**Pintos Project 3: Threads**

담당 교수 : 김영재

조 / 조원 : 20211524 김준우

개발 기간 : 11/5 ~ 11/19

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술.

project 2에서 구현한 프로그램에 추가적으로 thread와 관련된 추가적인 기능들을 구현한다. alarm clock, priority scheduling, advanced scheduler(BSD scheduler)를 구현한다.

1. **개발 범위 및 내용**
   1. **개발 범위**

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

thread가 일정시간 동안 작동을 하지 않으면 timer\_sleep 함수를 호출해 thread를 sleep 시키고 while문으로 thread\_yield()를 호출해 thread가 Running과 ready 두 상태만 반복하게 한다. 이렇게 되면 비효율적이기 때문에 block 상태를 추가적으로 만들어 시간에 맞게 상태를 바꿔 효율적으로 실행되도록 한다.

* 1. Priority Scheduling

pintos에서 기본적으로 RR(round robin)으로 scheduling되어 proecess가 동작한다. 그래서 thread가 실행되는 함수가 호출되었을 때, 우선순위에 관계없이 ready\_list의 가장 끝에 추가되어 실행을 기다리게 된다. 여기에 priority를 고려한 scheduling system을 만들어 우선순위가 높은 thread를 먼저 실행 할 수 있도록 ready\_list에 삽입되게 하고, 현재 running 중인 thread보다 우선순위가 높다면 먼저 실행 될 수 있도록 해줘야 한다.

이때 낮은 우선순위의 thread들은 starvation이 발생하기 때문에, aging을 고려해 시간이 지남에 따라 우선순위를 높아지도록 구현한다.

* 1. Advanced Scheduler (추가구현을 한 경우)

nice, recent cpu, load average 를 추가로 선언해 이것을 계산해서 mlfq로 각 queue의 우선순위를 정하고 각 queue는 RR로 실행되도록 한다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술

1. Blocked 상태의 스레드를 어떻게 깨울 수 있는지 서술.

block 상태의 thread들을 관리하기 위해 thread에 expired time을 선언해 block 상태를 언제 block상태를 해제 할 것인지 알 수 있게 하고, blocked 상태의 thread들을 관리할 list를 새로 만들어서 매 tick마다 이 list에 있는 thread들의 expired time을 확인해서 시간이 된 thread들을 unblock해서 list에서 제거하고, Ready\_list에 다시 넣어주도록 한다.

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

running 중인 thread보다 높은 우선순위의 thread가 들어오면 현재 thread를 멈추고 ready queue에 다시 삽입하고 새로 들어온 thread를 실행시켜준다. 이때 중지한 thread는 우선순위에 맞게 ready queue에 삽입해줘야 한다.

1. Advanced Scheduler에서 priority 계산에 필요한 각 요소를 서술. (추가구현을 한 경우)

BSD scheduler를 구현하기 위해서 nice, recent\_cpu, load\_avg가 필요하다.nice값은 -20~20까지의 값을 가진다. thread를 init할 때 nice와 recent\_cpu값을 초기화 한다. child thread는 parent의 nice와 recent\_cpu값을 받아온다. nice의 값이 양수가 되면 우선순위가 낮아진다. 따라서 nice의 값이 바뀌면 thread의 priority를 새로 계산해줘야 한다. priority 계산 식은 아래와 같다.

* + - *priority* = PRI\_MAX – (*recent\_cpu* / 4) - (*nice* \* 2)

recent\_cpu는 thread의 cpu\_time을 측정한다. 만약 thread에서 time interrupt가 발생하면

running thread의 recent cpu 값이 1이 증가하도록 하고 매초마다 모든 thread의 recent

cpu가 다시 계산되도록 한다. recent\_cpu의 계산식은 아래와 같다.

* + *recent\_cpu* = (2 \* load\_avg) / (2 \* load\_avg + 1 ) \* *recent\_cpu* + nice

load\_avg는 전역변수로 ready queue에 있는 thread의 평균 개수를 측정하는 값이다. 가장 처음 실행되면 0으로 초기화 되고, 매초 update 되도록 한다. load\_avg의 식은 아래와 같다.

