**Pintos Project 3: Threads**

담당 교수 : 박성용 교수님

학번 / 이름 : 20181684 / 정성원

개발 기간 : 22.10.27 ~ 22.11.12

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술.

-Alarm Clocking

-Process Priority Scheduling

-BSD Scheduler

1. **개발 범위 및 내용**
   1. **개발 범위**

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

Pintos의 time clocking 은 무한 loop를 돌며 확인하는 busy waiting 방식을 사용하고 있다. 이는 CPU의 낭비이기 때문에 이를 해결한다.

* 1. Priority Scheduling

Process의 priority를 고려하여 process를 실행한다.

* 1. Advanced Scheduler (추가구현을 한 경우)

MLFQ 구조를 갖는 BSD Scheduler를 구현하여 Priority에 따라 process를 실행할 수 있게 한다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술

1. Blocked 상태의 스레드를 어떻게 깨울 수 있는지 서술.

Thread에 일어날 시간을 추가하고(int64\_t wake\_tick), 시간을 체크하여 thread가 일어날 시간에 Block을 풀고 Ready 상태로 바꿔 준다.

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

현재 실행 중인 thread를 중지하고 thread\_yield(void)를 실행시킨다. thread\_yield(void)는 현재 실행 중인 thread를 ready queue에 넣고 다시 scheduling 하는데, 이를 통해서 priority가 더 큰 thread가 실행될 수 있게 한다.

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

Recent cpu ; thread의 cpu time을 측정한 값

Load average ; ready 상태의 thread의 개수

Nice ; thread가 갖고 있는 값으로, nice 값이 높을수록 priority는 낮아진다.

Load average와 nice를 통해서 Recent cpu를 계산하고, 구한 Recent cpu와 nice를 이용하여 priority를 구해준다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**
   2. **개발 방법**

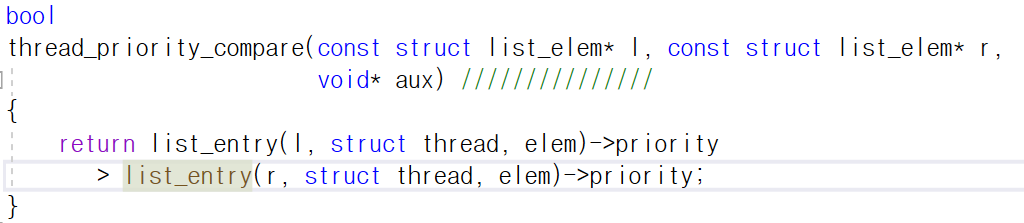
* II. B.의 개발 내용을 구현하기 위해 각각에 대해 다음 사항들을 포함하여 설명
  + 수정해야하는 소스코드

Void thread\_yield (void)

텍스트이(가) 표시된 사진

자동 생성된 설명

기본적으로 pintos에서 제공하는 thread\_yield()는 list\_push\_back을 사용했으나, priority를 위해서 list\_insert\_ordered를 사용한다.

bool thread\_priority\_compare(const struct list\_elem\* l, const struct list\_elem\* r, void\* aux) 

두 thread의 priority를 비교한다.

Void thread\_set\_priority (int new\_priority)텍스트이(가) 표시된 사진

자동 생성된 설명

Thread의 priority를 설정한다. 이 때 새 priority가 현재 thread의 priority보다 작다면, thread\_yield를 다시 실행해 준다.

Int thread\_get\_priority (void)

텍스트이(가) 표시된 사진

자동 생성된 설명

현재 실행 중인 thread의 priority를 구해준다.

Void thread\_set\_nice (int nice UNUSED)

텍스트이(가) 표시된 사진

자동 생성된 설명

Thread의 nice를 설정한다. 이렇게 설정된 nice는 이 후, priority를 계산하는데 이용된다.

Int thread\_get\_max\_priority(void)

텍스트이(가) 표시된 사진

자동 생성된 설명

list entry에 있는 thread들 중 priority가 가장 큰 thread를 찾아 그 thread의 priority를 return 한다.

Int thread\_get\_nice (void)

텍스트이(가) 표시된 사진

자동 생성된 설명

현재 thread의 nice를 return 한다.

Int thread\_get\_load\_avg

텍스트이(가) 표시된 사진

자동 생성된 설명

현재 thread의 load average를 구해준다. 이를 위해 fixed\_point 헤더를 이용한다.

Int thread\_get\_recent\_cpu

텍스트이(가) 표시된 사진

자동 생성된 설명

현재 thread의 recent cpu timedmf 구해준다. 이를 위해 fixed\_point 헤더를 이용한다.

void larc\_renew(void)

텍스트이(가) 표시된 사진

자동 생성된 설명

List entry를 돌면서 ready list에 저정된 thread들의 Load average와 recent cpu를 수정한다.

void renew\_priority()

텍스트이(가) 표시된 사진

자동 생성된 설명

All list에 저장된 무든 thread들의 priority를 갱신해준다. 이후 intr\_yield\_on\_return 을 통해 yield를 실행 시켜 준다. -

**timer.c**

static void timer\_interrupt (struct intr\_frame \*args UNUSED)

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

List entry를 돌면서, thread가 일어날 시간이 됐다면 interrupt를 일으켜 준다.

