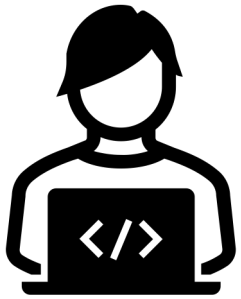
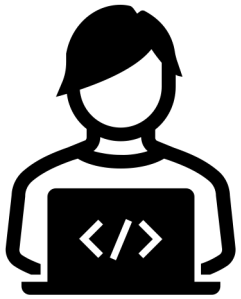
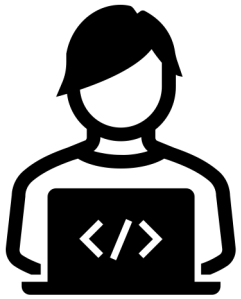
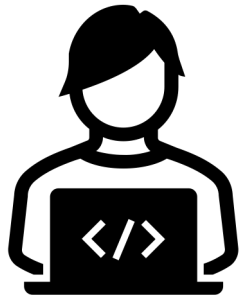
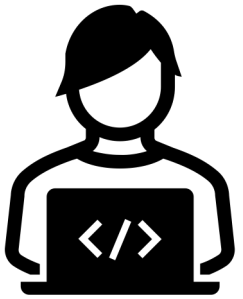
운영체제

학기말 팀 프로젝트

2020156038 주찬영
2020150017 박경은
2020152032 이현동
2020156022 송영주
2020156013 문예서

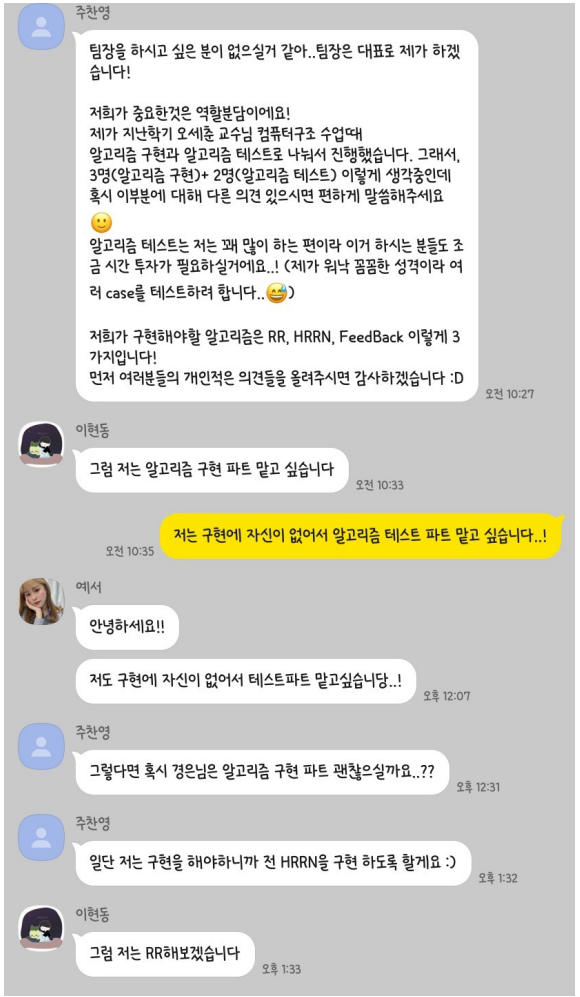
목차

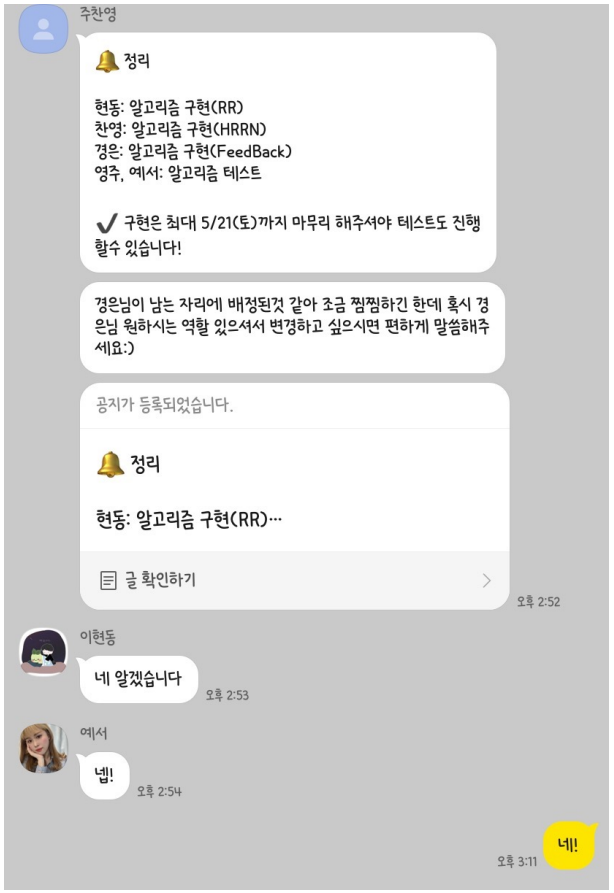
업무 분장

주찬영(팀장)	이현동	박경은	송영주	문예서
				
HRRN 알고리즘 구현	RR 알고리즘 구현	FeedBack 알고리즘 구현	알고리즘 테스트 및 보고서 작성	알고리즘 테스트 및 PPT 작성

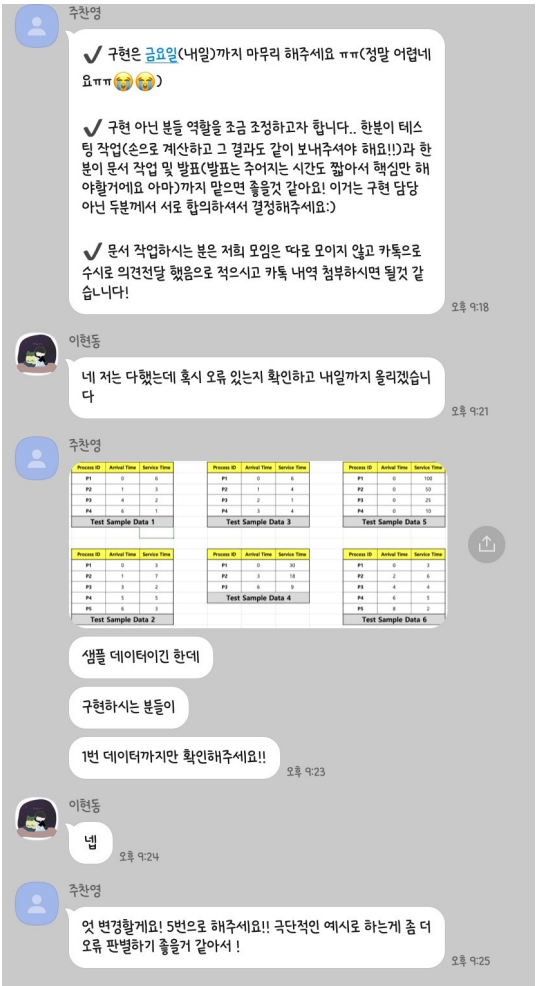
미팅 내용

업무 분담





알고리즘 구현 마무리, 테스트와 문서 작업 구체화



프로그램 핵심 구조

HRRN

process_id, arrive_time, service_time 리스트를 순회하면서 해당되는 프로세스의 객체를 만들어 프로세스를 담는 우선순위 큐(processQueue)에 프로세스를 대입하고 나서 스케줄링 알고리즘을 구현

```
// 프로세스 Input을 받아와 Arrive Time(도착 시간) 기준으로 정렬하여 큐에 저장 -> 우선순위 큐 이용
// 이 부분 코드는 쉽게 말해 도착 시간 순으로 Process 객체들을 정렬하는 부분을 담당하는 코드임
PriorityQueue<Process_Info> processQueue = new PriorityQueue<>(new Comparator<Process_Info>() {
    @Override
    public int compare(Process_Info ps1, Process_Info ps2) {
        if(ps1.getArriveTime() < ps2.getArriveTime()) { return -1; }
        else if(ps1.getArriveTime() > ps2.getArriveTime()) { return 1; }
        else { return 0; }
    }
});

// 프로세스 객체를 생성해서 processQueue에 추가
for (int i=0; i<process_id.size(); i++){

    Process_Info ps = new Process_Info(process_id.get(i), arrive_time.get(i), service_time.get(i));

    processQueue.add(ps);
    num_Process++;

}
```

RR

Round-Robin이 활성화되어 있으면서 시간할당량이 다 찼으면 실행하여
프로세스가 죽지 않고 가장 오래 대기한 프로세스를 가져오는 코드

```
        if(readyQueue.get(i).run && cycle) {
            if(readyQueue.get(i).service > 0 && wait[readyQueue.get(i).id] >= wait[readyQueue.get(idx).id]) {
                idx = i;
            }
        }
    }
```

