2018008059 컴퓨터 소프트웨어학부 김은수

운영 체제 HW#6

제출 일자: 2020/05/02

A. 과제 A

1. 자료구조 설명

Critical section에서 접근될 변수 cur_writer와 cur_count를 선언해준다. 해당 writer가 critical section에 접근한 횟수를 알기 위해서 int 배열 writer_access_count[5]를 선언한다. Critical section에 진입한 reader 스레드의 개수를 세는 readcount를 선언한다. Readcount는 Critical section에 reader가 하나이상 진입하는 것을 허용하기 위해서 사용하는 변수이다. 세마포어 S와 wrt를 선언해준다. S는 readcount를 수정할 때 readcount에 대한 race를 방지하기 위해 사용하는 세마포어이다. Wrt는 reader와 writer간의 접근을 상호배제하기 위한 용도로 사용되는 세마포어이다.

2. 함수 설명

Writer함수

Writer함수는 인자로 스래드의 번호를 받는다. (ex. 0번 프로세서는 0을 writer함수의 인자로 보낸다.) 받은 인자를 int형식의 writer_iid에 할당한다. 그 다음 COUNTING_NUMBER만큼 loop가 실행되고 writer가 Critical section에 들어가기 전에 sem_wait(&wrt)를 적어서 reader나 다른 writer가 Critical seaction에 없을 때, Critical section에 접근할 수 있게 해준다. 그 후 cur_writer에 writer를 식별할 수 있는 writer_iid를 써준다. Cur_count에는 해당 writer가 접근한 횟수를 쓰는 것이므로 writer_access_count[writer_iid]를 할당해준다. 이때 접근한 횟수는 증가함으로 writer_access_count[writer_iid]++를 해준다. 그 다음 이 writer가 Critical_section에서 나왔다는 것을 나타내기 위해 sem_post(&wrt)를 써준다.

Reader함수

reader함수는 cur_writer와 cur_count를 출력하는 함수이다. COUNTING_NUMBER만큼 loop을 돈다. Readcount변수의 값을 수정하기 전에 race를 방지하기 위해서 sem_wait(&S)를 적어서 S값을 0으로 바꿔준다. 그 다음에 readcount를++로 증가시켜준다. if문을 통해서, Readcount가 1이라면, 즉 Critical section에 진입할 reader가 있다면 sem_wait(&wrt)로 wrt의 개수를 감소시켜준다. 이는 나중에 4번에서 다시 설명하겠지만, reader가 있을 경우 reader의 Critical section접근은 허용하지만, writer의 접근은 막는 역할을 한다. 그 다음 readcount를 증가시켜줬으므로 sem_post(&S)를 통해 S값을 1로 다시

만들어준다. 그 다음 "The most recent writer id: [] , count:[]" 문구를 통해 cur_writer와 cur_count를 출력해준다. 출력을 마친 뒤 Reader가 critical section에서 나올 것이므로, readcount를 감소시킨다. 이때도 마찬가지로 readcount 수정에 대한 race를 방지하기 위해 sem_wait(&S)를 적어준다. 그리고 감소시켜준 뒤 if문을 통해 readcount가 0이라면 즉, critical section에 있는 reader가 없다면 sem_post(&wrt)를 해준다. Readcount에 대한 감소가 마쳤으므로 sem_post(&S)를 해준다.

3. 프로그램 구조 설명

main함수 안에서 세마포어 S와 wrt를 sem_init이라는 함수를 통해 1로 초기화해준다. 그 다음 reader 스레드 2개와 writer 스레드 5개를 pthread_create함수를 통해서 만들어준다. Reader스레드는 create된 후 reader함수를 실행하고 인자는 없다. writer스레드는 create된 후 writer함수를 실행한다. 그리고 인자로 각 스레드를 구별할 수 있는 parameter값을 넣어준다. 이 parameter는 int parameter[5]={0,1,2,3,4}로 pthread_create함수 전에 선언해주었다. 그 다음 pthread_join을 통해 스레드 종료를 기다린다. 그 다음 sem_destrot()함수를 통해 다 사용한 세마포어를 파괴한다.

