운영체제 텀프로젝트 보고서

2020320135 김윤서

서론

cpu 스케줄러의 개념

운영체제에서 CPU 스케줄링은 핵심적인 역할을 수행하는 기능 중 하나로, 여러 개의 프로세스가 동시에 실행을 요청할 때 CPU 자원을 효율적으로 분배하는 알고리즘을 의미한다. 현대의 컴퓨터 시스템은 다수의 프로세스가 병렬적으로 실행되며, 이들 프로세스는 모두 CPU 자원을 공유한다. 이때 어떤 프로세스에게 먼저 CPU를 할당할지, 얼마나 오랫동안 실행시킬지를 결정하는 것이 바로 CPU 스케줄러의 역할이다. 스케줄링 알고리즘의 성능은 시스템의 응답 시간, 처리량, 대기 시간, CPU 활용률 등에 직접적인 영향을 미치기 때문에, 상황과 목적에 따라 다양한 스케줄링 방식이 존재한다.

일반적으로 많이 사용되는 스케줄링 알고리즘에는 FCFS(First-Come-First-Serve), SJF(Shortest Job First), PSJF(Preemptive SJF), Priority, Preemptive Priority, RR(Round Robin) 등이 있다. 이들 알고리즘은 각기 다른 기준으로 프로세스의 실행 순서를 결정하며, 특정 상황에서는 한 알고리즘이 다른 알고리즘보다 더 효율적인 성능을 보이기도 한다. 예를 들어 SJF는 평균 대기 시간을 최소화할 수 있지만, 짧은 작업에 편향되어 긴 작업이 무기한 대기하는 starvation 문제가 발생할 수 있다. 반면 RR 알고리즘은 공정성을 보장하지만 컨텍스트 스위칭이 잦아질 수 있는 단점이 있다.

스케줄러 요약 설명

본 프로젝트에서는 이러한 다양한 CPU 스케줄링 알고리즘을 직접 구현하고, 동일한 조건에서의 시뮬레이션을 통해 그 성능을 비교할 수 있는 시뮬레이터를 개발하였다. 시뮬레이션의 기반이 되는 프로세스들은 무작위로 생성되며, 이를 통해 다양한 실행 시간, 도착 시간, 우선순위 등의 조건을 실험적으로 재현할 수 있도록 하였다. 구체적으로는 다음과 같은 과정으로 스케줄링이 진행된다.

먼저 create_processes(n) 함수를 통해 총 n개의 프로세스를 무작위 속성으로 생성한다. 이렇게 생성된 기본 프로세스 목록(base)은 clone_processes 함수를 통해 각 스케줄링 알고리즘별로 복제된다. 이를 통해 동일한 입력 조건 하에서 알고리즘 간의 성능 비교가 가능하도록 하였다. 예를 들어 fcfs_list, sjf_list, psjf_list, prio_list, pprio_list, rr_list는 각각 FCFS, SJF, PSJF, Priority, Preemptive Priority, Round Robin 알고리즘에 따라 독립적으로 동작하며, 각기 다른 방식으로 프로세스를 스케줄링하게 된다.

이러한 구조를 통해 사용자는 다양한 알고리즘을 손쉽게 비교하고, 각 방식의 특징과 장단점을 명확히 파악할 수 있도록 하였다. 본 보고서에서는 이 시뮬레이터의 구성과 각 알고리즘의 동작 방식을 자세히 설명하고, 시뮬레이션 결과를 바탕으로 성능을 정량적으로 비교하였다.

본론

다른 cpu 스케줄러에 대한 소개

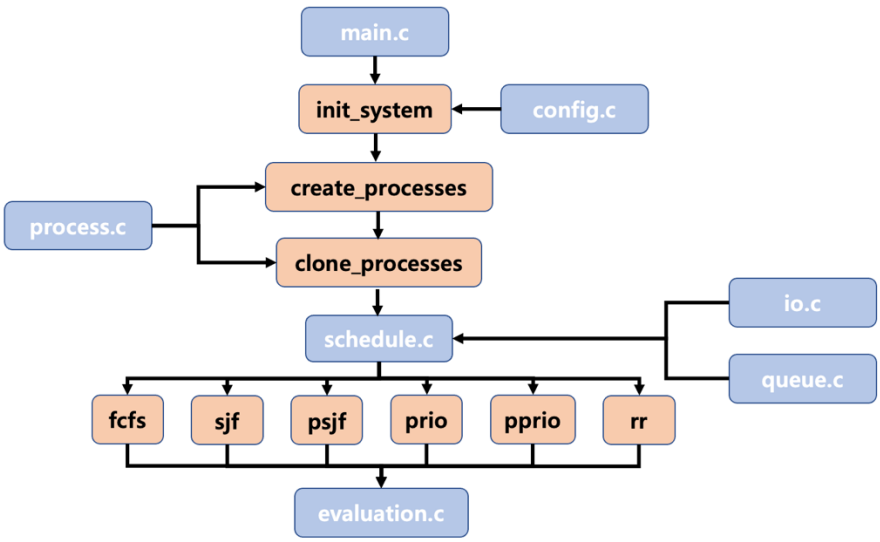
현대 운영체제 교육에서는 다양한 CPU 스케줄링 시뮬레이터가 사용되고 있으며, 이들은 각기 다른 목적과 특징을 가지고 설계되어 있다. 대표적인 예로는 교육용 운영체제 시뮬레이터인 OSSim이 있다. OSSim은 CPU 스케줄링뿐만 아니라 메모리 관리, 파일 시스템, 디스크 스케줄링 등 운영체제의 여러 요소를 통합적으로 학습할 수 있는 도구로, 주로 대학교 운영체제 강의에서 활용된다. 특히 그래픽 기반의 사용자 인터페이스를 통해 프로세스의 상태 전이와 CPU 할당 과정을 시각적으로 표현해줌으로써 학습자에게 직관적인 이해를 제공한다.

또한 웹 기반의 시뮬레이터도 널리 활용되고 있다. 예를 들어 CS-Scheduler, GeeksforGeeks에서 제공하는 CPU Scheduling Simulator와 같은 도구들은 사용자가 웹 브라우저에서 직접 프로세스 도착 시간, 실행 시간 등을 입력하면 FCFS, SJF, RR 등의 결과를 실시간으로 시각화해주는 기능을 제공한다. 이러한 웹 도구들은 접근성이 뛰어나며, 간단한 실험과 비교 분석에 적합하다.

