**Pintos Project 3: Threads**

담당 교수 : 문의현

학번 / 이름 : 20171300/지용환

개발 기간 : 11/17 ~ 12/04

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술.

thread와 관련된 기능을 구현하는 것으로, 원래는 running, ready를 반복하는 방식으로 구현되어있는 alarm clock을 효율적으로 변경하는 것이다. 또한 round-robin방법을 사용한 스케쥴링에 우선순위를 부여, 우선순위 스케쥴링이 되도록한다. 추가적으로는 multi-level feedback queue를 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 작동하니 않을시 sleep을 시킬 때, 다시 깨우는 alarm clock은 현재 비효율 적인busy-waiting으로 구현되어 있는데, 이를 수정하여 잠든 threadd를 block state로 두고, 깨울 시간에 ready로 변경시켜준다. 즉 효율성이 높아진다.

* 1. Priority Scheduling

원래는 우선순위에 관계없이 thread는 ready list의 끝에 삽입되었는데, 우선순위 스케쥴링을 통해 높은 우선순위를 가진 것을 list에 삽입시킨다.

* 1. Advanced Scheduler (추가구현을 한 경우)

추가적으로 구현을 하는 것은 multilevel feedback 큐로써, BSD 스케쥴러를 구현하는 것이다. 여러 큐가 ready 큐가 존재하며 각 큐는 서로 다른 우선순위를 가지고 있다. 당연히 우선순위가 높은 큐가 먼저 실행이 된다. 우선운위에 따라 실행되는 순서가 정해지는 큐는 각각 round-robin을 따른다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술

1. Blocked 상태의 스레드를 어떻게 깨울 수 있는지 서술.

먼저 일어날 시간을 확인하는 방법을 추가하기 위해 int형 변수를 thread에 주가시켜 준다. 이 일어날 시간이 오지않은 thread는 blocked상태이므로, 하나의 리스트를 생성하고 모든 blocked된 스레드를 이 리스트에 저장을 시켜준 이후, 매 tick마다 이 리스트의 thread를 검사하며 일어날 시간이 되면, 리스트에서 제거 및 unblock으로 바꿔주고(즉 깨워준 이후) ready 큐에 넣어주면 해결된다.

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

ready list에 높은 우선순위를 가진 thread가 들어오면 높은 우선순위의 thread가 실행되어야 하는 것이 우선순위 스케쥴링이므로, 현재 실행중인 thread의 상태를 ready로 바꾼 이후 ready list에 우선순위 순서대로 정렬을 시켜준다. 마지막으로 ready list에서 그냥 높은 우선순위를 가진 thread를 실행시켜주면 된다.

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

먼저 우선순위가 priority=pri\_max – (recent\_cpu/4) – (nice\*2)의 방법으로 계산되므로 알아야 할 면수는 pri\_max, recent\_cpu, nice에 대해 알아야 한다.

Pri\_max는 핀토스에서 미리 세팅된 값으로 스레드가 가질수 있는 최대 우선순위 값이다.

nice는 쓰레드 마다 가지는 값으로 이 값이 클수록 다른 쓰레드의 우선순위를 낮춰버린다. 이렇게 되면 우선순위가 낮아지는 쓰레드의 cpu time값이 낮아지는 결과를 야기한다.

recent\_cpu는 최근에 cpu에 의해 얼마나 사용된지에 대한 값으로 (2 \* load\_avg)/(2 \* load\_avg + 1 )\* recent\_cpu+nice 의 값을 따른다. 이때 load\_avg는 전역변수로 ready 상태 thread의 평균을 의미하는 값이다. load\_avg의 계산은 다음과 같다. =(59/60)\*load\_avg +(1/60)\*ready\_thraeds 주의할 점은 핀토스에 소수점 연산을 하지 않기 때문에 이 연산을 위해 추가적인 방법이 필요하다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

11/17~11/20 alarm clock

11/21~11/25 priority scheduling

11/25 ~12/1 advanced scheduling

12/3~12/4 보고서

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

1. alarm clock

alarm clock을 구현하기 위해서는 timer.c, thread.h, thread.c를 수정해야 하는데, 먼저 timer.c에서는 잠든 쓰레드를 확인하기 위한 blocked\_list를 선언해 여기 block상태인 thread를 관리한다. timer\_sleep함수에서 스레드의 일어날 시간을 계산하고 block\_sleep리스트에 넣어주는 과정을 구현하고, timer\_interrupt에서 이 잠든 thread가 일어날 시간이 됬으면 깨워주는 과정을 구현한다. 당연히 thread.h에서 일어날 시간을 계산하기 위한 int형 변수를 구조체도 추가해준다.

