1. 목적

본 과제는 리눅스의 환경에서, Time Slice에 따른 프로세스들의 CPU 스케줄링 성능을 이해한다. 구체적으로, time slice에 따른 유저 프로그램의 성능 및 문맥전환(Context switching) 오버헤드를 측정하고 분석한다.

Round Robin 스케줄링은 빠른 응답 시간(response time)을 요구하는 Interactive 프로세스에 적합한 기법이다. 본 기법은 프로세스를 전환할 때마다 context switching이 발생하고, 이에 따라 오버헤드가 달라지기 때문에 time slice를 어떻게 설정하는 지에 따라 성능이 달라진다. Time slice가 작을 경우 작업을 빈번하게 전환하기 때문에 context switching이 많이 발생하게 되고, 이는 상당한 오버헤드를 발생시킨다. 반면, time slice가 클 경우, response time이 느려지기 때문에 interactive 프로세스를 만족시키지 못한다.

본 과제에서는 Time Slice를 1ms, 10ms, 100ms 로 변화시켜가며 이에 따른 성능 변화를 관찰한다. 이를 위해 요구하는 본 과제의 내용은 다음과 같다.

(1) 성능 분석을 위한 프로그램을 구현하여, 프로그래밍 역량을 기른다.

(2) 스케줄링 Time slice (time quantum)에 따른 스케줄링 동작을 이해하고, 오버헤드를 분석하는 방법을 살펴본다.

(3) 리눅스 커널 프로그래밍을 통해 성능을 분석하는 방법을 살펴본다.

2. 과제 목표

A. 성능 분석을 위한 유저 프로그램 구현

- CPU 사용을 위해 단순 행렬 연산을 수행하는 C 프로그램을 구현한다 (행렬연산: tutorial 자료 확인)
- 프로그램의 입력으로는 생성할 연산 프로세스의 개수와 각 프로세스가 수행될 시간을 받는다. 이 입력값을 바탕으로 연산 프로세스들을 생성하고 각 프로세스는 입력된 시간만큼 연산을 수행한다.
- 출력값으로, 실행 중에 각 프로세스 별로 행렬 연산을 시작한 시점부터 100ms마다 각 epoch에서 실행한 연산 횟수와 각 epoch의 실제 실행 시간을 출력하고, 해당 프로세스가 종료될 때 해당 프로세스가 연산한 총 횟수와 시간을 출력한다.
- 별도로 CTRL + C를 눌렀을 때 작업 내용을 출력하고 종료하는 Signal handling을 구현했을 경우에는 추가 점수를 부여한다.

B. Time Slice에 따른 행렬연산 성능변화 분석

- 생성한 프로그램에 대한 CPU 스케줄링 방식을 Round Robin으로 변경하고, RR 스케줄러의 Time Slice를 1ms, 10ms, 100ms로 변경하면서 Time Slice 변화에 따른 성능 차이를 분석한다.
  - i. 구체적으로 A에서 구현한 프로그램에서 sched_setattr() 시스템 콜을 사용하여 RT-RR로 변경한다.
  - ii. 특히, 스케줄링 정책은 fork() 이전에 변경하여 child 프로세스가 모두 RT-RR로 스케줄링 되도록 해야 한다.
- 실험: 단위 시간(30초) 동안 fork된 각 프로세스가 수행한 행렬 연산 횟수의 합 측정
  - i. 성능을 명확하게 관찰하기 위해 실험에 사용하는 core 수를 1개로 조정한다.
  - ii. 이상치에 의한 실험 결과 오류를 방지하기 위해 각 time Slice 환경에서 최소 5회 이상 실험을 진행하고 분석한다.
  - iii. 실험 결과를 Figure 1, 2와 같이 표 및 그래프의 형태로 정리하고 분석한다.

RR Time slice	1ms	10ms	100ms
	# of calc.	# of calc.	# of calc.
Process #0	3300	3369	3421
Process #1	3278	3392	3423
Total calc.	6578	6761	6844

