**Pintos Project 3: Threads**

담당 교수 : 김영재 교수님

조 / 조원 : 정은교

개발 기간 : 2023.11.1 ~ 2023.11.18

1. **개발 목표**

프로젝트 1, 2에서 구현한 프로그램을 바탕으로 thread와 관련된 alarm clock, priority scheduling, advanced scheduler를 구현한다.

1. **개발 범위 및 내용**
   1. **개발 범위**
   2. Alarm Clock

thread가 작동을 멈추고 대기 상태로 전환되는 시간이 일정하게 유지될 때, 특정 시간이 지나지 않았을 때 while문으로 계속해서 thread\_yield() 함수를 호출하여 running과 ready 상태를 반복하는 것은 효율적이지 않다. 이 문제를 해결하기 위해 timer\_sleep() 함수를 사용하여 sleep 시킬 때, thread를 효율적으로 대기시키는 방법을 찾아야한다.

이러한 문제를 해결하기 위해 대부분의 운영체제는 sleep 상태에 있는 thread를 특정 시간이 지나면 자동으로 깨우는 기능을 제공한다. 이를 이용하면 while문과 반복 호출을 피할 수 있다. 일정 시간이 지난 후에 스스로 깨어나는 기능을 이용하여 불필요한 호출을 줄일 수 있다.

* 1. Priority Scheduling

pintos에서 round-robin 스케줄링이 사용되었다. 이는 thread\_yield() 함수나 thread\_unblock() 함수 호출 시 현재 스레드나 unblock된 스레드가 우선순위와 상관없이 ready 리스트의 끝에 추가된다는 뜻이다. 하지만 이번 프로젝트에서는 우선순위를 고려한 스케줄러를 도입한다. 다시 말해, 현재 스레드의 우선순위보다 높은 우선순위를 가진 스레드가 있다면 해당 스레드를 실행시킨다.

이뿐만 아니라 priority 스케줄링은 낮은 우선순위를 가진 스레드가 starvation 현상을 겪을 수 있는 취약점이 있기에, aging도 구현할 것이다. 시간이 흐를수록 우선순위를 높여주어 기아 현상을 방지하는 역할을 한다.

* 1. Advanced Scheduler (추가구현을 한 경우)

이번에는 BSD 스케줄러를 구현한다. 이를 위해 MLFQ(Multi-Level Feedback Queue)를 활용하고, 각 우선순위에 따라 큐를 생성한다. 함수 호출 시 가장 높은 우선순위를 가진 큐에서 스레드가 선택될 건데, 선택된 스레드들은 round-robin 방식으로 실행될 것이다.

* 1. **개발 내용**

1. Blocked 상태의 스레드를 어떻게 깨울 수 있는지 서술.

timer\_sleep 함수는 스레드의 wakeuptime을 설정하고, 해당 시간까지 스레드를 블록시키는 역할을 한다. Timer\_interrupt 함수는 타이머 인터럽트가 발생할 때 마다, MLFQ 스케줄러와 관련된 처리를 수행한다. 먼저, 시간이 흐를 때마다 CPU 부하 평균 값을 계산하고, 모든 스레드의 recent\_cpu 값을 업데이트한다. 또한, 일정 시간마다 모든 스레드의 우선순위를 업데이트한다. 그리고 blocked 상태의 스레드들을 관리하는 새로운 리스트를 만들어 해당 시간에 깨워져야 하는 스레드들을 찾아 unblocked 상태로 전환시키고 ready queue에 넣어주는 역할을 한다. 이렇게 함으로써 blocked 상태의 스레드들을 효율적으로 깨울 수 있게 된다.

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

우선순위가 높은 스레드가 ready 리스트에 들어오면, 현재 실행 중인 스레드를 ready 상태로 변경한 후, ready 리스트에 우선순위 순서대로 삽입한다. 그리고 ready 리스트에서 우선순위가 가장 높은 스레드를 선택하여 실행하는 방식으로 동작한다. 이렇게 함으로써 우선순위가 높은 스레드가 우선적으로 실행되도록 보장된다.

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

먼저, 스레드가 생성될 때 nice와 recent\_cpu 값은 0으로 초기화되고, 자식 스레드는 부모 스레드의 값을 받는다. Nice는 -20에서 20 사이의 값을 갖으며, 높은 Nice 값은 우선 순위를 낮추므로 변경 시 스레드의 우선순위를 재계산해야 한다.

Recent\_cpu는 스레드의 CPU 사용량을 추정한다. 타이머 인터럽트마다 running 상태의 스레드의 recent\_cpu 값은 1 증가하며, 매초마다 모든 스레드의 recent\_cpu가 재계산된다. 아래 식으로 계산된다:

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

Load\_avg는 ready 상태의 스레드 수의 평균을 나타내는 전역 변수로, 초기에는 0으로 설정된다. 매 초마다 update되며 아래와 같이 계산된다:

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

최종적으로 priority는 아래 식으로 계산된다:

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

이때 priority 값은 0 (PRI\_MIN)부터 63 (PRI\_MAX) 사이의 값을 갖는다. Pintos에서는 부동 소수점 연산을 지원하지 않기 때문에 fixed-point 연산으로 구현해야 한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

11/1 – 11/4 alarm clock 구현

11/5 – 11/10 priority scheduling 구현

1/11 – 11/18 advanced scheduling 구현

11/19 – 11/19 보고서 작성

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

Blocked 상태의 스레드를 관리하기 위해 sleeping\_thread\_list 자료구조를 추가하고, timer\_sleep() 함수에서는 스레드를 깨울 시점을 저장하는 wakeup\_tick 변수를 스레드 구조체에 추가합니다. 스레드를 깨우는 것은 timer\_interrupt() 함수 내에서 구현한다. 매 tick마다 호출되는 timer\_interrupt() 함수 내에서 실행되어 스레드를 깨웁니다.

