Pintos Project - Threads

1. 프로젝트 개요

1.1. Pintos 커널의 `timer_sleep()` 함수를 개선한다.

1.2. Pintos 에 우선순위 스케줄러(priority scheduler)를 구현한다. 또한 우선순위 스케줄링 시 발생할 수 있는 우선순위 역전 (priority inversion) 을 방지할 수 있는 priority donation 기능을 구현한다.

1.3. [Extra credit] 다단계 피드백 큐 스케줄러 (multilevel feedback queue scheduler)를 구현한다.

2. 프로젝트 내용

2.1. Alarm clock 의 개선

- ✓ [문제정의] '`devices/timer.c`'에 정의되어 있는 `timer_sleep()` 함수를 개선한다. 현재의 `timer_sleep()`은 기능적으로는 정상 동작하나, 정해진 시간이 경과할 때까지 `thread_yield()`를 반복 호출하며 busy waiting 하도록 구현되어 있다. 이를 busy waiting 없이 수행하도록 수정하여 성능을 개선한다.
- ✓ [해결] `void timer_sleep (int64 t ticks)`를 호출한 쓰레드를 현재 시각 기준으로 `ticks` 시간이 경과할 때까지 block 시키도록 수정한다 (`ticks` 는 timer tick 단위로 표시). 이후 `ticks` timer tick 이상이 경과하면 해당 thread 를 ready 상태로 전이시키면 된다. 이를 위해 `timer_sleep()`에 의해 block 된 스레드들이 대기할 수 있는 리스트를 만들고, 매 번 timer interrupt 가 발생할 때마다 (즉 timer interrupt handler 함수 `timer_interrupt()`에서) timer tick 이 `ticks` 이상 지났는지 검사하는 방식으로 구현한다.
- ✓ [테스트] 구현된 `timer_sleep()` 함수가 제대로 동작하는지 여부는 테스트 `alarm-single`, `alarm-multiple`, `alarm-simultaneous`, `alarm-priority`, `alarm-zero`, `alarm-negative` 를 사용하여 시험한다 (이 테스트들은 모두 `timer_sleep()` 함수를 사용하는 프로그램들이다. 따라서 원본 pintos 커널뿐만 아니라 `timer_sleep()` 함수가 수정된 후에도 정상 동작하여야 한다.

2.2. 우선순위 스케줄러 구현

- ✓ [문제정의] ⑴ 현재 pintos 에는 라운드로빈 방식의 스케줄러가 구현되어 있다. 이를 스레드 별 우선순위에 따라 스케줄링 할 수 있는 우선순위 스케줄러를 새로 구현한다. ⑵ 우선순위 스케줄링 방식의 문제점은 우선순위 역전 현상이 일어날 수 있다는 것이다. 이를 방지할 수 있도록 우선순위 스케줄러에 priority donation 기능을 구현한다.
- ✓ [해결] 각 스레드가 자신의 우선순위를 확인하고 우선순위를 변경할 수 있도록 두 가지 함수 `void thread_set_priority(int new_priority)`와 `int thread_get_priority(void)`를 구현한다.
- ✓ [테스트] 구현된 우선순위 스케줄러가 제대로 동작하는지 여부는 테스트 `priority-change`, `priority-preempt`, `priority-fifo`, `priority-sema`, `priority-convar`, `priority-donate-one`, `priority-donate-multiple`, `priority-donate-multiple2`, `priority-donate-nest`, `priority-donate-chain`, `priority-donate-sema`, `priority-donate-lower` 를 사용하여 시험한다.

2.3. [Extra credit] Multi-level feedback queue scheduler 구현

- ✓ [문제정의] 평균 응답 시간의 개선을 위해 4.4BSD 운영체제의 스케줄러와 유사한 동작을 하는 multilevel feedback queue scheduler(MLFQS)를 구현한다.
- ✓ [해결] MLFQS 도 우선순위 스케줄러와 유사하게 스레드의 우선순위 기반으로 동작하나 priority

donation 은 사용하지 않는다. 따라서 이 스케줄러의 구현을 시작할 때에는 donation 기능이 없는 상태의 우선순위 스케줄러로부터 시작하는 것이 좋다. 이 스케줄러는 pintos 를 시작할 때 –mlfqs 옵션을 주면 동작하도록 구현한다. MLFQS 는 각 스레드의 우선순위를 스케줄러가 정하기 때문에 스레드 생성 시 사용자가 지정한 priority 값은 무시되며, 스레드가 `thread_set_priority()`나 `thread_get_priority()`를 호출한 경우에도 단순히 현재의 스레드 우선순위를 되돌려 주도록 처리한다. 구체적인 MLFQ 스케줄링 알고리즘은 수업 시간에 다룬 MLFQ 스케줄러에 대한 기본 내용을 기반으로 게시판에 첨부한 4.4BSD 운영체제의 MLFQ 스케줄러에 대한 내용을 참고로 한다.

