교차로 문제

Operating System Project#2

중앙대학교 CHUNG-ANG UNIVERSITY

Computer System Lab.

프로젝트 개요

◆ Deadlock

- 정의
  - 프로세스들이 상호간의 리소스에 대해 경쟁하면서 영구적으로 블록되는 상태

◆ Deadlock 발생 조건

- Mutual Exclusion
- Hold and wait
- No preemption
- Circular wait

◆ Deadlock 처리 방법

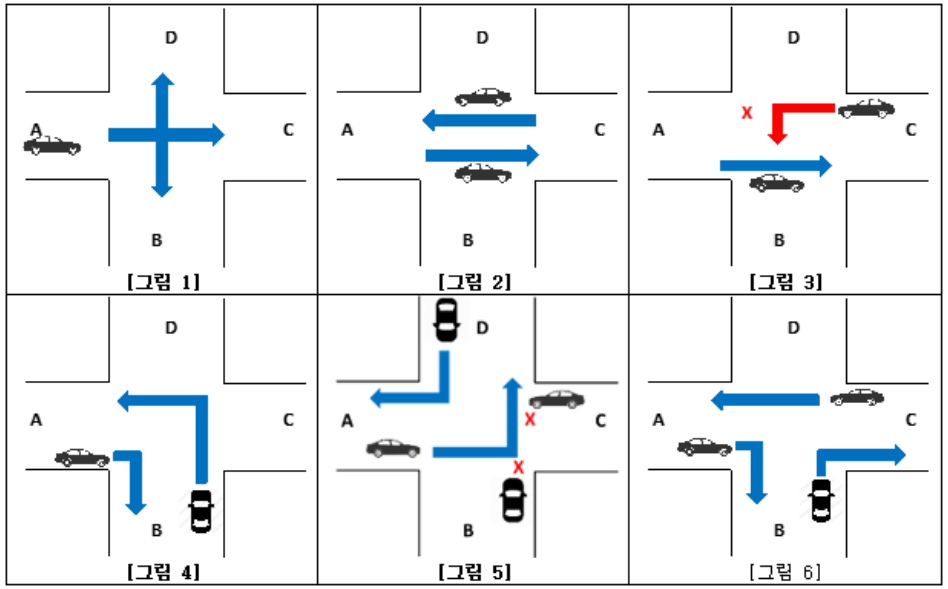
- Deadlock prevention
- Deadlock avoidance
- Deadlock detection

교차로 문제

◆ 프로젝트 개요

- 최대 한 칸씩 움직이는 차량(스레드)들이 교차로에 진입하여 출발지(A,B,C,D)에서 도착지(A,B,C,D) 까지 이동
- 교차로를 지나갈 때 Deadlock이 발생하지 않도록 해야함

X (0,0)	X (0,1)	D Src (0,2)	- (0,3)	D Dst (0,4)	X (0,5)	X (0,6)
X (1,0)	X (1,1)	(1,2)	- (1,3)	(1,4)	X (1,5)	X (1,6)
A Dst (2,0)	(2,1)	(2,2)	- (2,3)	(2,4)	(2,5)	C Src (2,6)
- (3,0)	- (3,1)	- (3,2)	X (3,3)	- (3,4)	- (3,5)	- (3,6)
A Src (4,0)	(4,1)	(4,2)	- (4,3)	(4,4)	(4,5)	C Dst (4,6)
X (5,0)	X (5,1)	(5,2)	- (5,3)	(5,4)	X (5,5)	X (5,6)
X (6,0)	X (6,1)	B Dst (6,2)	- (6,3)	B Src (6,4)	X (6,5)	X (6,6)



