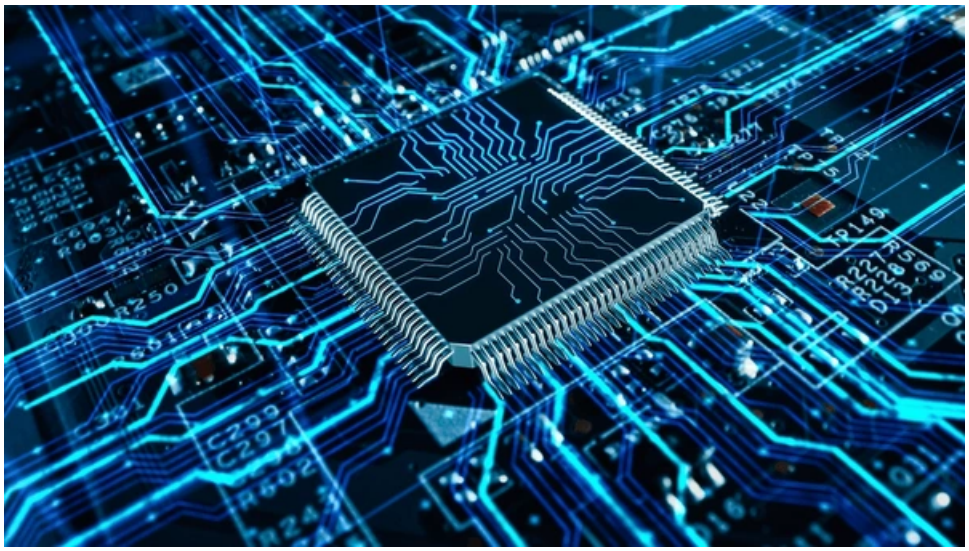
Term project: CPU Scheduler 구현

2025년 6월 1일

학번: 2020250148 이름: 심환석



1. 서론

1.1 CPU 스케줄링의 개념

CPU 스케줄링은 컴퓨터 운영체제의 핵심 기능 중 하나로, 다중 프로그래밍 환경에서 여러 프로세스가 CPU 자원을 효율적으로 공유할 수 있도록 관리하는 메커니즘이다. 현대의 컴퓨터 시스템은 동시에 수많은 프로세스를 처리해야 하지만, 단일 프로세서 환경에서는 특정 시점에 오직 하나의 프로세스만이 CPU를 사용할 수 있다. 따라서 운영체제는 CPU 스케줄러를 통해 어떤 프로세스가 언제 CPU를 사용할지 결정하고, 이를 통해 시스템의 전반적인 성능을 최적화한다.

CPU 스케줄링의 주요 목표는 프로세스 간 공정한 CPU 시간 분배, 시스템 처리량 최대화, 응답 시간 최소화, 그리고 전체적인 시스템 효율성 향상이다. 이러한 목표를 달성하기 위해 다양한 스케줄링 알고리즘이 개발되었으며, 각 알고리즘은 서로 다른 특성과 장단점을 가지고 있다.

1.2 프로젝트의 목적 및 개요

본 프로젝트는 운영체제의 CPU 스케줄링 알고리즘을 직접 구현하고 시뮬레이션함으로써, 각 알고리즘의 작동 원리와 특성을 깊이 있게 이해하는 것을 목적으로 한다. 이론적 학습에서 나아가 실제 구현을 통해 스케줄링 과정에서 발생하는 다양한 상황들을 직접 관찰하고, 각 알고리즘의 성능을 정량적으로 비교 분석할 수 있다.

프로젝트에서는 대표적인 CPU 스케줄링 알고리즘인 FCFS, SJF, Priority, Round Robin을 구현하였으며, 각각에 대해 선점형(Preemptive)과 비선점형(Non-preemptive) 방식을 모두 다루었다. 또한 Linux 환경에서 C 언어를 사용하여 구현함으로써 시스템 프로그래밍 능력 향상도 함께 도모하였다.

1.3 구현한 시뮬레이터의 주요 특징

본 프로젝트에서 구현한 CPU 스케줄링 시뮬레이터는 다음과 같은 주요 특징을 가진다:

1) 다양한 스케줄링 알고리즘 지원

- 총 8가지 알고리즘 구현: FCFS, Non-preemptive SJF, Preemptive SJF, Non-preemptive Priority, Preemptive Priority, Round Robin, 그리고 추가적으로 Non-preemptive/Preemptive LISJF (Longest I/O Shortest Job First)

2) 현실적인 프로세스 모델링

- 각 프로세스는 PID, 도착 시간, CPU 버스트 시간, I/O 버스트 시간, 우선순위 등의 속성을 가짐
- I/O 작업을 포함한 프로세스 실행 과정을 시뮬레이션하여 더욱 현실적인 스케줄링 환경 구현

3) 시각적 실행 과정 표시

- Gantt Chart 형태로 각 시간 단위별 프로세스 실행 상태를 출력
- 프로세스의 상태 변화(running, terminated, I/O request, I/O complete, CPU idle)를 명확히 표시

4) 정량적 성능 평가

- 각 알고리즘별로 평균 대기 시간(Average Waiting Time)과 평균 반환 시간(Average Turnaround Time) 계산
- 모든 알고리즘의 성능을 종합적으로 비교하여 순위 표시

5) 유연한 시스템 구성

- 랜덤 데이터 생성을 통한 다양한 시나리오 테스트 가능
- 프로세스 개수, 시간 범위 등의 파라미터 조정 가능

이러한 특징들을 통해 본 시뮬레이터는 CPU 스케줄링 알고리즘의 동작을 효과적으로 시각화하고, 각 알고리즘의 성능 차이를 명확하게 분석할 수 있는 교육적 도구로서의 역할을 수행한다.

2. 이론적 배경

2.1 CPU 스케줄링 알고리즘

CPU 스케줄링 알고리즘은 ready queue에 있는 프로세스들 중 어떤 프로세스에게 CPU를 할당할지 결정하는 정책이다. 각 알고리즘은 서로 다른 기준과 목표를 가지고 있으며, 시스템의 요구사항에 따라 적절한 알고리즘을 선택해야 한다.

2.1.1 First Come First Served (FCFS)

FCFS는 가장 단순한 스케줄링 알고리즘으로, 프로세스가 ready queue에 도착한 순서대로 CPU를 할당한다. 비선점형 방식으로 작동하며, 한 번 CPU를 할당받은 프로세스는 작업을 완료할 때까지 CPU를 독점한다. 구현이 간단하고 공정하다는 장점이 있지만, convoy effect(호송 효과)가 발생할 수 있다는 단점이 있다. 즉, CPU burst time이 긴 프로세스가 먼저 도착하면 뒤따르는 짧은 프로세스들이 오래 대기하게 된다.

2.1.2 Shortest Job First (SJF)

SJF는 CPU burst time이 가장 짧은 프로세스에게 우선적으로 CPU를 할당하는 알고리즘이다. 이론적으로 평균 대기 시간을 최소화할 수 있는 최적(optimal) 알고리즘이다. 비선점형과 선점형 두 가지 방식으로 구현할 수 있으며, 선점형 SJF는 SRTF(Shortest Remaining Time First)라고도 불린다. 주요 단점은 실제 시스템에서 프로세스의 CPU burst time을 정확히 예측하기 어렵다는 점과, CPU burst time이 긴 프로세스의 기아(starvation) 현상이 발생할 수 있다는 점이다.

2.1.3 Priority Scheduling

Priority 스케줄링은 각 프로세스에 우선순위를 부여하고, 가장 높은 우선순위를 가진 프로세스에게 CPU를 할당하는 방식이다. 우선순위는 내부적 또는 외부적 요인에 의해 결정될 수 있다. SJF와 마찬가지로 비선점형과 선점형으로 구현 가능하며, 중요한 작업을 우선적으로 처리할 수 있다는 장점이 있다. 그러나 낮은 우선순위 프로세스의 기아 현상이 발생할 수 있으며, 이를 해결하기 위해 aging 기법을 사용하기도 한다.