✓ [테스트] 구현된 MLFQS 가 제대로 동작하는지 여부는 테스트 `mlfqs-load-1, mlfqs-load-60, mlfqs-load-avg, mlfqs-recent-1, mlfqs-fair-2, mlfqs-fair-20, mlfqs-fair-2, mlfqs-fair-10, mlfqs-block` 을 사용하여 시험한다.

3. 팀구성

3.1. 프로젝트는 팀을 구성하여 진행한다. 각 팀은 최대 3 인 이내로 자유롭게 구성한다.

3.2. 팀 구성과 과제 평가

✓ 과제 종료 시 각 팀원의 기여도를 자체 평가하여 각 팀에 주어지는 ticket 을 나누어 가진다. 과제 수행에 많은 기여를 한 팀원일수록 많은 수의 ticket 으 가지도록 한다. 기여도의 근거는 작성한 코드 라인 수, 이론적인 기여도, 작성한 보고서 분량 등을 팀원들 간에 의논하여 결정하되, 단순한 문서 작업만으로는 **절대** 기여도를 인정할 수 없다. 팀원 별 ticket 수는 과제 결과 보고서에 꼭 명시하도록 한다.

✓ 각 팀마다 할당되는 ticket 의 수는 3 명인 팀은 9 장, 2 명인 팀은 7 장, 1 명인 팀은 4 장이다. 이를 기여도에 따라 나누어 가지되, (1) 한 사람이 최대 4 장까지 가질 수 있고, (2) 동일한 수의 ticket 을 가진 팀원은 2 명까지로 제한하며, (3) ticket 을 한 장도 못 받는 팀원도 있을 수 있다. 예를 들어 팀원이 3 명인 경우 (4, 3, 2)나 (4, 4, 1)은 가능하지만 (3, 3, 3)으로는 할 수 없다.

✓ 과제 평가 시 팀에 주어진 프로젝트 점수 중 60%는 모든 팀원에게 동일하게 주어지고, 나머지 40%는 자체 평가한 기여도에 따라 팀원마다 달라진다. 예를 들어, ticket 이 4 장인 팀원은 자기 팀의 점수를 모두 다 받고, ticket 이 1 장인 팀원은 자기 팀 점수의 70%만 받을 수 있다. 단, ticket 을 한 장도 못 받은 팀원은 팀 점수 60%의 1/3 에 해당하는 점수만 받을 수 있다.

✓ 각 팀원의 과제 점수는 티켓 수에 비례하여 부여한다.

4. Guideline

✓ 이 프로젝트는 pintos 소스 코드 트리 중 pintos/src/threads 디렉토리에서 진행한다.

✓ 반드시 Pintos 문서 2 장 내용을 이해한 후 설계, 구현을 시작한다.

✓ 이번 과제의 배점은 전체 성적 중 (100 점 만점 기준) 50 점에 해당한다; 2.1 과 2.2 는 서로 별개로 진행될 수 있으며, 분리해서 제출 가능하다. 2.1 번과 2.2 번의 배점은 각각 15 점과 35 점이다; 2.3 번의 MLFQ 스케줄러 구현은 선택 사항이며, 구현 시 15 점의 가산점이 부여된다.

✓ 코드 작성 시 들여쓰기(indentation)를 철저히 할 것; 들여쓰기가 되어있지 않은 소스 코드는 채점하지 않음

✓ 보고서와 수정한 소스코드를 e-campus 를 통해 제출한다

✓ 제출 마감은 2016 년 12 월 7 일 자정까지이며, 1 일 제출 지연 시 30%, 2 일 지연 시 70% 감점하며, 이후에는 미제출로 간주한다. 기말고사 전 주의 지정보강일에는 제출한 보고서를 기반으로 프로젝트 수행 결과에 대한 발표 및 평가가 진행된다. 특히 이 시간에는 자체적으로 평가한 팀원 간의 기여도에 대한 검증이 이루어진다.