

2023년 2학기 스터디그룹 주간학습보고서

그룹 명	MaveOS		
날 짜	11월 8일	시 간	12시 00분 ~ 14시 45분
스터디장소	50주년 기념관 516호	회차	__8__ 회차 모임
수강과목	운영체제	담당교수	강지훈
스터디	리더: 한아림		
참석자	그룹원: 손효림, 이경화		

학습문제

<동기화>

- 배경
- 임계구역 문제
- Peterson의 해결안
- 동기화를 위한 하드웨어 지원
- Mutex Locks
- Semaphores

학습문제 해결과정

<동기화>

#1. 배경

- 협력적 프로세스는 상호 작용하며 데이터를 공유
- 프로세스는 병렬 실행되거나 중단될 수 있음
- 여러 프로세스의 데이터 동시 접근은 일관성과 무결성을 훼손
- 경쟁 상황(race condition)에서는 실행 결과가 항상 같지 않고 접근 순서에 의존

- 한 순간에 하나의 프로세스만 변수에 접근할 수 있도록 하여 일관성 유지

#2. 임계구역

1) 임계구역(Critical Section)

- 임계구역에선 하나 이상의 다른 프로세스와 공유하는 데이터에 접근하고 갱신할 수 있음
- 한 프로세스가 수행하는 동안 다른 프로세스들은 그 임계구역을 실행할 수 없음
- 진입 허가를 요청해야 진입할 수 있음
- 진입 구역(entry section), 퇴출 구역(exit section), 나머지 구역(remainder section)으로 분류

2) 임계구역 문제(해결을 위한 요구조건)

- 상호 배제(Mutual exclusion) : 하나의 프로세스가 임계구역에서 실행되면 다른 프로세스들은 자신의 임계구역에서 실행될 수 없음
- 진행(Progress) : 프로세스 중 누가 먼저 들어갈 것인가 유한 시간 내에 결정해야 함
- 한정된 대기(Bounded waiting) : 프로세스가 진입 요청을 한 후부터 요청 허용까지 유한 시간 내에 임계구역에 들어갈 수 있어야 함
- 사용 가능한 프로세스 ID를 나타내는 커널 변수 next_available_pid는 경쟁 상황 유발 → fork() 호출해 생성한 자식 프로세스 ID를 부모 프로세스에게 반환

3) 단일 코어 환경 vs 멀티 코어 환경

- 단일 코어 환경
 - 공유 변수를 수정하는 동안 인터럽트 발생을 막으면 임계구역 문제 해결 가능
 - 명령어들이 선점 없이 순서대로 실행되는 것을 보장
- 멀티 코어 환경
 - 공유 변수를 수정하는 동안 모든 코어의 인터럽트를 비활성화 해야 함
 - 자원 효율성이 떨어짐
 - 임계구역 진입 메시지를 모든 코어에 전달하는 시간 소요

#3. Peterson 해결안

1) Peterson 해결안

- 프로세스가 임계구역과 나머지 구역을 번갈아 가며 실행하는 두 개의 프로세스가 있는 환경으로 한정
- 프로세서 또는 컴파일러가 종속성 없는 읽기 및 쓰기 작업에 대한 순서를 재정렬하기 때문에 최신 컴퓨터 아키텍처에서 사용할 수 없음
- 소프트웨어 기반(software-based) 해결책

2) Peterson 해결안 코드

int turn;

boolean flag[2];

- turn 변수 : 임계구역에 들어갈 차례
- flag 변수 : 임계구역으로 진입할 준비가 되었는지 여부

#4. 하드웨어 지원

1) 하드웨어 지원

- 메모리 장벽(Memory barriers) : 후속 적재 또는 저장 연산이 수행되기 전에 모든 적재 및 저장이 완료되도록 함
- 하드웨어 명령어
 - test_and_set() - TAS: 검사와 지정
 - compare_and_swap() - CAS: 비교와 교환
- 원자적 변수(atomic variable) : 카운터가 갱신되는 동안 단일 변수에 대한 데이터 경쟁 상황에 상호 배제를 보장하는데 사용

#5. Mutex Locks

1) Mutex Locks(=spinlock)

- 임계구역 문제를 해결하는데 가장 간단하게 사용할 수 있는 도구
- 프로세스 사이의 경쟁 상태를 방지하기 위해 사용
- busy waiting을 사용해 루프를 돌며 대기한다는 단점
- CPU 시간을 낭비할 수 있는 단점
- block으로 인한 컨텍스트 스위칭이 발생하지 않음