Figure1. 프로세스 별 Time Slice에 따른 행렬 연산 횟수

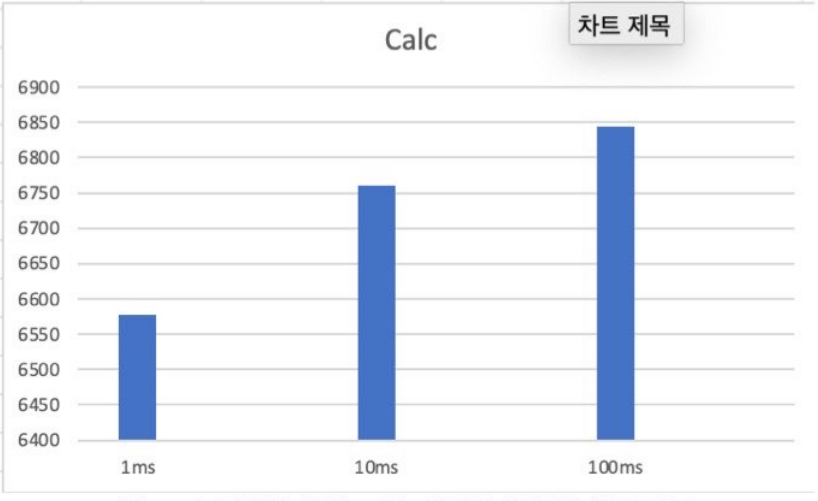

Figure 2. 프로세스 별 Time Slice에 따른 행렬 연산 횟수 그래프

C. CPU Burst Time을 활용한 성능 변화 분석

- 보다 정확한 성능 변화를 분석하기 위해 CPU Burst Time을 바탕으로 단위 시간 당 각 행렬 연산 프로세스가 수행한 행렬 연산량을 분석해보자.
- CPU 수행 시간을 측정하기 위해, 커널 레벨(내부)에서 CPU Burst Time을 측정한다.
  - i. 커널 내부의 sched_info_depart 메소드를 수정하여 내부 CPU Burst Time을 구할 수 있다.
- RR 스케줄러의 Time Slice를 1ms, 10ms, 100ms로 변경하면서 Time Slice 변화에 따른 성능 차이를 분석한다.
- 커널 내부에서는 printk()를 이용해 CPU burst를 출력하고, 그 결과를 dmesg를 통해 확인한다.
- pid를 이용해 프로세스를 구분하고, 엑셀 등의 프로그램을 이용해 프로세스별 CPU Burst의 합계를 계산한다.
- 실험: 단위시간(CPU Burst Time) 대비 행렬 연산 횟수 측정
  - i. 이상치에 의한 실험 결과 오류를 방지하기 위해 각 Time Slice 환경에서 최소 5회 이상 실험을 진행하고 분석한다.
  - ii. Time Slice가 각각 1ms, 10ms, 100ms인 경우에 대해 ./cpu 2 30을 수행하여 행렬 연산 횟수, CPU Burst의 합계(초 단위)를 계산한다.
  - iii. Calculations per minute = # of calc / Time (second) 를 통해 단위 초 당 행렬 연산 횟수를 계산한다.
  - iv. 이를 통해 Time Slice 에 따라 단위 초 당 행렬 연산 횟수가 어떻게 나타나는지 관찰하고, 이를 통해 Context switching 오버헤드를 파악한다.
- 실험 결과를 Figure 3, 4과 같이 표 및 그래프의 형태로 정리하고 분석한다.

RR Time slice	1ms		10ms		100ms	
	# of calc.	Time (s)	# of calc.	Time (s)	# of calc.	Time (s)
Process #0	89957	30.01052501	89684	30.01191832	91032	30.01129014
Process #1	89923	30.01412704	89802	30.01521622	91000	30.0113264
Total calc. and Time	179880	60.02465205	179486	60.02713454	182032	60.02261654

RR Time slice	1ms	10ms	100ms
Calulations per minute	2920	2990.081092	3032.723505
Baseline=1ms	100.00%	102.40%	103.86%
Baseline=10ms	97.66%	100.00%	101.43%

Figure 3. Time Slice에 따른 성능 변화

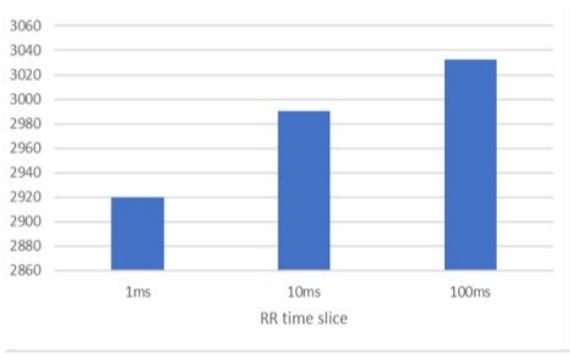

Figure 4. Time Slice에 따른 성능 변화 그래프

3. 제출 방법
   - **A.** Blackboard를 통하여 파일을 제출한다.
   - **B.** 보고서에 아래 내용을 반드시 포함하도록 한다. (보고서는 표지 제외 7페이지를 넘지 않도록 한다)
     - 학과, 학번, 이름, 제출 날짜, Freeday 사용 일수 (표지)
     - 과제 개요 및 Round Robin 스케줄링에서 Time Slice의 영향 설명
     - 작성한 모든 소스코드 및 작업 내용에 대한 설명
       i. 보고서에서 작성한 코드에 대한 설명은 2 페이지 이하로 한다. (위반 시 감점)
       ii. 소스를 그대로 첨부하지 말고, 전체적인 흐름과 수행한 작업을 설명한다.
     - Time Slice 의 변화에 따른 성능을 관찰한 표 및 그림, 결과에 대한 분석
     - 과제 수행 시의 문제점과 해결 과정 또는 의문 사항
     - 각 단계별로 수행한 내용까지라도 반드시 제출할 것
   - **C.** 제출할 파일 목록
     - 보고서
     - 직접 작성, 수정한 소스 파일 전체

- ❍ 결과 표와 그래프 원본 파일 (e.g., 엑셀)

D. 파일 제출 방법

1) 위 3개의 파일을 더하여 “os2_학번_이름.zip“으로 제출한다.
2) 소스파일은 절대로 리눅스 커널 소스 전체를 제출하지 않도록 한다.
3) 소스파일에서 수정한 부분은 “#[학번] [이름]” 형태로 주석을 달아 수정한 부분을 알아볼 수 있도록 표시한다.
4) 동일한 환경에서 컴파일과 실행이 가능해야 한다.

4. 제한사항
   A. 리눅스 배포판 및 커널 버전은 과제에서 명시된 버전을 사용한다.

5. 비고
   A. 프리데이를 초과한 지각제출의 경우, 지각제출 1일당 과목 전체 점수에서 1점씩 감점됨
   B. 2차과제 마감일 1주일 이후부터는 과제제출이 불가함