Operating Systems

Spring, 2018

School of Software, CAU

Project #2

- Thread Scheduling and Memory Allocation -

Due: June 12th, 2018

1. 프로젝트 개요

프로젝트 #2의 목표는 Pintos를 이용하여 스케줄링 알고리즘과 메모리 할당 알고리즘을 구현하는데 있다. Pintos는 기본적으로 Round Robin(RR) 스케줄러를 사용하는데, 여러분은 관련 코드의 수정 및 구현을 통해 RR 스케줄러를 Multi-level Feedback Queue (MFQ) 스케줄러로 교체해야 한다. 또한, Pintos는 기본적으로 메모리 할당 알고리즘으로 First Fit을 사용하는데, 관련 코드의 수정 및 구현을 통해 Pintos 부팅 시 다른 메모리 할당 알고리즘을 사용할 수 있도록 해야 한다. 마지막으로, 프로젝트 #2에서는 여러분의 프로젝트 수행을 위해 필요한 관련 Pintos 코드를 분석하고, 분석 결과를 리포트로 제출해야 한다.

2. 프로젝트 수행 준비

(1) Pintos 환경 구축

프로젝트 #2의 수행 환경은 프로젝트 #1과 동일하다. 프로젝트 #2 수행을 위해 아래의 링크를 이용해 변경된 Pintos 소스코드를 다운로드 받은 후, 프로젝트 #1의 문서를 참고하여 호스트 환경을 구축한다.

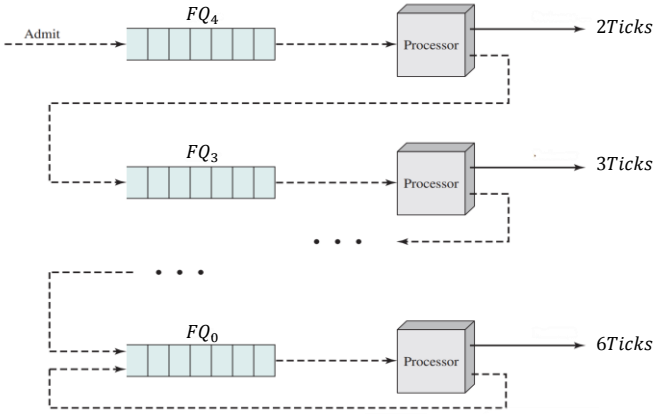
※ **cau15841-pintos2-qemu 아카이브** [다운로드]

(2) Multi-level Feedback Queue 스케줄러

프로젝트#2에서 구현해야 할 MFQ 스케줄러는 5단계 Feedback Dispatcher 방식으로 스레드를 스케줄링 한다. 이 스케줄러의 기본 할당 시간(Time Slice)는 Virtual Time 으로 1틱(Tick)이고, Feedback Queue(FQ)인 FQ_4(Highest Priority), FQ_3, FQ_2, FQ_1, FQ_0(Lowest Priority)의 할당 시간은 각각 2틱, 3틱, 4틱, 5틱, 6틱이다. 스레드는 생성될 때 할당된 우선순위에 따라 해당 FQ로 이동한다. 예를 들어, 스레드 생성 시 할당된 우선순위가 4(Highest Priority)인 경우, 이 스레드는 FQ_4에 배치되며 2틱의 실행 시간을 보장받는다.

MFQ 스케줄러는 가장 우선순위가 높은 스레드들을 모두 실행시킨 결과 해당 큐가 비게 되면, 다음 우선순위의 FQ

에서 대기 중인 스레드들을 실행시킨다. 각 스레드가 실행을 시작한 후, 자신의 할당 시간을 모두 소모하면, 우선순위를 한 단계 낮추어 해당 FQ로 이동한다. 예를 들어, 현재 FQ_3에 배치된 스레드가 실행을 시작한 후, 이 스레드의 할당 시간인 3틱을 모두 소진하면, 이 스레드의 우선순위는 2로 낮아지고 FQ_2로 이동한다. 5개의 FQ 중에서 가장 높은 우선순위 큐에 있는 모든 스레드들의 실행이 완료되어 큐가 비게 될 경우, 그 다음 높은 (그리고 비어 있지 않은) 큐에 있는 스레드들이 실행된다.



