Producers And Consumers

과목명	운영체제
담당교수	최종무 교수님
학과	소프트웨어학과
학번	32191105
이름	김지민
제출일자	21.04.23

목차

1. 문제정의

Produce and Consumer(생산자/소비자)는 대표적인 병행성 문제 중 하나이다. 생산자는 버퍼에 데이터를 생산하고 소비자는 그 버퍼에 있는 데이터를 소비한다. 즉 공유자원을 주고받는다. 여기서 생산자와 소비자는 여러명이 될 수 있고 따라서 자원에 대한 상호배제가 보장되어야 한다. 그리고 버퍼가 비었을 때, 버퍼가 꽉 찼을 때를 해결해 주기 위해 생산자, 소비자의 순서(동기화)를 보장해 주어야한다.
이때 사용하는 것이 lock과 조건변수이다. lock을 걸면 다른 쓰레드가 들어올 수 없기 때문에 상호배제를 보장해주고 조건변수를 통해 wait을 걸면 signal을 받을 때 깨어나기 때문에 순서관계를 보장해준다.
이번 과제는 이러한 생산자 소비자 문제를 구현하는 것이다. 차 5종류(1,2,3,4,5)를 버퍼에 생산하는 생산자 쓰레드 하나와 버퍼에 생산된 차를 소비하는 소비자 쓰레드 5개를 만들어 lock을 걸었을 때와 걸지 않았을 때, fine-grained lock일 때와 coarse-grained lock일 때를 구현하였다. 버퍼(큐)의 최대 용량은 10으로 하고 구매자 1은 차의 종류가 1인 것만 구매할 수 있도록 설정했다.

2. 구현 설명

1) 사용한 구조체

① car_produce : 차 정보 관련 구조체(차 생산 시작 시간, 차의 종류, 종류별 차의 개수). RR스케줄링 기법을 사용하여 차를 생산하려고 만든 구조체인데 결국 구현하지 못했다.
② Node : 노드 정보 구조체(car_produce(data), 다음 주소의 노드(next))
③ car_queue(CQ) : 큐(버퍼) 구조체(큐의 차량 개수, 큐의 front/rear, lock)
④ th_struct2 : 소비자 쓰레드가 사용하는 함수에 전달하는 인자 구조체(CQ, 소비자 종류

2) 사용한 함수

① 큐 관련 함수
- InitQueue :큐, lock 초기화
- IsEmpty : 큐가 비었는지 확인
- Enqueue : 큐에 노드(차) 삽입
- Dequeue : 큐에있는 노드(차) 삭제

② 생산자/소비자
- producer : 생산자 함수, 큐(버퍼)에 차가 꽉 차면 wait, lock을 건 쓰레드가 없고 큐가 꽉 차지 않을 경우 Enqueue

- consumer : 소비자 함수, 큐(버퍼)에 차가 없으면 wait, lock을 건 쓰레드가 없고 큐가 비어있지 않으며 차의 종류가 자신과 같을 경우 Dequeue

③ main함수

- gettimeofday : 락을 걸지 않았을 때, fine-grained lock일 때, coarse-grained lock일 때의 수행 시간을 측정하기 위해 사용
- pthread_create : 생산자 쓰레드 1개, 소비자 쓰레드 5개 생산
- pthread_join : main쓰레드가 생산한 쓰레드들 기다리도록

3) 공부한 코드

① **pthread에 여러개의 인자 전달** : pthread_create의 네 번째 인자가 쓰레드가 수행할 함수의 인자를 전달하는 것인데, 함수의 인자가 여러개여서 처음에 그냥 ','로 구분했다. 이 방법은 인자가 너무 많다는 오류가 발생해서 검색해본 결과 함수의 인자를 여러개 전달할 때는 구조체를 사용해야 한다는 것을 알았다.

② **캐스팅** : 처음에는 main함수에 큐를 선언하고 그것을 쓰레드의 인자에 전달하면 쓰레드끼리 큐가 공유되는 줄 알았다. 그런데 자꾸 구매자 종류와 차의 종류가 같으면 다음 코드를 수행하는 if문에 들어가지 않고 무한 반복해서 헤맸다. 한줄 한줄 수행해보면서 큐가 공유되지 않는다는 것을 알았고,
CQ *q_=(CQ *)q; 이런 식으로 캐스팅해주어야 한다는 것을 깨달았다.

③ **동적할당, 포인터등의 주소 관리** : 동적할당의 개념을 잘 모르고 사용해서 쓸데없는 동적할당을 많이 해주었다. 그 바람에 main에서 할당해놓고 함수에서 또 할당을 해 제대로 사용하지 못하는 상황이 생겼다. 동적할당을 정확히 알고 사용해야 한다는 것을 깨달았다. 그리고 포인터를 통해 큐의 주소를 전달해야 같은 큐를 공유한다는 것을 알았다.