* + *load\_avg* = (59/60) \* *load\_avg* + (1/60) \* *ready\_threads*

이때 pintos에서는 floating point 연산이 불가능 해서 fixed point 연산을 하도록 구현한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성
* 11/5~ 11/6 alarm clock 구현
* 11/10 ~ 11/12 priority scheduling 구현
* 11/13 ~ 11/17 advanced scheduling 구현
* 11/18 ~ 11/19 오류 수정 및 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

-alarm clock

timer.c에서 blocked된 Thread들을 저장할 새로운 list인 block\_list를 새로 선언하고 init해준다. timer\_sleep 함수를 수정해 thread의 expired time을 계산해주고 block\_list를 관리한다. 그리고 time\_interrupt함수에서 block된 상태의 thread를 깨워주는 코드를 추가한다.thread.h에서 thread struct에 expired\_time을 선언해서 시간을 체크 할 수 있도록 한다.

-priority scheduling

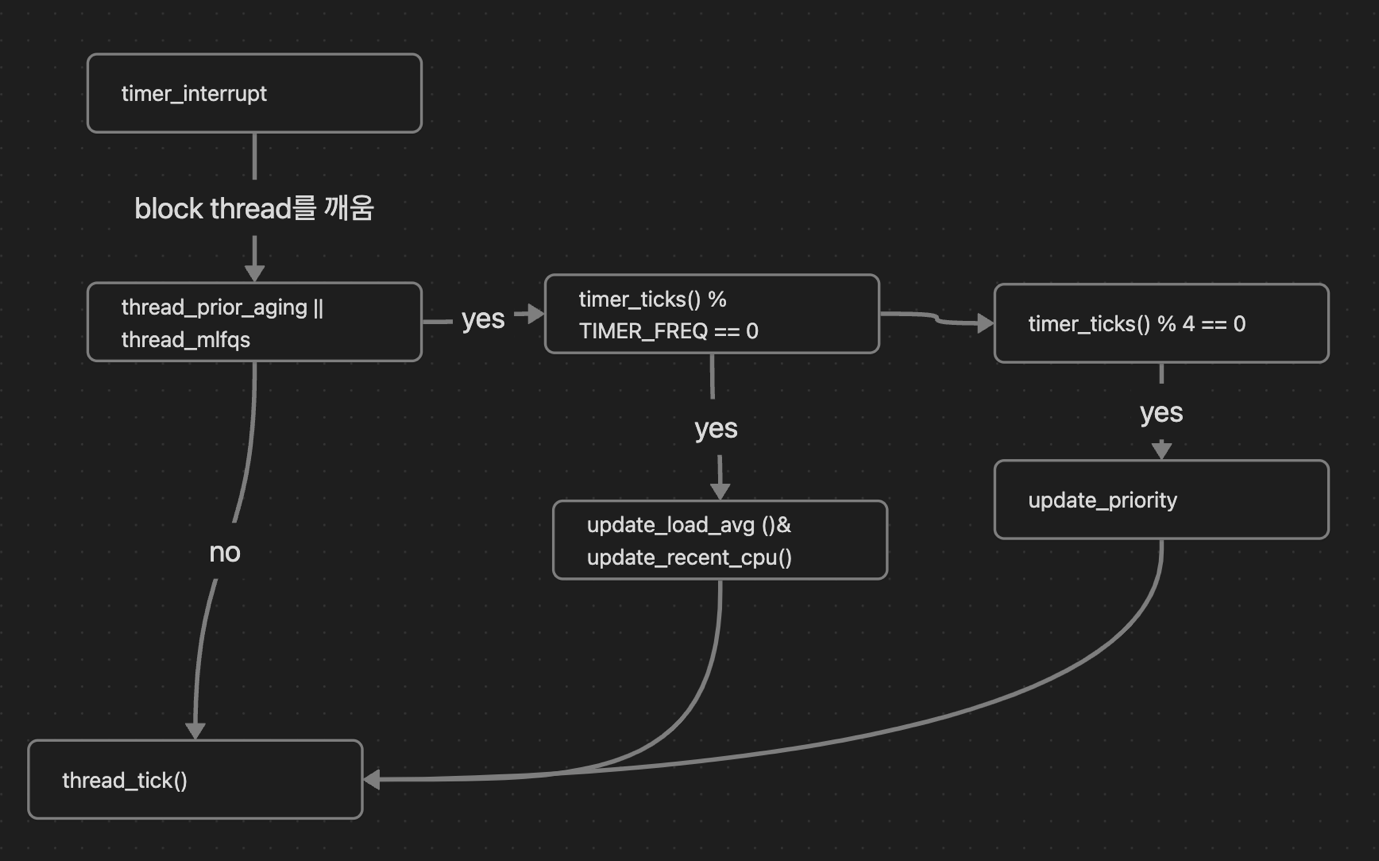
우선순위를 고려해서 ready queue에 thread들을 정렬해줘야 한다. 그리고 새로 create된thread의 경우에는 현재 실행중인 thread보다 우선순위가 높다면 실행중인 thread를 중단하고 새로 생성된 thread가 실행되어야 한다. thread\_create함수에서 priority 인자를 이용해 현재 running 상태의 thread보다 우선순위가 높으면 실행을 해주도록 thread\_yield 함수를 호출한다. ready queue에 정렬되어 thread를 삽입하기 위해 thread 끼리의 우선순위를 비교해주는 함수를 만들어 thread\_yield와 thread\_unblock 함수에서 이용해준다. 그리고 semaphore에서도 priority를 적용시켜주기 위해서 sema\_up함수를 수정해준다.

