

Analyzing Scheduling

Algorithms, Deeply studying Pintos to implement Priority Scheduling



Analyzing Scheduling Algorithms, Deeply studying Pintos to implement Priority Scheduling

Contents



**Introduction** **2**

**Problem Statement** **2**

**Methods Created & Used** **3**

**Result** **4**

Running Hello World Program 4

C Code Implementation 4

Implementation of Priority in Pintos 5

Passing all the three test Cases 6

**Summary** **8**

**References** **8**

1 

Analyzing Scheduling Algorithms, Deeply studying Pintos to implement Priority Scheduling



Introduction



Pintos are a simple instructional operating system framework for the 80x86 architecture. The software supports kernel threads, loading and running user programs, and a file system, but it implements all of these in a very simple way. Pintos is capable of running on actual x86 hardware, though it is often run on top of an x86 emulator, such as Bochs or QEMU.

Our aim is to incorporate a better way to handle the thread scheduling by implementing priority scheduling algorithm by changing the default First Come First Serve algorithm present in PintOS.

Analyzing three scheduling algorithms -: Shortest job first, First come first serve and Priority scheduling by creating a c code and running sample datasets on them and visualizing the result using R programming.

When a new thread is added to the ready list that has a higher priority than the currently running thread, the current thread should stop and yield the processor for the new higher priority thread that has come in. Also when threads are waiting for a lock, semaphore or a condition variable, the thread waiting with the highest priority should be awakened first.

Problem Statement



1. Testing Hello World Program
2. C Code implementation
3. Implementation of Priority in Pintos
4. Passing all the three test Cases:
   * Pintos Priority -Change
   * Pintos Priority-Preempt
   * Pintos Priority-Sema

2 

Analyzing Scheduling Algorithms, Deeply studying Pintos to implement Priority Scheduling

Methods Created & Used



Pintos main program in init.c file where thread is created and initialized. We have to send its priority in thread\_create () function. The DEFAULT\_PRIORITY of a thread is set to 31. Likewise the MIN\_PRIORITY and MAX\_PRIORITY are set to 0 and 63 respectively. When a new thread is added to the ready list that has a higher priority than the currently running thread, the current thread should stop and yield the processor for the new higher priority thread that has come in. Also when threads are waiting for a lock, semaphore or a condition variable, the thread waiting with the highest priority should be awakened first.

* **void list\_insert\_ordered (struct list \*, struct list\_elem \*, list\_less\_func \*, void \*aux);**

Inserts ELEM in the proper position in LIST, which must be sorted

according to LESS function. Runs in O (n) average case in the number of elements in LIST. The LESS function used is „thread\_sort\_less‟.

* **void list\_insert (struct list\_elem \*before, struct list\_elem \*elem);**

Inserts ELEM just before in the list and hence gives us the sorted list in descending order.

* **static bool thread\_sort\_less (const struct list\_elem \*lhs, const struct list\_elem \*rhs, void \*aux UNUSED);**

Tells list\_insert\_ordered how to insert element in the ready list, we keep the thread who has outstanding priority at the head of the list and the sort is stable. It is the LESS function passed as parameter in list\_insert\_ordered function.

* **void thread\_yield\_head (struct thread \*cur);**

This function is called when a low priority thread needs to be preempted by the higher priority one. The thread which should preempt must be passed as a parameter in this function. After this schedule function is called.

* **static void schedule (void);**

Schedules a new process. The running thread‟s state must have been changed from running to some other state. This function finds another thread to run and switches to it.

3 

Analyzing Scheduling Algorithms, Deeply studying Pintos to implement Priority Scheduling



Result

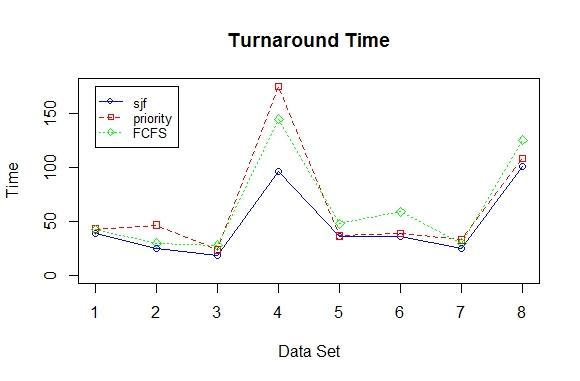


**Testing Hello World Program**



**C Code implementation**

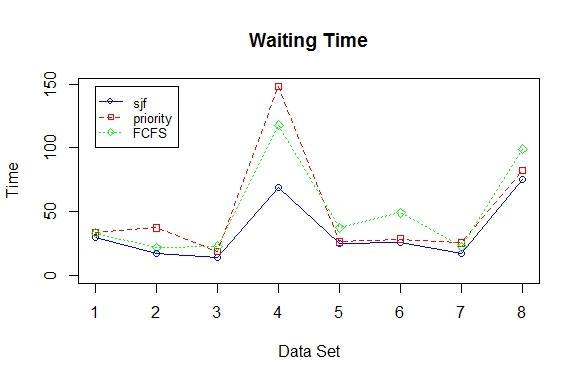
Graph of average turnaround time for the three scheduling algorithms



