Final Project Proposal - Priority Scheduling

David Chu (chud@carleton.edu)
Luisa Escosteguy (escosteguyl@carleton.edu)

Overview

We have decided to implement a priority based scheduler. More specifically, the following features in sched.c

- If a higher priority thread is added to the ready list (thread.node), the current thread should yield the processor to the new thread.
- Priority donation
    - When to use: when a thread wants to acquire a resource currently being held, its priority is donated to the resource holder.
    - Needed to handle the issue of “priority inversion”.
    - Multiple donation: multiple priorities are donated to a single thread
    - Nested donation: when threads recursively wait for other threads holding locks. We need to impose the maximum depth of nested priority donation.
- Priority tie-breaking
    - If all the threads have the highest priority, switch among the threads in round robin order (8 ms each).

Specifically, we will implement the following functions in thread.c:

- void **thread_set_priority** (int priority)
    - Set thread.priority = priority.
    - Check if the thread still has the highest priority; if not, yields.
- int **thread_get_priority** (void)
    - Returns thread.priority or if it is a priority donation, return max donated priority.

Also, we will modify sched_sched and sched in sched.c to implement priority.

Initial Risk Analysis/Potential Problems

- We are unsure about implementing synchronization and concurrency/multi-processing
- We don’t know where and what additional helper functions are implemented. This reference might be a helpful source for the potential list of useful helper functions.
- We have to think more carefully about different cases that priority donation is required.
- What would be a good set of test cases for this project? We are thinking of priority-simple (2 or 3 threads, high, low and medium priorities), priority-tie and donation-test.

Helpful resources you've found to research topic

- Pintos Projects: Project 1--Threads
    - 2.2.3 Priority Scheduling
    - 2.3.2 Priority Scheduling FAQ
- sched.h(0p) - Linux manual page
- Scheduling Priorities - Win32 apps | Microsoft Docs

Minimum Viable Goal

Our minimum viable goal would be to implement priority scheduling and handle the basic priority inversion cases and priority ties. We will test and demonstrate the correctness of our implementation by writing test cases.

Stretch Goal

- Handle more complex cases of priority donation, like nested donation and multiple donations.
- Build a more holistic and robust testing infrastructure that can test synchronization and edge cases.
- Improve the tie-breaking technique (using other algorithms instead of round-robin to increase efficiency).