-advanced scheduler

thread.h에서 recent\_cpu와 nice를 thread struct에 선언하고 init할 때 초기화 해준다. thread\_get\_nice, thread\_get\_load\_avg, thread\_get\_recent\_cpu를 구현하고, 이 값들을 업데이트 해줄 함수들을 추가로 만들어서 구현하고, timer.c의 timer\_interrupt 함수에서 값을 시간에 맞춰 갱신해준다.

1. **연구 결과**
   1. **Flow Chart**

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

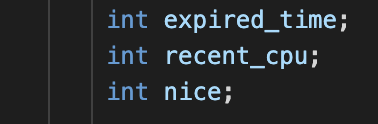
****

**스크린샷, 도표, 평면도, 디자인이(가) 표시된 사진

자동 생성된 설명**

* 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.alarm clock



threads/thread.h의 thread struct에 thread가 얼마나 block되었는지 저장 할 수 있도록 한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

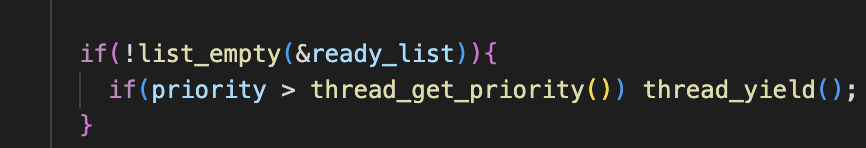
최근의 thread의 expired\_time을 계산해주고 block list에 추가하고 thread\_block함수를 호출해서 block해준다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

timer\_interrupt에서 block\_list를 돌면서 expired\_time을 체크해서 block된 thread를 해제해준다.