#6. Semaphores

1) Semaphores : 두 개의 원자적 함수로 제어되는 정수. 변수를 통해 공유 자원에 대한 접근을 제어


2) Mutex vs Semaphores

- Mutex
 - Lock이 하나
 - 동기화 대상이 하나일 때
 - lock 획득한 프로세스가 lock 해제해야 다른 프로세스가 사용 가능
- Semaphores
 - Lock이 여러 개일 때
 - 동기화 대상이 1개 이상일 때

- 프로세스가 변수를 확인하고 변경 가능

#7. Liveness

- 1) Liveness : 프로세스가 실행되는 수명 주기 동안 진행(Progress)되는 것을 보장하기 위해 시스템이 충족해야 하는 속성
 - 교착 상태(Deadlock) : 둘 이상의 프로세스가 자원을 사용하기 위해 상대방을 무한정 기다리는 문제
 - 우선순위 역전 : 우선순위가 높은 프로세스가 lock으로 인해 우선순위가 낮은 프로세스를 기다려야 하는 문제
 - Liveness 실패(ex. 무한 루프) : 실패를 방지하기 위해서는 진행(Progress)과 한정된 대기(bounded waiting)를 제공해야 함

학습성찰	
학습내용 이해도	97 %
<p>학습활동 돌아보기</p> <p>(좋았던 점, 아쉬운 점 활동모습 사진 추가)</p>	<p>- 한아림</p> <p>이번 스터디 주제는 동기화였다. 스터디 초반에는 먼저 임계구역의 문제와 Peterson 해결안에 대한 학습을 진행하였다. 동기화 기능은 평소 노트북을 사용할 때 자주 접해왔던 기능인데, 그 기능이 탄생하게 된 배경과 그 필요성을 이론적으로 학습하니 감회가 새로웠다. 또 관련 개념들 중 '데이터 일관성 훼손' 문제가 동기화가 탄생하게 된 가장 큰 이유임을 배울 수 있었는데, 내가 생각했던 것보다 데이터의 무결성이 컴퓨터 시스템에서 매우 중요한 요소였음을 깨달을 수 있었다. 이어 프로세스들이 서로 버퍼를 공유하는 환경을 조성하여 데이터 생산자 역할하는 것을 확인하였다. 이 부분은 직접 코드를 구현하여 실습하였는데, 글로만 접했을 때보다 직접 코딩하여 실습해보니 이해가 더 수월하였다.</p> <p>- 손효림</p> <p>이번 스터디에서는 동기화에 대한 내용을 전반적으로 다루며 세부적으로는 임계구역 문제, peterson의 해결안, 하드웨어 지원, Mutex Locks, Semaphores 등에 대한 내용을 학습했다. 생소한 단어가 많아 쉽지 않았지만 함께 차근차근 공부한 덕분에 이해도를 높일 수 있었다. 뿐만 아니라 얼마 전 공유메모리와 메시지 큐를 연결하는 과제를 수행했는데 스터디를 열심히 한 덕분에 수고를 덜었던 듯 하다. 여러모로 유익한 점이 많다.</p> <p>- 이경화</p> <p>이번 주차에는 동기화를 주제로 공부하였다. 프로세스가 협력적으로 실행되며 데이터를 공유하는 환경에서 경쟁 상황이 발생할 수 있고, 그럴 때 실행 결과가 항상 같지 않고 접근 순서에 의존하게 된다는 것을 알게 되었다. 이를 바탕으로 임계구역이라는 개념을 공부했는데, 배경을 충분히 짚고 넘어갔기 때문에 조금 헛갈릴 수 있는 임계구역의 개념을 잘 이해할 수 있었다. Mutex Lock, Semaphores에 대해서는 코드를 살펴보고 공부했는데, 아직 코드에 대한 이해도는 높지 않은 것 같다. 다만 Mutex와 Semaphores를 비교하며 차이점을 중심으로 살펴보면서 각각의 특징에 대해서는 충분히 이해할 수 있었다. 배우면 배울수록 운영체제는 매우 섬세하게 설계되었다는 생각이 든다. 그래서 스터디원들과 함께 서로가 이해한 부분을 제시하면서 토론하는 것이 매우 흥미롭다.</p> 
다음 학습계획	일정 : 11월 15일 12:00~14:45 (50주년기념관 516호)