3. 구현 결과

1) 차량 총 생산 수 10일 때

```
root@jimin-VirtualBox:/home/jimin/project/2021_DKU_OS/lab2_sync# ./lab2_sync -c=10 -q=1
===Vehicle Production Problem ===
==================(1) No Lock Experiment=================
Experiment Info
        Total Produce Number = 10
        Final Balance Value = 0
        Execution time = 0.095076sec

=================(2) Fine-grained Experiment================
Experiment Info
        Total Produce Number = 10
        Final Balance Value = 0
        Execution time = 0.135737sec

=================(3) Coarse-grained Lock Experiment=================
Experiment Info
        Total Produce Number = 10
        Final Balance Value = 0
        Execution time = 0.099819sec
```

2) 차량 총 생산 수 100일 때

```
root@jimin-VirtualBox:/home/jimin/project/2021_DKU_OS/lab2_sync# ./lab2_sync -c=100 -q=1
===Vehicle Production Problem ===
==================(1) No Lock Experiment=================
Experiment Info
        Total Produce Number = 100
        Final Balance Value = 0
        Execution time = 0.991487sec

=================(2) Fine-grained Experiment================
Experiment Info
        Total Produce Number = 100
        Final Balance Value = 0
        Execution time = 1.276602sec

=================(3) Coarse-grained Lock Experiment=================
Experiment Info
        Total Produce Number = 100
        Final Balance Value = 0
        Execution time = 1.598574sec
```

3) 차량 총 생산 수 1000개일 때

```
root@jimin-VirtualBox:/home/jimin/project/2021_DKU_OS/lab2_sync# ./lab2_sync -c=1000 -q=1
===Vehicle Production Problem ===
==================(1) No Lock Experiment=================
Experiment Info
        Total Produce Number = 1000
        Final Balance Value = 0
        Execution time = 12.082478sec

==================(2) Fine-grained Experiment==================
Experiment Info
        Total Produce Number = 1000
        Final Balance Value = 0
        Execution time = 11.335536sec

==================(3) Coarse-grained Lock Experiment==================
Experiment Info
        Total Produce Number = 1000
        Final Balance Value = 0
        Execution time = 11.150991sec
```

4) 차량 수를 크게 설정할 때 segmentation fault가 발생하는 경우가 있었다.

```
==================(2) Fine-grained Experiment==================
Segmentation fault (core dumped)
root@jimin-VirtualBox:/home/jimin/project/2021_DKU_OS/lab2_sync# ./lab2_sync -c=1000 -q=1
===Vehicle Production Problem ===
==================(1) No Lock Experiment=================
Experiment Info
        Total Produce Number = 1000
        Final Balance Value = 0
        Execution time = 11.549151sec

==================(2) Fine-grained Experiment==================
Segmentation fault (core dumped)
```

4. 논의

이번 과제는 이전에 했던 과제들보다 시간을 훨씬 많이 투자한 과제였다. 머리로는 이해가 되는데 코드로 구현해내는 것이 정말 어려웠다. 그리고 이번 과제를 통해 동적할당, 포인터, 쓰레드 등 애매했던 개념에 대해 제대로 알게 되었다. 애매한 개념은 이상한 결과로 이어진다는 것을 몸소 깨달았다. 특히 강의들을 때는 lock과 쓰레드의 관계에 대해서 완벽히 이해했다고 생각했는데 전혀 아니었다. 문제 설명 피피티를 읽을 때마다, 그걸 구현할 때마다 새롭게 깨닫는 기분이었다. 덕분에 쓰레드와 lock에 대해서는 나중에 취업해도 기억할 것 같다.
한 가지 아쉬운 점은 RR로 차를 생산하는 부분을 구현해내지 못한 것이다. 저번 과제에서 한번 해봤으니까 할 수 있을 것 같았는데 그걸 실제 쓰레드와 다른 함수들과 연결하려고 하니까 너무 복잡해서 힘들었다. 다음에 시간 될 때 꼭 다시 한 번 해봐야겠다고 다짐했다.
그리고 가끔 차량 총 생산 수 인자를 크게 주면 segmentation fault가 뜰 때가 있다. 쓰레드가 스케줄링 되는 순서에 따라 달라지는 것 아닐까 생각했는데 정확한 이유는 잘 모르겠다.
작년 시스템프로그래밍과 이번 운영체제 과제를 할 때마다 느끼는 건데 안되는 코드를 알아내고 고친 후 실행했을 때 되는 것을 보면 너무 뿌듯하고 기분이 좋다. 처음 시작할 때는 막막한데 하면 할수록 알고리즘을 생각하면서 코드를 짜고 있는 나를 보면 신기하다. 물론 잘하진 않는다. 그래도 과제를 하면서 더 열심히 공부하게 되고 실력이 늘어나는 것이 느껴져서 기분이 좋다,