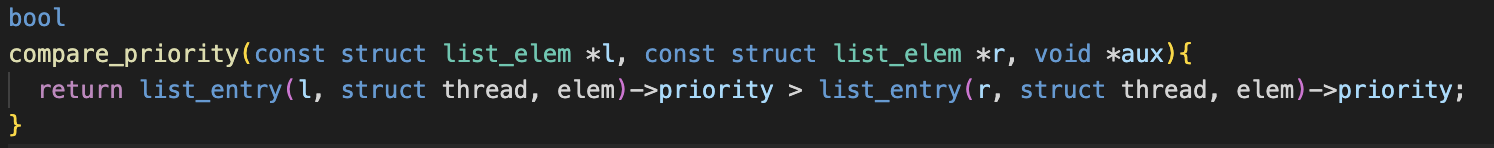
2.priority\_scheduling

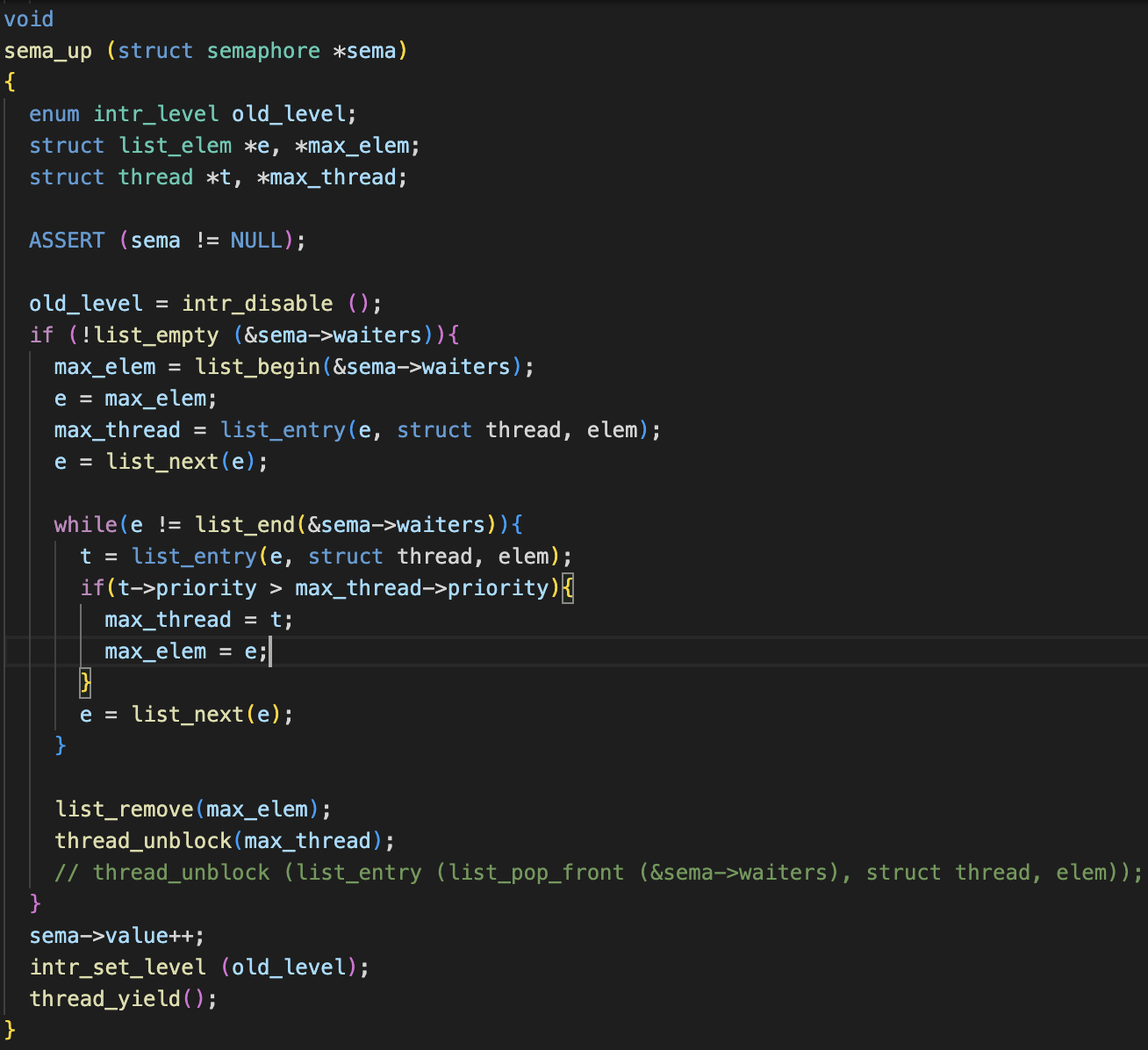


thread.c의 thread\_create함수에서 생성될 thread의 priority가 thread\_current()의 priority보다 크다면 생성된 thread를 thread\_yield 함수를 통해 실행시켜준다.