MFQ 스케줄러는 낮은 우선순위 스레드의 Starvation을 막기 위해 Aging 기법을 사용한다. FQ_i ($1 \le i \le 4$)에 속한 어떤 스레드 P가 실행 중일 때, 이 스레드보다 우선순위가 낮은 큐 FQ_j ($0 \le j < i \le 4$)에 있는 실행 준비 상태의 스레드 Q들의 Age는 매 틱 마다 1씩 증가된다. 만일 FQ_j ($0 \le j < i \le 4$)에 속한 어떤 스레드 P의 Age가 20에 도달하게 되면, 스레드 P의 우선순위가 한 단계 상승하여 상위 우선순위 큐 FQ_{j+1}로 이동되며 동시에 Age값이 0으로 초기화 된다.

(3) 메모리 할당 알고리즘

하나 이상의 연속된 메모리 블록을 할당하기 위해 필요한 공간을 효율적으로 탐색하는 것은 Pinto 메모리 관리자의 중요한 역할 중 하나로서 First Fit, Next Fit, Best Fit, Buddy System과 같은 기법이 사용될 수 있다.

First Fit의 경우, 메모리의 시작 지점부터 검색하여 필요한 크기 이상의 빈 공간이 나오는 최초 위치를 할당한다. Next Fit의 경우, 직전에 할당한 위치에서부터 검색하여 필요한 크기 이상의 빈 공간이 나오는 최초 위치를 할당한다. Best Fit의 경우, 메모리 전체를 검색하여 필요한 크기 보다 큰 공간 중에서 가장 적은 공간을 할당한다. 마지막으로, Buddy System은 메모리를 2^k 크기의 메모리 블록으로 할당하게 되는데, 요청 크기 s에 가장 가까운 빈 공간($2^{k-1} < s \le 2^k$)이 생성될 때까지 메모리 공간을 절반으로 계속 분할한다.

Pintos에서는 512개의 커널 페이지를 사용할 수 있기 때문에, Buddy System 구현을 위해 $(0 \le k \le 9)$를 사용하면 된다.

3. 프로젝트 요구 사항

(1) 스케줄링 알고리즘

- **요구사항 (1)-a: RR 스케줄러 관련 분석 리포트 작성**

프로젝트 디렉토리 threads/thread.c, threads/thread.h에 저장된 소스코드를 분석하고, Pintos에 기본으로 탑재된 RR 스케줄러의 동작 방법을 이해한 다음, 아래 질문에 대한 답변을 분석 리포트에 포함한다. 단, 질문에 대한 답변으로 관련 소스코드나 화면을 캡쳐해서는 안 된다.

① Pintos의 타이머 인터럽트는 1초에 100회 발생한다. 준비 큐에 있다가 Dispatcher에 의해 실행이 시작된 스레드는 다른 스레드가 실행되기 전까지 몇 초(또는 타이머 틱) 동안 실행되는가?

② RR 스케줄러는 어떤 자료구조를 이용하여 준비 큐에 머무르고 있는 스레드를 관리하는가?

③ threads/thread.c 소스 코드에서 어떤 함수들이 질문 ②에서 언급된 자료구조에 접근하는가?

④ RR 스케줄러는 threads/thread.h에 정의된 5가지의 스레드 우선순위를 사용하는가?

- **요구사항 (1)-b: MFQ 스케줄러 구현 및 구현 리포트 작성**

threads/thread.c와 threads/thread.h 소스코드 내에 MFQ 스케줄러를 구현한 다음, Pintos에 탑재된 RR 스케줄러를 MFQ 스케줄러로 교체한다. projects/scheduling/ 디렉토리에는 스케줄링 알고리즘을 테스트하는 샘플 코드가 추가되어 있고, 아래 명령어를 이용해 샘플 코드를 실행할 수 있다.

```
# ../../utils/pintos scheduling
```

여러분은 이 샘플 코드를 수정해 새롭게 구현한 MFQ 스케줄러가 올바르게 동작하는지 테스트할 수 있다. 프로젝트 채점 시에는 테스트 코드가 채점용 코드로 교체될 예정이다.

구현이 완료되면 다음 항목이 포함된 구현 리포트를 작성한다.