1. priority scheduling

thread.c와 synch.c, init.c의 수정이 필요하다.

먼저 thread\_yield와 thread\_unblock를 할때 thread를 우선순위에 따라 ready\_list에 넣어주는 과정을 추가해준다. 이때 결국엔 비교가 필요하기 때문에 비교함수를 하나 생성해 준다. 뿐만 아니라 새로 생성한 쓰레드가 현재 쓰레드의 우선순위보다 높으면 그 쓰레드를 yeild할수 있도록 thread\_create의 수정또한 필요하다. 이제 세마포어 문제를 해결하야 하는데 synch.c에서 우선순위에 대한 세마포어들을 수정해주면 된다. sema\_up을 할 때는 waiters(세마를 기다리고 있는 리스트)중에 우선순위가 가장큰 쓰레드를 시켜주면 되고, 이후 다시 schedunlig을 해주면 우선순위 스케쥴링은 해결된다.

3.

먼저 thread.c, thread.h, init.c에서 ppt처럼 추가를 해준 이후, thread의 구조체에 nice와 recent cpu를 선언해준다. 초기화는 자식 스레드인경우 부모의 값, 아니면 0으로 설정해 준다. 앞서서 설명한 식을 계산하기 위해 thread.c에 추가적인 함수(그리고 계산)를 구현해 주고 bit shift연산을 위해 FRACTION또한 정의해 준다. 이때 load\_avg는 말했듯이 전역변수로 선언해준다. 이 함수들을 통해 우선순위가 모두 계산되면 timer.c의 timer\_interrupt에서 4초마다 우선순위를 업데이트 그리고 timer가 timer\_FREQ만큼 tick이 흘렀다면 load\_avg와 recent\_cpu를 업데이트 하도록 한다.

1. **연구 결과**
   1. **Flow Chart**
2. **alarm clock**

텍스트이(가) 표시된 사진

자동 생성된 설명

1. **우선순위 스케쥴링**

텍스트이(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

1. **alarm clock**

**가장** 먼저 thread.h부분의 thread 구조체의 수정이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

일어날 시간을 계산하기 위한 int wakeup변수를 추가했다.  
thread.c의 thread\_init에서 0으로 초기화를 해주면 된다.



timer.c에서 가장먼저 해야할 일은 자고있는 스레드의 리스트를 만들어 주는것이다. 

초기화는 간단히 list\_init함수를 사용해 해주면 된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

본격적으로 timer\_sleep의 수정사항이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이전의 busy waiting방법을 지운후 thread의 wake시간을 계산하고, 현재 쓰레드의 wake요소에 넣어준다. 그후 block list에 추가하고 block상태로 만들어 준다.

timer\_interrupt의 수정사항이다.

텍스트, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

tick이 변할대마다 thread\_up함수(아래에 설명)을 이용해 일어날 thread를 파악해 그 쓰레드를 깨워준다. 나머지는 alarm clock과는 관계없다.

thread\_up함수는 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

일어날 시간을 계산하고 thread를 깨워주는 함수로, block\_list에서 처음부터 끝까지 탐색하고 만약 wakeup이 틱보다 작을 때, 일어날 시간, 즉 blocklist에서 제거하고 그 쓰레드를 unblock시켜준다.

1. priority scheduling

thread\_create함수의 수정이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

만약 쓰레드를 생성할 때, 현재 쓰레드 우선순위(thread\_get priority를 사용해 얻을수 잇다)와, 생성하는 쓰레드의 우선순위를 비교해, 생성 쓰레드의 우선순위가 높다면 yield를 호출한다. 먼저 thread\_get\_priority는 다음과 같다.(설명생략)

텍스트이(가) 표시된 사진

자동 생성된 설명

thread\_yield 함수는 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

간단히 list\_insert\_order함수를 이용해 그 순서에 맞게 ready list에 넣어주고 현재 쓰레드의 상태를 ready로 바꾼이후 스케쥴링 해주는 함수이다. 이때 비교함수 list\_pp는 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

프로젝트 0에서 사용한 함수로, 그냥 크기를 비교해주는 함수이다.

thread\_unblock의 수정사항이다



단순히 list\_push\_back을 했던 것 대신, 우선순위에 맞게 ready\_list에 들어가도록 수적해준다.

