**Pintos Project 3: Threads**

담당 교수 : 문의현

학번 / 이름 : 20180492 / 김범준

개발 기간 : 2022. 11. 16 ~ 2022. 12. 04

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술.

PintOS 프로젝트 3에서는 Alarm clock과 priority scheduling에서 기존에 busy waiting 방식으로 구현되어 있던 것을 우선순위를 고려하는 방법으로 더욱 효율적이게 개선한다. 또한, 어떤 쓰레드의 우선 순위가 낮아 발생할 수 있는 기아를 방지하기 위하여 priority aging 기법을 적용한다.

1. **개발 범위 및 내용**
   1. **개발 범위**

* 아래 각 항목 개발의 필요성 또는 개발 시 기대되는 결과를 간략히 서술
  1. Alarm Clock

기존 timer\_sleep과 timer\_interrupt에서는 쓰레드가 THREAD\_RUNNING과 THREAD\_READY 상태를 번갈아가는 busy waiting 방식으로, 비교적 비효율적으로 구현되어 있었다. 이것을 쓰레드가 깨어날 시간을 저장한 뒤 해당 시간만큼 시간이 지나면 interrupt에서 깨우는 방식을 사용하여, 더욱 효율적인 방법으로 개선한다.

* 1. Priority Scheduling

기존에 이미 쓰레드 구조체에 구현되어 있던 멤버변수 priority를 활용하여 우선순위를 고려하여 cpu를 분배하는 방식의 scheduling을 새롭게 구현한다. 또한 우선순위를 고려하는 scheduling을 적용하였을 경우 발생할 수 있는 문제인 기아를 방지하기 위하여, 일정 시간이 지날 때마다 우선순위를 한 단계 상승시키는 priority aging 기법을 적용한다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술

1. Blocked 상태의 스레드를 어떻게 깨울 수 있는지 서술.

block이 된 상태의 쓰레드에는 어느 시점까지 block되어야 하는지 나타내는 변수 block\_time이 새롭게 정의되어 있다. 쓰레드를 block할 때 기존의 ready\_list로 가는 것이 아닌, block되는 쓰레드들을 관리하는 새로운 리스트(큐) block\_queue를 선언하여 해당 큐에 push한다. 큐에 push될 때에는 block\_time이 같이 계산되어 저장된다. 이후, 모든 틱마다 큐 안에 있는 쓰레드가 깨어날지 확인하고, 깨어날 시간이 되었다면 다시 ready\_list에 쓰레드를 삽입한다.

1. Ready list에 running thread보다 높은 priority를 가진 thread가 들어올 경우 priority scheduling에 따르면 어떻게 해야하는지 서술.

ready\_list에 실행되는 쓰레드보다 높은 우선순위를 가진 쓰레드가 들어온다면 해당 쓰레드는 리스트 안에 있는 쓰레드들과 우선순위를 비교하게 되고, 최종적으로 현재 실행되고 있는 쓰레드보다 새로운 쓰레드의 우선순위가 높다면 새로운 쓰레드가 cpu를 할당받고 실행되고 있던 쓰레드는 다시 ready\_list에 진입한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

2022. 11. 16 – 2022. 11. 23 : alarm clock 구현

2022. 11. 24 ~ 2022. 12. 01 : priority scheduling 구현

2022. 12. 02 ~ 2022. 12. 04 : 보고서 작성

* 1. **개발 방법**
* II. B.의 개발 내용을 구현하기 위해 각각에 대해 다음 사항들을 포함하여 설명
  + 수정해야하는 소스코드
  + 수정하거나 추가해야 하는 자료구조
  + 수정하거나 추가해야 하는 함수

-timer.c

먼저 block된 쓰레드를 관리하는 새로운 리스트 block\_queue를 선언한다. timer\_sleep() 함수에서 thread를 잠재울 때 block되어야 하는 시간을 저장하는 멤버변수 block\_time에 새로운 값을 할당하고, block\_queue에 해당 쓰레드를 삽입한다. timer\_interrupt에서는 block\_queue에 들어가 있는 쓰레드를 탐색하며 깨어날 시간이 된 쓰레드를 unblock한다. 이후, time slice마다 쓰레드의 priority를 aging 기법을 이용하여 갱신한다.

-init.c

전체강의 강의자료와 같이 parse\_options() 함수에서 aging에 관련된 부분을 추가한다.

