운영체제 프로젝트2 결과보고서

수 강 과 목 : 운영체제
담 당 교 수 : 엄영익 교수님
학 과 : 소프트웨어학과
학 번 : 2020315798
이 름 : 최진우
제 출 일 : 2022년 10월 28일

1. 프로젝트 개요

1) 목표 : Deadlock Avoidance 구현
2) 세부 목표
    - Max-claim 행렬과 Current-alloc 행렬을 기준으로 deadlock avoidance 구현

2. 프로젝트 개발 환경

1) 개발 언어 : C
2) 개발 환경 : Ubuntu 18.04.5 LTS (성균관대학교 인의예지 클러스터)
3) 컴파일러 : gcc 7.5.0
4) 사용 프로그램 및 목적
    · Putty : 컴파일 및 vi에디터
    · FileZilla : SFTP를 이용한 백업 및 테스트 케이스 전송
    · EditPlus5 : 코드 작성
5) 라이브러리 및 목적
    · stdio.h : 표준 출력 및 파일 접근
    · stdlib.h : 동적 할당
    · string.h : input txt 파일의 띄어쓰기 토큰화

5. 프로젝트 설계 및 알고리즘

1) 설계 흐름도

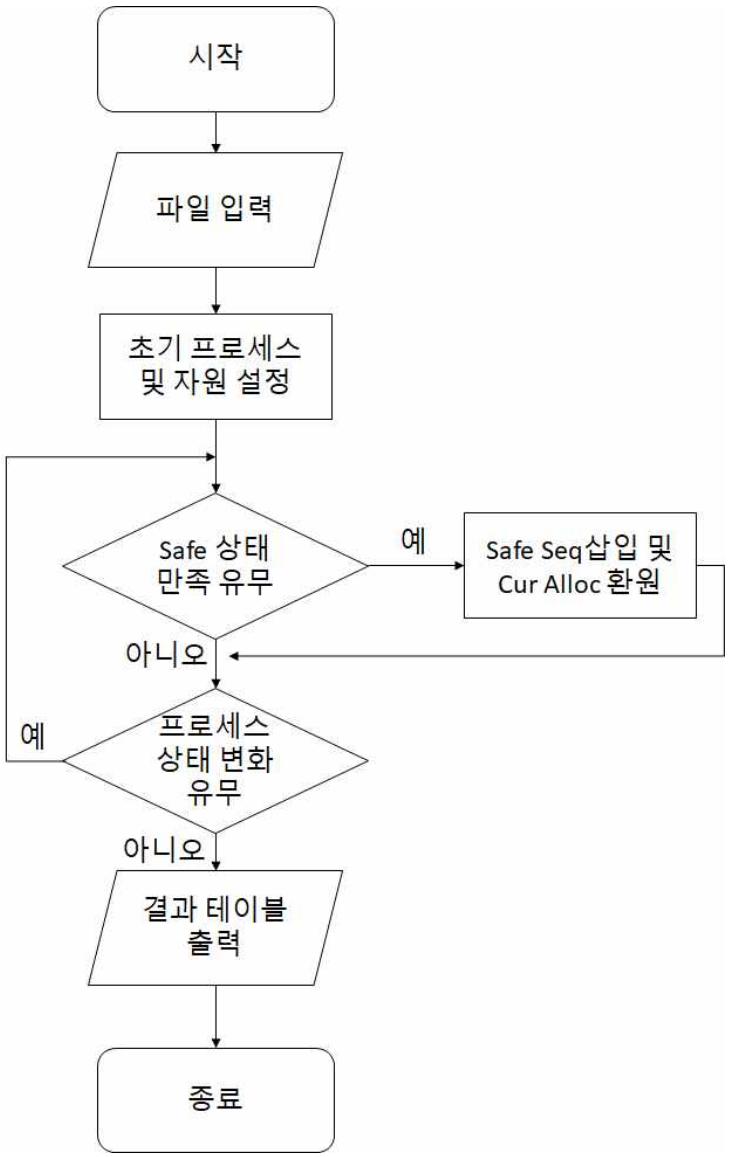
2) 설계/알고리즘 설명

강의를 통해 접한 Dijkstra와 Habermann의 알고리즘을 토대로 현재 주어진 상황이 Safe인지 Unsafe인지를 판별하는 프로그램을 구현했습니다. 과제안내서에 제시된 샘플 인풋 파일과 같은 형식으로 파일 입력을 받아 프로세스 수, 자원 종류 수, 자원 종류별 유닛 수, 프로세스별 자원 최대 요구량 등 문제 해결을 위한 배열을 동적으로 할당받습니다.