마지막으로 synch.c의 sema\_up의 함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

waiters에는 sema의 순위를 기다리는 thraed들이 존재하고있는 리스트인데 여기서 모든 쓰레드를 검사 하며 최대의 우선순위를 가진 쓰레드를 찾는다. 찾은 이후 list\_remove로 waiters에서 제거를 하고, thread\_unblock으로 해당 thread를 unblock상태로 변경해준다. 그 이후 thread\_yield를 통해 재 스케쥴링 해주는 함수이다.

1. Advanced scheduler

ppt에 있는 수정사항은 생략하겠다.

ppt에서 준 수정사항은 생략하도록 하겠다.

thread.h의 쓰레드 nn과 cpu가 nice와 recent\_cpu를 계산하기 위한 변수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이또한 thread.c에서 0으로 수정해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이때 만약 자식 쓰레드라면 부모의 값을 그대로 가져오도록 init\_thread에 추가를 해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

timer.c의 수정사항이다.

텍스트이(가) 표시된 사진

자동 생성된 설명]

매 tick마다 cpu의 값을 증가시켜주고 timer\_tick이 TIMER\_FREQ만큼 흘렀으면 nice와 cpu의 값을 4 tick마다는 우선순위를 업데이트 해준다.

thread.c의 update\_nn\_cpu함수는 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

만약 현재 쓰레드가 idle이 아니라면 ready\_list의 크기 +1의 값을 ready\_threads로 사용해서 load\_avg를 위에 설명한 식으로 구해줘 update를 해준다. 또한 all\_list의 모든 쓰레드들에게 idle이 아닌지 검사하며 만약 idle아니라면 해당 쓰레드의 recent\_cpu의 값도 재할당 해준다. 계산식은 위에서 설명한 식이다. 단순히 ppt의 식에 FRACTION을 추가해준 것인데 이유는 실수연산을 위해서이다. 정수의 모든 연산에 FRACTION을 곱해주면 되는데 32bit중 하위 14bit가 fraction이기(실수부분) 때문이다.

다음으로 update\_priority함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

all list를 검사하며 우선순위를 먼저 ppt의 식대로 계산해준다.(당연히 FRACTION추가) 이후, PRI\_MAX,PIR\_MIN의 범위밖으로 나가면 해당값으로 바꿔주고 그외 범위안에 들어온다면 걔산한 우선순위를 부여해주는 함수이다.

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

priority-lifo.c 코드중

텍스트, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명 부분이 의미하는 것은 우선순위를 설정해주는 것으로, 가장 먼저 PRI\_DEFAULT +THRAED\_CNT+1 설정해 주고 그 이후 FOR문 16개의 추가적인 쓰레드에 PRI\_DEFAULT+1+i의 우선순위를 부여하며 쓰레드를 생성해주는 것이다. 이때 쓰레드가 생성될 때마다 simpleThreadFunc가 실행되는데 여기서는 op라는 배열에 쓰레드의 id 만큼의 위치에(id의 행에) ITER\_CNT(16)만큼 데이터를 써주고 thread\_yield를 실행하며 가장 높은 우선순위를 가진 thread가 cpu를 가지게 해준다. 다시말해 li된(높은 id를 가진) 쓰레드가 ready-list의 front에 있는 것이다. 반복문의 종료후, 마지막으로 처음 thread를 다시 PRI\_DEFAULT로 낮춰서 ready\_list에 있는 나머지 thread에게 cpu를 넘겨주며 이젠 15의 우선순위를 가진 id가 선택이 된다. 그 다음 for문은 다음과 같은데, 현재 id=15임을 유의하자

텍스트이(가) 표시된 사진

자동 생성된 설명(일단 simpleThreadFunc함수인데, op에 ITER\_CNT만큼 데이터를 저장해주는 함수이다.)

텍스트이(가) 표시된 사진

자동 생성된 설명

위의 코드는 마지막 반복문은데, 그냥 output이 op를 for문으로 돌아가며 계속 d->id를 출력시켜준다. 즉 id가 15인 것이 ITER\_CNT번만큼 출력이 되게 된다. 결과를 보면 알수 있듯이 첫번째로 15가 16번 찍혀있다. 나머지의 결과도 우선순위에 따라 14 ~0의 순서로 16번씩 프린트 되는 결과를 보인다.

* 텍스트이(가) 표시된 사진

  자동 생성된 설명
* make check 수행 결과를 캡처하여 첨부
* 텍스트이(가) 표시된 사진

  자동 생성된 설명