-thread.h

전체강의 강의자료와 같이 aging 기법의 사용을 나타내는 플래그 변수 thread\_prior\_aging 변수를 선언하고, thread\_aging 함수의 프로토타입을 정의한다. 또한, 어떤 쓰레드가 block되었을 때 언제까지 block되는지 나타내는 멤버변수 block\_time을 쓰레드 구조체에 추가한다.

-thread.c

전체강의 강의자료와 같이 thread\_tick()함수에서 thread\_prior\_aging이 true일 경우 thread\_aging() 함수를 호출하는 부분을 추가한다. thread\_create()에서 새롭게 생성된 쓰레드의 우선순위가 현재 실행중인 쓰레드의 우선순위보다 높을 경우 선점할 수 있게 해주는 코드를 추가한다. thread\_unblock()과 thread\_yield() 함수에서 쓰레드가 새롭게 ready\_list에 들어갈 때 우선순위에 맞는 순서로 삽입될 수 있도록 코드를 수정한다. 새롭게 구현한 thread\_aging() 함수에서 ready\_list에 있는 쓰레드의 우선순위를 한 단계 상향조정하도록 코드를 구현한다.

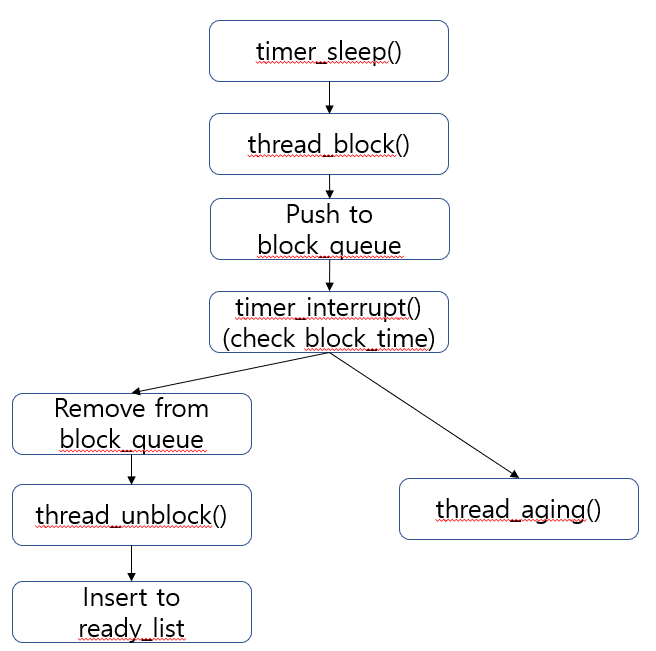
-synch.c

sema\_up() 함수에서 우선순위가 가장 높은 쓰레드가 unblock되도록 코드를 수정한다.

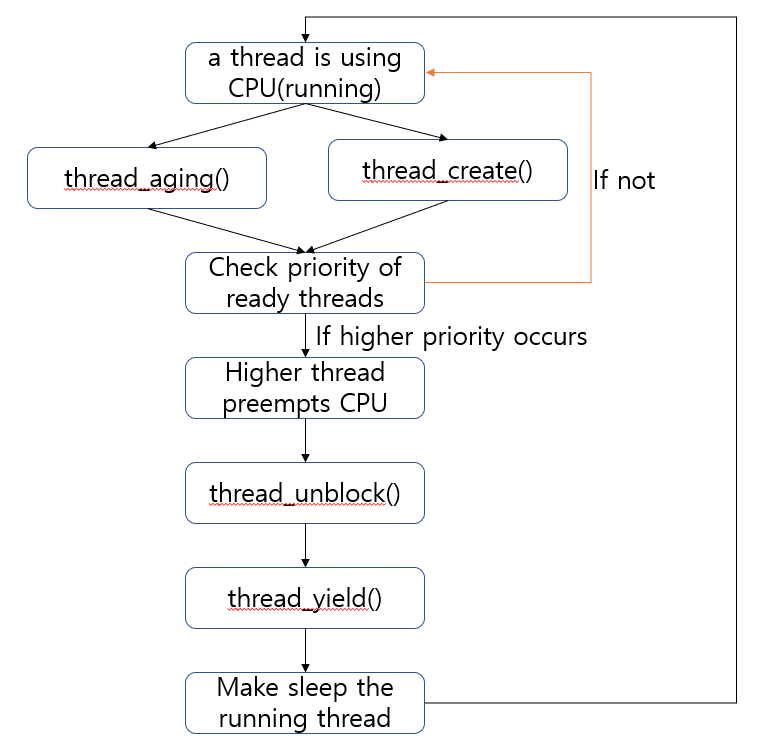
1. **연구 결과**
   1. **Flow Chart**

* II. B. 개발 내용의 각 항목에 대하여 Flow Chart 작성  
  (추가구현에 대해서는 flow chart를 작성하지 않아도 됨)