프로세스 1번부터 n번까지 입력 txt 파일의 정보를 불러와 현재 할당 가능한 여유 유닛 수와 프로세스별 필요(Additional need) 유닛 수를 자원 종류별로 비교해, 모두 만족할 경우 해당 프로세스는 Safe Sequence에 넣습니다. 또한, 해당 프로세스가 현재 할당받고 있던 유닛 개수(Cur. Alloc.)에 환원하여 다른 프로세스를 찾습니다.

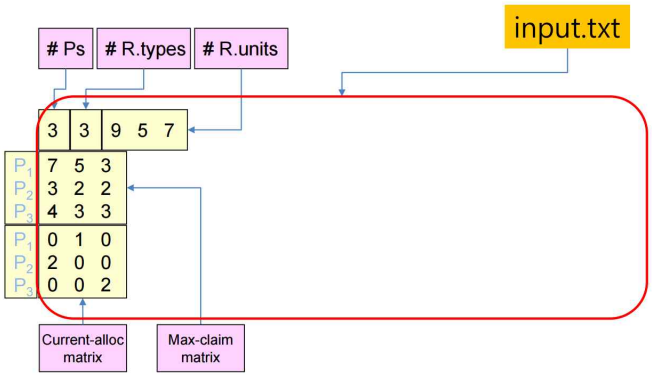
이번에 구현한 프로그램에선 루프마다 Safe 조건을 만족하는 하나의 프로세스 단위로 검사를 진행하며 어떤 프로세스가 Safe Sequence에 들어갔는지 전체 자원 유닛 정보 테이블과 함께 출력합니다. 또한, 전체적으로 Safe 여부가 결정된 후에는 Safe일 경우 Safe Sequence를, Unsafe일 경우 어떤 프로세스는 Safe 하지만 어떤 프로세스 때문에 Unsafe인지 각각 리스트를 출력합니다. 만약 동시에 여러 프로세스가 Safe 상태가 되는 상황이라면, 번호가 낮은 프로세스부터 먼저 해결하는 것을 원칙으로 합니다.

3) 알고리즘 흐름도


시작
파일 입력
초기 프로세스
및 자원 설정
Safe 상태
만족 유무
예
Safe Seq삽입 및
Cur Alloc 환원
아니오
프로세스
상태 변화
유무
예
아니오
결과 테이블
출력
종료