교차로 문제

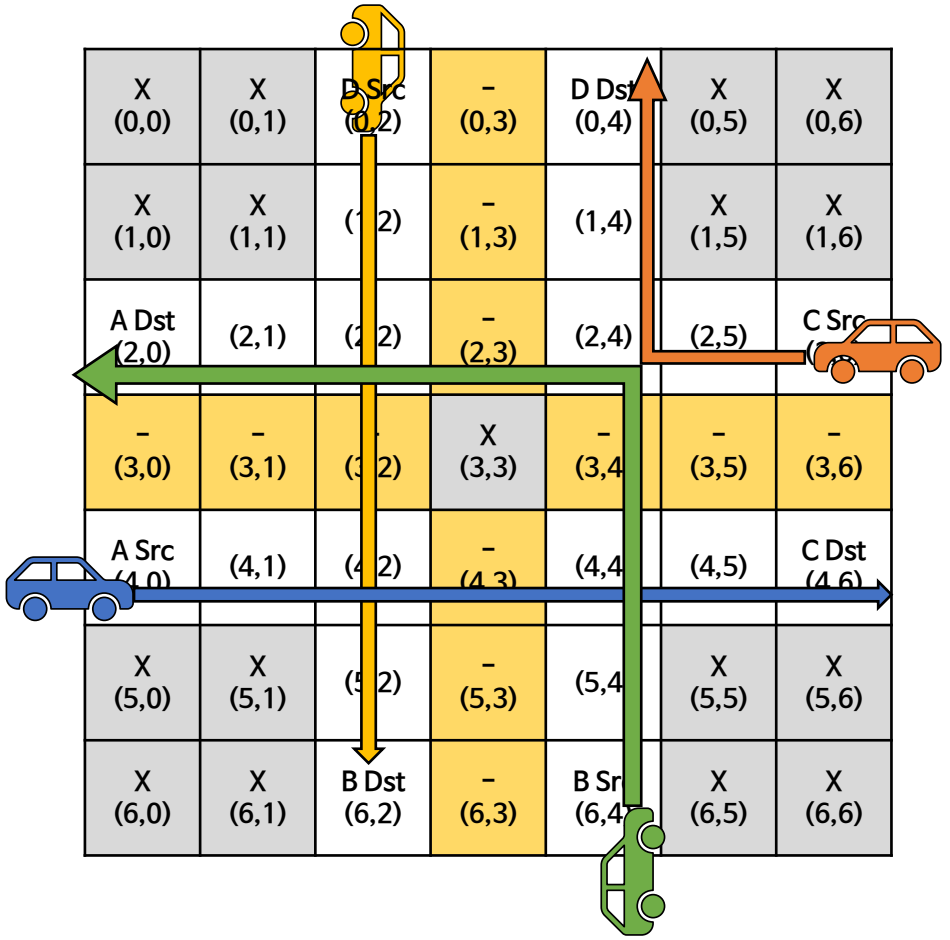
◆ 프로젝트 요구사항

- 0. Pintos 환경에서 빌드 및 실행
  - 본 프로젝트는 Pintos 기반으로 수행되어야 함
    - 소스코드 링크
      - https://drive.google.com/file/d/1Ob0Fn4DJo8rS5tsfCOUDRIHaQZLSauTO/view?usp=sharing
      - 또는 아래 명령어로 리눅스에서 다운받을 수 있음
        - wget --no-check-certificate 'https://docs.google.com/uc?export=download&id=1Ob0Fn4DJo8rS5tsfCOUDRIHaQZLSauTO' -O cau15841-pintos-qemuv2.zip
    - projects/crossroads 디렉토리의 파일을 수정 및 사용해야 하며 추가적인 파일 생성 가능
      - crossroads.c 파일의 run_crossroads 함수가 진입점
    - 개발 파일 외 다른 파일을 수정할 경우 수정한 근거를 레포트로 작성
      - **ats.h/.c 및 crossroads.h/.c 파일은 수정 절대 불가**
  - 본 프로젝트 실행 명령어 및 매개변수는 다음과 같음
    - ../../utils/pintos crossroads aAA:bBD:cCD:dDB
      - a 차량이 A에서 출발, A 도착
      - b 차량이 B에서 출발, D 도착
      - c 차량이 C에서 출발, D 도착
      - d 차량이 D에서 출발, B 도착