이외에도 LittleOS, NachOS와 같은 교육용 운영체제 프로젝트에서는 사용자가 직접 스케줄러 알고리즘을 구현하도록 유도한다. 이러한 프로젝트는 실제 운영체제 커널과 유사한 구조를 갖추고 있으며, 학생들이 프로세스 관리, 시스템 콜 처리, 스케줄링 등 저수준 시스템 구조를 직접 설계하고 수정할 수 있게 한다.

이러한 기존 도구들과 달리, 본인이 구현한 시뮬레이터는 스케줄링 알고리즘 자체의 논리와 성능 비교에 집중할 수 있도록 설계되었다. GUI 없이 CLI 환경에서 작동하며, 동일한 조건에서 여러 스케줄링 알고리즘을 동시에 실행하고 그 결과를 정량적으로 비교할 수 있도록 구성되어 있다. 특히 프로세스 생성의 무작위성과 모듈화된 알고리즘 구조를 통해 실험 중심의 학습에 적합하며, 알고리즘의 내부 동작을 직접 분석하고 수정하는 데 용이한 특징을 지닌다.

스케줄러 시스템 구성도



1. FCFS (First-Come First-Serve)

수도코드01 – 부록 참고

설명

FCFS는 도착한 순서대로 프로세스를 처리하는 가장 단순한 스케줄링 알고리즘이다. 구현은 시간 단위로 시뮬레이션이 진행되며, 현재 시간에 도착한 프로세스를 ready queue에 추가하고, CPU가 비어 있을 때 가장 먼저 도착한 프로세스를 선택하여 실행한다. 프로세스 실행 중 I/O 요청이 발생하면 해당 프로세스는 I/O 큐로 이동하며, 실행이 끝난 프로세스는 turnaround time과 waiting time이 기록된다. 이 알고리즘은 공정성은 보장되나, 짧은 작업이 긴 작업 뒤에 배치되었을 경우 기다려야 하는 문제가 있다.

2. SJF (Shortest Job First)

수도코드02 - 부록 참고

설명

SJF는 실행 시간이 가장 짧은 프로세스를 우선 선택하는 방식이다. 비선점형이므로, 한 번 선택된 프로세스는 종료될 때까지 CPU를 점유한다. 시뮬레이션은 FCFS와 유사하게 시간 단위로 진행되며, ready queue에 있는 프로세스 중에서 남은 실행 시간이 가장 짧은 것을 선택해 실행한다. 대기 시간은 실행 시작 시점과 도착 시점의 차이로 계산되며, 프로세스는 종료 시 turnaround time과 waiting time을 기록한다. SJF는 평균 대기 시간을 최소화할 수 있는 이점이 있지만, 긴 작업이 계속 밀려나는 starvation 문제가 존재한다.

3. PSJF (Preemptive SJF)

수도코드03 - 부록 참고

설명

PSJF는 SJF의 선점형 버전으로, 매 시간마다 ready queue를 확인하여 실행 시간이 가장 짧은 프로세스를 선택한다. 현재 실행 중인 프로세스보다 더 짧은 실행 시간을 가진 프로세스가 도착하면 현재 프로세스를 중단하고 새로운 프로세스로 교체된다. I/O 요청이나 종료 시점에도 마찬가지로 CPU가 비워지고, 다음 시간에 다시 가장 짧은 작업을 찾아 실행한다. 이 알고리즘은 작업 시간의 효율적 분배에 강점을 가지지만, 컨텍스트 스위칭이 자주 발생할 수 있어 오버헤드가 증가할 수 있다.

4. Priority Scheduling (Non-preemptive)

수도코드04 - 부록 참고

설명

Priority Scheduling은 프로세스마다 우선순위 값을 두고, 낮은 숫자를 더 높은 우선순위로 간주하여 높은 우선순위를 가진 프로세스부터 실행한다. 비선점형이기 때문에, 우선순위가 높은 프로세스가 ready queue에 있어도 현재 실행 중인 프로세스가 끝날 때까지 기다려야 한다. 각 시간마다 도착한 프로세스를 큐에 넣고, CPU가 비어 있으면 우선순위가 가장 높은 프로세스를 선택한다. 우선순위에 따라 CPU 자원을 배분하므로, 긴급성이 중요한 시스템에서 적합하지만, 우선순위가 낮은 프로세스는 장시간 대기할 수 있는 단점이 있다.

5. Preemptive Priority Scheduling

수도코드05 - 부록 참고

설명

선점형 우선순위 스케줄링은 도착한 프로세스 중에서 우선순위가 가장 높은 프로세스를 항상 실행하도록 하며, 기존 실행 중인 프로세스보다 더 높은 우선순위의 프로세스가 도착하면 즉시 선점한다. 매 시간마다 ready queue를 확인하며, 실행 중인 프로세스보다 높은 우선순위의 프로세스가 존재하는 경우 교체한다. 이를 통해 우선순위가 높은 작업이 즉시 처리될 수 있는 장점이 있으며, 응답성이 중요한 환경에서 효과적이다. 반면 우선순위가 낮은 작업은 실행 기회를 거의 갖지 못하는 starvation 문제가 발생할 수 있다.

6. Round Robin (RR)

수도코드06 - 부록 참고

설명

RR은 각 프로세스에 동일한 타임퀀텀(time quantum)을 부여하여 공정하게 CPU를 분배하는 선점형 알고리즘이다. 실행 도중 타임퀀텀이 소진되면 현재 프로세스를 ready queue의 뒤로 보내고, 다음 프로세스를 실행한다. 도착한 프로세스는 순서대로 큐에 들어가며, 매 시간마다 타임퀀텀을 감소시키고, 타임퀀텀이 0이 되면 교체가 발생한다. 이 알고리즘은 공정성과 응답 시간을 고르게 유지할 수 있으나, 타임퀀텀이 너무 작으면 컨텍스트 스위칭이 과도하게 발생하여 효율이 저하될 수 있다.