5. 프로젝트 결과 정책

1) input.txt



2022-2 운영체제 과제2 안내서

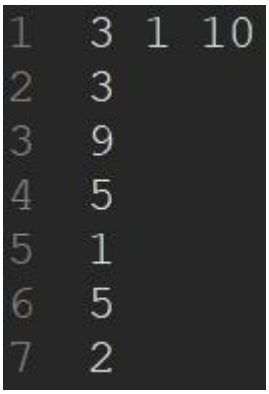
2) 콘솔 결과

· 입력 데이터에 대한 초기 상태 출력
· 루프 단위로 Safe 상태로 변한 프로세스 존재 시 테이블 출력
· 진행 후 프로세스 상태 변화가 없으면 결과 출력
· Safe일 경우 : Safe Sequence 출력
· Unsafe일 경우 :
    1) Safe 상태에 놓인 프로세스 별도 출력
    2) 어떤 프로세스 때문에 Unsafe인지 리스트 출력

```
>> Initial State
+-----------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-----------------------------------------------------+
| Process  1 |  8  5  3   |  0  1  0    |  8  4  3  |
| Process  2 |  3  2  2   |  2  0  0    |  1  2  2  |
| Process  3 | 11  0  2   |  3  0  2    |  8  0  0  |
| Process  4 |  2  2  2   |  2  1  1    |  0  1  1  |
| Process  5 |  4  3  7   |  0  0  2    |  4  3  5  |
+-----------------------------------------------------+
| Total          Unit #  :  10  5  7                  |
+-----------------------------------------------------+
| Available      Unit #  :   3  3  2                  |
+-----------------------------------------------------+

>> Loop 1 : Process 2 became safe
+-----------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-----------------------------------------------------+
| Process  1 |  8  5  3   |  0  1  0    |  8  4  3  |
| Process  2 |  3  2  2   |  0  0  0    |  0  0  0  |
| Process  3 | 11  0  2   |  3  0  2    |  8  0  0  |
| Process  4 |  2  2  2   |  2  1  1    |  0  1  1  |
| Process  5 |  4  3  7   |  0  0  2    |  4  3  5  |
+-----------------------------------------------------+
| Total          Unit #  :  10  5  7                  |
+-----------------------------------------------------+
| Available      Unit #  :   5  3  2                  |
+-----------------------------------------------------+

>> Loop 2 : Process 4 became safe
+-----------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-----------------------------------------------------+
| Process  1 |  8  5  3   |  0  1  0    |  8  4  3  |
| Process  2 |  3  2  2   |  0  0  0    |  0  0  0  |
| Process  3 | 11  0  2   |  3  0  2    |  8  0  0  |
| Process  4 |  2  2  2   |  0  0  0    |  0  0  0  |
| Process  5 |  4  3  7   |  0  0  2    |  4  3  5  |
+-----------------------------------------------------+
| Total          Unit #  :  10  5  7                  |
+-----------------------------------------------------+
| Available      Unit #  :   7  4  3                  |
+-----------------------------------------------------+

>> Result
UNSAFE CONDITION

>> Safe Process List
Process 2
Process 4

>> Unsafe Process List
Process 1
Process 3
Process 5
```

6. 프로젝트 결과 케이스 분석

1) 단일 자원 : Safe 일 경우(OS07-S28-37의 6번 슬라이드 예시)




```
>> Initial State
+-------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-------------------------------------------------+
| Process  1 |  3   |  1   |  2  |
| Process  2 |  9   |  5   |  4  |
| Process  3 |  5   |  2   |  3  |
+-------------------------------------------------+
| Total         Unit #  :  10
+-------------------------------------------------+
| Available     Unit #  :   2
+-------------------------------------------------+

>> Loop 1 : Process 1 became safe
+-------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-------------------------------------------------+
| Process  1 |  3   |  0   |  0  |
| Process  2 |  9   |  5   |  4  |
| Process  3 |  5   |  2   |  3  |
+-------------------------------------------------+
| Total         Unit #  :  10
+-------------------------------------------------+
| Available     Unit #  :   3
+-------------------------------------------------+

>> Loop 2 : Process 3 became safe
+-------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-------------------------------------------------+
| Process  1 |  3   |  0   |  0  |
| Process  2 |  9   |  5   |  4  |
| Process  3 |  5   |  0   |  0  |
+-------------------------------------------------+
| Total         Unit #  :  10
+-------------------------------------------------+
| Available     Unit #  :   5
+-------------------------------------------------+

>> Loop 3 : Process 2 became safe
+-------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-------------------------------------------------+
| Process  1 |  3   |  0   |  0  |
| Process  2 |  9   |  0   |  0  |
| Process  3 |  5   |  0   |  0  |
+-------------------------------------------------+
| Total         Unit #  :  10
+-------------------------------------------------+
| Available     Unit #  :  10
+-------------------------------------------------+

>> Result
SAFE CONDITION

>> Safe Sequence
Process 1 -> Process 3 -> Process 2
```

운영체제 강의자료 OS07-S28-37의 6번 슬라이드에 나온 예시 케이스입니다.
단일 종류 자원의 경우로 Safe Condition을 만족하는 경우입니다. 초기 상태부터 루프마다 상태가 변한 프로세스가 있으면 출력합니다. 최종적으로 Safe Sequence를 출력합니다.
출력 테이블은 자원의 종류가 3개일 때로 최적화되어 있으므로 스타일이 어긋나게 출력되어 있습니다.