2.1.4 Round Robin (RR)

Round Robin은 시분할 시스템을 위해 설계된 선점형 스케줄링 알고리즘이다. 각 프로세스는 동일한 크기의 시간 할당량(time quantum)을 받으며, 이 시간이 만료되면 ready queue의 맨 뒤로 이동한다. 모든 프로세스가 공평하게 CPU 시간을 할당받을 수 있어 응답 시간이 좋고 기아 현상이 발생하지 않는다. 그러나 time quantum의 크기 설정이 중요하며, 너무 작으면 context switch overhead가 증가하고, 너무 크면 FCFS와 유사해진다.

2.2 스케줄링 평가 지표

스케줄링 알고리즘의 성능을 평가하기 위해서는 객관적이고 정량적인 지표가 필요하다. 본 프로젝트에서는 다음 두 가지 주요 지표를 사용하여 알고리즘의 성능을 평가한다.

2.2.1 Average Waiting Time

대기 시간(Waiting Time)은 프로세스가 ready queue에서 CPU를 기다린 총 시간을 의미한다. I/O 작업 시간은 포함되지 않으며, 오직 CPU를 할당받기 위해 대기한 시간만을 측정한다. 평균 대기 시간은 모든 프로세스의 대기 시간을 합산한 후 프로세스 수로 나눈 값이다. 이 값이 작을수록 프로세스들이 CPU를 빠르게 할당받았음을 의미하며, 시스템의 반응성이 좋다고 평가할 수 있다.

2.2.2 Average Turnaround Time

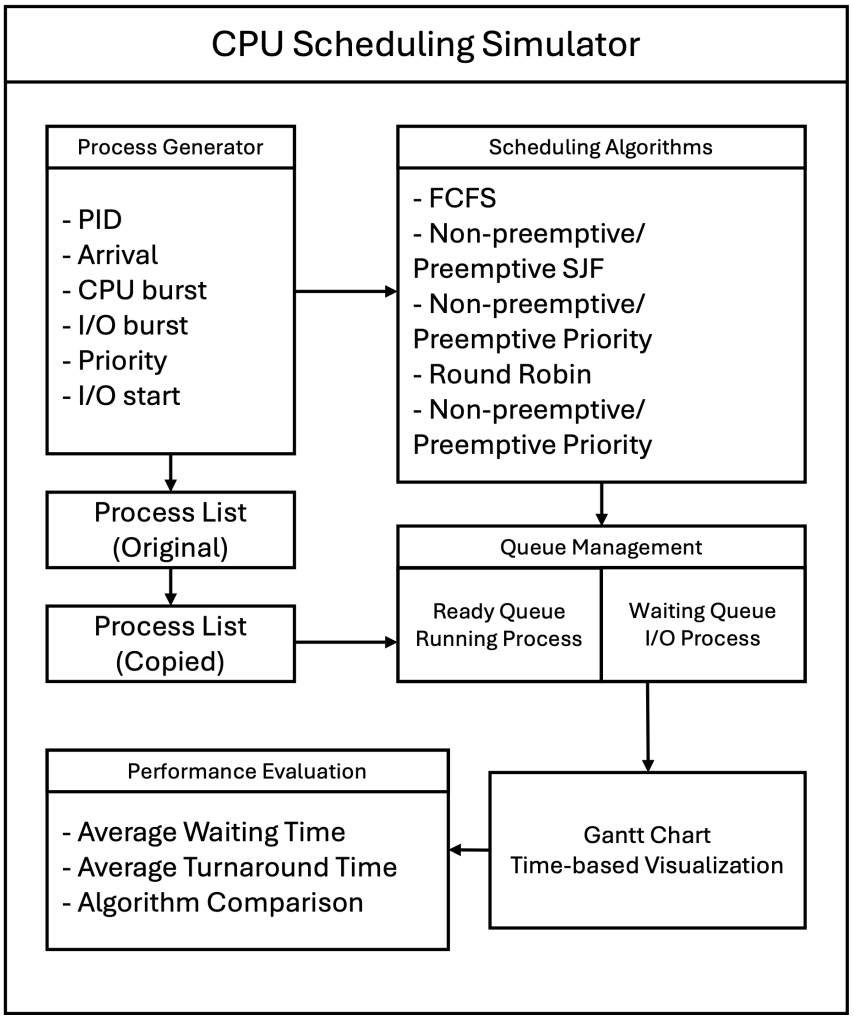
반환 시간(Turnaround Time)은 프로세스가 시스템에 도착한 시점부터 작업을 완료하고 종료될 때까지의 총 시간이다. 이는 대기 시간, CPU burst time, I/O burst time을 모두 포함한 값이다. 평균 반환 시간은 모든 프로세스의 반환 시간을 합산한 후 프로세스 수로 나눈 값이다. 이 지표는 프로세스가 시스템에서 얼마나 빨리 처리되는지를 나타내며, 작을수록 시스템의 처리 효율이 높다고 볼 수 있다.

두 지표는 서로 보완적인 관계에 있으며, 스케줄링 알고리즘의 종합적인 성능을 평가하기 위해서는 두 지표를 모두 고려해야 한다. 특정 알고리즘이 한 지표에서는 우수하지만 다른 지표에서는 부진할 수 있기 때문에, 시스템의 목적과 요구사항에 따라 적절한 균형점을 찾는 것이 중요하다.

3. CPU 스케줄링 시뮬레이터 설계 및 구현

3.1 시스템 구성도

본 CPU 스케줄링 시뮬레이터의 전체적인 시스템 구성도는 다음과 같다.



시스템은 프로세스 생성부터 최종 평가까지 순차적으로 진행되며, 각 알고리즘은 동일한 프로세스 세트에 대해 독립적으로 실행된다.

3.2 주요 자료구조

3.2.1 프로세스 구조체

```
struct proc {
    int pid;// 프로세스 ID (1부터 시작)
    int arrive;// 도착 시간
    int cpu_burst;// CPU 버스트 시간
    int io_burst;// I/O 버스트 시간
    int prior;// 우선순위 (낮을수록 높은 우선순위)
    int io_start;// I/O 시작 시점 (CPU burst 중 언제 I/O가 발생하는지)
    int cpu_left;// 남은 CPU 시간
    int io_left;// 남은 I/O 시간
    int wait;// 총 대기 시간
    int turnaround;// 총 반환 시간
};
```

프로세스 구조체는 시뮬레이션에 필요한 모든 정보를 담고 있다. 특히 `io_start`는 I/O 작업이 CPU burst 중 언제 발생하는지를 나타내며, 이를 통해 더욱 현실적인 스케줄링 시뮬레이션이 가능하다. `cpu_left`와 `io_left`는 실시간으로 업데이트되며, 프로세스의 진행 상황을 추적하는 데 사용된다.

3.2.2 큐 구조 (Ready Queue, Waiting Queue)

```
struct proc** ready_q = malloc(sizeof(struct proc*) * process_num);
struct proc** waiting_q = malloc(sizeof(struct proc*) * process_num);
int ready_cnt = 0;// ready queue에 있는 프로세스 수
int waiting_cnt = 0;// waiting queue에 있는 프로세스 수
```

큐는 동적 배열로 구현되었으며, 프로세스 구조체의 포인터를 저장한다. 이는 메모리 효율성을 높이고 프로세스 이동 시 복사 오버헤드를 줄인다. 각 큐는 해당하는 카운터 변수로 현재 큐의 크기를 관리한다.

3.2.3 평가 결과 저장 구조체

```
struct result {
    double wait_avg;// 평균 대기 시간
    double turn_avg;// 평균 반환 시간
    char alg_name[name_size];// 알고리즘 이름
};
struct result* results[total_alg];// 모든 알고리즘의 결과 저장
```

각 알고리즘의 성능 평가 결과를 저장하여 최종적으로 비교 분석할 때 사용한다.