① threads/thread.c와 threads/thread.h에서 RR 스케줄러를 교체하기 위해 수정 및 추가한 부분

② 준비 큐에 머무르고 있는 스레드를 관리하기 위해 사용한 자료구조

③ 스레드 Aging 구현 방법

④ MFQ 스케줄러의 동작을 확인하기 위한 테스트 방법

(2) 메모리 할당

- **요구사항 (2)-a: 메모리/페이지 할당 관련 분석 리포트 작성**

프로젝트 디렉토리 내의 threads/malloc.c, threads/palloc.c, 그리고 lib/kernel/bitmap.c에 저장된 소스코드를 분석하고, Pintos의 메모리/페이지 할당 방법을 이해한 다음, 아래 질문에 대한 답변을 분석 리포트에 포함한다. 단, 질문에 대한 답변으로 관련 소스코드나 화면을 캡쳐해서는 안 된다.

① Pintos 메모리 시스템의 기본 페이지 크기는 몇 바이트인가?

② threads/malloc.c에서 기본 페이지보다 작은 크기의 메모리 영역을 할당하기 위해 어떤 방법이 사용되고 있는가?

③ threads/palloc.c와 lib/kernel/bitmap.c 내의 어떤 함수에 페이지 할당 알고리즘(First Fit)이 구현되어 있는가?

- **요구사항 (2)-b: 메모리/페이지 할당 알고리즘 구현 및 구현 리포트 작성**

프로젝트 디렉토리 내의 threads/palloc.c와 lib/kernel/bitmap.c 소스코드를 수정한 후, Next Fit, Best Fit, Buddy System과 같은 3가지 메모리/페이지 할당 알고리즘을 구현한다. Pintos 실행 시 '-ma=번호' 옵션을 추가하면 페이지 할당 알고리즘을 선택하는 변수가 변경된다. 이 변수를 이용해 Pintos 실행 시 선택한 페이지 할당 알고리즘을 동작 시킬 수 있도록 한다. 변수 변경과 관련된 사항은 threads/init.c의 259번 라인, threads/palloc.h의 7번 라인과 threads/palloc.c의 44번 라인을 참고한다.

projects/memalloc/ 디렉토리 내에는 페이지 할당 알고리즘을 테스트하는 샘플 코드가 추가되어 있고, 다음 명령어를 이용해 샘플 코드를 실행할 수 있다.

```
#  ../../utils/pintos -ma=0 memalloc
```

위 명령어는 페이지 할당 알고리즘 번호를 0(First Fit)으로 설정하고, 이를 테스트하는 샘플 코드를 실행한다. 샘플 코드는 현재 512개 커널 페이지 할당 상황을 출력하도록 작성되어 있기 때문에, 여러분은 이를 통해 페이지 할당 알고리즘이 올바르게 동작하는지 테스트할 수 있다. 프로젝트 채점 시에는 테스트 코드가 채점용 코드로 교체될 예정이다.

구현이 완료되면 다음 항목이 포함된 구현 리포트를 작성한다.

① threads/palloc.c와 lib/kernel/bitmap.c 에서 페이지 할당 알고리즘 구현을 위해 수정 및 추가한 부분

② 각 페이지 할당 알고리즘의 구현 방법

③ 페이지 할당 알고리즘의 동작을 확인하기 위한 테스트 방법

4. 기타

(1) 팀 구성

각 팀은 같은 반에 속한 3명으로 구성되어야 한다. 팀이 구성되면 5월 30일까지 수업 조교(이재환 박사과정, jhlee@cslab.cau.ac.kr)에게 팀장과 팀원 명단, 그리고 팀장 연락처를 통보해야 한다. 그때까지 팀원을 구하지 못한 학생은 본인 연락처를 메일로 수업 조교에게 보내야 하며, 이 경우 수업 조교가 팀을 임의로 구성할 수 있다.

(2) 최종 제출물 및 제출 시한

① 'make clean' 명령어를 통해 빌드 파일을 모두 제거한 뒤 관련 소스코드를 포함하여 압축한 디렉토리 압축 파일 (※ 소스코드는 제3자가 이해할 수 있도록 상세히 주석 처리가 되어야 하며, 프로젝트 채점 시 주석 처리의 수준이 반영될 예정임)

② “3. 프로젝트 요구 사항”에서 요청한 분석 리포트

③ 최종 제출문은 수업 조교에게 2018년 6월 12일 오후 11시 59분까지 이메일로 제출되어야 함