**Pintos Project 3: Threads**

담당 교수 : 문의현

학번 / 이름 : 20181536 / 엄석훈

개발 기간 : 2022.11.15 ~ 2022.11.29

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술.

이번 프로젝트에서는 운영체제에서 thread와 관련되어 alarm clock, priority scheduling 기법과 aging, bsd scheduler를 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 사용되지 않을 때 sleep상태가 된다. 이때 원하는 시간만큼 지났는지 매번 확인하는 과정에서 running state와 ready state를 반복하는데 이러한 비효율적인 방법이 아닌 thread가 sleep되면 block state로 바꾸고 저장해둔 뒤 원하는 시간이 지난 뒤 다시 꺼내 ready state로 이동할 수 있도록 해준다.

* 1. Priority Scheduling

현재 pintos에서 thread\_yield()나 thread\_unblock()이 호출되어 새로운 thread가 ready queue에 들어오면 무조건 뒤로 들어가서 순서를 기다리는 round-robin 스케줄링 기법을 사용하고 있다. 이를 개선하여 각 thread별로 우선순위를 부여하여 우선순위를 기준으로 ready queue에 들어갈 수 있도록 priority 스케줄링 기법을 구현한다. 이 때 우선순위가 낮은 thread는 영원히 호출되지 못하는 starvation이 발생할 수 있음으로 aging기법을 구현하여 시간이 지날수록 thread의 priority가 증가할 수 있도록 해준다.

* 1. Advanced Scheduler (추가구현을 한 경우)

추가 스케줄러로 BSD스케줄러를 구현한다. 이는 multi-level feedback queue 또는 multi-level ready queue를 이용하여 각 priority별로 ready queue를 구현하여 관리해주는 스케줄러이다. 이때 이를 관리하기 위해서 nice, priority, recent\_cpu, load\_avg값을 계산하면서 스케줄러가 동작할 수 있도록 해주어야 한다. 그러나 이번 구현에서는 더 쉽게 구현하기 위해서 하나의 큐에서 모든 thread를 관리하도록 해주었다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술

1. Blocked 상태의 스레드를 어떻게 깨울 수 있는지 서술.

쓰레드가 sleep하여 blocked 상태가 되면 이를 block된 thread만 모아두는 큐에 저장해준다. 그리고 매번 이 큐를 확인하여 깨워야 할 시간이 된 thread를 깨워서 ready queue에 넣어준다.

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

원래의 running thread는 실행을 중단하고 priority 순서에 맞춰서 다시 ready queue에 다시 삽입해주고 새로운 thread를 ready list에서 꺼내어 실행시켜준다.

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

Priority 계산을 위해서는 -20부터 20의 값을 가지는 nice가 필요하다. Nice값이 양수라면 priority는 감소한다. 다음으로 recent\_cpu값이 필요한데 해당 thread가 소모한 cpu time으로 매 tick마다 running thread의 recent\_cpu값은 1씩 증가한다. 그리고 상태와 상관없이 모든 thread는 매 초 recent\_cpu = (2 \* load\_avg) / (2 \* load\_avg + 1) \* recent\_cpu + nice 라는 공식에 따라서 다시 계산된다. 마지막으로 load\_avg값은 ready 상태인 thread 개수의 평균으로 처음에는 0으로 시작하여 매 초 (59/60)\*load\_avg + (1/60)\*ready\_thread 라는 공식에 따라서 다시 계산된다. 그리고 마지막으로 매 4tick마다 priority는 PRI\_MAX – (recent\_cpu / 4) – (nice \* 2) 라는 공식에 따라서 다시 계산된다. 그리고 이때 priority는 0과 63사이의 값만 가질 수 있음으로 이를 벗어낫다면 마지막으로 조정된다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

11.15 ~ 11.18 : alarm clock 구현

11.19 ~ 11.23 : priority scheduling 구현 (aging 제외)

11.24 ~ 11.26 : aging 구현

11.27 ~ 11.28 : advanced scheduler 구현

11.29 : 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

1) alarm clock

thread.h : thread 구조체에 해당 thread가 sleep할 시간을 저장하는 변수 sleeptime을 선언해준다.

thread.c : thread\_init()함수에서 sleeptime변수를 0으로 초기화 해주고blocked\_queue를 초기화 해준다. 또한 sleep할 때와 wake할 때 필요한 thread\_sleep()함수와 thread\_awake함수를 만들어준다.

timer.c : timer\_inturrupt()함수에서 thread\_awake를 호출하도록 해주고 timer\_sleep()함수에서는 thread\_sleep()함수를 사용해 thread를 재우도록 해준다.