3.3 주요 모듈 설명

3.3.1 프로세스 생성 모듈 (make_processes)

```
void make_processes() {
// 프로세스 리스트 동적 할당// 랜덤 시드 설정// 각 프로세스의 속성을 랜덤하게 생성://   - PID: 1부터 N까지
```

```
순차적 할당//   - Arrival time: 0~10 범위//   - CPU burst time: 5~20 범위//   - I/O burst time: 0~5 범위//   - Priority: 1~N 범위//   - I/O start: 1~(CPU burst time-1) 범위// 도착 시간 순으로 정렬// 생성된 프로세스 정보 출력
}
```

이 모듈은 시뮬레이션에 사용될 프로세스들을 생성한다. 랜덤 데이터를 사용하여 다양한 시나리오를 테스트할 수 있으며, I/O 작업이 있는 경우 CPU burst 중간에 발생하도록 설계했다. 이는 실제 시스템에서 프로세스가 I/O 요청 후 다시 CPU 작업을 계속하는 상황을 모델링한 것이다.

3.3.2 정렬 모듈

시뮬레이터는 네 가지 정렬 함수를 제공하며, 각각 다른 스케줄링 알고리즘에서 사용된다:

- **sort_by_arrive_time()**: FCFS에서 사용. 도착 시간이 빠른 순으로 정렬하며, 동일한 경우 PID가 작은 순으로 정렬
- **sort_by_cpu_time()**: SJF에서 사용. 남은 CPU 시간이 적은 순으로 정렬
- **sort_by_priority()**: Priority 스케줄링에서 사용. 우선순위 값이 낮은(우선순위가 높은) 순으로 정렬
- **sort_by_io()**: LISJF에서 사용. I/O 시간이 긴 순으로 정렬한 후, 동일하면 CPU 시간이 짧은 순으로 정렬

모든 정렬 함수는 기준 값이 동일한 경우를 위한 2차, 3차 정렬 기준을 가지고 있어 일관된 결과를 보장한다.

3.3.3 프로세스 복사 모듈 (copy_proc_list)

```
struct proc** copy_proc_list() {
// 새로운 프로세스 리스트 할당// 원본 프로세스의 모든 속성을 복사// 복사된 리스트 반환
}
```

각 알고리즘은 동일한 프로세스 세트로 테스트되어야 공정한 비교가 가능하다. 이 모듈은 원본 프로세스 리스트를 복사하여 각 알고리즘이 독립적으로 실행될 수 있도록 한다.

3.3.4 평가 모듈

- **calculate_result()**: 개별 알고리즘의 실행 결과를 계산하고 출력한다. 각 프로세스의 대기 시간과 반환 시간을 표시하고, 평균값을 계산하여 저장한다.
- **compare_all_algorithms()**: 모든 알고리즘의 결과를 비교하여 평균 대기 시간과 평균 반환 시간 기준으로 순위를 매긴다.

3.4 스케줄링 알고리즘 구현

각 스케줄링 알고리즘의 기본 구조는 다음과 같은 단계로 이루어진다:

```
1. 시간 = 0으로 초기화
2. 모든 프로세스가 종료될 때까지 반복:
   a. 현재 시간에 도착한 프로세스를 ready queue에 추가
   b. 알고리즘에 따라 실행할 프로세스 선택
   c. 프로세스 실행 (CPU burst 감소)
   d. 종료된 프로세스 처리
   e. ready queue의 프로세스들 대기 시간 증가
   f. waiting queue의 I/O 작업 처리
   g. I/O 요청이 필요한 프로세스를 waiting queue로 이동
```

```
   h. 시간 증가
3. 결과 계산 및 출력
```

3.4.1 FCFS 알고리즘

비선점형 알고리즘으로, ready queue를 도착 시간 순으로 정렬한 후 가장 앞의 프로세스를 실행한다. 실행 중인 프로세스는 종료되거나 I/O 요청을 할 때까지 CPU를 독점한다. 구현이 단순하며 공정성을 보장하지만, convoy effect가 발생할 수 있다.

3.4.2 Non-preemptive SJF

ready queue를 남은 CPU burst time 순으로 정렬하여 가장 짧은 프로세스를 선택한다. 비선점형이므로 한 번 시작된 프로세스는 중단되지 않는다. 평균 대기 시간을 최소화할 수 있지만, 긴 프로세스의 starvation 문제가 있다.

3.4.3 Preemptive SJF

새로운 프로세스가 도착할 때마다 현재 실행 중인 프로세스와 비교하여, 더 짧은 remaining time을 가진 프로세스가 있으면 context switch를 수행한다. 다음과 같은 추가 로직이 필요하다:

```
if(running != NULL && ready_cnt > 0) {
   if(running→cpu_left > ready_q[0]→cpu_left) {
// 현재 프로세스를 ready queue에 되돌리고// 새 프로세스 실행
   }
}
```

3.4.4 Non-preemptive Priority

ready queue를 우선순위 순으로 정렬하여 가장 높은 우선순위(낮은 priority 값)를 가진 프로세스를 실행한다. 중요한 작업을 우선 처리할 수 있지만, 낮은 우선순위 프로세스의 starvation이 발생할 수 있다.

3.4.5 Preemptive Priority

더 높은 우선순위를 가진 프로세스가 도착하면 현재 프로세스를 선점한다. 우선순위가 같은 경우에는 context switch overhead를 줄이기 위해 현재 프로세스를 계속 실행한다.

3.4.6 Round Robin

각 프로세스는 time quantum(본 구현에서는 5로 설정) 동안만 실행되며, 시간이 만료되면 ready queue의 맨 뒤로 이동한다. FIFO 방식으로 관리되며 별도의 정렬이 필요없다:

```
if(time_slice >= time_q && ready_cnt > 0) {
// 현재 프로세스를 ready queue 맨 뒤로// ready queue 맨 앞 프로세스 실행
   time_slice = 0;
}
```

3.4.7 추가 알고리즘 (Non-preemptive/Preemptive LISJF)

Longest I/O Shortest Job First는 본 프로젝트에서 추가로 구현한 알고리즘이다. I/O burst time이 긴 프로세스를 우선 실행하여 I/O 작업을 빨리 시작하게 함으로써 CPU idle time을 줄이는 것이 목표다. I/O time이 같으면 CPU burst time이 짧은 프로세스를 우선 선택한다.

3.5 I/O 작업 처리 구현

본 시뮬레이터의 I/O 처리는 다음과 같은 특징을 가진다:

1. **I/O 시작 시점 관리**: 각 프로세스는 `io_start` 값을 가지며, CPU burst를 이만큼 수행한 후 I/O 작업을 시작한다.
2. **I/O 요청 처리**

```
if(running != NULL && running→io_left > 0) {
    int cpu_done = running→cpu_burst - running→cpu_left;
    if(cpu_done == running→io_start) {
// 프로세스를 waiting queue로 이동// running process를 NULL로 설정
    }
}
```

1. **I/O 작업 진행**: waiting queue에 있는 모든 프로세스의 I/O burst를 매 시간 단위마다 감소시킨다. I/O 작업 중에도 turnaround time은 증가하지만 waiting time은 증가하지 않는다.
2. **I/O 완료 처리**: I/O 작업이 완료된 프로세스는 다시 ready queue로 이동한다. Round Robin의 경우 queue의 맨 뒤로, 다른 알고리즘의 경우 정렬이 필요하다.

이러한 I/O 처리 메커니즘을 통해 실제 시스템에서 발생하는 CPU-I/O 버스트 사이클을 효과적으로 시뮬레이션할 수 있다.