교차로 문제

◆ 프로젝트 요구사항

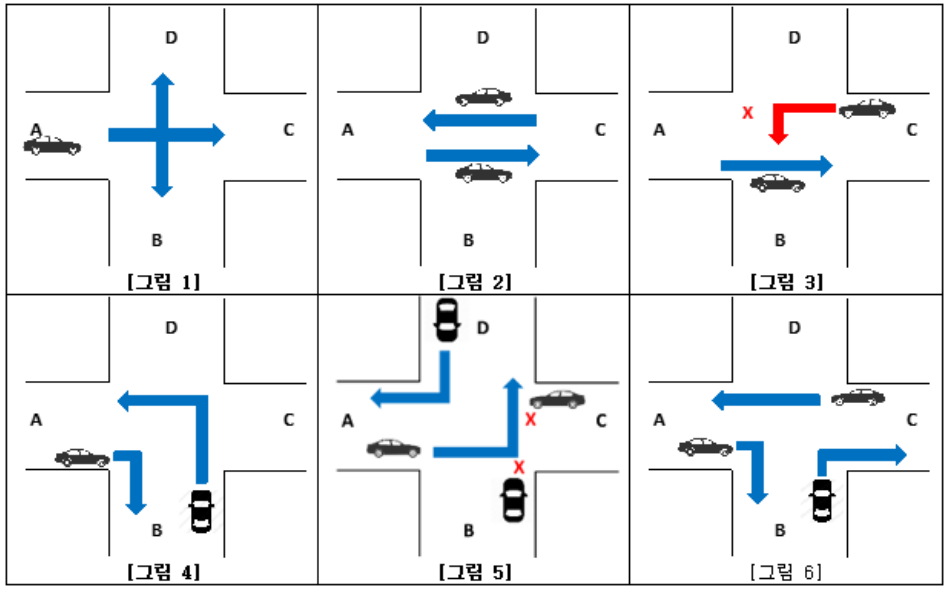
- 1. 차량이 교차로에서 Deadlock이 발생하지 않도록 해야함
  - 7x7 크기의 맵이 주어짐
    - 각 칸은 한번에 하나의 차량만이 이동이 가능
    - X칸은 진입 불가
  - 초기 입력된 매개변수에 따라 차량 스레드들이 생성 및 실행됨
  - 각 차량은 A,B,C,D에서 시작하여 A,B,C,D 중 한 곳으로 이동함
    - 우측통행을 전제로 하며 각 차량이 이동할 경로는 vehicle.c 에 정의되어 있음
      - 경로는 수정 불가
      - 각 차량은 출발지와 도착지가 일치 할 수 있음
        - ex) A->A일 경우 교차로를 돌아서 이동
    - 각 차량은 단위 스텝 당 최대 한 칸 전진할 수 있음