실행 결과

1. 프로세스 생성

```
yoonseo@gim-yunseoui-MacBookPro-3 cpu-scheduling % ./cpu_scheduler
Enter number of processes: 3
[Process 1] Arrival Time: 6, CPU Burst Time: 13, Priority: 3, IO Count: 2
    └ IO Request 1: Time = 6, Burst = 1
    └ IO Request 2: Time = 13, Burst = 3
[Process 2] Arrival Time: 5, CPU Burst Time: 6, Priority: 1, IO Count: 1
    └ IO Request 1: Time = 4, Burst = 1
[Process 3] Arrival Time: 2, CPU Burst Time: 14, Priority: 2, IO Count: 3
    └ IO Request 1: Time = 3, Burst = 2
    └ IO Request 2: Time = 6, Burst = 3
    └ IO Request 3: Time = 5, Burst = 1
```

2. FCFS (First-Come First-Serve)

```
================ FCFS Scheduling ================
time 0: [Ready Queue] (empty)
time 0: CPU is idle
time 1: [Ready Queue] (empty)
time 1: CPU is idle
time 2: Process 3 arrived
time 2: [Ready Queue] P3(RT=14, Pri=2)
time 2: Process 3 starts executing
time 3: [Ready Queue] (empty)
time 4: [Ready Queue] (empty)
time 5: Process 2 arrived
time 5: [Ready Queue] P2(RT=6, Pri=1)
time 5: Process 3 requests I/O(duration 2)
time 5: Process 2 starts executing
time 6: Process 1 arrived
time 6: [Ready Queue] P1(RT=13, Pri=3)
time 7: Process 3 completes I/O and returns to Ready Queue
time 7: [Ready Queue] P1(RT=13, Pri=3) P3(RT=11, Pri=2)
time 8: [Ready Queue] P1(RT=13, Pri=3) P3(RT=11, Pri=2)
time 9: [Ready Queue] P1(RT=13, Pri=3) P3(RT=11, Pri=2)
time 9: Process 2 requests I/O(duration 1)
time 9: Process 1 starts executing
time 10: Process 2 completes I/O and returns to Ready Queue
time 10: [Ready Queue] P3(RT=11, Pri=2) P2(RT=2, Pri=1)
time 11: [Ready Queue] P3(RT=11, Pri=2) P2(RT=2, Pri=1)
time 12: [Ready Queue] P3(RT=11, Pri=2) P2(RT=2, Pri=1)
time 13: [Ready Queue] P3(RT=11, Pri=2) P2(RT=2, Pri=1)
time 14: [Ready Queue] P3(RT=11, Pri=2) P2(RT=2, Pri=1)
time 15: [Ready Queue] P3(RT=11, Pri=2) P2(RT=2, Pri=1)
time 15: Process 1 requests I/O(duration 1)
time 15: Process 3 starts executing
time 16: Process 1 completes I/O and returns to Ready Queue
time 16: [Ready Queue] P2(RT=2, Pri=1) P1(RT=7, Pri=3)
time 17: [Ready Queue] P2(RT=2, Pri=1) P1(RT=7, Pri=3)
time 17: Process 3 requests I/O(duration 1)
time 17: Process 2 starts executing
time 18: Process 3 completes I/O and returns to Ready Queue
time 18: [Ready Queue] P1(RT=7, Pri=3) P3(RT=9, Pri=2)
time 19: Process 2 completed(turnaround: 14, waiting: 7)
time 19: [Ready Queue] P1(RT=7, Pri=3) P3(RT=9, Pri=2)
time 19: Process 1 starts executing
time 20: [Ready Queue] P3(RT=9, Pri=2)
time 21: [Ready Queue] P3(RT=9, Pri=2)
time 22: [Ready Queue] P3(RT=9, Pri=2)
time 23: [Ready Queue] P3(RT=9, Pri=2)
time 24: [Ready Queue] P3(RT=9, Pri=2)
time 25: [Ready Queue] P3(RT=9, Pri=2)
time 26: Process 1 completed(turnaround: 20, waiting: 6)
time 26: [Ready Queue] P3(RT=9, Pri=2)
time 26: Process 3 starts executing
time 27: [Ready Queue] (empty)
time 27: Process 3 requests I/O(duration 3)
time 27: CPU is idle
time 28: [Ready Queue] (empty)
time 28: CPU is idle
time 29: [Ready Queue] (empty)
time 29: CPU is idle
time 30: Process 3 completes I/O and returns to Ready Queue
time 30: [Ready Queue] P3(RT=8, Pri=2)
time 30: Process 3 starts executing
time 31: [Ready Queue] (empty)
time 32: [Ready Queue] (empty)
time 33: [Ready Queue] (empty)
time 34: [Ready Queue] (empty)
time 35: [Ready Queue] (empty)
time 36: [Ready Queue] (empty)
time 37: [Ready Queue] (empty)
time 38: Process 3 completed(turnaround: 36, waiting: 16)
```

3. SJF (Shortest Job First)

```
================ Non-preemptive SJF Scheduling ================
time 0: [Ready Queue] (empty)
time 0: CPU is idle
time 1: [Ready Queue] (empty)
time 1: CPU is idle
time 2: Process 3 arrived
time 2: [Ready Queue] P3(RT=14, Pri=2)
time 2: Selected P3 as shortest job (RT=14)
time 2: Process 3 starts executing
time 3: [Ready Queue] (empty)
time 4: [Ready Queue] (empty)
time 5: Process 2 arrived
time 5: [Ready Queue] P2(RT=6, Pri=1)
time 5: Process 3 requests I/O(duration 2)
time 5: Selected P2 as shortest job (RT=6)
time 5: Process 2 starts executing
time 6: Process 1 arrived
time 6: [Ready Queue] P1(RT=13, Pri=3)
time 7: Process 3 completes I/O and returns to Ready Queue
time 7: [Ready Queue] P1(RT=13, Pri=3) P3(RT=11, Pri=2)
time 8: [Ready Queue] P1(RT=13, Pri=3) P3(RT=11, Pri=2)
time 9: [Ready Queue] P1(RT=13, Pri=3) P3(RT=11, Pri=2)
time 9: Process 2 requests I/O(duration 1)
time 9: Selected P3 as shortest job (RT=11)
time 9: Process 3 starts executing
time 10: Process 2 completes I/O and returns to Ready Queue
time 10: [Ready Queue] P1(RT=13, Pri=3) P2(RT=2, Pri=1)
time 11: [Ready Queue] P1(RT=13, Pri=3) P2(RT=2, Pri=1)
time 11: Process 3 requests I/O(duration 1)
time 11: Selected P2 as shortest job (RT=2)
time 11: Process 2 starts executing
time 12: Process 3 completes I/O and returns to Ready Queue
time 12: [Ready Queue] P1(RT=13, Pri=3) P3(RT=9, Pri=2)
time 13: Process 2 completed(turnaround: 8, waiting: 1)
time 13: [Ready Queue] P1(RT=13, Pri=3) P3(RT=9, Pri=2)
time 13: Selected P3 as shortest job (RT=9)
time 13: Process 3 starts executing
time 14: [Ready Queue] P1(RT=13, Pri=3)
time 14: Process 3 requests I/O(duration 3)
time 14: Selected P1 as shortest job (RT=13)
time 14: Process 1 starts executing
time 15: [Ready Queue] (empty)
time 16: [Ready Queue] (empty)
time 17: Process 3 completes I/O and returns to Ready Queue
time 17: [Ready Queue] P3(RT=8, Pri=2)
time 18: [Ready Queue] P3(RT=8, Pri=2)
time 19: [Ready Queue] P3(RT=8, Pri=2)
time 20: [Ready Queue] P3(RT=8, Pri=2)
time 20: Process 1 requests I/O(duration 1)
time 20: Selected P3 as shortest job (RT=8)
time 20: Process 3 starts executing
time 21: Process 1 completes I/O and returns to Ready Queue
time 21: [Ready Queue] P1(RT=7, Pri=3)
time 22: [Ready Queue] P1(RT=7, Pri=3)
time 23: [Ready Queue] P1(RT=7, Pri=3)
time 24: [Ready Queue] P1(RT=7, Pri=3)
time 25: [Ready Queue] P1(RT=7, Pri=3)
time 26: [Ready Queue] P1(RT=7, Pri=3)
time 27: [Ready Queue] P1(RT=7, Pri=3)
time 28: Process 3 completed(turnaround: 26, waiting: 6)
time 28: [Ready Queue] P1(RT=7, Pri=3)
time 28: Selected P1 as shortest job (RT=7)
time 28: Process 1 starts executing
time 29: [Ready Queue] (empty)
time 30: [Ready Queue] (empty)
time 31: [Ready Queue] (empty)
time 32: [Ready Queue] (empty)
time 33: [Ready Queue] (empty)
time 34: [Ready Queue] (empty)
time 35: Process 1 completed(turnaround: 29, waiting: 15)
```