Void timer\_sleep (int64\_t ticks)

텍스트이(가) 표시된 사진

자동 생성된 설명

-thread를 disable상태로 바꾼 후, sleep상태로 바꿔 준다.

Fixed\_point.h in threads

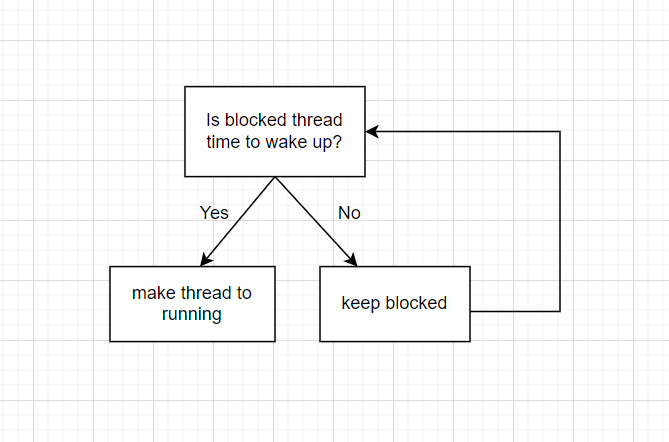
텍스트이(가) 표시된 사진

자동 생성된 설명

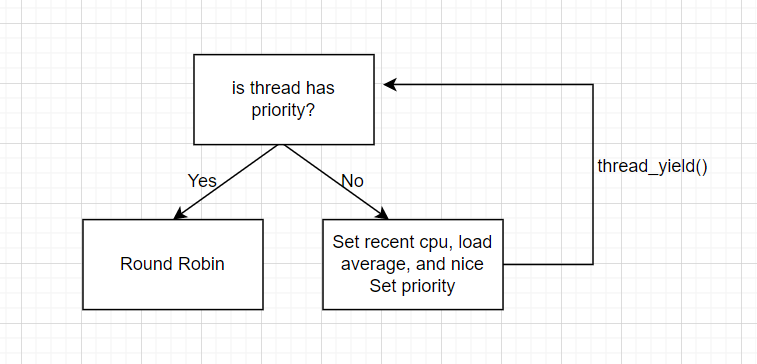
-priority를 계산하기 위해서 만든 헤더파일로, recent cpu, load average, nice를 이용해서 priority를 계산한다.

1. **연구 결과**
   1. **Flow Chart**

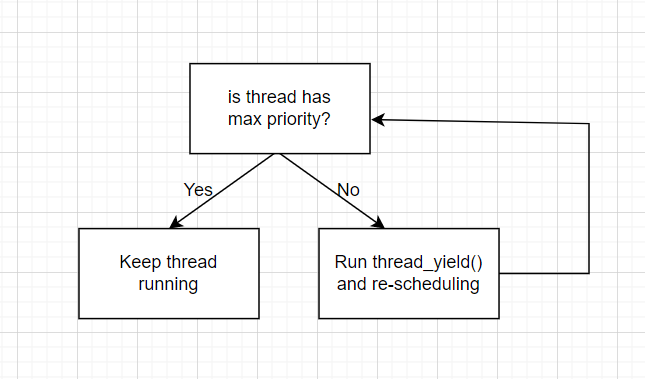
* II. B. 개발 내용의 각 항목에 대하여 Flow Chart 작성  
  (추가구현에 대해서는 flow chart를 작성하지 않아도 됨)
* **Alarm Clocking**

****

* **Priority scheduling**

****

* **BSD scheduler**

****

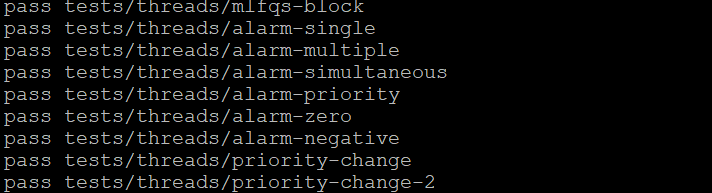
* 1. **제작 내용**
* II. B. 개발 내용의 각 항목에 대하여 실질적으로 구현한 코드의 관점에서 작성 (구현 내용, 알고리즘 등을 명확히 서술할 것)
  + 구현에 있어 Pintos에 내장된 라이브러리나 자체 제작한 함수를 사용한 경우 이에 대해서도 설명
* 개발 중 발생한 문제나 이슈가 있으면 이를 간략히 설명하고 해결한 방식에 대해 설명
  1. **시험 및 평가 내용**
* priority-lifo.c 코드 및 priority-lifo 테스트 결과 분석

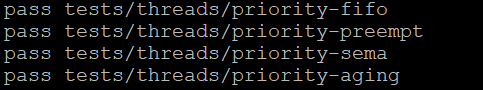
텍스트이(가) 표시된 사진

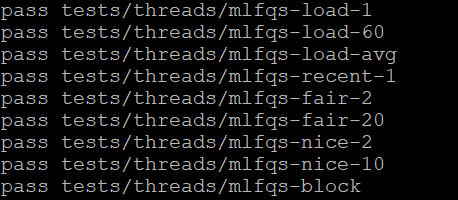
자동 생성된 설명

15부터 차례대로 0까지 16번을 반복하는 것을 볼 수 있다.

* make check 수행 결과를 캡처하여 첨부







-모두 Pass한 것을 확인할 수 있다.