운영체제 Lab1

32203660 이현정

1) 목표

스케줄링에 사용하는 알고리즘인 FCFS, SPN, HRRN, RR(q=1, 4), FEEDBACK(q=1, 2i)를 구현해 보는 것을 목표로 하고 있으며, 해당하는 알고리즘에 따라 어떠한 방식으로 프로세서들이 동작하는 출력을 통해 확인을 해보려고 합니다.

2) 설계 과정

각 프로세스는 도착시각, 수행시간의 배열을 통해 받는다고 것이 필요하다고 생각하였습니다. 기본적으로 {도착시간, 수행시간}의 배열 값들이 배열 내에 들어오는 이중 배열 구조하고 생각을 하였고, 추가로 각 프로세스의 동작을 조정과 초기화를 위해서 해당 배열의 맨 앞에 {0,0}를 추가해서 {{0,0,},{0,2}, ...} 형태로 프로세스에 대한 정보를 받아서 알고리즘마다 해당 함수가 돌아가게 구현해보았습니다.

각 실행마다 원하는 알고리즘에 따라서 다르게 동작하기 위해서 실행 입력을 받으면 해당 알고리즘의 이름을 통해서 받는 것으로 하였으며, 해당 내용으로는 FCFS, RR, SPN, HRRN, feedback 함수가 있습니다. 모든 함수는 각 프로세스가 언제 수행되는지 확인의 출력을 위해서 해당 프로세스가 돌아갈 때에는 result 함수의 각 프로세스 순서를 1~N까지 표현한 인덱스의 해당 위치에 들어가게 구현하였습니다. 기본적으로 for 문의 i로 시간마다 프로세스 j가 언제 동작하는지 확인하는 방식입니다.

FCFS : for문의 i라는 시간을 기준으로 현재보다 이전에 도착하고 아직 동작해야 할 시간이 남았으며, 현재 동작할 프로세스가 없는 경우(state == 0)는 가장 빠르게 도착한 프로세스의 번호가 state에 들어가서 결과를 담을 result 배열에서 해당 프로세스의 시간 부분에 1을 담고 수행 시간을 1줄입니다. 이러한 내용은 마지막으로 들어오는 프로세스가 마지막에 끝나므로, 마지막 프로세스의 수행시간이 0일 때 종료하게 구현하였습니다.

SPN : 기본적이니 동작은 위와 비슷하며, 동시에 도착해도 짧은 실행시간을 가진 것이 먼저 끝날 수 있도록 시간마다 남은 시간이 가장 짧은 것을 선택 후 해당 시간을 준 만큼을 차지할 수 있도록 하였습니다. 맨 끝 프로세스가 반드시 마지막에 끝나지 않으므로 모든 프로세스의 종료 확인을 위한 time_end로 끝난 프로세서 개수 확인을 진행하였습니다.

HRRN : (기다리는 시간 + 수행 시간) / 수행 시간이 클수록 더 높은 우선순위를 가지게 되므로 현재 시각에 사용되는 프로세스(state)가 가장 먼저 만난 값을 넣은 뒤에 각 시간마다 해당 기준 값에 대해서 비교한 후에 더 큰 값을 가지는 프로세스의 번호를 state에 넣어서 먼저 돌아갈 수 있도록 구현하였습니다. 수행시간만큼 해당 프로세스가 수행돼야 한다는 측면에 대해서는 while 문으로 수행시간만큼 돌아갔으면 현재 시간 i을 그만큼 늘리고 해당 범위 동안은 결과에 값이 들어감과 동시에 수행 시간이 줄어들게 하였습니다. 모든 프로세스가 종료했는지는 sum에 값을 누적시켜서 해당하는 개수와 같은지 확인하게 하였습니다.

RR : 각 프로세스가 언제 도착하는지 뿐만 아니라 처리가 되어서 다시 자신의 차례를 기다리는 것까지를 고려해야 하므로 order 배열을 이용해서 각 프로세스가 들어와서 기다리게 될때마다 해당 값이 증가하는 방식으로 만들었습니다. 먼저 조금 전에 완료된 프로세스는 새로 들어오는 프로세스보다 늦게 들어온다는 가정 속에서 last 변수의 값을 1로 변경시켜서 추후 처리가 되도록 프로세스의 순번이 결정되게 하였고, 그다음에 현재 동작해야 하는 프로세스 범위에 존재하는 경우에는 order에 값이 가장 작은 것부터 동작하게 될 수 있도록 처음 만난 범위에 존재하는 값을 넣었는데, 더 작은 값이 뒤에서 나온다면 해당 값으로 현재 시간에 사용되는 프로세스(state)가 변경되도록 하였습니다. state 값이 존재하면 해당 값에 대해서 result에 기록되게 되며, 주어진 Time Quantum에 따라 동작하기 위해서 t_use를 이용해서 직전에 사용한 프로세스가 얼마큼 동작했는지 확인해서 더 사용을 해야 한다면 동작할 수 있게 하였으며, 모든 프로세스의 수행시간이 남아있지 않았을 때 종료됩니다.