2) priority scheduling

init.c : pintos를 실행할 때 -aging이 입력되면 thread\_prior\_aging을 true가 되도록 해준다.

thread.h : synch.h 파일을 include하고 aging기법에 사용할 thread\_prior\_aging을 선언해준다.

thread.c : aging기법에 사용할 thread\_prior\_aging을 선언해주고 thread\_yield()함수와 thread\_unblock()함수에서 list\_push\_back대신 list\_insert\_ordered함수로 대치해주고 이 때 비교할 때 사용할 thread\_check\_priority()함수를 만들어준다. 또한 마지막으로 thread\_create()될 때 priority를 비교하여 thread를 실행할 수 있도록 해준다.

synch.c : semaphore를 사용할 때도 list\_push\_back대신 list\_insert\_ordered함수를 사용하도록 바꿔준다.

float.h : 실수 계산을 위한 각종 함수들을 만들어준다.

timer.c : timer\_inturrupt함수에서 thread\_prior\_aging이 true일 때 priority가 갱신되도록 해준다.

3) advanced scheduler

thread.h : thread구조체에 recent\_cpu와 nice를 포함시킨다.

thread.c : thread\_init()함수에서 recent\_cpu와 nice값을 초기화 시켜준다. 또한 init\_thread에서는 자식 thread가 부모의 recent\_cpu와 nice값을 상속받도록 해준다. load\_avg값을 전역변수로 선언하고 초기화 해준다. 그리고 bsd 스케줄러에서 필요한 각종 계산식을 담당하는 함수들을 만들어준다. 마지막으로는 thread\_set\_priority함수를 동작하지 않도록 바꿔주고 thread\_get\_priority, thread\_set\_nice, thread\_get\_nice, thread\_get\_load\_avg함수들이 역할을 하도록 수정해준다.

timer.c : timer\_inturrupt()함수에서 thread\_mlfqs가 true일 때 각 시간에 맞게 load\_avg, nice, recent\_cpu, priority를 다시 계산하도록 해준다.

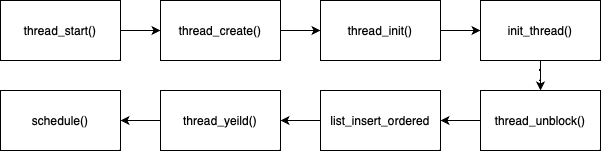
1. **연구 결과**
   1. **Flow Chart**

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

1) alarm clock



2) 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) alarm clock

텍스트이(가) 표시된 사진

자동 생성된 설명

thread 구조체에 sleeptime변수를 포함시켜준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

blocked\_queue를 저장하는 list를 thread.c에 선언해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

thread\_init()함수에서 sleeptime변수와 blocked\_queue을 초기화한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

thread가 sleep할 때 해당 thread를 blocked\_queue에 넣어서 sleep상태로 만들어 주는 thread\_sleep()함수와 blocked\_queue를 돌면서 sleeptime을 지난 thread를 찾아서 thread\_unblock시켜 깨워주는 thread\_wakeup()함수를 만들어준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

timer\_sleep()함수에서 기존처럼 시간이 다 될때까지 thread\_yield()시키는 방식이 아닌 위에서 만든 thread\_sleep()함수를 사용하도록 코드를 수정해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

timer\_interrupt()함수에서 매 틱마다 thread\_wakeup()함수를 호출해서 깨울 시간이 된 thread를 확인하여 깨워주도록 만들어준다.

2) priority scheduling

텍스트이(가) 표시된 사진

자동 생성된 설명

thread.h에서 aging기법을 위해서 synch.h를 include하고 thread\_prior\_aging변수를 선언해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

또한 aging기법을 위해서 init.c의 parse\_option()함수에서 -aging이라는 옵션이 걸리면 thread\_prior\_aging이 true가 되도록 해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

마지막으로 thread.c에서도 aging을 위해서 thread\_prior\_aging변수를 선언해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

thread\_create()에서 thread가 생성될 때 해당 thread가 현재 실행되는 thread보다 우선순위가 높은지 확인하여 교체하도록 해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

thread\_yield()함수와 thread\_unblock()함수에서 list\_push\_back대신 list\_insert\_ordered()함수를 사용하여 ready\_list에 우선순위별로 thread를 정렬하여 집어넣어 주도록 한다. 이를 통해서 ready\_list의 맨 앞에서 thread를 꺼내면 가장 우선순위가 높은 thread가 꺼내지는 효과를 낼 수 있다.



list\_insert\_ordered()함수에서 사용할 우선순위를 비교하는 함수를 만들어준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

현재의 thread의 우선순위와 ready\_list가장 앞 thread의 우선순위를 비교하여 만약 ready\_list의 thread가 실행될 수 있도록 하는 함수를 만들어준다.

