체, shared memory 변수는 parent.c 와 같습니다.**
3. **Localcount, Myorder**

****

**Localcount 변수는 실제로 critical\_section\_variable 값을 증가하는 횟수를 세기 위한 변수이며, 정상적으로 작동하면 2000000입니다.**

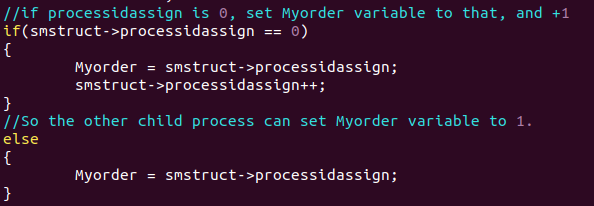
**Myorder는 smstruct의 processidassign 멤버변수를 저장합니다.**

1. **동기화 방법 설명**

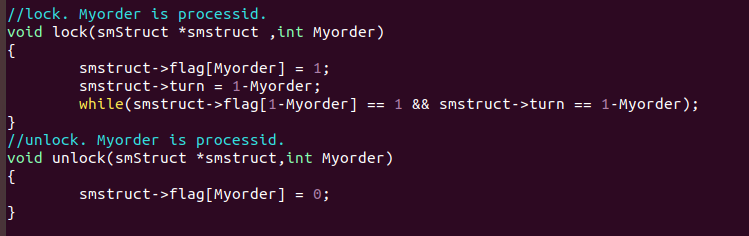
**동기화 방법은 다음과 같습니다.**

**Smstruct에 있는 processidassign 멤버변수를 통해 각 프로세스 마다 다른 processid를 배정합니다. 이를 통해서 프로세스 id가 구별되고, Peterson’s algorithm을 구현함으로써 동기화 하였습니다.**

**Myorder라는 변수를 child.c에 선언하여, processidassign 의 값을 저장해주었습니다.**

****

**0번 id가 다른 프로세스에서 지정되었다면 값을 올려줍니다. 따라서 다른프로세스는 if문에 의해서 1을 저장하게 되어 각 프로세스를 구분하는 것이 가능합니다.**

****

**이렇게 구분된 processid를 기반으로, Peterson’s algorithm을 구현합니다.**

**해당 함수는 child.c에만 구현하였습니다.**

1. **프로그램 구조 설명**
2. **Parent.c**

**부모 프로세스는 fork함수를 통해 2개의 자식프로세스를 생성합니다. execl함수를 사용하여 child 실행파일을 실행할 수 있도록 하였고,**

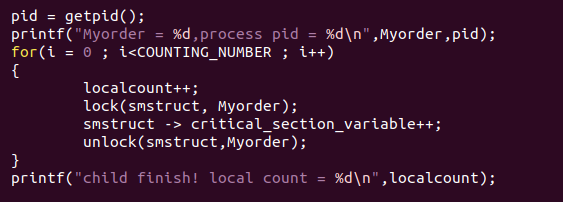
**2개의 자식 프로세스를 모두 생성하고 난 뒤에 waitpid 함수를 통해서 자식프로세스가 모두 종료될 때 까지 기다립니다.**

**만약 모두 종료되었다면, 실제값과 예상된 값을 출력합니다. 그 이후 shared memory를 detach 하고 delete합니다. Delete는 자식프로세스에서 진행하지 않고 오직 부모프로세스에서 진행합니다.**

1. **Child.c**

**Shard memory id를 얻고 attach한 후에 shmaddr를 smstruct에 저장하여 멤버변수 값에 접근합니다.**

**이후 동기화 과정을 거친 후에**

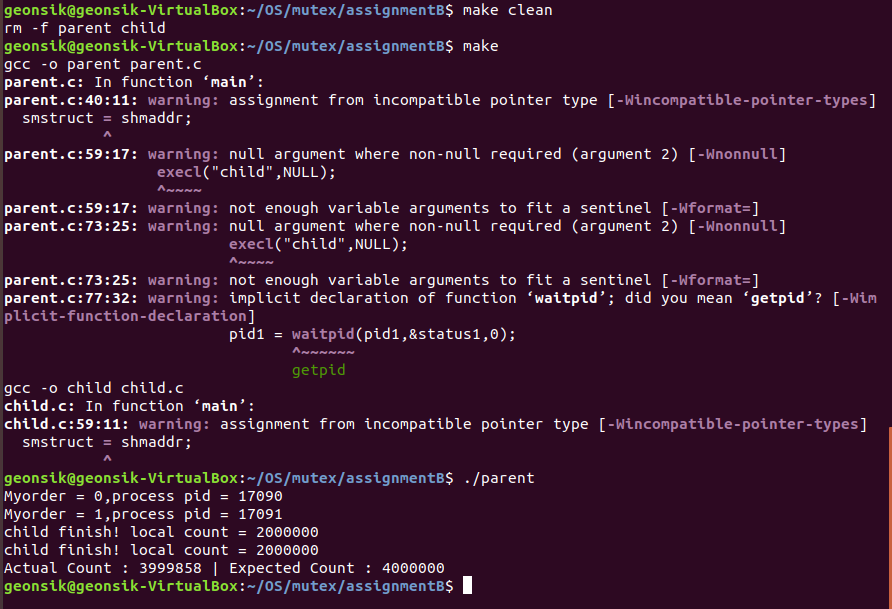
****

**자신이 부모로부터 부여받은 Myorder의 값과 pid를 출력합니다.**

**그 이후 critical\_section\_variable 값을 증가시키고(자식 프로세스에서만 증가시킵니다.), localcount값을 출력하면서 실제로 얼마나 수행되었는지를 출력합니다.**

**이 후 shared memory를 detach 합니다.**

**실행결과는 다음페이지에 있습니다.**



**동기화가 이루어져 실제 계산된 값이 예상된 값과 거의 비슷하게 계산된 것을 확인할 수 있습니다.**