list\_insert\_ordered (&ready\_list, &t->elem, compare\_priority, NULL)

3번째 인자로 비교함수를 받아서 삽입해주는 list\_insert\_ordered 함수를 이용해서 ready queue에 priority에 맞게 삽입해준다. 이는 thread가 unblock되거나 yield될 때 필요하다. 그리고 우선순위를 비교해주는 compare\_priority 함수는 아래와 같다.



synch.c의 sema\_up 함수에서도 priority 값이 가장 큰 thread를 sema\_up을 해줘야 한다 

semaphore로 기다리고 있는 thread들의 list를 탐색하면서 priority가 가장 큰 thread를 찾아내고 그 thread를 list에서 제거하고 unblock 해준다. 현재 thread의 우선순위가 새로운 thread의 수선순위보다 크면 다시 thread의 우선순위를 조정하기 위해 thread\_yield를 호출해준다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

그리고 starvation을 방지하기위해 threads/init.c에 bool 타입의 thread\_prior\_aging을 선언해준다. thread.c에도 동일하게 선언해준다. 그리고 priority를 업데이트 해주는 함수를 만들고 timer\_interrupt에서 4tick 마다 이 함수를 호출해서 우선순위를 업데이트 해준다.



스크린샷, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명

3.BSD

fixed point 연산을 위해서 FRACTION을 thread.h에 정의해준다.

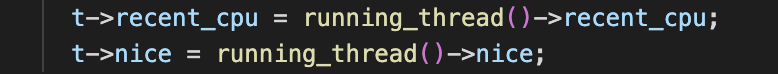
폰트, 그래픽, 텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

thread의 구조체에는 recent\_cpu와 nice를 선언해주고 thread\_init 함수에서 초기화 하고 thread\_init으로 자식 thread는 부모 thread의 이 값들을 받아오도록 한다.

텍스트, 폰트, 스크린샷, 그래픽이(가) 표시된 사진

자동 생성된 설명

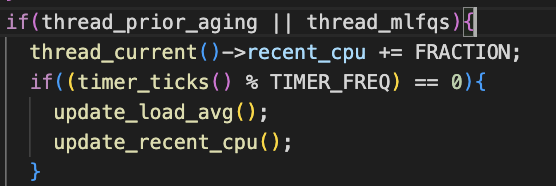


그리고 load\_avg를 전역변수로 thread.c 파일에 선언해주고 thread\_init 함수에서 초기화 해준다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

주어진 공식을 이용해 recent\_cpu와 load\_avg를 update를 해주는 함수다. recent\_cpu는 thread 마다 있기 때문에 thread가 들어있는 list를 전체 탐색하면서 계산해준다. update\_load\_avg 함수에서 ready\_thread가 idle thread가 아니면 1을 더해준다. 이들을 매초마다 업데이트 해줘야해서 timer interrupt 함수에서 매초 호출해주면 된다.



텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

nice, load\_avg, recent\_cpu를 얻어오는 함수들이다. load\_avg와 recent\_cpu는 100을 곱한 값을 반환해준다.

* 1. **시험 및 평가 내용**
* priority-lifo.c 코드 및 priority-lifo 테스트 결과 분석

텍스트, 스크린샷, 흑백, 패턴이(가) 표시된 사진

자동 생성된 설명

우선순위가 일정하게 출력되면 성공한 것인데 우선순위가 일정하게 출력되는 모습을 확인 할 수 있다.

* make check 수행 결과를 캡처하여 첨부

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명