4. PSJF (Preemptive SJF)

```
============== Preemptive SJF Scheduling ==============

time 0: [Ready Queue] (empty)
time 0: CPU is idle
time 1: [Ready Queue] (empty)
time 1: CPU is idle
time 2: Process 3 arrived
time 2: [Ready Queue] P3(RT=14, Pri=2)
time 2: Selected P3 as shortest job (RT=14)
time 2: Process 3 starts executing
time 3: [Ready Queue] (empty)
time 4: [Ready Queue] (empty)
time 5: Process 2 arrived
time 5: [Ready Queue] P2(RT=6, Pri=1)
time 5: Process 3 requests I/O(duration 2)
time 5: Selected P2 as shortest job (RT=6)
time 5: Process 2 starts executing
time 6: Process 1 arrived
time 6: [Ready Queue] P1(RT=13, Pri=3)
time 6: Selected P1 as shortest job (RT=13)
time 6: Process 1 starts executing
time 7: Process 3 completes I/O and returns to Ready Queue
time 7: [Ready Queue] P2(RT=5, Pri=1) P3(RT=11, Pri=2)
time 7: Selected P2 as shortest job (RT=5)
time 7: Process 2 starts executing
time 8: [Ready Queue] P3(RT=11, Pri=2) P1(RT=12, Pri=3)
time 8: Selected P3 as shortest job (RT=11)
time 8: Process 3 starts executing
time 9: [Ready Queue] P1(RT=12, Pri=3) P2(RT=4, Pri=1)
time 9: Selected P2 as shortest job (RT=4)
time 9: Process 2 starts executing
time 10: [Ready Queue] P1(RT=12, Pri=3) P3(RT=10, Pri=2)
time 10: Selected P3 as shortest job (RT=10)
time 10: Process 3 starts executing
time 11: [Ready Queue] P1(RT=12, Pri=3) P2(RT=3, Pri=1)
time 11: Process 3 requests I/O(duration 1)
time 11: Selected P2 as shortest job (RT=3)
time 11: Process 2 starts executing
time 12: Process 3 completes I/O and returns to Ready Queue
time 12: [Ready Queue] P1(RT=12, Pri=3) P3(RT=9, Pri=2)
time 12: Process 2 requests I/O(duration 1)
time 12: Selected P3 as shortest job (RT=9)
time 12: Process 3 starts executing
time 13: Process 2 completes I/O and returns to Ready Queue
time 13: [Ready Queue] P1(RT=12, Pri=3) P2(RT=2, Pri=1)
time 13: Process 3 requests I/O(duration 3)
time 13: Selected P2 as shortest job (RT=2)
time 13: Process 2 starts executing
time 14: [Ready Queue] P1(RT=12, Pri=3)
time 14: Selected P1 as shortest job (RT=12)
time 14: Process 1 starts executing
time 15: [Ready Queue] P2(RT=1, Pri=1)
time 15: Selected P2 as shortest job (RT=1)
time 15: Process 2 starts executing
time 16: Process 2 completed(turnaround: 11, waiting: 14)
time 16: Process 3 completes I/O and returns to Ready Queue
time 16: [Ready Queue] P1(RT=11, Pri=3) P3(RT=8, Pri=2)
time 16: Selected P3 as shortest job (RT=8)
time 16: Process 3 starts executing
time 17: [Ready Queue] P1(RT=11, Pri=3)
time 17: Selected P1 as shortest job (RT=11)
time 17: Process 1 starts executing
time 18: [Ready Queue] P3(RT=7, Pri=2)
time 18: Selected P3 as shortest job (RT=7)
time 18: Process 3 starts executing
time 19: [Ready Queue] P1(RT=10, Pri=3)
time 19: Selected P1 as shortest job (RT=10)
time 19: Process 1 starts executing
time 20: [Ready Queue] P3(RT=6, Pri=2)
time 20: Selected P3 as shortest job (RT=6)
time 20: Process 3 starts executing
time 21: [Ready Queue] P1(RT=9, Pri=3)
time 21: Selected P1 as shortest job (RT=9)
time 21: Process 1 starts executing
time 22: [Ready Queue] P3(RT=5, Pri=2)
time 22: Selected P3 as shortest job (RT=5)
time 22: Process 3 starts executing
time 23: [Ready Queue] P1(RT=8, Pri=3)
time 23: Selected P1 as shortest job (RT=8)
time 23: Process 1 starts executing
time 24: [Ready Queue] P3(RT=4, Pri=2)
time 24: Process 1 requests I/O(duration 1)
time 24: Selected P3 as shortest job (RT=4)
time 24: Process 3 starts executing
time 25: Process 1 completes I/O and returns to Ready Queue
time 25: [Ready Queue] P1(RT=7, Pri=3)
time 25: Selected P1 as shortest job (RT=7)
time 25: Process 1 starts executing
time 26: [Ready Queue] P3(RT=3, Pri=2)
time 26: Selected P3 as shortest job (RT=3)
time 26: Process 3 starts executing
time 27: [Ready Queue] P1(RT=6, Pri=3)
time 27: Selected P1 as shortest job (RT=6)
time 27: Process 1 starts executing
time 28: [Ready Queue] P3(RT=2, Pri=2)
time 28: Selected P3 as shortest job (RT=2)
time 28: Process 3 starts executing
time 29: [Ready Queue] P1(RT=5, Pri=3)
time 29: Selected P1 as shortest job (RT=5)
time 29: Process 1 starts executing
time 30: [Ready Queue] P3(RT=1, Pri=2)
time 30: Selected P3 as shortest job (RT=1)
time 30: Process 3 starts executing
time 31: Process 3 completed(turnaround: 29, waiting: 60)
time 31: [Ready Queue] P1(RT=4, Pri=3)
time 31: Selected P1 as shortest job (RT=4)
time 31: Process 1 starts executing
time 32: [Ready Queue] (empty)
time 33: [Ready Queue] (empty)
time 34: [Ready Queue] (empty)
time 35: Process 1 completed(turnaround: 29, waiting: 76)
```

