**Project #2**

**Thread Scheduling and Memory Allocation**

**Team 9**

**20132335 김태홍**

**20160030 심대범**

**20163704 박주현**

**Contents**

1. **스케줄링 알고리즘**

**(1) – a : RR 스케줄러 관련 분석 리포트 작성**

1. **– b : MFQ 스케줄러 구현 및 구현 리포트 작성**
2. **메모리 할당**

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

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

**<스케줄링 알고리즘 >**

1. **RR 스케줄러 관련 분석 리포트 작성**
2. Pintos의 타이머 인터럽트는 1초에 100번 발생한다. 준비 큐에 있다가 Dispatcher에 의해 실행이 시작된 스레드는 다른 스레드가 실행되기 전까지 몇 초(또는 타이머 틱) 동안 실행되는가?

* thread\_tick() 함수를 통해 4틱만큼 사용됨을 확인할 수 있다.

미리 정의 된 TIME\_SLICE 만큼의 타이머 틱동안 실행이 되는데, 해당 값은 4로 정의되어 있다.

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

* thread.c 파일의 선언에 따르면, RR 스케줄러는 list 구조체를 사용하여 준비 큐에 머무르고 있는 스레드를 관리하고 있다.

Struct list의 정의된 코드를 찾기 위해 <list.h>파일을 참조한 결과, 해당 list 구조체는 Doubly Linked List로 구현된 큐임을 정확하게 확인할 수 있었다.

1. Threads/thread.c 소스 코드에서 어떤 함수들이 질문 2에서 언급된 자료구조에 접근하는가?

* thread\_init() 함수에서는 list\_init(&ready\_list) 함수를 통해 접근함을 알 수 있다.

Thread\_unblock()함수에서는 list\_push\_back(&ready\_list, &t->elem) 함수를 통해 접근함을 알 수 있다.

thread\_yield() 함수에서는 list\_push\_back(&ready\_list, &cur->elem) 함수를 통해 접근함을 알 수 있다.

Next\_thread\_to\_run() 함수에서는 list\_empty(&ready\_list) 함수를 통해 접근함을 알 수 있다.

1. RR 스케줄러는 threads/thread.h에 정의된 5가지 스레드 우선순위를 사용하는가?

* RR 스케줄러는 스레드 우선순위를 사용하지 않는다. 생성된 스레드를 확인해보면 모두 기본값을 가질 뿐
* 아니라 입력된 순서대로 선점 방식을 사용하여 같은 list에서 관리되기 때문에 우선순위를 사용하지 않음을 확인할 수 있다.

