Università della Svizzera italiana

Faculty of Informatics

Operating Systems

PintOS: Report Project 2
Schedulers

Group #11

Daniele del Pozzo, Jora Zeneli, Simone Ferreira Machado

Spring Semester, 2025

1 Files changed

- **pintos0/pintos-env/pintos/threads/thread.c** (*modified*)
- **pintos0/pintos-env/pintos/threads/thread.h** (*modified*)

2 Changes

pintos0/pintos-env/pintos/threads/thread.c

- **thread_priority_cmp(...)** (*added*):

  A comparator used for sorting the ready list in descending order of priority. Ensures correct scheduling behavior by selecting the most urgent task first.

- **thread_create** (*updated*): Now includes logic to yield the CPU if a newly created thread has a higher priority than the currently executing one. This ensures the immediate execution of the highest-priority thread for proper preemptive scheduling.

- **update_thread_priority(struct thread *t)** (*added*):

  Computes an individual thread's priority using the formula:

  $$\text{priority} = \text{PRI\_MAX} - \frac{\text{recent\_cpu}}{4} - (\text{nice} \times 2)$$

  This is used in periodic updates and when a thread's `nice` value changes.

- **update_priority_all** (*added*): Recalculates the priority of every thread based on its `recentCPU` and `nice` values. This ensures continuous adjustments to thread priority when using MLFQS.
- **update_recent_cpu_all(void)** (*added*): Updates CPU usage metrics for all threads, applying decay to old values while accounting for nice levels, ensuring fair scheduling.
- **update_load_avg(void)** (*added*): The system-wide load average is computed once per second following the formula from the PintOS documentation:

  $$\text{load\_avg} = \frac{59}{60} \times \text{load\_avg} + \frac{1}{60} \times \text{threadsReady}$$

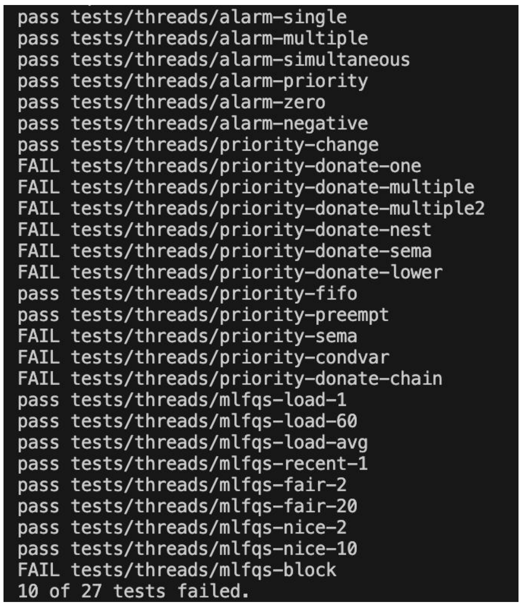
  This helps track system load for scheduling adjustments.
- **thread_set_priority(int)** (*updated*): Function that updates the current thread's priority and yields CPU control if higher-priority threads are waiting. It ensures urgent threads get immediate attention by checking the ready list after each priority change. The system maintains responsiveness by preempting lower-priority threads when needed.
- **next_thread_to_run(void)** (*updated*): Function that selects the highest-priority thread from the ready queue to run next. If no threads are ready, it returns the idle thread to keep the CPU active. The chosen thread is removed from the ready list before execution.
- **thread_yield(void)** (*updated*): Threads are now inserted in priority order via `list_insert_ordered()`, keeping the ready list sorted. This ensures the scheduler can always quickly access the highest-priority thread.
- **thread_unblock(struct thread *)** (*updated*): The ready list now maintains priority order by using `list_insert_ordered()` instead of `list_push_back()`. This prevents priority inversion by ensuring higher-priority threads remain at the front.
- **thread_tick(void)** (*updated*): Introduced logic to increment `recentCPU`, adjust `load_avg`, and periodically update priorities when `thread_mlfqs` is active. This enables automatic tracking of CPU usage and fair thread scheduling.
- **thread_init(void)** (*updated*): `load_avg` is now initialized to zero at startup, ensuring all threads have default values for `nice` and `recentCPU`. This prepares the system for MLFQS-based scheduling when activated.
- **init_thread(...)** (*updated*): Every newly created thread is now initialized with `nice = 0` and `recentCPU = 0`, ensuring proper setup for MLFQS scheduling.
- **update_recent_cpu_all(void)** (*added*): Updates `recentCPU` for all active threads using the formula:

  $$\text{recent\_cpu} = \frac{2 \times \text{load\_avg}}{2 \times \text{load\_avg} + 1} \times \text{recent\_cpu} + \text{nice}$$

  This ensures a fair distribution of CPU time among threads.
- **thread_set_nice(int)** (*updated*): Allows a thread to modify its `nice` value and updates its priority accordingly. The thread will yield if another thread now has a higher priority.
- **thread_get_nice(void)** (*updated*): Returns the current thread's `nice` value. Provides API support for querying thread niceness.
- **thread_get_load_avg(void)** (*updated*): Returns the system-wide load average multiplied by 100. Allows monitoring of system load trends.
- **thread_get_recent_cpu(void)** (*updated*): Returns the `recentCPU` value of the current thread, scaled by 100. Useful for tracking CPU usage at the thread level.

pintos0/pintos-env/pintos/threads/thread.h

- **struct thread** (*updated*): Introduced two new fields: `int nice`, which represents how willing a thread is to give up CPU time in the range of -20 to 20, and `FPReal recentCPU`, which tracks the amount of CPU time recently used, utilizing fixed-point arithmetic. These additions are necessary for implementing MLFQS-based scheduling.
- **void thread_set_nice(int)** (*updated*): Declared this function to enable modification of the thread's nice value.
- **int_thread_get_recent_cpu(void)** (*updated*): Declared this function to return the recent CPU usage multiplied by 100.
- **int_thread_get_load_avg(void)** (*updated*): Declared this function to return the system load average multiplied by 100.
- **int_thread_get_nice(void)** (*updated*): Declared this function to return the current thread's nice value.



```
pass tests/threads/alarm-single
pass tests/threads/alarm-multiple
pass tests/threads/alarm-simultaneous
pass tests/threads/alarm-priority
pass tests/threads/alarm-zero
pass tests/threads/alarm-negative
pass tests/threads/priority-change
FAIL tests/threads/priority-donate-one
FAIL tests/threads/priority-donate-multiple
FAIL tests/threads/priority-donate-multiple2
FAIL tests/threads/priority-donate-nest
FAIL tests/threads/priority-donate-sema
FAIL tests/threads/priority-donate-lower
pass tests/threads/priority-fifo
pass tests/threads/priority-preempt
FAIL tests/threads/priority-sema
FAIL tests/threads/priority-condvar
FAIL tests/threads/priority-donate-chain
pass tests/threads/mlfqs-load-1
pass tests/threads/mlfqs-load-60
pass tests/threads/mlfqs-load-avg
pass tests/threads/mlfqs-recent-1
pass tests/threads/mlfqs-fair-2
pass tests/threads/mlfqs-fair-20
pass tests/threads/mlfqs-nice-2
pass tests/threads/mlfqs-nice-10
FAIL tests/threads/mlfqs-block
10 of 27 tests failed.
```

Figure 1: Tests