5. Priority Scheduling (Non-preemptive)

```
================ Non-preemptive Priotiry Scheduling ================
time 0: [Ready Queue] (empty)
time 0: CPU is idle
time 1: [Ready Queue] (empty)
time 1: CPU is idle
time 2: Process 3 arrived
time 2: [Ready Queue] P3(RT=14, Pri=2)
time 2: Process 3 starts executing (Priority 2)
time 3: [Ready Queue] (empty)
time 4: [Ready Queue] (empty)
time 5: Process 2 arrived
time 5: [Ready Queue] P2(RT=6, Pri=1)
time 5: Process 3 requests I/O(duration 2)
time 5: Process 2 starts executing (Priority 1)
time 6: Process 1 arrived
time 6: [Ready Queue] P1(RT=13, Pri=3)
time 7: Process 3 completes I/O and returns to Ready Queue
time 7: [Ready Queue] P1(RT=13, Pri=3) P3(RT=11, Pri=2)
time 8: [Ready Queue] P1(RT=13, Pri=3) P3(RT=11, Pri=2)
time 9: [Ready Queue] P1(RT=13, Pri=3) P3(RT=11, Pri=2)
time 9: Process 2 requests I/O(duration 1)
time 9: Process 3 starts executing (Priority 2)
time 10: Process 2 completes I/O and returns to Ready Queue
time 10: [Ready Queue] P1(RT=13, Pri=3) P2(RT=2, Pri=1)
time 11: [Ready Queue] P1(RT=13, Pri=3) P2(RT=2, Pri=1)
time 11: Process 3 requests I/O(duration 1)
time 11: Process 2 starts executing (Priority 1)
time 12: Process 3 completes I/O and returns to Ready Queue
time 12: [Ready Queue] P1(RT=13, Pri=3) P3(RT=9, Pri=2)
time 13: Process 2 completed(turnaround: 8, waiting: 1)
time 13: [Ready Queue] P1(RT=13, Pri=3) P3(RT=9, Pri=2)
time 13: Process 3 starts executing (Priority 2)
time 14: [Ready Queue] P1(RT=13, Pri=3)
time 14: Process 3 requests I/O(duration 3)
time 14: Process 1 starts executing (Priority 3)
time 15: [Ready Queue] (empty)
time 16: [Ready Queue] (empty)
time 17: Process 3 completes I/O and returns to Ready Queue
time 17: [Ready Queue] P3(RT=8, Pri=2)
time 18: [Ready Queue] P3(RT=8, Pri=2)
time 19: [Ready Queue] P3(RT=8, Pri=2)
time 20: [Ready Queue] P3(RT=8, Pri=2)
time 20: Process 1 requests I/O(duration 1)
time 20: Process 3 starts executing (Priority 2)
time 21: Process 1 completes I/O and returns to Ready Queue
time 21: [Ready Queue] P1(RT=7, Pri=3)
time 22: [Ready Queue] P1(RT=7, Pri=3)
time 23: [Ready Queue] P1(RT=7, Pri=3)
time 24: [Ready Queue] P1(RT=7, Pri=3)
time 25: [Ready Queue] P1(RT=7, Pri=3)
time 26: [Ready Queue] P1(RT=7, Pri=3)
time 27: [Ready Queue] P1(RT=7, Pri=3)
time 28: Process 3 completed(turnaround: 26, waiting: 6)
time 28: [Ready Queue] P1(RT=7, Pri=3)
time 28: Process 1 starts executing (Priority 3)
time 29: [Ready Queue] (empty)
time 30: [Ready Queue] (empty)
time 31: [Ready Queue] (empty)
time 32: [Ready Queue] (empty)
time 33: [Ready Queue] (empty)
time 34: [Ready Queue] (empty)
time 35: Process 1 completed(turnaround: 29, waiting: 15)
```

6. Preemptive Priority Scheduling

```
================ Preemptive Priority Scheduling ================
time 0: [Ready Queue] (empty)
time 0: CPU is idle
time 1: [Ready Queue] (empty)
time 1: CPU is idle
time 2: Process 3 arrived
time 2: [Ready Queue] P3(RT=14, Pri=2)
time 2: Process 3 starts executing (Priority 2)
time 3: [Ready Queue] (empty)
time 4: [Ready Queue] (empty)
time 5: Process 2 arrived
time 5: [Ready Queue] P2(RT=6, Pri=1)
time 5: Process 3 requests I/O(duration 2)
time 5: Process 2 starts executing (Priority 1)
time 6: Process 1 arrived
time 6: [Ready Queue] P1(RT=13, Pri=3)
time 7: Process 3 completes I/O and returns to Ready Queue
time 7: [Ready Queue] P1(RT=13, Pri=3) P3(RT=11, Pri=2)
time 8: [Ready Queue] P1(RT=13, Pri=3) P3(RT=11, Pri=2)
time 9: [Ready Queue] P1(RT=13, Pri=3) P3(RT=11, Pri=2)
time 9: Process 2 requests I/O(duration 1)
time 9: Process 3 starts executing (Priority 2)
time 10: Process 2 completes I/O and returns to Ready Queue
time 10: [Ready Queue] P1(RT=13, Pri=3) P2(RT=2, Pri=1)
time 10: Process 2 starts executing (Priority 1)
time 11: [Ready Queue] P1(RT=13, Pri=3) P3(RT=10, Pri=2)
time 12: Process 2 completed(turnaround: 7, waiting: 0)
time 12: [Ready Queue] P1(RT=13, Pri=3) P3(RT=10, Pri=2)
time 12: Process 3 starts executing (Priority 2)
time 13: [Ready Queue] P1(RT=13, Pri=3)
time 13: Process 3 requests I/O(duration 1)
time 13: Process 1 starts executing (Priority 3)
time 14: Process 3 completes I/O and returns to Ready Queue
time 14: [Ready Queue] P3(RT=9, Pri=2)
time 14: Process 3 starts executing (Priority 2)
time 15: [Ready Queue] P1(RT=12, Pri=3)
time 15: Process 3 requests I/O(duration 3)
time 15: Process 1 starts executing (Priority 3)
time 16: [Ready Queue] (empty)
time 17: [Ready Queue] (empty)
time 18: Process 3 completes I/O and returns to Ready Queue
time 18: [Ready Queue] P3(RT=8, Pri=2)
time 18: Process 3 starts executing (Priority 2)
time 19: [Ready Queue] P1(RT=9, Pri=3)
time 20: [Ready Queue] P1(RT=9, Pri=3)
time 21: [Ready Queue] P1(RT=9, Pri=3)
time 22: [Ready Queue] P1(RT=9, Pri=3)
time 23: [Ready Queue] P1(RT=9, Pri=3)
time 24: [Ready Queue] P1(RT=9, Pri=3)
time 25: [Ready Queue] P1(RT=9, Pri=3)
time 26: Process 3 completed(turnaround: 24, waiting: 7)
time 26: [Ready Queue] P1(RT=9, Pri=3)
time 26: Process 1 starts executing (Priority 3)
time 27: [Ready Queue] (empty)
time 28: [Ready Queue] (empty)
time 28: Process 1 requests I/O(duration 1)
time 28: CPU is idle
time 29: Process 1 completes I/O and returns to Ready Queue
time 29: [Ready Queue] P1(RT=7, Pri=3)
time 29: Process 1 starts executing (Priority 3)
time 30: [Ready Queue] (empty)
time 31: [Ready Queue] (empty)
time 32: [Ready Queue] (empty)
time 33: [Ready Queue] (empty)
time 34: [Ready Queue] (empty)
time 35: [Ready Queue] (empty)
time 36: Process 1 completed(turnaround: 30, waiting: 36)
```