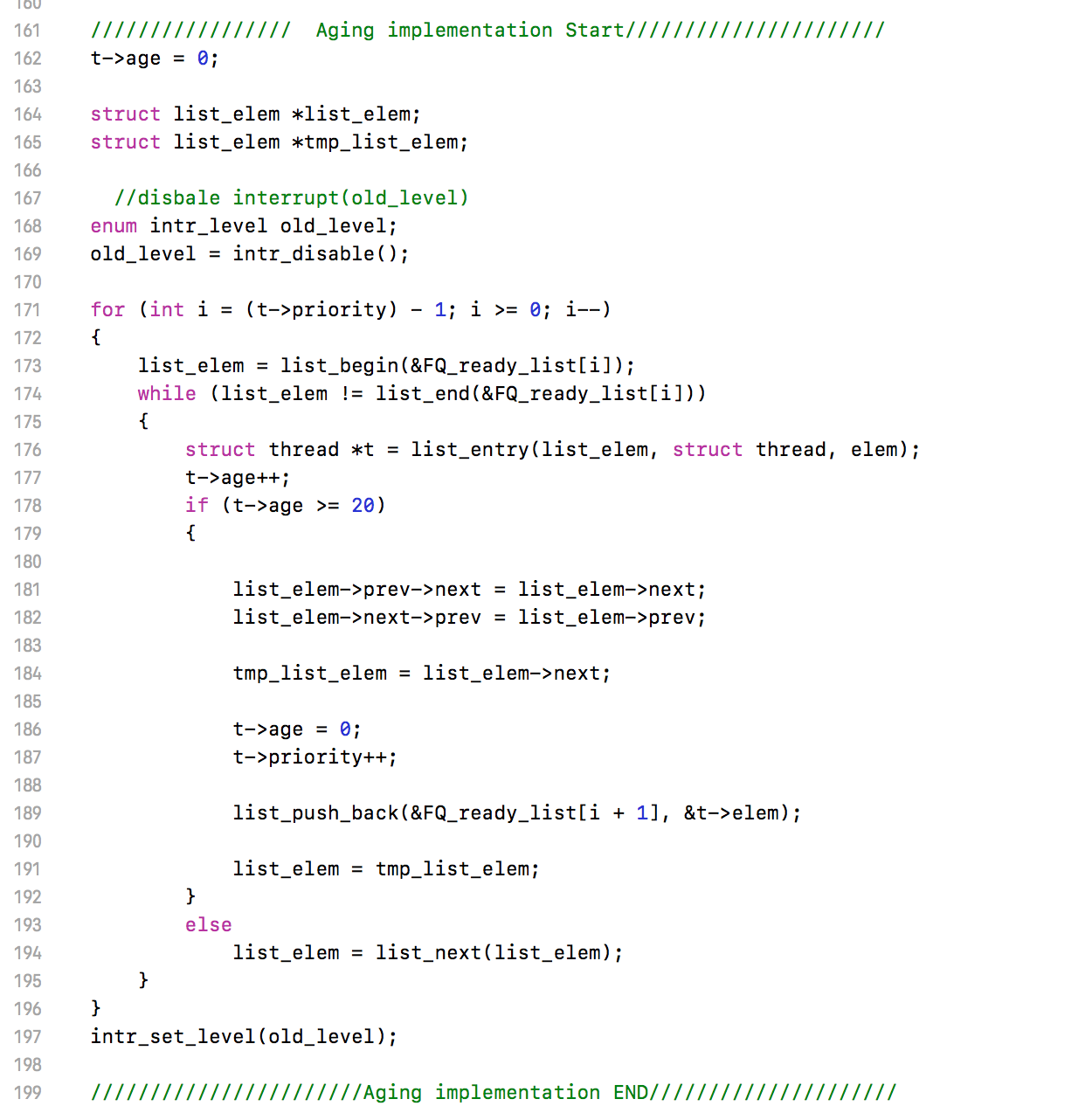
1. **MFQ 스케줄러 구현 및 구현 리포트 작성**
2. Threads/thread.c와 threads/thread.h에서 RR 스케줄러를 교체하기 위해 수정 및 추가한 부분

-static struct list FQ\_ready\_list[5]:FQ 리스트 추가(우선순위가 0부터 4까지 있는 쓰레드 관리)

이에 따라 각 list들을 초기화하는 코드 역시 추가/ list\_push\_back 부분 수정.

-MFQ 에서 사용되는 aging 기법을 위한 코드 추가.(그림1-1 참조)

- next\_thread\_to\_run() 수정:MFQ 스케줄러가 구현되기 위해서 우선순위가 높은 리스트를 먼저

검사하고, 해당 리스트에 준비된 쓰레드가 있다면 그 쓰레드를 실행시키게 했다.

<그림 1-1>

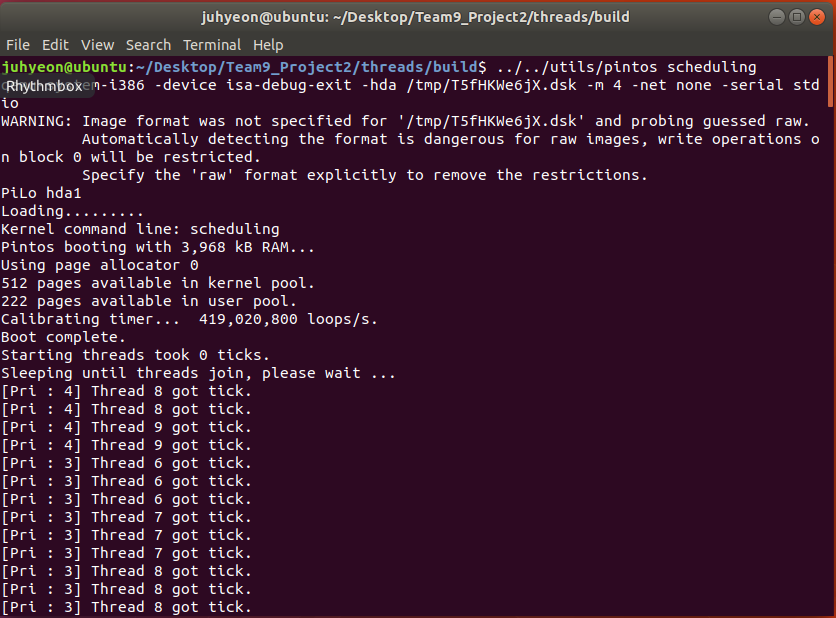
1. 준비 큐에 머무르고 있는 스레드를 관리하기 위해 사용한 자료구조

- static struct list FQ\_ready\_list[5] 를 추가해서 사용함으로써 각 큐를 기반으로 scheduling 함.