실행중인 프로세스의 서비스 시간이 남아있으면서 실행된 시간이 시간 할당량과 다르면 활성화시키는 코드
아닐 경우 실행 -> 실행중인 프로세스의 서비스 시간이 끝났을 경우 실행시키는 코드

```
if(readyQueue.get(idx).service > 0 && temp != timeSlice) {
    cycle = false;
} else {
    if(readyQueue.get(idx).service == 0) {
        readyQueue.get(idx).run = false;
        cnt++;
```

FeedBack

프로세스는 CPU를 차지할 때까지 큐를 통하여 FCFS에 의하여 이동되며, 할당된 시간이 만료, 입출력, 돌발사건 등으로 인하여 CPU를 양도할 경우 그 작업은 그 다음 하위 단계의 큐로 이동되어 재배치된다.

마지막 단계의 큐에서는 그 프로세스가 완료될 때까지 RR로 순환된다.

프로세스가 하위 단계의 큐로 옮겨갈수록 각 큐의 할당된 시간은 2의 제곱 형태로 커지게 설정된다.

임의 큐에 있는 프로세스는 모든 상위 단계의 큐가 비어 있지 않으면 수행될 수 없으며, 수행 중인 프로세스라도 더 높은 단계의 큐에 있는 프로세스에 의해 선점될 수 있다.

Program Testing

Program의 유효성을 검증하기 위해 여러 다른 데이터들을 이용하여 검증하였음. (결과는 보고서에 첨부)

Process ID	Arrival Time	Service Time
P1	0	6
P2	1	3
P3	4	2
P4	6	1

Test Sample Data 1

Process ID	Arrival Time	Service Time
P1	0	3
P2	1	7
P3	3	2
P4	5	5
P5	6	3

Test Sample Data 2

Process ID	Arrival Time	Service Time
P1	0	6
P2	1	4
P3	2	1
P4	3	4

Test Sample Data 3

Process ID	Arrival Time	Service Time
P1	0	30
P2	3	18
P3	6	9

Test Sample Data 4

Process ID	Arrival Time	Service Time
P1	0	100
P2	0	50
P3	0	25
P4	0	10

Test Sample Data 5

Process ID	Arrival Time	Service Time
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2

Test Sample Data 6

소감 및 에피소드

주찬영

이번 운영체제 팀프로젝트를 통해서 프로세스 스케줄링 알고리즘을 한번 더 생각해볼 수 있어서 나름 의미 있던 팀 프로젝트였다만, 일단 구현하는데 있어서 너무 어려웠다... 하지만 해결하고 나니 나름 뿌듯함이 있어서 한편으로는 좋았던 팀 프로젝트 시간이였다고 생각한다.

박경은

이번 운영체제 팀프로젝트 과제를 통해서 피드백 스케줄링이라는 알고리즘을 처음 구현해 보았는데, 수업에서 배울 때 머리로는 이해를 했는데 이번 과제에서 이것을 구현하는게 너무 어려웠다... 여러 참고 문헌들과 교재를 참고하면서 직접 프로그래밍해보는 과정에서 여러 시행착오가 있었고 한계가 있었지만, 팀원분들께서 도와주신 덕분에 이 정도까지 한 것 같다. 이번 프로젝트를 통해서 다음에 이러한 팀프로젝트 과제가 존재한다면 그때는 팀원들에게 도움이 될 수 있는 실력을 가지기 위해 지금부터라도 열심히 노력해야겠다는 마음을 먹게 되었다.

이현동

이번 운영체제 팀프로젝트의 과제를 통해 라운드-로빈이라는 스케쥴링 알고리즘을 처음 만들어보았는데, 수업 내용에 있던 스케쥴링을 직접 프로그래밍하고, 실행을 시켰을 때 이론에서만 배웠던 내용이 직접 결과로 나타나니 매우 신기한 경험이었다.

송영주

운영체제 팀프로젝트를 통해 수업 시간에 이론으로 배웠던 알고리즘들을 팀원들과 구현해보고, 구현한 것을 바탕으로 테스트해보며 직접 경험해봐서 신기하기도 하고 좋은 경험이었다.

문예서

처음 운영체제 팀 프로젝트를 할 때는 코딩 실력이 너무 부족해서 많이 걱정했었다. 하지만 코딩 실력을 고려하여 편성된 팀 덕분에 실력 있는 여러 팀원들과 프로젝트를 수행했다. 이번에는 팀원들에게 많은 도움이 되지는 못했지만 이번 경험을 바탕으로 코딩 실력을 더 키워 보탬이 되는 팀원이 되고자 마음먹게 되었다.

감사합니다