7. Round Robin (RR)

```
========= Round Robin Scheduling (Time Quantum = 2) =========
time 0: [Ready Queue] (empty)
time 0: CPU is idle
time 1: [Ready Queue] (empty)
time 1: CPU is idle
time 2: Process 3 arrived
time 2: [Ready Queue] P3(RT=14, Pri=2)
time 2: Process 3 starts executing
time 3: [Ready Queue] (empty)
time 4: [Ready Queue] (empty)
time 5: Process 2 arrived
time 5: [Ready Queue] P2(RT=6, Pri=1)
time 5: Process 3 requests I/O(duration 2)
time 5: Process 2 starts executing
time 6: Process 1 arrived
time 6: [Ready Queue] P1(RT=13, Pri=3)
time 7: Process 3 completes I/O and returns to Ready Queue
time 7: [Ready Queue] P1(RT=13, Pri=3) P3(RT=11, Pri=2)
time 7: Time quantum expired. Switching from Process 2
time 7: Process 1 starts executing
time 8: [Ready Queue] P3(RT=11, Pri=2) P2(RT=4, Pri=1)
time 9: [Ready Queue] P3(RT=11, Pri=2) P2(RT=4, Pri=1)
time 9: Time quantum expired. Switching from Process 1
time 9: Process 3 starts executing
time 10: [Ready Queue] P2(RT=4, Pri=1) P1(RT=11, Pri=3)
time 11: [Ready Queue] P2(RT=4, Pri=1) P1(RT=11, Pri=3)
time 11: Process 3 requests I/O(duration 1)
time 11: Process 2 starts executing
time 12: Process 3 completes I/O and returns to Ready Queue
time 12: [Ready Queue] P1(RT=11, Pri=3) P3(RT=9, Pri=2)
time 13: [Ready Queue] P1(RT=11, Pri=3) P3(RT=9, Pri=2)
time 13: Process 2 requests I/O(duration 1)
time 13: Process 1 starts executing
time 14: Process 2 completes I/O and returns to Ready Queue
time 14: [Ready Queue] P3(RT=9, Pri=2) P2(RT=2, Pri=1)
time 15: [Ready Queue] P3(RT=9, Pri=2) P2(RT=2, Pri=1)
time 15: Time quantum expired. Switching from Process 1
time 15: Process 3 starts executing
time 16: [Ready Queue] P2(RT=2, Pri=1) P1(RT=9, Pri=3)
time 16: Process 3 requests I/O(duration 3)
time 16: Process 2 starts executing
time 17: [Ready Queue] P1(RT=9, Pri=3)
time 18: Process 2 completed(turnaround: 13, waiting: 8)
time 18: [Ready Queue] P1(RT=9, Pri=3)
time 18: Process 1 starts executing
time 19: Process 3 completes I/O and returns to Ready Queue
time 19: [Ready Queue] P3(RT=8, Pri=2)
time 20: [Ready Queue] P3(RT=8, Pri=2)
time 20: Process 1 requests I/O(duration 1)
time 20: Process 3 starts executing
time 21: Process 1 completes I/O and returns to Ready Queue
time 21: [Ready Queue] P1(RT=7, Pri=3)
time 22: [Ready Queue] P1(RT=7, Pri=3)
time 22: Time quantum expired. Switching from Process 3
time 22: Process 1 starts executing
time 23: [Ready Queue] P3(RT=6, Pri=2)
time 24: [Ready Queue] P3(RT=6, Pri=2)
time 24: Time quantum expired. Switching from Process 1
time 24: Process 3 starts executing
time 25: [Ready Queue] P1(RT=5, Pri=3)
time 26: [Ready Queue] P1(RT=5, Pri=3)
time 26: Time quantum expired. Switching from Process 3
time 26: Process 1 starts executing
time 27: [Ready Queue] P3(RT=4, Pri=2)
time 28: [Ready Queue] P3(RT=4, Pri=2)
time 28: Time quantum expired. Switching from Process 1
time 28: Process 3 starts executing
time 29: [Ready Queue] P1(RT=3, Pri=3)
time 30: [Ready Queue] P1(RT=3, Pri=3)
time 30: Time quantum expired. Switching from Process 3
time 30: Process 1 starts executing
time 31: [Ready Queue] P3(RT=2, Pri=2)
time 32: [Ready Queue] P3(RT=2, Pri=2)
time 32: Time quantum expired. Switching from Process 1
time 32: Process 3 starts executing
time 33: [Ready Queue] P1(RT=1, Pri=3)
time 34: Process 3 completed(turnaround: 32, waiting: 33)
time 34: [Ready Queue] P1(RT=1, Pri=3)
time 34: Process 1 starts executing
time 35: Process 1 completed(turnaround: 29, waiting: 48)
```