FEEDBACK : 사용되지 않은 경우에는 가진 순서의 값을 유지해야 하는 부분에서 다 완성하지 못했습니다.각 q의 값으로 주어지는 것에 따라서 1이면 1번만 해당 프로세스가 사용되고 넘어가게 하였고, 2i 값을 가지면 프로세스가 몇 번째 큐에 있는지에 따라서 2^(큐 위치 : 0 ~)으로 해결을 하면서, 주어지는 인자에 따라서 다르게 동작하는 1개의 함수로 만들어보려고 하였습니다. 순서를 Q마다 계산해야 하므로 order를 2차원 배열로 만들어서 초기화를 100으로 진행합니다. 먼저 순서를 확인하기 위해서 현재 큐에 들어가 있는 프로세스인지 확인하면서 새로운 프로세스이면 첫 번째 큐에 담고, 직전에 사용했다면 자신이 있던 위치의 값을 200으로 변화시킨 다음에 아직 수행시간이 남아있으면 그 다음 Q로 이동해서 cnt에 담겨있는 q마다 돌아야 하는 순번을 입력하게 하였습니다. 그리고 현재 돌게 될 프로세스를 찾기 위해서 order를 맨 위 큐의 위치부터 비교하면서 만약 가장 작은 값을 발견하면 state에 저장하는데, 같은 큐 중에서 더 작은 값이 있으면 그것을 선택하게 구현하였으며, 가장 높은 큐에서 점점 낮은 큐로 돌아가게 구성하였습니다.

3) 결과1 : 스크린샷

p.24의 예시자료를 이용한 출력을 해보았습니다.

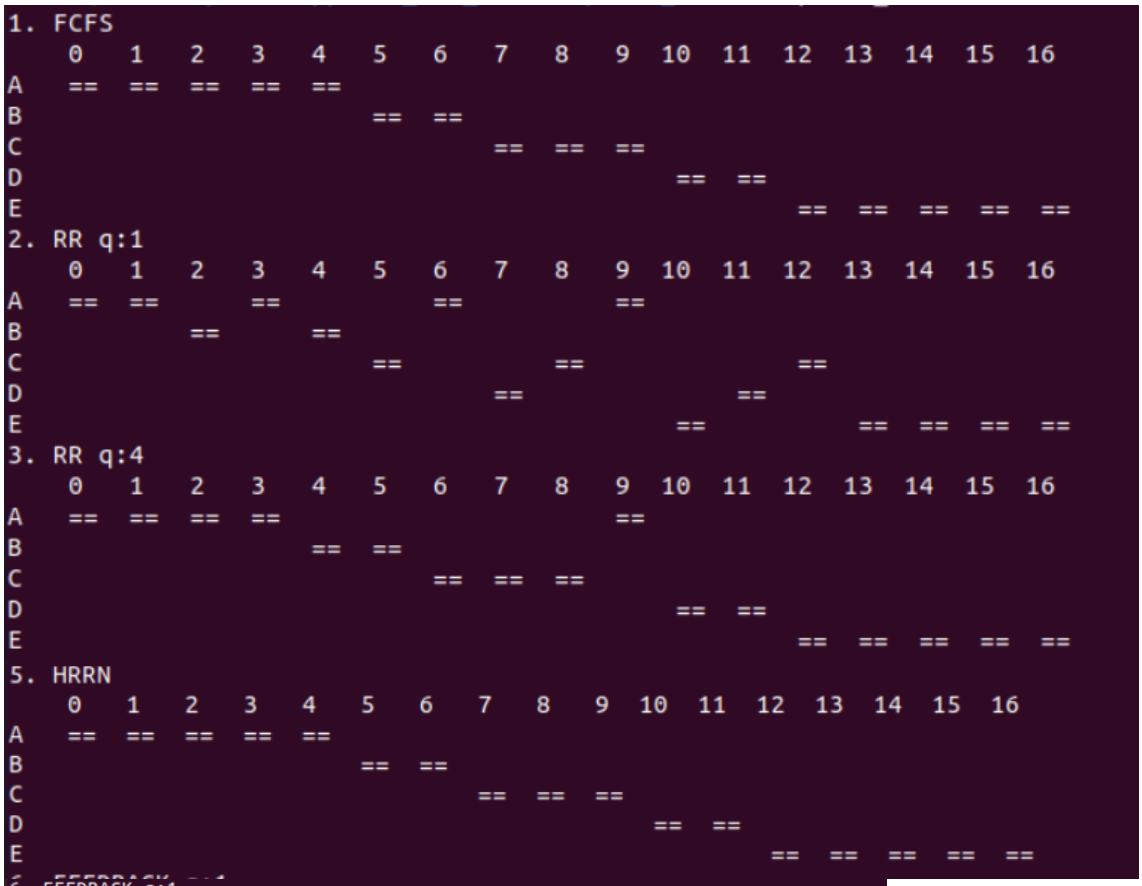
int a[6][2] = {{0,0},{0,3},{2,6},{4,4},{6,5},{8,2}};