교차로 문제

◆ 프로젝트 요구사항

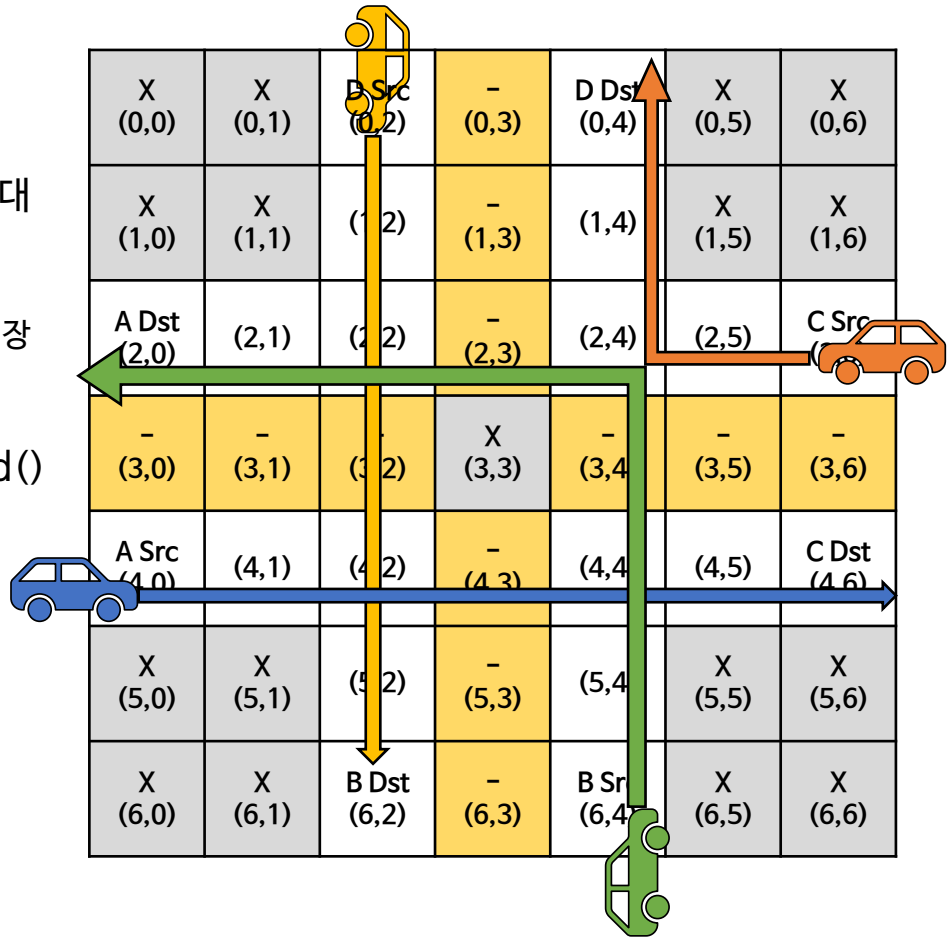
- 1. 차량이 교차로에서 Deadlock이 발생하지 않도록 해야함
  - 각 차량 스레드가 교차로에 진입하는 방식을 제어하여 잠재적으로 발생 가능한 Deadlock 문제를 해결해야 함
    - Deadlock prevent, avoidance 등의 방식을 사용
  - 교차로 통과시 [그림2], [그림4], [그림6] 같은 케이스는 가능
    - 동시에 같은 칸을 지나가는 케이스가 아닐 경우 가능
  - [그림 3], [그림 5] 같은 케이스는 불가능
    - D->A와 A->D는 동시에 진입 가능하지만 B->D, C->A 차량이 동시에 진입할 경우 충돌



교차로 문제

◆ 프로젝트 요구사항

- 1. 차량이 교차로에서 Deadlock이 발생하지 않도록 해야함
  - 각 차량 스레드는 단위 스텝 당 최대 한 칸 전진할 수 있도록 제한하되, 모든 차량이 최대 한 칸 전진할 때까지 단위 스텝이 증가하지 않아야 함
    - 반드시 이동할 수 있는 모든 차량이 이동한 후에 단위스텝을 증가시켜야 함
    - 단위 스텝은 반드시 'crossroads.h' 헤더 파일에 정의된 'int crossroads_step' 변수에 저장되어야 함
    - 'int crossroads_step' 변수는 항상 증가하여야 하며, 감소할 수 없음
  - 'int crossroads_step' 변수가 변경될 때 마다 'ats.h'에 정의된 unitstep_changed() 함수를 호출하여야 하며, 이 함수는 모든 스레드로부터 보호되어야 함
    - unitstep_changed()는 공유 리소스에 대한 접근보호가 보장되어야 함
  - 작성한 코드에서 임의의 시간대기가 발생하지 않도록 해야함
    - timer_msleep() 과 같은 방식을 사용하지 말 것



교차로 문제

◆ 프로젝트 요구사항

- 2. 과제 제출 요구사항
  - 보고서 요구사항
    - 교차로 문제를 해결하기 위해 사용한 기법 및 결과 설명
  - 결과물 제출 요구사항
    - 프로젝트 결과물 및 보고서 압축 후 이클래스로 제출
      - make clean 후 제출 필수

교차로 문제

◆ 과제 관련 문의

- 조민규 조교 (mgjo@cslab.cau.ac.kr)
- 연구실 위치 : 208-530