알고리즘간 성능 비교

1. FCFS (First-Come First-Serve)

```
======== Waiting Times ========
Waiting Time of Process 1: 6
Waiting Time of Process 2: 7
Waiting Time of Process 3: 16

======== Turnaround Times ========
Turnaround Time of Process 1: 20
Turnaround Time of Process 2: 14
Turnaround Time of Process 3: 36

======== [FCFS] Average Times ========
Average Waiting Time: 9.67
Average Turnaround Time: 23.33
```

2. SJF (Shortest Job First)

```
======== Waiting Times ========
Waiting Time of Process 1: 15
Waiting Time of Process 2: 1
Waiting Time of Process 3: 6

======== Turnaround Times ========
Turnaround Time of Process 1: 29
Turnaround Time of Process 2: 8
Turnaround Time of Process 3: 26

======== [Non-preemptive SJF] Average Times ========
Average Waiting Time: 7.33
Average Turnaround Time: 21.00
```

3. PSJF (Preemptive SJF)

```
======== Waiting Times ========
Waiting Time of Process 1: 76
Waiting Time of Process 2: 14
Waiting Time of Process 3: 60

======== Turnaround Times ========
Turnaround Time of Process 1: 29
Turnaround Time of Process 2: 11
Turnaround Time of Process 3: 29

======== [Preemptive SJF] Average Times ========
Average Waiting Time: 50.00
Average Turnaround Time: 23.00
```

4. Priority Scheduling (Non-preemptive)

```
======== Waiting Times ========
Waiting Time of Process 1: 15
Waiting Time of Process 2: 1
Waiting Time of Process 3: 6

======== Turnaround Times ========
Turnaround Time of Process 1: 29
Turnaround Time of Process 2: 8
Turnaround Time of Process 3: 26

======== [Non-preemptive Priotiry] Average Times ========
Average Waiting Time: 7.33
Average Turnaround Time: 21.00
```

5. Preemptive Priority Scheduling

```
======== Waiting Times ========
Waiting Time of Process 1: 36
Waiting Time of Process 2: 0
Waiting Time of Process 3: 7

======== Turnaround Times ========
Turnaround Time of Process 1: 30
Turnaround Time of Process 2: 7
Turnaround Time of Process 3: 24

======== [Preemptive Priority] Average Times ========
Average Waiting Time: 14.33
Average Turnaround Time: 20.33
```

6. Round Robin (RR)

```
======== Waiting Times ========
Waiting Time of Process 1: 48
Waiting Time of Process 2: 8
Waiting Time of Process 3: 33

======== Turnaround Times ========
Turnaround Time of Process 1: 29
Turnaround Time of Process 2: 13
Turnaround Time of Process 3: 32

======== [Round Robin] Average Times ========
Average Waiting Time: 29.67
Average Turnaround Time: 24.67
```

7. 분석

실험 결과를 통해 각 스케줄링 알고리즘의 성능을 정량적으로 비교할 수 있었다. 주요 평가지표로는 평균 대기 시간(Average Waiting Time)과 평균 반환 시간(Average Turnaround Time)을 사용하였으며, 이는 프로세스가 얼마나 오랫동안 CPU를 기다렸는지와 전체 수행 완료까지 걸린 총 시간을 각각 나타낸다.

가장 기본적인 방식인 FCFS(First-Come First-Serve) 알고리즘은 평균 대기 시간이 9.67, 평균 반환 시간이 23.33으로 측정되었다. 이는 순서대로 실행함으로써 구현이 간단하고 공정성을 보장하지만, 짧은 작업이 긴 작업 뒤에 배치될 경우 오랫동안 대기해야 하는 단점이 있음을 보여준다.

Non-preemptive SJF(Shortest Job First) 알고리즘은 평균 대기 시간 7.33, 평균 반환 시간 21.00으로 FCFS보다 더 나은 성능을 보였다. 이는 짧은 작업을 우선 실행함으로써 전체적인 대기 시간을 줄이는 효과가 있었기 때문이며, 작업 시간이 미리 알려진 상황에서는 효율적인 선택임을 나타낸다.

하지만 Preemptive SJF의 경우, 예상과 달리 평균 대기 시간이 50.00으로 매우 높게 나타났고, 평균 반환 시간도 23.00으로 그다지 효율적이지 못한 결과를 보였다. 이는 선점이 과도하게 발생하거나 I/O 요청으로 인해 큐가 불안정해질 경우 오히려 전체 처리 흐름이 지연될 수 있음을 시사한다. 또한 컨텍스트 스위칭의 오버헤드가 영향을 미쳤을 가능성도 존재한다.

Non-preemptive Priority 알고리즘은 SJF와 유사한 결과를 보였으며, 평균 대기 시간 7.33, 평균 반환 시간 21.00으로 측정되었다. 우선순위를 기준으로 처리함으로써 중요한 작업을 빠르게 실행할 수 있는 장점이 있으나, 선점이 없는 특성상 높은 우선순위가 즉시 반영되지 못할 수 있다.

Preemptive Priority는 평균 대기 시간 14.33, 평균 반환 시간 20.33으로 대기 시간은 다소 증가했지만 반환 시간은 모든 알고리즘 중 가장 낮았다. 이는 높은 우선순위의 프로세스를 즉시 처리함으로써 응답 시간을 개선하는 효과가 있었음을 의미한다. 그러나 낮은 우선순위의 프로세스가 계속 밀려나는 starvation 현상이 발생할 수 있는 구조이다.

마지막으로, Round Robin은 평균 대기 시간 29.67, 평균 반환 시간 24.67로 측정되어 전반적으로 다소 낮은 효율을 보였다. 이는 모든 프로세스에게 동일한 실행 시간을 보장해 공정성은 확보되지만, 타임 퀀텀이 짧아질수록 컨텍스트 스위칭 횟수가 증가하여 오버헤드가 커지기 때문이다.

전체적으로 살펴보면, SJF 및 Non-preemptive Priority 방식은 비교적 안정적으로 낮은 대기 시간과 반환 시간을 제공하였고, Preemptive Priority는 응답 성능이 우수하나 낮은 우선순위 프로세스에게 불리할 수 있었다. 반면, Preemptive SJF와 RR은 상황에 따라 오히려 처리 효율이 감소할 수 있음을 보여주었다.