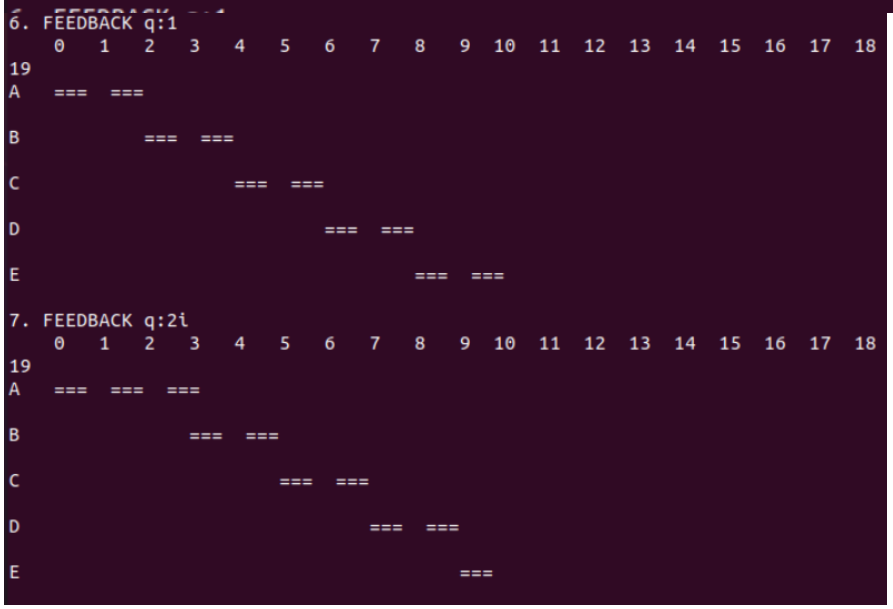
-> 사진 모두 19이후 부분은 좁아서 아래쪽으로 내려왔습니다!

```
1. FCFS
    0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
A   ==  ==  ==
B             ==  ==  ==  ==  ==  ==
C                                 ==  ==  ==  ==
D                                             ==  ==  ==  ==  ==
E                                                                 ==  ==
2. RR q:1
    0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
A   ==  ==      ==
B         ==      ==      ==          ==              ==              ==
C                     ==          ==              ==              ==
D                             ==              ==              ==          ==  ==
E                                     ==              ==
3. RR q:4
    0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
A   ==  ==  ==
B             ==  ==  ==  ==                              ==  ==
C                             ==  ==  ==  ==
D                                         ==  ==  ==  ==                  ==
E                                                                 ==  ==
4. SPN
    0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
A   ==  ==  ==
B             ==  ==  ==  ==  ==  ==
C                                         ==  ==  ==  ==
D                                                     ==  ==  ==  ==  ==
E                                     ==  ==
5. HRRN
    0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
A   ==  ==  ==
B             ==  ==  ==  ==  ==  ==
C                                 ==  ==  ==  ==
D                                                 ==  ==  ==  ==  ==
E                                                     ==  ==
6. FEEDBACK q:1
    0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
A   ===  ===
B         ===  ===
C               ===  ===
D                     ===  ===
E                           ===  ===
7. FEEDBACK q:2i
    0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
A   ===  ===  ===
B             ===  ===  ===
C                         ===  ===
D                               ===  ===
E
```

3-2) 결과2 : 스크린샷

int a[6][2] = {{0,0},{0,5},{2,2},{4,3},{6,2},{8,5}}; 를 이용하였습니다.





4) 어려웠던 점

우선 어떤 구조를 가져야 할지 각각의 자료구조를 선정하고, 어떤 식으로 동작하게 만들어야 할지 틀을 만드는 것에서 어려움을 느꼈지만, 어떤 부분이 필요한지에 관한 생각을 기반으로 동작을 구현해보았습니다. 또한, 각 프로세스가 처한 상황마다 다르게 동작을 구현해야 한다는 점에서 오류를 고치고 수정하는 것에 시간이 오래 걸렸으며, 어떤 상황이 존재할 수 있을지에 대해서 모두를 고려해야 한다는 점이 가장 어려웠다고 생각합니다.