운영체제 Lab02

2015140124
전자공학과
진우빈

1. 프로그램의 구조

Instance, allocations, max 배열을 만들어 각각 자원 개수 만큼의 값을 입력 받습니다.
Availables, needs 배열에는 입력 받은 값에 따라 자동으로 계산되어 값이 들어가게 됩니다.
만약 Allocation 이나 Max의 값이 잘못 입력되었을 시 함수를 종료합니다.
이후 가능한 sequence가 있는 지 계산하는 함수를 통해 Safety Sequence의 종류를 출력합니다.

2. 사용 함수

print_safe_sequence(allocations, needs, availables, sequence):
입력 받은 값을 통해 생성된 allocations needs, available 배열을 parameter로 받습니다. 프로세스의 개수, instance의 개수 만큼 for문을 돌면서 safe한 sequence가 있는 지 탐색합니다. safe 할 시 sequence배열에 추가로 해당 프로세스를 더한 후 재귀 함수를 통해 반복하게 됩니다. 만약 총 프로세스의 개수만큼 sequence배열이 채워지면 Safety한 sequence란 뜻이므로 해당 sequence 배열을 return하게 됩니다.

3. 실행 화면

아래 사진을 예시로 하여 진행 하였습니다.

- 5 processes P_0 through P_4
  - ✓ 3 resource types *A* (10 instances), *B* (5 instances), and *C* (7 instances)

- Snapshot at time T_0:

	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

<Process 및 Resource의 개수, 각각 Instance, Allocation, Max 입력 받기>

```
====== Process 갯수 입력 ======
process 갯수를 입력해주세요(2~10) : 5

====== Resource 갯수 입력 ======
resource 갯수를 입력해주세요(2~5) : 3
====== Instance 입력 ======
1번째 자원의 instance 갯수를 입력해주세요(0~99) : 10
2번째 자원의 instance 갯수를 입력해주세요(0~99) : 5
3번째 자원의 instance 갯수를 입력해주세요(0~99) : 7

====== Allocation 입력 ======
process0의 자원을 할당해주세요(띄어쓰기로 구분 ex) 0 1 2 ...) : 0 1 0
process1의 자원을 할당해주세요(띄어쓰기로 구분 ex) 0 1 2 ...) : 2 0 0
process2의 자원을 할당해주세요(띄어쓰기로 구분 ex) 0 1 2 ...) : 3 0 2
process3의 자원을 할당해주세요(띄어쓰기로 구분 ex) 0 1 2 ...) : 2 1 1
process4의 자원을 할당해주세요(띄어쓰기로 구분 ex) 0 1 2 ...) : 0 0 2

====== Max 입력 ======
process0의 자원의 최대치를 입력해주세요(띄어쓰기로 구분 ex) 0 1 2 ...) : 7 5 3
process1의 자원의 최대치를 입력해주세요(띄어쓰기로 구분 ex) 0 1 2 ...) : 3 2 2
process2의 자원의 최대치를 입력해주세요(띄어쓰기로 구분 ex) 0 1 2 ...) : 9 0 2
process3의 자원의 최대치를 입력해주세요(띄어쓰기로 구분 ex) 0 1 2 ...) : 2 2 2
process4의 자원의 최대치를 입력해주세요(띄어쓰기로 구분 ex) 0 1 2 ...) : 4 3 3
```

1. 프로세스와 자원의 개수를 입력 받습니다.
2. 입력 받은 자원의 개수만큼 Instance 개수를 입력 받습니다.
3. 입력 받은 프로세스의 개수만큼 Allocation을 입력 받습니다.
4. 입력 받은 프로세스의 개수만큼 Max를 입력 받습니다.

<입력받은 정보들을 출력>

```
[출력]
====== Instance ======
Instances : 10 5 7
Available : 3 3 2

====== Allocation ======
process0 : 0 1 0
process1 : 2 0 0
process2 : 3 0 2
process3 : 2 1 1
process4 : 0 0 2

====== Max ======
process0 : 7 5 3
process1 : 3 2 2
process2 : 9 0 2
process3 : 2 2 2
process4 : 4 3 3

====== Need ======
process0 : 7 4 3
process1 : 1 2 2
process2 : 6 0 0
process3 : 0 1 1
process4 : 4 3 1
```

입력받은 정보에 따라 Instance, Allocation, Max, Need를 Process 별로 출력합니다.

<가능한 Saftey Sequence를 모두 출력>

```
====== 가능한 safe sequence 종류들 ======
시작 => process1 => process3 => process0 => process2 => process4 => 종료
시작 => process1 => process3 => process0 => process4 => process2 => 종료
시작 => process1 => process3 => process2 => process0 => process4 => 종료
시작 => process1 => process3 => process2 => process4 => process0 => 종료
시작 => process1 => process3 => process4 => process0 => process2 => 종료
시작 => process1 => process3 => process4 => process2 => process0 => 종료
시작 => process1 => process4 => process3 => process0 => process2 => 종료
시작 => process1 => process4 => process3 => process2 => process0 => 종료
시작 => process3 => process1 => process0 => process2 => process4 => 종료
시작 => process3 => process1 => process0 => process4 => process2 => 종료
시작 => process3 => process1 => process2 => process0 => process4 => 종료
시작 => process3 => process1 => process2 => process4 => process0 => 종료
시작 => process3 => process1 => process4 => process0 => process2 => 종료
시작 => process3 => process1 => process4 => process2 => process0 => 종료
시작 => process3 => process4 => process1 => process0 => process2 => 종료
시작 => process3 => process4 => process1 => process2 => process0 => 종료
```

가능한 Safety Sequence를 모두 출력하고 종료합니다.

Jupyter notebook 환경에서 실행하였습니다. 따라서 ipynb파일로 소스코드를 첨부하였지만 다른 환경에서 실행할 수 있게끔 py파일도 추가로 첨부하였습니다.