4. 프로그램이 어떻게 First Reader-Writers Problem을 해결하는지 설명

R1 R2 W3 R4 순서로 들어왔다고 할 때 R1 은 reader 함수에서 sem_wait 를 만난다. 이때, S=1 이기때문에 통과하고 다음에 readcount 는 1 로 증가한다. Readecount 가 1 이므로 if 문에 충족하니까 sem_wait(wrt)함수를 실행한다. Wrt 는 0 으로 값이 바뀐다. 그 다음, sem_post(&S)를 통해서 S 를 1 로 만들어 준다. (이는 readcount 값 수정에서 race 를 방지하기 위해서이다.) R1 은 Critical section 에 들어간다. R2 가 입장하면 readcount 가 증가하는데, readcount 가 2 이기 때문에 if 안 코드를 실행하지는 않는다. R2 도 critical section 에 들어가 코드를 수행한다. Writer 가 등장하면, writer 함수를 실행하는데 Reader 가 critical section 에 있으면 readcount 가 감소하지 않고, (즉 0 이 아니고) sem_post(wrt)를 실행하지 않아서 critical section 에 들어가지 못한다. Reader 가 모두 critical section 을 빠져나가야 지만 sem_post(&wrt)가 실행되어서 wrt 값이 1 로 증가하고 critical section 에 들어갈 수 있다. Writer 가 reader 가 모두 빠져나간 뒤 critical section 에 있으면 wrt 가 0 이다. 그리고 reader 가 reader 함수를 실행할 때 reader 가 빠져나간 뒤 들어온 첫 reader 이므로 readcount 가 1 이다. 따라서 if 문을 실행하고 sem_wait(wrt)에서 wrt 가 0 이므로 critical section 에 진입할 수 없다. 이런식으로 first reader- writers problem 을 해결한다.

5. 컴파일 방법

Make file을 만들고 gcc -o A_assignment A_assignment.c -lpthread로 A_assignment.c 코

드에 대한 실행파일 A_assignment를 만들었다.

과제 B

1. 자료구조 설명

스레드가 함수실행 시 넘겨줄 인자를 배열 phil로 설정한다. Phil[n] = {0,1,2,3,4,5}로 값이 할당되어 있다. 젓가락이 6개이므로 각 젓가락에 중복된 접근을 막기위해 세마포어 6개를 sem_t S[N]으로 설정해준다.

2. 함수 설명

Func 함수는 스레드가 생성되면 실행하는 함수이다. 이 함수는 각 스레드마다 philosopher함수를 몇 번 실행할 것인지 for문으로 정해주는 함수이다. 스레드를 구별하는 인자를 받고 그대로 philosopher함수의 인자로 넣어준다.

Philosopher함수

이 함수는 각 철학자들이 젓가락 오른쪽 왼쪽을 가져가고 밥 먹는 것을 실행하는 함수이다. 우선 스레드를 구별하는 인자 num을 받고 philosopher_number에 할당한다. 이 숫자는 왼쪽 젓가락의 숫자이기도 하다. 철학자는 0번부터 5번까지 총 6명이다. 0번철학자는 왼쪽으로 0번 젓가락 오른쪽으로 1번 젓가락을 가진다. 5번 철학자는 왼쪽으로 5번 젓가락 오른쪽으로 0번젓가락을 가진다. 오른쪽 젓가락을 나타내는 int right를 선언한다. 이는 (philosopher_number+1)%N의 값을 가진다. 철학자가 짝수번이라면 즉, philosopher_number%2가 0이라면, 오른쪽 젓가락을 먼저 가져간다. 가져가기 전에 sem_wait(&S[right])을 통해 오른쪽 젓가락이 0이 아니여야(0번 젓가락이 사용 중이 아니여야) 가져갈 수 있고 "philosopher 몇 번 pick up 오른쪽 chopstick"문구를 출력할 수 있다. 그 다음 sem_wait(&S[philosopher_number])를 통해 왼쪽 chopstick을 사용할 수 있으면 가져가고 "philosopher 몇번 pick up 왼쪽 chopstick"을 출력한다. 여기까지 실행 시 철학자가 두 젓가락을 가져갔으므로, 밥을 먹는다. "philosopher 몇번 starting eating"을 출력한다. sleep(2)만큼 대기했다가 "philosopher 몇 번 finishes eating"을 출력해 밥을 다 먹었다는 것을 알려준다. 그 다음, sem_post(&S[philosopher_number])를 통해 왼쪽 fork를 내려놓는다. 이는 왼쪽 젓가락변수를 1로 만드는 것으로 다른 철학자가 이 젓가락에 접근 할 수 있게 한다. 마찬가지로 오른쪽 젓가락도 sem_post(&S[right])를 통해 내려놓는다. 철학자가 홀수번이라면 sem_wait(&S[philosopher_number])를 통해 왼쪽 젓가락을 먼저 거져간다. 왼쪽 젓가락 변수가 0이라면 가져갈 수 없다. 왼쪽 젓가락을 가져갔으면 그에 상응하는 문구를 출력한다. 그 다음 오른쪽 젓가락을 접근하기 위해 sem_wait(&S[right])를 사용한다. 여기서도 오른쪽 젓가락이 0이라면, 즉 누군가 사용하고 있고 반납을 안 했다면 젓가락을 가져갈 수 없다. 두 젓가락을 가져왔으면 밥을 먹고 sleep(2)만큼 대기한 후 밥을 다 먹었다는 문구를 출력한다. 그 다음 sem_post를 통해 두