결론

구현한 스케줄러에 대한 정리

본 프로젝트에서는 다양한 CPU 스케줄링 알고리즘을 통합적으로 비교 분석할 수 있는 시뮬레이터를 직접 구현하였다. 시뮬레이터는 모듈화된 구조로 설계되어 있어, 각 구성 요소가 명확히 분리되어 있으며 유지보수 및 확장이 용이하다. main.c에서는 전반적인 실행 흐름을 관리하며, config.c를 통해 초기 설정값을 로드하고, process.c에서 무작위로 생성된 프로세스들을 기반으로 실험을 시작한다. 생성된 프로세스들은 clone_processes 함수를 통해 복사되어 각각의 스케줄링 알고리즘에서 동일한 입력 조건 하에 독립적으로 실행된다. 스케줄링 알고리즘은 schedule.c 내부에 구현되어 있으며, FCFS, Non-preemptive SJF, Preemptive SJF, Non-preemptive Priority, Preemptive Priority, Round Robin 등 총 여섯 가지 방식이 포함되어 있다. 각 알고리즘은 프로세스의 도착 시간, 실행 시간, 우선순위 등을 고려하여 스케줄링을 수행하고, queue.c와 io.c 모듈을 통해 ready queue와 I/O 큐의 상태를 관리한다. I/O 요청은 실시간으로 반영되며, 해당 프로세스는 waiting queue로 이동 후, 일정 시간이 지난 후 다시 ready queue에 복귀하는 방식으로 처리된다. 모든 알고리즘의 실행이 종료되면, evaluation.c를 통해 각 프로세스의 대기 시간과 반환 시간 등 주요 성능 지표가 계산된다. 이를 통해 알고리즘 간의 효율성과 특징을 정량적으로 비교할 수 있도록 하였다.

프로젝트 수행 소감 및 향후 발전방향

이번 프로젝트를 통해 CPU 스케줄링 알고리즘의 내부 동작 원리를 직접 구현해보며, 운영체제의 핵심 개념 중 하나인 프로세스 스케줄링에 대한 이해를 한층 깊이 있게 다질 수 있었다. 단순히 이론적으로 알고 있던 스케줄링 알고리즘들을 실제 코드로 작성하고, 각 알고리즘이 어떻게 프로

세스를 선택하고 실행 순서를 결정하는지 체계적으로 분석함으로써, 각 알고리즘이 가지는 구조적 특징과 장단점을 명확히 체감할 수 있었다.

특히, 동일한 조건에서 여러 알고리즘을 비교하기 위해 프로세스를 무작위로 생성하고 이를 동일하게 복제하여 각 알고리즘에 적용하는 구조를 설계한 점은, 실험의 공정성과 결과 분석의 신뢰성을 높이는 데 기여하였다. 또한, I/O 요청 처리, ready queue와 waiting queue 간의 전이, 선점 조건 구현 등 실제 운영체제와 유사한 상황을 다루는 과정에서 시스템 자원 관리의 복잡성과 중요성을 실감할 수 있었다.

향후 발전 방향으로는, 시뮬레이터에 GUI(그래픽 사용자 인터페이스)를 추가하여 프로세스의 상태 전이를 시각적으로 표현함으로써, 학습 도구로서의 활용성을 높일 수 있을 것이다. 또한 다중 CPU 환경에서의 멀티코어 스케줄링, 실시간 스케줄링 알고리즘(RMS, EDF 등), 또는 aging 기법과 같은 starvation 방지 기법을 추가하여 보다 실제적인 시나리오를 반영한 실험이 가능하도록 확장할 수 있다.

전반적으로 이번 프로젝트는 운영체제 이론과 실제 구현 간의 간극을 효과적으로 좁힐 수 있는 유익한 경험이었다. 복잡한 시스템 구조를 직접 설계하고 시뮬레이션해보는 과정을 통해 문제 해결 능력은 물론, 시스템 전반에 대한 구조적 사고력도 함께 키울 수 있었다.

참고문헌

1. Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts* (10th ed.). Wiley.
2. Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems* (4th ed.). Pearson.
3. GeeksforGeeks. "CPU Scheduling Algorithms."
https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/
4. TutorialsPoint. "Operating System - CPU Scheduling."
https://www.tutorialspoint.com/operating_system/os_process_scheduling.htm
5. GitHub – OSSim: Operating System Simulation Tool.
https://github.com/ossimlabs/ossim

부록

수도코드01

```
function run_fcfs(processes):
    time ← 0
    while all processes not completed:
        for p in processes:
            if p.arrival_time == time:
                enqueue ready_queue with p
        update_waiting_queue(time)
        if current_process requests I/O:
            move to waiting queue
        if no current_process and ready_queue not empty:
            current_process ← dequeue from ready_queue
        execute current_process for 1 unit
        if current_process completed:
            record turnaround and waiting time
            current_process ← NULL
        time++
```

수도코드02

```
function run_sjf(processes):
    time ← 0
    while all processes not completed:
        for p in processes arriving at time:
            enqueue ready_queue with p
        update_waiting_queue(time)
        if current_process requests I/O:
            current_process ← NULL
        if no current_process:
            current_process ← process with shortest remaining time
        execute current_process
        if current_process completed:
            record stats
            current_process ← NULL
        time++
```

수도코드03

```
function run_psjf(processes):
    time ← 0
    while all processes not completed:
        for p in processes arriving at time:
            enqueue ready_queue with p
        update_waiting_queue(time)
        if current_process requests I/O:
            current_process ← NULL
        shortest ← process with least remaining_time
        if shortest != current_process:
            requeue current_process if needed
            current_process ← shortest
        execute current_process
        if current_process completed:
            record stats
            current_process ← NULL
        time++
```

수도코드04

```
function run_priority(processes):
    time ← 0
    while all processes not completed:
        for p in processes arriving at time:
            enqueue ready_queue with p
        update_waiting_queue(time)
        if current_process requests I/O:
            current_process ← NULL
        if no current_process:
            current_process ← highest_priority_process
        execute current_process
        if current_process completed:
            record stats
            current_process ← NULL
        time++
```

수도코드05

```
function run_preemptive_priority(processes):
    time ← 0
    while all processes not completed:
        for p in processes arriving at time:
            enqueue ready_queue with p
        update_waiting_queue(time)
        if current_process requests I/O:
            current_process ← NULL
        highest ← process with highest priority
        if highest ≠ current_process:
            requeue current_process if needed
            current_process ← highest
        execute current_process
        if current_process completed:
            record stats
            current_process ← NULL
        time++
```

수도코드06

```
function run_rr(processes):
    time ← 0
    quantum ← TIME_QUANTUM
    while all processes not completed:
        for p in processes arriving at time:
            enqueue ready_queue with p
        update_waiting_queue(time)
        if current_process requests I/O or quantum == 0:
            enqueue current_process
            current_process ← NULL
        if no current_process and ready_queue not empty:
            current_process ← dequeue
            quantum ← TIME_QUANTUM
        execute current_process
        quantum--
        if current_process completed:
            record stats & current_process ← NULL
            quantum ← 0
        time++
```