2018008331 컴퓨터소프트웨어학부 박민경

운영 체제 HW#6

제출 일자 : 2020/05/08

A. 과제 A

1. 자료구조 설명

First readers-writers problem을 구현하기 위해 2명의 reader와 5명의 writer을 pthread_t consumer, producer 배열로 선언하였다. 또한 각 thread의 count 값을 담기 위해 int count 배열을 선언하고, threadID 배열을 순서대로 0, 1, 2, 3, 4로 초기화하여 thread를 생성할 때 실행 함수 인자로 넘겨주어 cur_writer 값을 갱신할 수 있게 한다.

2. 함수 설명

main( ) – semaphore을 초기화하고, consumer와 producer 스레드를 생성한다. producer 스레드 생성 시 해당 스레드를 식별할 id인 threadId[i] 값을 인자로 넘겨 writer 함수에서 넘겨받을 수 있도록 한다. 각 스레드의 모든 실행이 끝날 때까지 기다리고, semaphore을 파괴하며 종료한다.

writer(void *num) – writer은 cur_writer, cur_count 값을 갱신하는 함수이다. 인자로 num을 받아 writer_id에 저장한다. for문을 돌며 critical section에서 cur_writer의 값을 writer_id 값으로 갱신해주고, 해당 스레드의 count 값을 1 증가시켜 cur_count 변수에 저장한다.

reader( ) – reader은 cur_writer, cur_count 값을 읽어 출력하는 함수이다. for문을 돌며 readcount를 1 증가시키고, readcount가 1일 경우 critical section에 writer가 없을 때까지 wait한 후 cur_writer와 cur_cout 값을 읽어 출력한다. 그리고 readcount를 1 감소시키고, readcount가 0일 경우 critical section에 작업중인 reader가 없음을 알린다.

3. 프로그램 구조 설명

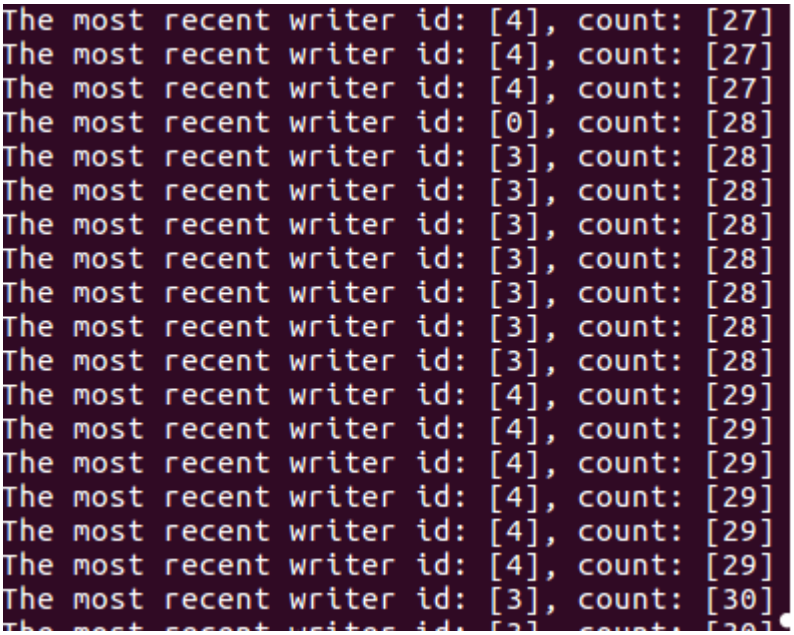
first readers-writers 문제를 해결하기 위해 semaphore을 이용한다. main에

서 consumer와 producer 스레드를 생성하고, consumer은 writer 함수, producer은 reader 함수를 실행한다. 편의상 consumer을 reader, producer을 writer라 하면, writer는 값을 수정하고, reader는 가장 최근에 수정된 값을 읽어 들여 출력하는 역할을 한다. COUNTING_NUMBER 만큼 for문을 반복하고 나면, main 스레드는 모든 스레드의 종료를 기다린 후 semaphore을 파괴하고 종료한다.

4. 프로그램이 어떻게 First Reader-Writers Problem을 해결하는지 설명

semaphore을 이용하여 writer와 reader가 critical section에 진입하는 것을 제어한다. writer가 critical section에 있을 때는 다른 writer와 reader가 접근하는 것을 모두 막아야 한다. 반면, reader는 공유 데이터를 읽어 들이기만 하므로 다른 reader가 함께 critical section에 접근하는 것을 허용한다. 하지만 읽는 도중 writer가 접근하여 데이터를 수정하면 안 되므로 writer의 접근은 차단한다. reader가 critical section에 접근해 있을 경우 reader의 수는 더 늘어날 수 있지만, writer은 모든 reader가 작업을 끝냈을 경우에만 critical section에 접근할 수 있으므로 writer의 starvation 문제가 발생할 수 있다.

<실행결과>



B. 과제 B

1. 자료구조 설명

6개의 chopstick과 6명의 philosopher을 가정한다. 6개의 chopstick은 semaphore로, 6명의 philosopher을 배열로 선언하였다. reader와 5명의 writer을 pthread_t consumer, producer 배열로 선언하였다. philoID 배열을 순서대로 0, 1, 2, 3, 4, 5로 초기화하여 thread를 생성할 때 실행 함수 인자로 넘겨주어 id 값을 갱신할 수 있게 한다.

2. 함수 설명

main( ) –semaphore인 stick 배열을 초기화하고 각 철학자에 해당하는 스레드를 생성한다. 이 때 인자로 philoId[i] 값을 넘겨주어 dining 함수에서 id 값을 갱신할 수 있도록 한다. 모든 스레드의 실행이 종료되기를 기다린 후 모든 semaphore을 파괴하고 종료한다.

dining( ) – 철학자들이 식사하는 것을 보여주는 함수이다. id를 선언하여 인자로 받은 num을 int형으로 저장하고, for문을 5번 돌며 각 철학자들이 총 5번씩 식사를 할 수 있도록 한다. for문 내에서는, id값이 짝수이면 오른쪽 젓가락부터, 홀수이면 왼쪽 젓가락부터 집어 든다. 모든 젓가락을 든 후에 식사를 하고, sleep(1) 후 식사가 완료되었다는 메시지를 출력한다.

3. 프로그램 구조 설명

dining-philosophers 문제를 해결하기 위해 semaphore을 이용한다. 철학자에 대한 정보는 스레드와 같고, 젓가락에 대한 접근은 양 옆 철학자에 대해 공유 데이터에 접근하는 것과 같다. 따라서 임계구역에 접근하는 것과 마찬가지로 한 명의 철학자가 젓가락을 사용하면 다른 철학자는 그 젓가락에 접근할 수 없는 것이다. 젓가락과 철학자에 대해 번호를 할당하고, 각 젓가락 자체를 semaphore로 하여 접근 제한을 만든다. main에서 스레드와 semaphore을 생성한 후, 각 철학자가 dining 함수를 실행함으로써 총 5번 식사를 하는 과정을 출력한다.

4. 프로그램이 어떻게 Dining-Philosophers Problem을 해결하는지 설명

각 철학자가 왼쪽의 젓가락을 들고 그 다음 오른쪽의 젓가락을 들어서 식사하는 알고리즘은, 철학자 모두가 무한정 서로를 기다리는 교착 상태에 빠질 수 있다. 이 프로그램에서는 LR solution을 채택함으로써 그것을 방지한다. 모든 철학자가 자신의 왼쪽 or 오른쪽 젓가락을 드는 사태는 발생하지 않기 때문에 교착 상태가 발생하지 않는다.

<실행 결과>