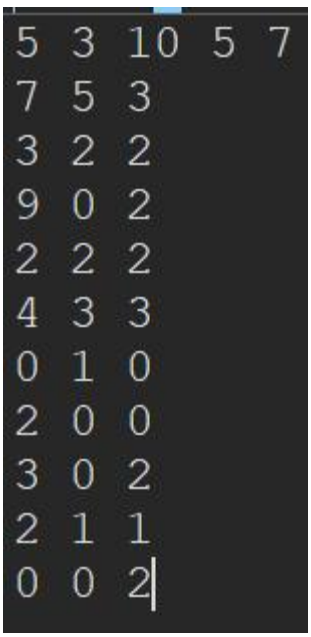
2) 단일 자원 : Unsafe 일 경우(OS07-S28-37의 7번 슬라이드 예시)

```
3 1 10
5
9
7
1
5
2
```

```
>> Initial State
+-------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-------------------------------------------------+
| Process  1 |  5   |  1    |  4  |
| Process  2 |  9   |  5    |  4  |
| Process  3 |  7   |  2    |  5  |
+-------------------------------------------------+
| Total         Unit #  :  10
+-------------------------------------------------+
| Available     Unit #  :   2
+-------------------------------------------------+

>> Result
UNSAFE CONDITION

>> Unsafe Process List
Process 1
Process 2
Process 3
```

운영체제 강의자료 OS07-S28-37의 7번 슬라이드에 나온 예시 케이스입니다.
단일 종류 자원의 경우로 Safe Condition을 만족하지 못하는는 경우입니다. 최종적으로 Safe 상태로 변한 프로세스와 어떤 프로세스 때문에 Unsafe인지(Unsafe Process List)를 출력합니다.
출력 테이블은 자원의 종류가 3개일 때로 최적화되어 있으므로 스타일이 어긋나게 출력되어 있습니다.

3) 여러 종류의 자원 : Safe 일 경우 (OS07-S38-42의 3번 슬라이드 예시)




```
| Available     Unit #  :   5  3  2
+-------------------------------------------------+

>> Loop 2 : Process 4 became safe
+-------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-------------------------------------------------+
| Process  1 |  7  5  3   |  0  1  0    |  7  4  3  |
| Process  2 |  3  2  2   |  0  0  0    |  0  0  0  |
| Process  3 |  9  0  2   |  3  0  2    |  6  0  0  |
| Process  4 |  2  2  2   |  0  0  0    |  0  0  0  |
| Process  5 |  4  3  3   |  0  0  2    |  4  3  1  |
+-------------------------------------------------+
| Total         Unit #  :  10  5  7
+-------------------------------------------------+
| Available     Unit #  :   7  4  3
+-------------------------------------------------+

>> Loop 3 : Process 1 became safe
+-------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-------------------------------------------------+
| Process  1 |  7  5  3   |  0  0  0    |  0  0  0  |
| Process  2 |  3  2  2   |  0  0  0    |  0  0  0  |
| Process  3 |  9  0  2   |  3  0  2    |  6  0  0  |
| Process  4 |  2  2  2   |  0  0  0    |  0  0  0  |
| Process  5 |  4  3  3   |  0  0  2    |  4  3  1  |
+-------------------------------------------------+
| Total         Unit #  :  10  5  7
+-------------------------------------------------+
| Available     Unit #  :   7  5  3
+-------------------------------------------------+

>> Loop 4 : Process 3 became safe
+-------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-------------------------------------------------+
| Process  1 |  7  5  3   |  0  0  0    |  0  0  0  |
| Process  2 |  3  2  2   |  0  0  0    |  0  0  0  |
| Process  3 |  9  0  2   |  0  0  0    |  0  0  0  |
| Process  4 |  2  2  2   |  0  0  0    |  0  0  0  |
| Process  5 |  4  3  3   |  0  0  2    |  4  3  1  |
+-------------------------------------------------+
| Total         Unit #  :  10  5  7
+-------------------------------------------------+
| Available     Unit #  :  10  5  5
+-------------------------------------------------+

>> Loop 5 : Process 5 became safe
+-------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-------------------------------------------------+
| Process  1 |  7  5  3   |  0  0  0    |  0  0  0  |
| Process  2 |  3  2  2   |  0  0  0    |  0  0  0  |
| Process  3 |  9  0  2   |  0  0  0    |  0  0  0  |
| Process  4 |  2  2  2   |  0  0  0    |  0  0  0  |
| Process  5 |  4  3  3   |  0  0  0    |  0  0  0  |
+-------------------------------------------------+
| Total         Unit #  :  10  5  7
+-------------------------------------------------+
| Available     Unit #  :  10  5  7
+-------------------------------------------------+

>> Result
SAFE CONDITION

>> Safe Sequence
Process 2 -> Process 4 -> Process 1 -> Process 3 -> Process 5
```