4. 실행 결과 및 성능 분석

4.1 시뮬레이션 환경

본 시뮬레이션은 다음과 같은 환경에서 수행되었다.

- 프로세스 개수: 5개
- 랜덤 데이터 범위:
  - Arrival time: 0~10
  - CPU burst time: 5~20
  - I/O burst time: 0~5
  - Priority: 1~5 (숫자가 작을수록 높은 우선순위)
  - Time quantum (Round Robin): 5

시뮬레이션에 사용된 프로세스 정보는 다음과 같다.

- Pid1: arrive=1, cpu_burst=18, io_burst=1, priority=3, io_start=13
- Pid2: arrive=10, cpu_burst=8, io_burst=0, priority=5, io_start=-1
- Pid3: arrive=8, cpu_burst=10, io_burst=5, priority=5, io_start=2
- Pid4: arrive=2, cpu_burst=16, io_burst=1, priority=1, io_start=14
- Pid5: arrive=3, cpu_burst=13, io_burst=0, priority=3, io_start=-1

4.2 알고리즘별 실행 결과

4.2.1 Gantt Chart

각 알고리즘의 실행 과정을 Gantt Chart 형태로 시각화하였다. 주요 이벤트는 다음과 같이 표시된다:

- `[running]` : 해당 시간 동안 프로세스가 실행됨
- `[terminated]` : 프로세스가 종료됨
- `[I/O request]` : I/O 작업을 위해 waiting queue로 이동
- `[I/O complete]` : I/O 작업 완료 후 ready queue로 복귀
- `[CPU idle]` : 실행 가능한 프로세스가 없음

예시 (FCFS 일부, 전체 출력은 부록에 별도 첨부):

```
T 0: [CPU idle],
T 1: P1 [running],
T 2: P1 [running],
...
T 13: P1 [running], P1 [I/O request],
T 14: P4 [running], P1 [I/O complete],
```

4.2.2 프로세스별 대기시간 및 반환시간

각 알고리즘별로 프로세스의 대기시간과 반환시간을 측정한 결과는 다음과 같다:

FCFS (First Come First Served)

- P1: wait_time=13, turnaround_time=32
- P2: wait_time=40, turnaround_time=48
- P3: wait_time=43, turnaround_time=58
- P4: wait_time=16, turnaround_time=33
- P5: wait_time=32, turnaround_time=45

Non-preemptive SJF

- P1: wait_time=7, turnaround_time=26
- P2: wait_time=4, turnaround_time=12
- P3: wait_time=27, turnaround_time=42
- P4: wait_time=48, turnaround_time=65
- P5: wait_time=26, turnaround_time=39

Preemptive SJF

- P1: wait_time=47, turnaround_time=66
- P2: wait_time=6, turnaround_time=14
- P3: wait_time=16, turnaround_time=31
- P4: wait_time=31, turnaround_time=48
- P5: wait_time=0, turnaround_time=13

Non-preemptive Priority

- P1: wait_time=13, turnaround_time=32
- P2: wait_time=40, turnaround_time=48
- P3: wait_time=43, turnaround_time=58
- P4: wait_time=16, turnaround_time=33
- P5: wait_time=32, turnaround_time=45

Preemptive Priority

- P1: wait_time=28, turnaround_time=47
- P2: wait_time=40, turnaround_time=48
- P3: wait_time=43, turnaround_time=58
- P4: wait_time=0, turnaround_time=17
- P5: wait_time=27, turnaround_time=40

Round Robin (Time Quantum=5)

- P1: wait_time=41, turnaround_time=60
- P2: wait_time=31, turnaround_time=39
- P3: wait_time=41, turnaround_time=56
- P4: wait_time=47, turnaround_time=64
- P5: wait_time=40, turnaround_time=53

Non-preemptive LISJF

- P1: wait_time=15, turnaround_time=34
- P2: wait_time=35, turnaround_time=43
- P3: wait_time=22, turnaround_time=37
- P4: wait_time=18, turnaround_time=35
- P5: wait_time=50, turnaround_time=63

Preemptive LISJF

- P1: wait_time=17, turnaround_time=36
- P2: wait_time=35, turnaround_time=43
- P3: wait_time=22, turnaround_time=37
- P4: wait_time=13, turnaround_time=30
- P5: wait_time=50, turnaround_time=63

4.3 성능 비교 분석

4.3.1 평균 대기시간 비교

평균 대기시간 기준 순위:

1. **Preemptive SJF: 20.00** ← 최적
2. Non-preemptive SJF: 22.40

3. Preemptive LISJF: 27.40
4. Preemptive Priority: 27.60
5. Non-preemptive LISJF: 28.00
6. Non-preemptive Priority: 28.80
7. FCFS: 28.80
8. Round Robin: 40.00

분석: Preemptive SJF가 가장 짧은 평균 대기시간을 보였다. 이는 짧은 작업을 우선 처리하면서도 더 짧은 프로세스가 도착하면 즉시 전환하는 특성 때문이다. 반면 Round Robin은 모든 프로세스를 공평하게 처리하지만, 빈번한 context switch로 인해 가장 긴 대기시간을 기록했다.

4.3.2 평균 반환시간 비교

평균 반환시간 기준 순위:

1. **Preemptive SJF: 34.40** ← 최적
2. Non-preemptive SJF: 36.80
3. Preemptive LISJF: 41.80
4. Preemptive Priority: 42.00
5. Non-preemptive LISJF: 42.40
6. Non-preemptive Priority: 43.20
7. FCFS: 43.20
8. Round Robin: 54.40

분석: 평균 반환시간에서도 Preemptive SJF가 최적의 성능을 보였다. SJF 계열 알고리즘이 상위권을 차지한 것은 짧은 프로세스를 빨리 완료시켜 전체적인 처리 시간을 단축시키기 때문이다.

4.4 결과 해석 및 고찰

4.4.1 알고리즘별 특성 분석

1. **SJF 알고리즘의 우수성**: Preemptive와 Non-preemptive SJF 모두 시간 측면에서 우수한 성능을 보였다. 특히 Preemptive SJF는 두 평가 지표 모두에서 1위를 차지했다. 이는 SJF가 이론적으로 평균 대기시간을 최소화하는 최적 알고리즘임을 실험적으로 입증한다.
2. **Preemptive vs Non-preemptive**: 대부분의 경우 선점형이 비선점형보다 좋은 성능을 보였다. 특히 Priority 스케줄링에서 P4(priority=1)의 경우, 선점형에서는 대기시간이 0이지만 비선점형에서는 16으로 큰 차이를 보였다.
3. **FCFS와 Priority의 유사성**: 본 시뮬레이션에서 Non-preemptive FCFS와 Non-preemptive Priority가 동일한 평균값을 보였다. 이는 우연의 일치로, 프로세스들의 도착 순서와 우선순위가 비슷하게 배치되었기 때문이다.
4. **Round Robin의 특성**: RR은 공정성을 보장하지만 시간 효율성 측면에서는 가장 낮은 성능을 보였다. 그러나 모든 프로세스의 대기시간이 31~47 범위로 비교적 균등하게 분포되어 있어, 특정 프로세스의 starvation을 방지한다는 장점을 확인할 수 있다.

4.4.2 I/O 처리의 영향

1. **CPU Idle 발생**: FCFS에서 Time 64에 CPU idle이 발생했다. 이는 P4의 I/O 작업이 진행 중일 때 ready queue가 비어있었기 때문이다. 이러한 상황은 I/O bound 프로세스의 배치에 따라 시스템 효율성이 크게 달라질 수 있음을 보여준다.
2. **LISJF 알고리즘의 효과**: I/O burst time이 긴 프로세스를 우선 처리하는 LISJF는 중간 정도의 성능을 보였다. Preemptive LISJF가 Non-preemptive보다 약간 좋은 성능을 보인 것은 I/O 작업을 더 효율적으로 스케줄링했기 때문이다.