1. **Alarm Clock**

****

1. **Priority scheduling**

****

* 1. **제작 내용**
* II. B. 개발 내용의 각 항목에 대하여 실질적으로 구현한 코드의 관점에서 작성 (구현 내용, 알고리즘 등을 명확히 서술할 것)
  + 구현에 있어 Pintos에 내장된 라이브러리나 자체 제작한 함수를 사용한 경우 이에 대해서도 설명
* 개발 중 발생한 문제나 이슈가 있으면 이를 간략히 설명하고 해결한 방식에 대해 설명

1. 전체강의 자료에 따른 코드 수정

-thread.h

텍스트이(가) 표시된 사진

자동 생성된 설명



텍스트이(가) 표시된 사진

자동 생성된 설명

-init.c

텍스트이(가) 표시된 사진

자동 생성된 설명

-thread.c

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

다음은 전체강의 자료에 따라 추가된 코드 부분이다. thread.h의 쓰레드 구조체에서 block되어야 할 시간을 나타내는 멤버변수 block\_time을 추가로 선언해주었다.

2. timer.c



block된 쓰레드를 관리하는 리스트 자료구조 block\_queue를 선언하였다.

-timer\_sleep()

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

block되는 쓰레드의 block\_time에 ticks + start 값을 할당하고, block\_queue에 쓰레드의 list\_elem을 삽입한 뒤 thread\_block() 함수를 호출한다.

-timer\_interrupt()

텍스트이(가) 표시된 사진

자동 생성된 설명

block\_queue에 있는 쓰레드들을 탐색하면서, block\_time이 ticks보다 작거나 같으면, 즉 깨어나야할 시간이 경과했으면 unblock해준다. round-robin 기법의 time slice가 4이므로, 4틱이 지날 때마다 thread\_aging() 함수를 호출하여 쓰레드의 priority를 갱신한다.

2. thread.c

-thread\_create()

텍스트이(가) 표시된 사진

자동 생성된 설명

현재 실행되고 있는 쓰레드의 우선순위보다 새로운 쓰레드의 우선순위가 높을 경우 thread\_yield() 함수를 호출한다.

-thread\_unblock()

텍스트이(가) 표시된 사진

자동 생성된 설명

쓰레드를 unblock하여 ready\_list에 삽입할 때 우선순위에 맞도록 삽입한다. 먼저, ready\_list를 탐색하면서 삽입하려는 쓰레드보다 우선순위가 낮은 쓰레드를 찾고, list\_insert() 함수를 호출하여 해당 위치에 쓰레드를 삽입한다.

-thread\_yield()

텍스트이(가) 표시된 사진

자동 생성된 설명

thread\_unblock()과 마찬가지의 방법으로, 우선순위에 맞게 쓰레드가 ready\_list에 삽입되도록 한다.

-thread\_aging()

텍스트이(가) 표시된 사진

자동 생성된 설명

thread\_aging() 함수에서는 ready\_list를 순회하며 리스트 안에 있는 모든 쓰레드의 우선순위를 PRI\_MAX보다 작은 경우 한 단계 상향시킨다.

3. synch.c

-sema\_up()

텍스트이(가) 표시된 사진

자동 생성된 설명

sema\_up() 함수에서 대기 큐에서 우선순위가 가장 높은 쓰레드가 unblock되도록 구현하였다. 처음에 해당 부분을 구현하지 않았을 때 priority-sema 테스트케이스가 통과되지 못하였다. 해당 코드를 구현한 후에도 계속 테스트케이스가 통과되지 못하였으나, 코드 마지막 두 번째 줄에 thread\_yield() 함수를 호출하는 부분을 추가하여 priority-sema 테스트케이스를 통과할 수 있었다.

* 1. **시험 및 평가 내용**
* priority-lifo.c 코드 및 priority-lifo 테스트 결과 분석
* make check 수행 결과를 캡처하여 첨부

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

make check 실행 시 priority-change, priority-change-2, priority-fifo가 통과되지 못하였다. 이 중 priority-change와 priority-change-2의 경우 네 번째 줄에 예상치 못하게 줄바꿈이 되는 현상이 발생하였으나, 이유를 찾지 못하였다.