4) 여러 종류의 자원 : Unsafe 일 경우 (OS07-S38-42의 5번 슬라이드 예시)

```
5 3 10 5 7
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
0 4 0
2 0 0
3 0 2
2 1 1
0 0 2
```

```
>> Initial State
+-----------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-----------------------------------------------------+
| Process  1 |  7  5  3   |  0  4  0    |  7  1  3  |
| Process  2 |  3  2  2   |  2  0  0    |  1  2  2  |
| Process  3 |  9  0  2   |  3  0  2    |  6  0  0  |
| Process  4 |  2  2  2   |  2  1  1    |  0  1  1  |
| Process  5 |  4  3  3   |  0  0  2    |  4  3  1  |
+-----------------------------------------------------+
| Total          Unit #  :  10  5  7                  |
+-----------------------------------------------------+
| Available      Unit #  :   3  0  2                  |
+-----------------------------------------------------+

>> Result
UNSAFE CONDITION

>> Unsafe Process List
Process 1
Process 2
Process 3
Process 4
Process 5
```

5) 여러 종류의 자원 : Unsafe 일 경우 (Safe 프로세스 일부 포함)

```
5 3 10 5 7
8 5 3
3 2 2
11 0 2
2 2 2
4 3 7
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
```

```
>> Initial State
+-------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-------------------------------------------------+
| Process  1 |  8  5  3   |  0  1  0    |  8  4  3  |
| Process  2 |  3  2  2   |  2  0  0    |  1  2  2  |
| Process  3 | 11  0  2   |  3  0  2    |  8  0  0  |
| Process  4 |  2  2  2   |  2  1  1    |  0  1  1  |
| Process  5 |  4  3  7   |  0  0  2    |  4  3  5  |
+-------------------------------------------------+
| Total         Unit #  :  10  5  7               |
+-------------------------------------------------+
| Available     Unit #  :   3  3  2               |
+-------------------------------------------------+

>> Loop 1 : Process 2 became safe
+-------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-------------------------------------------------+
| Process  1 |  8  5  3   |  0  1  0    |  8  4  3  |
| Process  2 |  3  2  2   |  0  0  0    |  0  0  0  |
| Process  3 | 11  0  2   |  3  0  2    |  8  0  0  |
| Process  4 |  2  2  2   |  2  1  1    |  0  1  1  |
| Process  5 |  4  3  7   |  0  0  2    |  4  3  5  |
+-------------------------------------------------+
| Total         Unit #  :  10  5  7               |
+-------------------------------------------------+
| Available     Unit #  :   5  3  2               |
+-------------------------------------------------+

>> Loop 2 : Process 4 became safe
+-------------------------------------------------+
| Process-ID | Max. Claim | Cur. Alloc. | Add. Need |
+-------------------------------------------------+
| Process  1 |  8  5  3   |  0  1  0    |  8  4  3  |
| Process  2 |  3  2  2   |  0  0  0    |  0  0  0  |
| Process  3 | 11  0  2   |  3  0  2    |  8  0  0  |
| Process  4 |  2  2  2   |  0  0  0    |  0  0  0  |
| Process  5 |  4  3  7   |  0  0  2    |  4  3  5  |
+-------------------------------------------------+
| Total         Unit #  :  10  5  7               |
+-------------------------------------------------+
| Available     Unit #  :   7  4  3               |
+-------------------------------------------------+

>> Result
UNSAFE CONDITION

>> Safe Process List
Process 2
Process 4

>> Unsafe Process List
Process 1
Process 3
Process 5
```

7. 프로젝트 오류 대응

1) 파일이 없는 경우

```
2020315798@swui:/home/2020315798/2022-2/OS/proj2$ ./m
Failed to open input file.
```

2) 입력 파일의 행 수가 프로세스 수 * 2 + 첫 줄을 만족하지 못하는 경우

```
2020315798@swui:/home/2020315798/2022-2/OS/proj2$ ./m exception1.txt
Exception : Input value is incorrect.
```

3) Max Claim, Cur. Alloc. 등 프로세스 및 자원 수가 음수인 경우

```
2020315798@swui:/home/2020315798/2022-2/OS/proj2$ ./m exception2.txt
Exception : Number of additional need cannot be a negative value.
```