4.4.3 실제 시스템 적용 시 고려사항

1. **예측의 한계**: SJF가 최적의 성능을 보였지만, 실제 시스템에서는 프로세스의 CPU burst time을 정확히 예측하기 어렵다. 따라서 과거 실행 이력을 바탕으로 한 예측 기법이 필요하다.
2. **Starvation 문제**: SJF와 Priority 스케줄링에서 긴 프로세스나 낮은 우선순위 프로세스의 starvation이 관찰되었다. 예를 들어, Non-preemptive SJF에서 P4는 48의 대기시간을 가졌다. 실제 시스템에서는 aging 기법 등의 보완책이 필요하다.
3. **Context Switch Overhead**: 본 시뮬레이션에서는 context switch 시간을 0으로 가정했지만, 실제로는 상당한 overhead가 발생한다. 특히 Round Robin과 Preemptive 알고리즘들은 빈번한 context switch로 인해 실제 성능이 더 낮을 수 있다.
4. **시스템 목적에 따른 선택**:
   - 배치 시스템: SJF나 FCFS가 적합
   - 대화형 시스템: Round Robin이 적합
   - 실시간 시스템: Priority 기반 스케줄링이 필수

이러한 분석을 통해 각 스케줄링 알고리즘이 특정 상황과 요구사항에 따라 선택되어야 함을 확인할 수 있다. 최적의 알고리즘은 시스템의 목적, 프로세스의 특성, 그리고 성능 지표의 우선순위에 따라 달라진다.

5. 결론

5.1 프로젝트 수행 결과 요약

본 프로젝트를 통해 운영체제의 핵심 기능인 CPU 스케줄링 알고리즘을 직접 구현하고 비교 분석하였다. 총 8가지 스케줄링 알고리즘(FCFS, Non-preemptive SJF, Preemptive SJF, Non-preemptive Priority, Preemptive Priority, Round Robin, Non-preemptive LISJF, Preemptive LISJF)을 C 언어로 구현하였으며, 각 알고리즘의 특성과 성능을 정량적으로 평가하였다.

시뮬레이션 결과, Preemptive SJF가 평균 대기시간(20.00)과 평균 반환시간(34.40) 모두에서 최적의 성능을 보였다. 이는 SJF가 이론적으로 최적 알고리즘임을 실험적으로 검증한 것이다. 반면 Round Robin은 공정성을 보장하지만 시간 효율성 측면에서는 가장 낮은 성능을 보였다. 이를 통해 각 알고리즘이 추구하는 목표와 트레이드오프 관계를 명확히 확인할 수 있었다.

또한 I/O 작업을 포함한 현실적인 프로세스 모델을 구현함으로써, CPU-I/O 버스트 사이클이 스케줄링에 미치는 영향을 관찰할 수 있었다. 특히 I/O 작업의 타이밍과 빈도가 전체 시스템 성능에 중요한 영향을 미친다는 점을 확인하였다.

5.2 구현 과정에서의 어려움 및 해결 방법

5.2.1 I/O 작업 시점 모델링

초기에는 I/O 작업을 고정된 시간에 처리하도록 구현했으나, 이는 현실성이 부족했다. 이를 해결하기 위해 `io_start` 변수를 추가하여 각 프로세스마다 CPU burst 중 랜덤한 시점에 I/O 작업이 발생하도록 수정하였다. 이를 통해 더욱 다양한 스케줄링 시나리오를 시뮬레이션할 수 있었다.

5.2.2 선점 조건 처리

Preemptive 알고리즘 구현 시 선점 조건을 정확히 처리하는 것이 어려웠다. 특히 비교 기준 값이 동일한 경우의 처리가 중요했는데, 무의미한 context switch를 방지하기 위해 동일한 경우에는 현재 실행 중인 프로세스를 계속 진행하도록 설계하였다. 이는 실제 시스템에서의 overhead를 고려한 현실적인 선택이었다.

5.2.3 메모리 관리

동적 메모리 할당을 사용하면서 메모리 누수를 방지하는 것이 중요했다. 각 알고리즘 실행 후 할당된 메모리를 적절히 해제하고, 프로세스 복사 시에는 포인터만 복사하여 메모리 효율성을 높였다.

5.2.4 공정한 비교 환경 구성

모든 알고리즘이 동일한 조건에서 비교되도록 하는 것이 중요했다. 이를 위해 원본 프로세스 리스트를 유지하고 각 알고리즘 실행 전에 복사하는 방식을 채택했다. 또한 정렬 함수에서 2차, 3차 정렬 기준을 명확히 정의하여 일관된 결과를 보장하였다.

5.3 프로젝트 수행 소감

본 프로젝트는 운영체제 수업에서 배운 이론적 지식을 실제 코드로 구현해보는 귀중한 경험이었다. 교과서에서 간단히 설명되는 스케줄링 알고리즘들이 실제로는 많은 세부사항과 예외 처리를 필요로 한다는 것을 깨달았다. 특히 다음과 같은 점들이 인상 깊었다:

1. **이론과 실제의 차이**: 개념적으로 이해하고 있던 알고리즘을 코드로 구현하면서, 고려해야 할 세부사항이 매우 많다는 것을 체감했다. 예를 들어, 큐가 비어있는 경우, 동일한 우선순위를 가진 프로세스들의 처리, I/O 작업 중 발생할 수 있는 다양한 시나리오 등을 모두 처리해야 했다.
2. **시각화의 중요성**: Gantt Chart 형태로 실행 과정을 출력하도록 구현한 것이 알고리즘의 동작을 이해하는 데 큰 도움이 되었다. 각 시간 단위마다 프로세스의 상태 변화를 추적할 수 있어 디버깅과 분석이 용이했다.
3. **성능 지표의 의미**: 평균 대기시간과 반환시간이라는 단순한 숫자 뒤에 숨겨진 의미를 깊이 이해할 수 있었다. 특정 알고리즘이 평균적으로는 우수하더라도 개별 프로세스 입장에서는 불공정할 수 있다는 점이 흥미로웠다.
4. **프로그래밍 능력 향상**: 복잡한 자료구조와 알고리즘을 구현하면서 C 프로그래밍 능력이 크게 향상되었다. 특히 포인터와 동적 메모리 관리, 구조체 활용 등에 대한 이해가 깊어졌다.

5.4 향후 발전 방향

본 프로젝트를 기반으로 다음과 같은 방향으로 시뮬레이터를 발전시킬 수 있을 것이다:

5.4.1 기능적 확장

1. **Multilevel Queue Scheduling**: 시스템 프로세스, 대화형 프로세스, 배치 프로세스 등을 구분하여 각각 다른 스케줄링 알고리즘을 적용하는 다단계 큐 스케줄링 구현
2. **Multiprocessor Scheduling**: 여러 개의 CPU를 가진 시스템에서의 스케줄링 시뮬레이션. 부하 균형(load balancing)과 프로세서 친화도(processor affinity) 등의 개념 추가
3. **Real-time Scheduling**: 데드라인을 가진 실시간 프로세스들의 스케줄링. EDF(Earliest Deadline First), Rate Monotonic 등의 알고리즘 구현

4. **Dynamic Priority with Aging**: Priority 스케줄링에서 starvation을 해결하기 위한 aging 기법 구현. 대기 시간에 따라 우선순위가 동적으로 변화하도록 개선

5.4.2 현실성 개선

1. **Context Switch Overhead**: 현재는 0으로 가정한 context switch 시간을 실제로 구현하여 더 현실적인 시뮬레이션 수행
2. **다양한 I/O 패턴**: 현재는 프로세스당 한 번의 I/O만 발생하지만, 여러 번의 I/O가 다양한 패턴으로 발생하도록 확장
3. **CPU Burst 예측**: SJF 알고리즘을 위한 exponential averaging 등의 예측 기법 구현
4. **메모리 제약 고려**: CPU 스케줄링과 메모리 관리를 연계하여 메모리 부족 시의 스케줄링 영향 분석

5.4.3 사용자 인터페이스 개선

1. **GUI 구현**: 텍스트 기반에서 벗어나 그래픽 인터페이스로 Gantt Chart를 실시간으로 시각화
2. **대화형 시뮬레이션**: 사용자가 실행 중에 프로세스를 추가하거나 우선순위를 변경할 수 있는 기능
3. **통계 분석 강화**: 다양한 성능 지표(응답 시간, CPU 이용률, 처리량 등)를 추가하고 그래프로 표시
4. **시나리오 저장/불러오기**: 특정 프로세스 세트를 저장하고 재사용할 수 있는 기능

이러한 발전 방향들은 시뮬레이터를 더욱 완성도 있는 교육 도구로 만들 수 있을 것이며, 운영체제의 복잡한 동작을 더 깊이 이해하는 데 도움이 될 것이다. 특히 현대의 멀티코어 프로세서와 복잡한 워크로드를 고려한 스케줄링 연구에 기여할 수 있을 것으로 기대한다.

