Pintos and Project One

Chia-Hui Tai

CS140 Autumn 07-08
Stanford University

Overview

- Typical OS structure

Adopted from Lecture Notes L1 p.14
Getting Started

• Stanford computing facility
  – UNIX Workstations: elaine, myth, vine, etc.

• Building and Running Pintos
  > set path = ( /usr/class/cs140/`uname -m`/bin $path )
  > zcat /usr/class/cs140/pintos/pintos.tar.gz | tar x
  > cd pintos/src/threads/
  > make
  > pintos -v (--bochs)-- run alarm-multiple

Project One: Threads

• What is a thread? pointer to instruction + state
• Threading System

[Diagram showing CPU, Ready list, Wait lists, Unblock / Wake up, Scheduling, Timer interrupt, Block / Sleep, Running]

Adopted from Forum Parmar’s slides in cs140, winter 2007
Part I – Alarm Clock

• Already in Pintos: devices/timer.c
• Redo timer_sleep() to avoid busy waiting:

```c
/* Suspends execution for approximately TICKS timer ticks. */
void timer_sleep (int64_t ticks){
    int64_t start = timer_ticks () ;
    ASSERT (intr_get_level () = NTR_ON);
    while (timer_elapsed (start) < ticks)
        thread_yield ();
}
```

• Requirement:
  – No busy waiting
  – Also, reduce the time spent in timer interrupt handler

Part II – (a) Priority Scheduling

• Priority Scheduling
  – Thread L yields as H added to ready list
  – Thread H wakes up first when H and L both waiting for a lock, a sema, etc

• Starting Point
  – Read Section 2.1.1
  – Go over the Pintos source code
    • When does content switching happen in Pintos?
    • When does your scheduler need to take action? Point it out from the code.

• Have this part working before doing Part III
Part II – (b) Priority Donation

- Priority inversion problem
  - Consider the scenario:
    - L holds Lock K, running
    - H comes in, kicking out L (L still holds Lock K)
    - M comes in ready list
    - H waits for Lock K; M starts running
    - Now: M runs, then L, and then H
  - What should we do?
    - Priority Donation
    - Required for locks
      - Optional for sema and conditional variable

Part III – Advanced Scheduler

- BCD Scheduler
  - Appendix B4.4 for multi-level feedback queue
  - Enabled when `thread_mlfqs == true`
  - Priority depends on:
    - niceness
    - recent_cpu
    - load_avg

- Fixed-Point Real Arithmetic
Grading

• 50% Design Document
  – Template, asking about:
    • Data structure
    • Algorithm
    • Synchronization
    • Rationale
  – Coding Standard
    • Comment your code!

• 50% Test suite
  – Run `make check` in `build/`
  – Test scripts in `pintos/src/tests`

Useful Tools

• cvs/svn
• cscope
• backtrace
• pintos-gdb
• Test cases
  – Run an individual test (e.g. alarm-multiple)
    `make build/tests/threads/alarm-multiple.result`, OR
    `pintos -v -- run alarm-multiple`
• Data structure
  – Provided in `pintos/src/lib/kernel/`
• Newsgroup
Some Advice

• Read the manual
• Start with Design Document
• Integrate early
• Spend lots of time reading the code
• Synchronization
  – Keep in mind: A thread can be interrupted by another thread