이 큐들은 next\_thread\_to\_run을 통해 검사되고 실행되고 수정되었다.

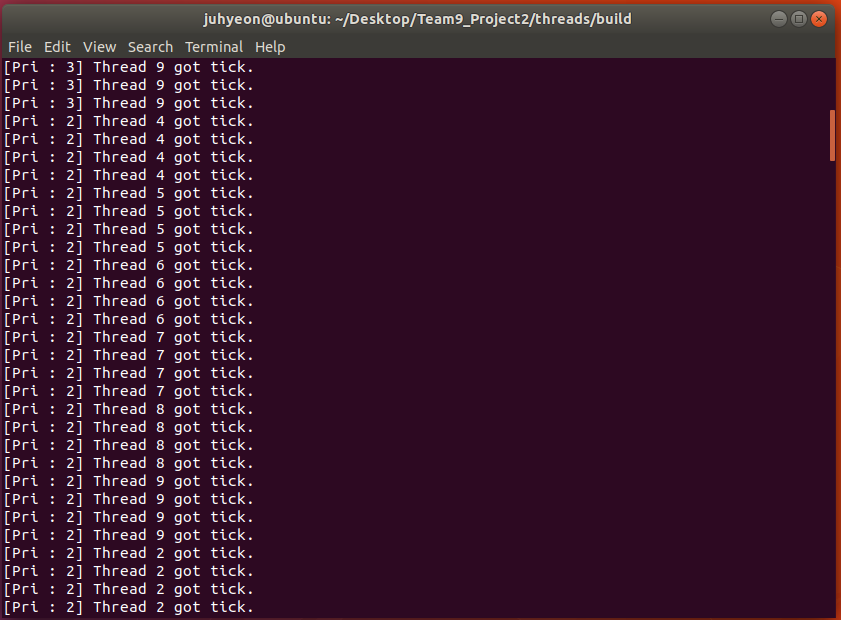
1. 스레드 Aging 구현 방법

- 먼저 스레드가 생성될 때나 최초 실행되었을 때 age=0으로 초기화 된다. 자세한 구현 방법은 그림1-1 을 보면 알 수 있다. 현재 스레드가 thread\_tick() 함수를 호출할 때 해당 스레드의 우선순위보다 낮은 모든 스레드의 age를 1씩 증가시킵니다. 이 후에 스레드의 age가 20 이상이 되었을 경우, age가 20이 된 쓰레드들은 현재의 우선순위보다 한 계단 높은 우선순위 리스트에 넣는다.

1. MFQ 스케줄러의 동작을 확인하기 위한 테스트 방법

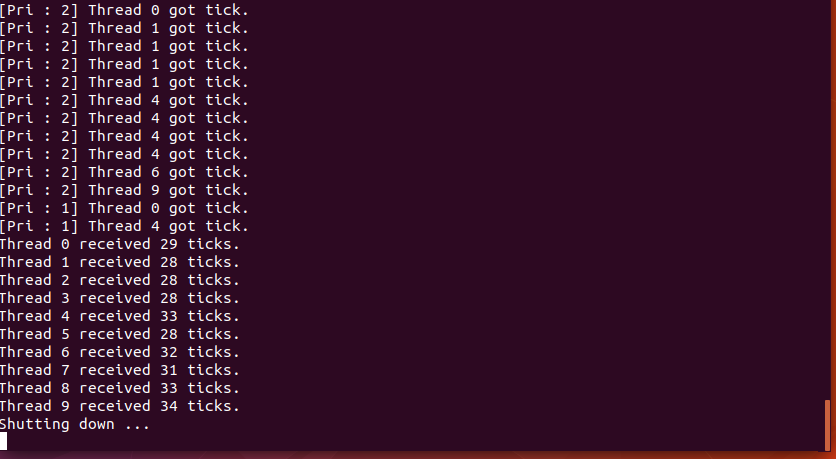
우선순위가 4인 thread들은 ticks이 2인 것을 알 수 있다.

또한 우선순위가 3인 쓰레드들의 tick이 3인것을 알 수 있다.



tick을 모두 소모한 Thread 9의 우선순위가 감소 한 것을 볼 수 있다.

aging을 통해 Thread 2의 우선순위가 증가 한 것을 볼 수 있다.



- 결과적으로 starvation이 발생 하지 않았다.