**Pintos Project 3: Threads**

담당 교수 : 김영재

조 / 조원 : 20211582 임나현

개발 기간 : 2024.11.18 ~11.25

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술.

이전까지는 round robin의 방식을 사용하여 각 process나 thread의 priority를 고려하지 않는 방식의 스케줄러를 사용했다면 이번 project 3에서는 여러 가지 스케줄러 중 MLFQ 방식으로 구현한다. 이를 위해 timer를 alarm clock을 구현하고 효율적으로 관리하고 priority를 각 thread 별로 부여하는 것이 필요하다.

1. **개발 범위 및 내용**
   1. **개발 범위**

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

기존 timer는 thread를 running state와 ready state를 반복하는 busy waiting 방식을 사용하였지만 이는 비효율적이다. 따라서, ready queue를 구현하여 일정 시간 이후에 queue에 삽입되어 대기할 수 있도록 구현한다.

* 1. Priority Scheduling

기존 round robin 방식을 개선하기 위해 각 thread마다 priority를 부여해야 한다. 우선 순위에 따라 여러 thread 중 가장 높은 우선 순위를 running 상태로 전환하는데 이는 ready queue의 front thread를 pop하는 방식을 사용한다. 나머지 thread들은 ready queue에 남아 있으며 새로운 thread가 ready queue에 들어오는 경우 우선순위에 따라 해당 위치에 삽입된다

* 1. Advanced Scheduler (추가구현을 한 경우)

Mlfq 방식을 통해 scheduler를 구현한다. Mlfq 방식은 priority aging을 통해 starvation 문제가 발생하지 않도록 priority를 지속적으로 관리하며, 여러 level을 이용하기 때문에 각 level마다 queue를 가지고 queue 내에서는 round robin 방식으로 thread를 관리한다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술

1. Blocked 상태의 스레드를 어떻게 깨울 수 있는지 서술.

Thread가 Running 상태에서 할당된 cpu time을 전부 사용하면 ready state로 전환되지 않고 blocked state로 전환한다. blocked state로 전환된 thread들을 관리하기 위한 sleep\_list list 구조체를 선언하고 thread가 sleep\_list에 삽입될 때 timer 또한 같이 저장한다. 매 timer tick이 흐를 때마다 timer\_interrupt 함수를 통해 현재 time과 저장되어 있는 timer 값을 확인하여 다시 ready queue에 들어가야 할 thread들을 탐색한다. 만약, wake해야 할 thread를 확인했다면 sleep\_list에서 제거하고 ready queue에 삽입한다.

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

만약 새로운 thread가 running thread보다 priority가 높다면 현재 running state였던 thread는 ready list에 추가되고 새로운 thread가 running state가 되어야 한다. list\_insert\_ordered 함수는 priority 순서대로 정렬하기 때문에 새로운 thread가 가장 front에 위치하게 되어 running state로 context switching이 일어난다.

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

nice 값 (-20~20) 을 통해 각각의 thread에 priority를 부여한다.

: thread가 사용한 CPU 시간



****

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

2024.11.18 - 2024.11.19: alarm clock 구현

2024.11.20 – 2024.11.22: priority scheduling & alarm priority 구현

2024.11.22 – 2024.11.24: mlfq 구현

2024.11.24: 보고서 작성

* 1. **개발 방법**
* II. B.의 개발 내용을 구현하기 위해 각각에 대해 다음 사항들을 포함하여 설명
  1. Alarm clock

현재는 busy waiting으로 구현이 되어 있지만, sleep list를 선언하여 block된 thread들을 관리한다. thread를 list에 저장할 때 sleep시간을 함께 저장한다. Timer\_interrupt 함수에서는 tick을 증가시키면서 sleep list에 있는 모든 thread를 탐색하여 ready state로 전환해야 하는 thread가 있는지 확인한다.

* 1. Priority Scheduling

Round robin 방식이 아닌 priority를 기반으로 하는 스케줄링을 구현해야 한다. 우선, 각 thread에 priority를 부여해야 하며, 가장 priority가 높은 thread가 ready list에 front에 있을 수 있도록 compare함수를 구현해야 한다. 만약, 새로운 thread의 priority가 현재 실행중인 thread의 priority보다 높은 경우 thread\_yield 함수를 통해 resecheduling해야 한다. 기존에는 list\_push\_back으로 ready list에 새로 들어오는 thread는 뒤에 삽입되도록 하였는데, list\_insert\_ordered함수로 전부 교체하여 들어올 때마다 priority 순서대로 정렬되도록 해야 한다.