3) advanced scheduler

텍스트이(가) 표시된 사진

자동 생성된 설명

Advanced scheduler인 bsd scheduler 에서 필요한 recent\_cpu와 nice를 thread.h의 thread 구조체에 선언해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

또한 load\_avg도 thread.c에 전역번수로 선언해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그리고 thread.c 의 thread\_init()함수에서 load\_avg, recent\_cpu와 nice값을 초기화 해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그리고 init\_thread()함수에서 recent\_cpu와 nice가 부모 thread로 부터 상속받을 수 있도록 해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그리고 recent\_cpu, nice, priority, load\_avg를 계산할 때 실수 계산이 필요함으로 부동소수점 계산이 필요함으로 이를 위한 함수들을 모아둔 float.h 파일을 만들고 thread.c에서 include해준다.

텍스트이(가) 표시된 사진

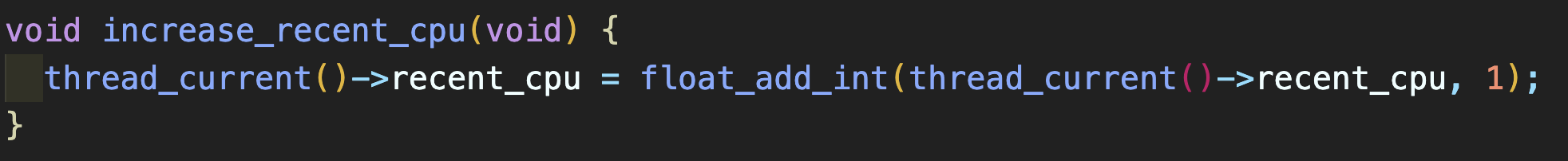
자동 생성된 설명텍스트, 화면이(가) 표시된 사진

자동 생성된 설명텍스트, 화면이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

float.h 파일에 선언된 함수들은 위와 같다. 각각의 함수는 pintos manual을 참고하여 만들어주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

그리고 위에서 만든 실수 계산과 pintos manual의 bsd scheduler의 각 항목들에 대한 계산 공식을 적용하여 계산해주는 함수를 만들었다. Increase\_recent\_cpu는 현재 실행중인 thread하나의 recent\_cpu값을 1 증가시켜주는 함수이다. 다음으로 update\_load\_avg, update\_recent\_cpu, update\_priority함수는 load\_avg값을 갱신해주고, 모든 thread의 recent\_cpu와 priority값을 다시 계산해주는 함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

추가적으로 thread\_set\_priority함수는 thread\_mlfqs가 true일 때는 사용되지 않도록 해주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

마지막으로 thread\_set\_nice 함수에서 thread의 nice값을 새로운 new\_nice로 바꾸고 이에 따라 해당 thread의 우선순위를 다시 계산해준 뒤 ready\_list의 다른 thread와의 우선순위를 비교하는 기능을 구현해주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

그리고 마지막으로 thread\_get\_nice함수에서는 현재 thread의 nice값을 리턴하는 기능을, thread\_get\_load\_avg는 load\_avg값을 100배 해서 return하는 함수를, thread\_get\_recent\_cpu는 현재 thread의 recent\_cpu를 100배 해서 return하는 기능을 구현해주었다.

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

텍스트이(가) 표시된 사진

자동 생성된 설명

Priority-lifo 테스트는 동일한 priority를 가진 thread를 16개 생성하여 동작시켜보는 테스트이다. 이 때 동일한 priority를 가지고 있음으로 각 thread가 돌아가면서 last-in-first-out을 지키며 수행되어야 한다. 그리고 그 결과 16개의 thread가 모두 16번씩 순서를 지켜가며 실행되는 것을 확인할 수 있다.

* make check 수행 결과를 캡처하여 첨부

텍스트이(가) 표시된 사진

자동 생성된 설명