6. 부록

6.1 Github 주소

https://github.com/Hwan0130/251RCOSE34102/tree/main

다음 깃헙 주소에서 본 프로젝트의 소스코드와 결과를 확인할 수 있다.

6.1 전체 출력

본 CPU 스케줄러는 다음과 같이 컴파일 후, 프로세스의 개수(아래 결과에서는 5개)를 지정하여 실행시킬 수 있다. 출력은 로그에서 확인할 수 있어 `nohup` 명령어를 통해 로그를 저장하였다.

```
gcc -o scheduler scheduler.c
nohup ./scheduler 5 > output1.out
```

```
Process List:
Pid1: arrive=1, cpu_burst=18, io_burst=1, priority=3, io_start=13
Pid4: arrive=2, cpu_burst=16, io_burst=1, priority=1, io_start=14
Pid5: arrive=3, cpu_burst=13, io_burst=0, priority=3, io_start=-1
Pid3: arrive=8, cpu_burst=10, io_burst=5, priority=5, io_start=2
Pid2: arrive=10, cpu_burst=8, io_burst=0, priority=5, io_start=-1

<First Come First Served Algorithm>
```

T 0: [CPU idle],
T 1: P1 [running],
T 2: P1 [running],
T 3: P1 [running],
T 4: P1 [running],
T 5: P1 [running],
T 6: P1 [running],
T 7: P1 [running],
T 8: P1 [running],
T 9: P1 [running],
T 10: P1 [running],
T 11: P1 [running],
T 12: P1 [running],
T 13: P1 [running], P1 [I/O request],
T 14: P4 [running], P1 [I/O complete],
T 15: P4 [running],
T 16: P4 [running],
T 17: P4 [running],
T 18: P4 [running],
T 19: P4 [running],
T 20: P4 [running],
T 21: P4 [running],
T 22: P4 [running],
T 23: P4 [running],
T 24: P4 [running],
T 25: P4 [running],
T 26: P4 [running],
T 27: P4 [running], P4 [I/O request],
T 28: P1 [running], P4 [I/O complete],
T 29: P1 [running],
T 30: P1 [running],
T 31: P1 [running],
T 32: P1 [running], P1 [terminated],
T 33: P4 [running],
T 34: P4 [running], P4 [terminated],
T 35: P5 [running],
T 36: P5 [running],
T 37: P5 [running],
T 38: P5 [running],
T 39: P5 [running],
T 40: P5 [running],
T 41: P5 [running],
T 42: P5 [running],
T 43: P5 [running],
T 44: P5 [running],
T 45: P5 [running],
T 46: P5 [running],
T 47: P5 [running], P5 [terminated],

```
T 48: P3 [running],
T 49: P3 [running], P3 [I/O request],
T 50: P2 [running],
T 51: P2 [running],
T 52: P2 [running],
T 53: P2 [running],
T 54: P2 [running], P3 [I/O complete],
T 55: P2 [running],
T 56: P2 [running],
T 57: P2 [running], P2 [terminated],
T 58: P3 [running],
T 59: P3 [running],
T 60: P3 [running],
T 61: P3 [running],
T 62: P3 [running],
T 63: P3 [running],
T 64: P3 [running],
T 65: P3 [running], P3 [terminated],

[pid: 1, wait_time: 13, turnaround_time: 32]
[pid: 4, wait_time: 16, turnaround_time: 33]
[pid: 5, wait_time: 32, turnaround_time: 45]
[pid: 3, wait_time: 43, turnaround_time: 58]
[pid: 2, wait_time: 40, turnaround_time: 48]

평균 대기 시간: 28.80
평균 반환 시간: 43.20
================================================================
<Non-preemptive Shortest Job First Algorithm>
T 0: [CPU idle],
T 1: P1 [running],
T 2: P1 [running],
T 3: P1 [running],
T 4: P1 [running],
T 5: P1 [running],
T 6: P1 [running],
T 7: P1 [running],
T 8: P1 [running],
T 9: P1 [running],
T 10: P1 [running],
T 11: P1 [running],
T 12: P1 [running],
T 13: P1 [running], P1 [I/O request],
T 14: P2 [running], P1 [I/O complete],
T 15: P2 [running],
T 16: P2 [running],
T 17: P2 [running],
T 18: P2 [running],
```

T 19: P2 [running],
T 20: P2 [running],
T 21: P2 [running], P2 [terminated],
T 22: P1 [running],
T 23: P1 [running],
T 24: P1 [running],
T 25: P1 [running],
T 26: P1 [running], P1 [terminated],
T 27: P3 [running],
T 28: P3 [running], P3 [I/O request],
T 29: P5 [running],
T 30: P5 [running],
T 31: P5 [running],
T 32: P5 [running],
T 33: P5 [running], P3 [I/O complete],
T 34: P5 [running],
T 35: P5 [running],
T 36: P5 [running],
T 37: P5 [running],
T 38: P5 [running],
T 39: P5 [running],
T 40: P5 [running],
T 41: P5 [running], P5 [terminated],
T 42: P3 [running],
T 43: P3 [running],
T 44: P3 [running],
T 45: P3 [running],
T 46: P3 [running],
T 47: P3 [running],
T 48: P3 [running],
T 49: P3 [running], P3 [terminated],
T 50: P4 [running],
T 51: P4 [running],
T 52: P4 [running],
T 53: P4 [running],
T 54: P4 [running],
T 55: P4 [running],
T 56: P4 [running],
T 57: P4 [running],
T 58: P4 [running],
T 59: P4 [running],
T 60: P4 [running],
T 61: P4 [running],
T 62: P4 [running],
T 63: P4 [running], P4 [I/O request],
T 64: [CPU idle], P4 [I/O complete],
T 65: P4 [running],
T 66: P4 [running], P4 [terminated],

```
[pid: 1, wait_time: 7, turnaround_time: 26]
[pid: 4, wait_time: 48, turnaround_time: 65]
[pid: 5, wait_time: 26, turnaround_time: 39]
[pid: 3, wait_time: 27, turnaround_time: 42]
[pid: 2, wait_time: 4, turnaround_time: 12]

평균 대기 시간: 22.40
평균 반환 시간: 36.80
================================================================
<Preemptive Shortest Job First Algorithm>
T 0: [CPU idle],
T 1: P1 [running],
T 2: P4 [running],
T 3: P5 [running],
T 4: P5 [running],
T 5: P5 [running],
T 6: P5 [running],
T 7: P5 [running],
T 8: P5 [running],
T 9: P5 [running],
T 10: P5 [running],
T 11: P5 [running],
T 12: P5 [running],
T 13: P5 [running],
T 14: P5 [running],
T 15: P5 [running], P5 [terminated],
T 16: P2 [running],
T 17: P2 [running],
T 18: P2 [running],
T 19: P2 [running],
T 20: P2 [running],
T 21: P2 [running],
T 22: P2 [running],
T 23: P2 [running], P2 [terminated],
T 24: P3 [running],
T 25: P3 [running], P3 [I/O request],
T 26: P4 [running],
T 27: P4 [running],
T 28: P4 [running],
T 29: P4 [running],
T 30: P4 [running], P3 [I/O complete],
T 31: P3 [running],
T 32: P3 [running],
T 33: P3 [running],
T 34: P3 [running],
T 35: P3 [running],
T 36: P3 [running],
```

```
T 37: P3 [running],
T 38: P3 [running], P3 [terminated],
T 39: P4 [running],
T 40: P4 [running],
T 41: P4 [running],
T 42: P4 [running],
T 43: P4 [running],
T 44: P4 [running],
T 45: P4 [running],
T 46: P4 [running], P4 [I/O request],
T 47: P1 [running], P4 [I/O complete],
T 48: P4 [running],
T 49: P4 [running], P4 [terminated],
T 50: P1 [running],
T 51: P1 [running],
T 52: P1 [running],
T 53: P1 [running],
T 54: P1 [running],
T 55: P1 [running],
T 56: P1 [running],
T 57: P1 [running],
T 58: P1 [running],
T 59: P1 [running],
T 60: P1 [running], P1 [I/O request],
T 61: [CPU idle], P1 [I/O complete],
T 62: P1 [running],
T 63: P1 [running],
T 64: P1 [running],
T 65: P1 [running],
T 66: P1 [running], P1 [terminated],

[pid: 1, wait_time: 47, turnaround_time: 66]
[pid: 4, wait_time: 31, turnaround_time: 48]
[pid: 5, wait_time: 0, turnaround_time: 13]
[pid: 3, wait_time: 16, turnaround_time: 31]
[pid: 2, wait_time: 6, turnaround_time: 14]

평균 대기 시간: 20.00
평균 반환 시간: 34.40
================================================================
<Non-preemptive Priority Algorithm>
T 0: [CPU idle],
T 1: P1 [running],
T 2: P1 [running],
T 3: P1 [running],
T 4: P1 [running],
T 5: P1 [running],
T 6: P1 [running],
```