* 1. Advanced Scheduler

Mlfq 방식을 사용하기 위해서는 recent cpu, load avg, nice 값을 계산해야 한다. 계산하는 공식은 명세서에 나와있으므로 이를 기반으로 구현한다. 그러나, pintos에서는 fixed point 연산을 하기 때문에 계산에 필요한 float & int 연산을 하는 함수를 추가해야 한다. Thread의 recent cpu, load avg, priority를 갱신할 수 있는 update 함수 또한 구현해야 한다.

1. **연구 결과**
   1. **Flow Chart**

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

텍스트, 도표, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

텍스트, 도표, 스크린샷, 라인이(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

****

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

timer.c에는 정적변수로 sleep\_list를 선언하고 thread 구조체 안에 sleep list에 저장될 thread들의 timer인 sleep\_timer를 선언한다.

**텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

timer\_sleep 함수에서는 해당하는 thread를 실행된 timer를 포함하여 list에 저장하도록 구현했다. List\_push\_back 함수를 통해 sleep\_list list에 pop한다.

**텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명**

timer\_interrupt에서는 sleep\_list 내에서 ready\_list로 이동해야하는 thread들이 있는지 매번 확인해줘야 한다.

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

thread의 priority의 크기를 비교하는 함수이다.

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

ready\_list에 삽입할 때 기존의 round robin에서 priority 순서대로 정렬될 수 있도록 compare함수를 이용한다.

**텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명**

만약, 새로운 priority가 현재 running thread의 priority보다 크면 context switching이 일어나야 하기 때문에 thread\_yield를 통해 다시 스케줄링한다.

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

thread\_set\_priority 함수에서는 현재 thread의 priority를 새로운 priority로 변경해주며, 만약 새로운 priority가 기존 priority보다 작은 경우 list에 있는 다른 thread들 중에서 priority가 더 큰 경우가 발생할 수 있으므로 thread\_yield함수를 불러온다.

**텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명**

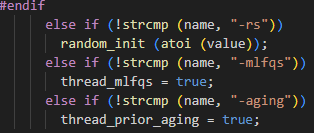
Sema\_up함수에서 list를 priority순서대로 정렬하여 가장 우선순위가 높은 thread를 list\_remove함수를 통해 pop하고 running state로 전환할 수 있도록 한다.

**스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명**

priority aging을 위해 pintos에서는 fixed point 연산을 구현해야 한다. 이번 과제에서는 load\_avg, recent\_cpu, priority 계산을 해야하므로 관련된 연산을 지원하는 함수들이다.





명세서에 나와있는대로 thread\_aging과 관련된 선언들을 작성한다.

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

max\_priority 함수에서는 priority로 정렬되어 있는 ready\_list에서 front를 pop하여 max priority를 얻는다.

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

update\_load\_avg 함수에서는 명세서에 나와있는 load avg 식을 이용하여 update하는 함수이다.

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

update\_priority, recent\_cpu 함수에서는 all list들을 탐색하면서, priority와 recent cpu를 update하는 함수이다.

**스크린샷, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명**thread\_aging 함수에서는 recent cpu를 1씩 증가시키고 load avc, recent cpu, priority를 update한다.

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

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

Priority lifo는 서로 다른 우선순위를 가질 때는 우선 순위가 높은 thread부터 실행되면 되지만, 만약 같은 priority를 가지는 여러 thread가 들어 온다면 어떻게 동작하는지 확인할 수 있는 테스트 케이스이다. 우선순위가 같은 경우는 다음과 같은 출력처럼 lifo(마지막 thread가 처음으로 실행)방식이 적용된다. 매 실행 횟수마다 동일한 출력값을 가진다는 점에서 제대로 출력된다는 것을 확인할 수 있다.

* make check 수행 결과를 캡처하여 첨부

텍스트, 폰트, 흑백, 스크린샷이(가) 표시된 사진

자동 생성된 설명