Ready 리스트에 현재 실행 중인 스레드보다 우선 순위가 높은 스레드가 들어오면 priority scheduling에 따라 처리해야 합니다. 기존에는 list\_push\_back() 함수를 통해 단순히 추가되었지만, 우선 순위에 따라 정렬하여 추가해야 합니다. 이를 위해 list\_insert\_ordered() 함수로 변경하여 우선 순위에 맞춰 정렬된 형태로 추가합니다. 이를 위해 priority\_compare 함수를 추가하여 우선 순위를 비교합니다. 이렇게 함으로써 ready\_list가 우선 순위에 맞게 관리됩니다.

고급 스케줄러에서 priority 계산에 필요한 요소들은 다음과 같습니다. 스레드 구조체에 recent\_cpu와 nice 변수를 추가하고, update\_priority()와 update\_recent\_cpu() 함수를 추가합니다. nice 값의 업데이트는 thread\_set\_nice() 함수에서, load\_avg 값과 recent\_cpu 값의 업데이트는 update\_recent\_cpu() 함수에서 이루어지며, 최종적인 priority 값의 업데이트는 update\_priority() 함수에서 실행됩니다. 이러한 추가 구현을 통해 스레드의 우선순위를 계산하고 관리합니다.

1. **연구 결과**
   1. **Flow Chart**
      1. **Alarm Clock**

**스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명**

* + 1. **Priority Scheduling**

**텍스트, 스크린샷, 폰트, 그래픽 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.h thread 구조체에 이번 프로젝트를 위한 변수 int wakeup\_tick, recent\_cpu, nice를 선언한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

devices/timer.c에서 sleeping thread를 관리할 queue를 선언한다. Timer\_sleep() 함수에서 busy waiting 부분을 주석으로 처리하고, 바꿔준다.

텍스트, 폰트, 스크린샷, 그래픽이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

timer\_init() 함수에서 sleeping\_thread\_list를 초기화한다.

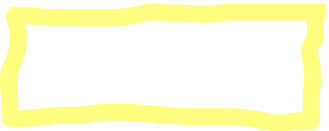
텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Timer\_interrupt() 함수에서 sleeping\_thread\_list를 돌면서 깨울 thread가 있는지 탐색한다. 이는 ticks이 증가할 때마다 wakeupt\_tick이 ticks보다 작을 경우 sleeping\_thread\_list에서 해당 thread를 삭제하고 다음 thread를 unblock하여 깨운다.

텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명



* 1. Priority Scheduling

텍스트이(가) 표시된 사진

자동 생성된 설명

Project 3 자료에 나온 변수들을 해당 파일에서 선언해준다.

threads/synch.c의 sema\_up()함수에서 새로운 thread들이 왔을 때 이를 정렬해주어야 한다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Threads/thread.c의 thread\_create(), thread\_yield(), thread\_unblock() 함수에서 thread의 priority에 따라 리스트에서 scheduling 되도록 해준다.



텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

List\_insert\_order의 3번째 인자로 주었던 priority\_compare() 함수는 threads/thread.c에 선언된다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

thread\_set\_priority() 함수는 current thread의 priority를 새로 업데이트한다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

thread\_get\_priority() 함수는 current thread의 priority를 반환한다.

텍스트, 폰트, 스크린샷, 일렉트릭 블루이(가) 표시된 사진

자동 생성된 설명

thread\_set\_nice() 함수는 현재 thread의 nice 값을 함수의 인자로 nice 값으로 업데이트 한다. 또한, priority도 새로 재설정하는데, 다음 식으로 계산한다.

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

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

thread\_get\_nice() 함수는 현재 thread의 nice 값을 반환한다.

텍스트, 스크린샷, 폰트, 일렉트릭 블루이(가) 표시된 사진

자동 생성된 설명

thread\_get\_load\_avg() 함수는 system의 load average에 100을 곱한 값을 반환한다.

텍스트, 폰트, 스크린샷, 일렉트릭 블루이(가) 표시된 사진

자동 생성된 설명

thread\_get\_recent\_cpu() 함수는 현재 thread의 recent\_cpu 값에 100을 곱한 값을 반환한다.

텍스트, 스크린샷, 폰트, 일렉트릭 블루이(가) 표시된 사진

자동 생성된 설명

update\_recent\_cpu() 함수는 load\_avg 값을 다시 계산하여, recent\_cpu 값 또한 재계산한다. 다음은 두 변수를 계산하는 식이다.

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

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

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

update\_priprity() 함수에서 nice와 recent\_cpu로 priority를 계산한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

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

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

priority\_lifo.c는 동일한 우선순위를 갖는 여러 개의 스레드를 생성한 후, 라운드 로빈 방식으로 동작하는지 확인한다. 모든 스레드가 동일한 우선순위를 가지므로 처음에는 main 스레드의 자식 스레드들이 실행되지 않는다. 이후, main 스레드의 우선순위를 PRI\_DEFAULT 값으로 설정하면 우선순위가 15인 스레드부터 순차적으로 실행된다. ready list에서 스레드가 나올 때 우선순위 순으로 정렬되므로 그 이후에도 우선순위에 따라 스레드가 실행된다. 따라서 반복해도 동일한 출력이 나오게 된다.

* make check 수행 결과를 캡처하여 첨부

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명