T 7: P1 [running],
T 8: P1 [running],
T 9: P1 [running],
T 10: P1 [running],
T 11: P1 [running],
T 12: P1 [running],
T 13: P1 [running], P1 [I/O request],
T 14: P4 [running], P1 [I/O complete],
T 15: P4 [running],
T 16: P4 [running],
T 17: P4 [running],
T 18: P4 [running],
T 19: P4 [running],
T 20: P4 [running],
T 21: P4 [running],
T 22: P4 [running],
T 23: P4 [running],
T 24: P4 [running],
T 25: P4 [running],
T 26: P4 [running],
T 27: P4 [running], P4 [I/O request],
T 28: P1 [running], P4 [I/O complete],
T 29: P1 [running],
T 30: P1 [running],
T 31: P1 [running],
T 32: P1 [running], P1 [terminated],
T 33: P4 [running],
T 34: P4 [running], P4 [terminated],
T 35: P5 [running],
T 36: P5 [running],
T 37: P5 [running],
T 38: P5 [running],
T 39: P5 [running],
T 40: P5 [running],
T 41: P5 [running],
T 42: P5 [running],
T 43: P5 [running],
T 44: P5 [running],
T 45: P5 [running],
T 46: P5 [running],
T 47: P5 [running], P5 [terminated],
T 48: P3 [running],
T 49: P3 [running], P3 [I/O request],
T 50: P2 [running],
T 51: P2 [running],
T 52: P2 [running],
T 53: P2 [running],
T 54: P2 [running], P3 [I/O complete],

```
T 55: P2 [running],
T 56: P2 [running],
T 57: P2 [running], P2 [terminated],
T 58: P3 [running],
T 59: P3 [running],
T 60: P3 [running],
T 61: P3 [running],
T 62: P3 [running],
T 63: P3 [running],
T 64: P3 [running],
T 65: P3 [running], P3 [terminated],

[pid: 1, wait_time: 13, turnaround_time: 32]
[pid: 4, wait_time: 16, turnaround_time: 33]
[pid: 5, wait_time: 32, turnaround_time: 45]
[pid: 3, wait_time: 43, turnaround_time: 58]
[pid: 2, wait_time: 40, turnaround_time: 48]

평균 대기 시간: 28.80
평균 반환 시간: 43.20
================================================================
<Preemptive Priority Algorithm>
T 0: [CPU idle],
T 1: P1 [running],
T 2: P4 [running],
T 3: P4 [running],
T 4: P4 [running],
T 5: P4 [running],
T 6: P4 [running],
T 7: P4 [running],
T 8: P4 [running],
T 9: P4 [running],
T 10: P4 [running],
T 11: P4 [running],
T 12: P4 [running],
T 13: P4 [running],
T 14: P4 [running],
T 15: P4 [running], P4 [I/O request],
T 16: P1 [running], P4 [I/O complete],
T 17: P4 [running],
T 18: P4 [running], P4 [terminated],
T 19: P1 [running],
T 20: P1 [running],
T 21: P1 [running],
T 22: P1 [running],
T 23: P1 [running],
T 24: P1 [running],
T 25: P1 [running],
```

```
T 26: P1 [running],
T 27: P1 [running],
T 28: P1 [running],
T 29: P1 [running], P1 [I/O request],
T 30: P5 [running], P1 [I/O complete],
T 31: P5 [running],
T 32: P5 [running],
T 33: P5 [running],
T 34: P5 [running],
T 35: P5 [running],
T 36: P5 [running],
T 37: P5 [running],
T 38: P5 [running],
T 39: P5 [running],
T 40: P5 [running],
T 41: P5 [running],
T 42: P5 [running], P5 [terminated],
T 43: P1 [running],
T 44: P1 [running],
T 45: P1 [running],
T 46: P1 [running],
T 47: P1 [running], P1 [terminated],
T 48: P3 [running],
T 49: P3 [running], P3 [I/O request],
T 50: P2 [running],
T 51: P2 [running],
T 52: P2 [running],
T 53: P2 [running],
T 54: P2 [running], P3 [I/O complete],
T 55: P2 [running],
T 56: P2 [running],
T 57: P2 [running], P2 [terminated],
T 58: P3 [running],
T 59: P3 [running],
T 60: P3 [running],
T 61: P3 [running],
T 62: P3 [running],
T 63: P3 [running],
T 64: P3 [running],
T 65: P3 [running], P3 [terminated],

[pid: 1, wait_time: 28, turnaround_time: 47]
[pid: 4, wait_time: 0, turnaround_time: 17]
[pid: 5, wait_time: 27, turnaround_time: 40]
[pid: 3, wait_time: 43, turnaround_time: 58]
[pid: 2, wait_time: 40, turnaround_time: 48]

평균 대기 시간: 27.60
```

```
평균 반환 시간: 42.00
================================================================
<Round Robin Algorithm (time quantum: 5)>
T 0: [CPU idle],
T 1: P1 [running],
T 2: P1 [running],
T 3: P1 [running],
T 4: P1 [running],
T 5: P1 [running],
T 6: P4 [running],
T 7: P4 [running],
T 8: P4 [running],
T 9: P4 [running],
T 10: P4 [running],
T 11: P5 [running],
T 12: P5 [running],
T 13: P5 [running],
T 14: P5 [running],
T 15: P5 [running],
T 16: P1 [running],
T 17: P1 [running],
T 18: P1 [running],
T 19: P1 [running],
T 20: P1 [running],
T 21: P3 [running],
T 22: P3 [running], P3 [I/O request],
T 23: P2 [running],
T 24: P2 [running],
T 25: P2 [running],
T 26: P2 [running],
T 27: P2 [running], P3 [I/O complete],
T 28: P4 [running],
T 29: P4 [running],
T 30: P4 [running],
T 31: P4 [running],
T 32: P4 [running],
T 33: P5 [running],
T 34: P5 [running],
T 35: P5 [running],
T 36: P5 [running],
T 37: P5 [running],
T 38: P1 [running],
T 39: P1 [running],
T 40: P1 [running], P1 [I/O request],
T 41: P3 [running], P1 [I/O complete],
T 42: P3 [running],
T 43: P3 [running],
T 44: P3 [running],
```

```
T 45: P3 [running],
T 46: P2 [running],
T 47: P2 [running],
T 48: P2 [running], P2 [terminated],
T 49: P4 [running],
T 50: P4 [running],
T 51: P4 [running],
T 52: P4 [running], P4 [I/O request],
T 53: P5 [running], P4 [I/O complete],
T 54: P5 [running],
T 55: P5 [running], P5 [terminated],
T 56: P1 [running],
T 57: P1 [running],
T 58: P1 [running],
T 59: P1 [running],
T 60: P1 [running], P1 [terminated],
T 61: P3 [running],
T 62: P3 [running],
T 63: P3 [running], P3 [terminated],
T 64: P4 [running],
T 65: P4 [running], P4 [terminated],

[pid: 1, wait_time: 41, turnaround_time: 60]
[pid: 4, wait_time: 47, turnaround_time: 64]
[pid: 5, wait_time: 40, turnaround_time: 53]
[pid: 3, wait_time: 41, turnaround_time: 56]
[pid: 2, wait_time: 31, turnaround_time: 39]

평균 대기 시간: 40.00
평균 반환 시간: 54.40
================================================================
<Non-preemptive Longest I/O Shortest Job First Algorithm>
T 0: [CPU idle],
T 1: P1 [running],
T 2: P1 [running],
T 3: P1 [running],
T 4: P1 [running],
T 5: P1 [running],
T 6: P1 [running],
T 7: P1 [running],
T 8: P1 [running],
T 9: P1 [running],
T 10: P1 [running],
T 11: P1 [running],
T 12: P1 [running],
T 13: P1 [running], P1 [I/O request],
T 14: P3 [running], P1 [I/O complete],
T 15: P3 [running], P3 [I/O request],
```