젓가락을 반납한다.

3. 프로그램 구조 설명

철학자가 6명이므로 스레드를 6개 만든다. 이는 pthread_t thread_id[N]으로 나타낼 수 있다. 또한, 각 젓가락에 대한 세마포어를 만들기 위해 sem_init(&S[i],0,1)을 통해서 각 세마포어를 1로 초기화 한다. 그 다음 pthread_create를 통해서 각 스레드를 만들고 이 스레드는 각 번호에 해당하는 number를 func함수의 매개변수로 넘긴다. func함수는 이 변수를 num에 할당한다. 그 다음에 for문에서 philosopher함수를 100번 돌리며, num을 매개변수로 넣어준다. 그러면 philosopher 함수는 스레드를 구별하는 변수를 philosopher_number에 할당하고 이 스레드가 짝수면 오른쪽 젓가락 먼저 가져가고 이 스레드가 홀수이면 왼쪽 젓가락을 먼저 가져간다. 이런 식으로 philosopher함수를 다 실행하면 pthread_join을 통해 스레드의 종료를 기다린다. 마지막으로 sem_destory를 통해 사용한 세마포어를 파괴한다.

4. 프로그램이 어떻게 Dining-Philosophers Problem을 해결하는지 설명

-LR solution과 deadlock, starvation

LR solution은 짝수 철학자와 그 옆 홀수철학자가 같은 젓가락을 먼저 잡는 것을 시도하는 solution이다. Dining-philosopher problem은 모든 철학자가 식사를 하지 못해 굶어 죽는 상황인 starvation인 상태가 발생한다. 이는 모든 철학자가 오른쪽(이나 왼쪽) 젓가락을 잡은 채로 모든 철학자가 다른 쪽 젓가락을 기다릴 때 발생한다. 즉, 1번 쓰레드가 2번 쓰레드의 영향으로 실행을 못하는 상황인데 2번 쓰레드는 1번 쓰레드에 의해 실행을 못하게 되면 두 쓰레드 모두 실행을 하지 못하게 되는 상황이다. 즉, deadlock에 빠지는 것이다. 근데 이 코드처럼 짝수는 오른쪽 젓가락 먼저 가져가고 홀수는 왼쪽 젓가락 먼저 가져간다고 하면 모두가 같은 방향의 (오른쪽이나 왼쪽의) 젓가락을 가지고 있어서 다른 젓가락을 기다리는 deadlock에 빠지지 않는다. 즉 단체로 한쪽에 젓가락을 가져가는 것을 방지하면서 dining-philosopher-problem을 해결할 수 있다.

5. 컴파일 방법

Makefile을 만들고 거기에 gcc -o B_assignment B_assignment.c -lpthread로 B_assignment.c 파일에 대한 B_assignmet 실행파일을 만들었다.