T 16: P4 [running],
T 17: P4 [running],
T 18: P4 [running],
T 19: P4 [running],
T 20: P4 [running], P3 [I/O complete],
T 21: P4 [running],
T 22: P4 [running],
T 23: P4 [running],
T 24: P4 [running],
T 25: P4 [running],
T 26: P4 [running],
T 27: P4 [running],
T 28: P4 [running],
T 29: P4 [running], P4 [I/O request],
T 30: P1 [running], P4 [I/O complete],
T 31: P1 [running],
T 32: P1 [running],
T 33: P1 [running],
T 34: P1 [running], P1 [terminated],
T 35: P4 [running],
T 36: P4 [running], P4 [terminated],
T 37: P3 [running],
T 38: P3 [running],
T 39: P3 [running],
T 40: P3 [running],
T 41: P3 [running],
T 42: P3 [running],
T 43: P3 [running],
T 44: P3 [running], P3 [terminated],
T 45: P2 [running],
T 46: P2 [running],
T 47: P2 [running],
T 48: P2 [running],
T 49: P2 [running],
T 50: P2 [running],
T 51: P2 [running],
T 52: P2 [running], P2 [terminated],
T 53: P5 [running],
T 54: P5 [running],
T 55: P5 [running],
T 56: P5 [running],
T 57: P5 [running],
T 58: P5 [running],
T 59: P5 [running],
T 60: P5 [running],
T 61: P5 [running],
T 62: P5 [running],
T 63: P5 [running],

```
T 64: P5 [running],
T 65: P5 [running], P5 [terminated],

[pid: 1, wait_time: 15, turnaround_time: 34]
[pid: 4, wait_time: 18, turnaround_time: 35]
[pid: 5, wait_time: 50, turnaround_time: 63]
[pid: 3, wait_time: 22, turnaround_time: 37]
[pid: 2, wait_time: 35, turnaround_time: 43]

평균 대기 시간: 28.00
평균 반환 시간: 42.40
================================================================
<Preemptive Longest I/O Shortest Job First Algorithm>
T 0: [CPU idle],
T 1: P1 [running],
T 2: P4 [running],
T 3: P4 [running],
T 4: P4 [running],
T 5: P4 [running],
T 6: P4 [running],
T 7: P4 [running],
T 8: P3 [running],
T 9: P3 [running], P3 [I/O request],
T 10: P4 [running],
T 11: P4 [running],
T 12: P4 [running],
T 13: P4 [running],
T 14: P4 [running], P3 [I/O complete],
T 15: P4 [running],
T 16: P4 [running],
T 17: P4 [running], P4 [I/O request],
T 18: P1 [running], P4 [I/O complete],
T 19: P1 [running],
T 20: P1 [running],
T 21: P1 [running],
T 22: P1 [running],
T 23: P1 [running],
T 24: P1 [running],
T 25: P1 [running],
T 26: P1 [running],
T 27: P1 [running],
T 28: P1 [running],
T 29: P1 [running], P1 [I/O request],
T 30: P4 [running], P1 [I/O complete],
T 31: P4 [running], P4 [terminated],
T 32: P1 [running],
T 33: P1 [running],
T 34: P1 [running],
```

```
T 35: P1 [running],
T 36: P1 [running], P1 [terminated],
T 37: P3 [running],
T 38: P3 [running],
T 39: P3 [running],
T 40: P3 [running],
T 41: P3 [running],
T 42: P3 [running],
T 43: P3 [running],
T 44: P3 [running], P3 [terminated],
T 45: P2 [running],
T 46: P2 [running],
T 47: P2 [running],
T 48: P2 [running],
T 49: P2 [running],
T 50: P2 [running],
T 51: P2 [running],
T 52: P2 [running], P2 [terminated],
T 53: P5 [running],
T 54: P5 [running],
T 55: P5 [running],
T 56: P5 [running],
T 57: P5 [running],
T 58: P5 [running],
T 59: P5 [running],
T 60: P5 [running],
T 61: P5 [running],
T 62: P5 [running],
T 63: P5 [running],
T 64: P5 [running],
T 65: P5 [running], P5 [terminated],

[pid: 1, wait_time: 17, turnaround_time: 36]
[pid: 4, wait_time: 13, turnaround_time: 30]
[pid: 5, wait_time: 50, turnaround_time: 63]
[pid: 3, wait_time: 22, turnaround_time: 37]
[pid: 2, wait_time: 35, turnaround_time: 43]

평균 대기 시간: 27.40
평균 반환 시간: 41.80
================================================================
<평균 대기 시간 순위>
1등. Preemptive Shortest Job First Algorithm: 20.00
2등. Non-preemptive Shortest Job First Algorithm: 22.40
3등. Preemptive Longest I/O Shortest Job First Algorithm: 27.40
4등. Preemptive Priority Algorithm: 27.60
5등. Non-preemptive Longest I/O Shortest Job First Algorithm: 28.00
6등. Non-preemptive Priority Algorithm: 28.80
```

7등. First Come First Served Algorithm: 28.80
8등. Round Robin Algorithm: 40.00

<평균 반환 시간 순위>
1등. Preemptive Shortest Job First Algorithm: 34.40
2등. Non-preemptive Shortest Job First Algorithm: 36.80
3등. Preemptive Longest I/O Shortest Job First Algorithm: 41.80
4등. Preemptive Priority Algorithm: 42.00
5등. Non-preemptive Longest I/O Shortest Job First Algorithm: 42.40
6등. Non-preemptive Priority Algorithm: 43.20
7등. First Come First Served Algorithm: 43.20
8등. Round Robin Algorithm: 54.40