4 

Analyzing Scheduling Algorithms, Deeply studying Pintos to implement Priority Scheduling

Graph of average waiting time for the three scheduling algorithms



1. We found out that total average waiting time and turnaround time would be minimum in sjf as compared to all scheduling algorithms.
2. In priority scheduling algorithm if high priority is given to the process having low burst time then average turnaround time and waiting time will decrease.
3. Out of the three scheduling algorithms sjf is optimal.

**Implementation of Priority in Pintos**

**Original Implementation:** Original Pintos follows First Come First Serve (FCFS)scheduling algorithms by storing process which are ready to run in „Ready List‟

(Here ready list is a doubly linked list). The newly created thread is inserted at the end of the list. So, when a process needs to be scheduled then the first element is poped from the ready list.

**Our Implementation:** We have implemented priority scheduling algorithm inthis. Whenever a new process is created it is inserted in the ready list according to its priority in descending order. We have implemented “list\_insert\_ordered”. By this implementation the ready list is in sorted order with thread of highest priority as its first element. So, when a threads need to be scheduled the thread with highest priority is poped out.

5 

Analyzing Scheduling Algorithms, Deeply studying Pintos to implement Priority Scheduling

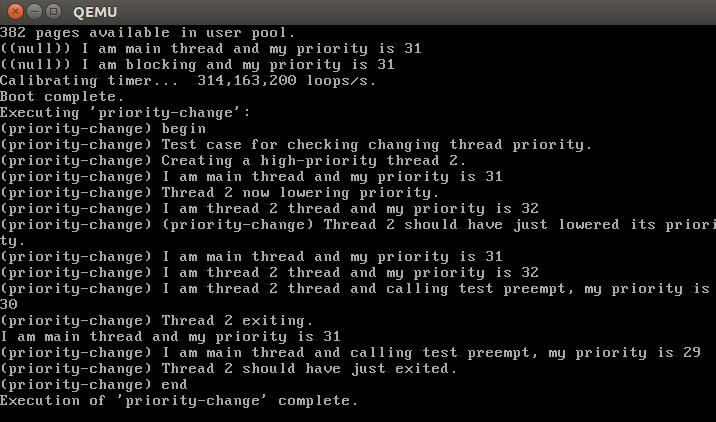


**Passing all the three test Cases**

We have passed all the test cases which are present in Pintos. We had printed console messages in some test cases to clearly understand which thread is running which part of the code and what is its name and priority. As we have implemented priority scheduling with preemption, meaning higher priority should preempt the lower priority thread, or main focus was on these three test cases:

 Pintos Priority –Change

Verifies that lowering a thread's priority so that it is no longer the highest-priority thread in the system causes it to yield immediately. Firstly, main thread with priority (31) creates a new thread with priority (32). The new created thread‟s priority is changed to 30 so that it is no longer the highest priority in the system and causes it to yield immediately. The priority is changed by passing the new priority in thread\_set\_priority function.

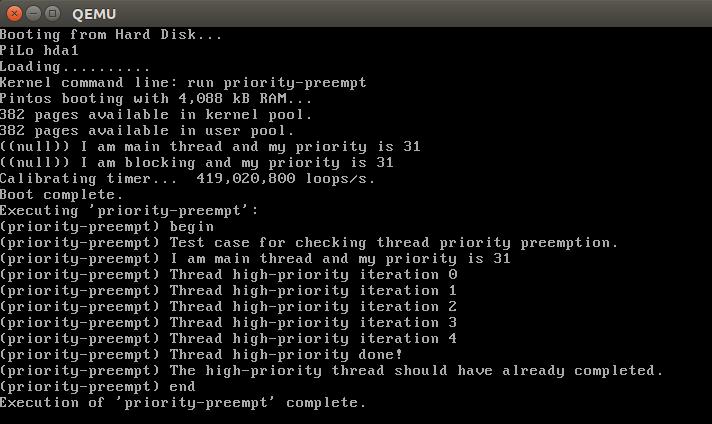


 Pintos Priority-Preempt

Ensures that a high-priority thread really preempts. This test simply shows that if a higher priority thread than main is created it will preempt the main thread and will execute itself and the main thread will continue its execution after this new thread has completed its execution. Similarly, if the thread with priority lower than main thread is created then the main thread is executed until completion after that new thread is executed.

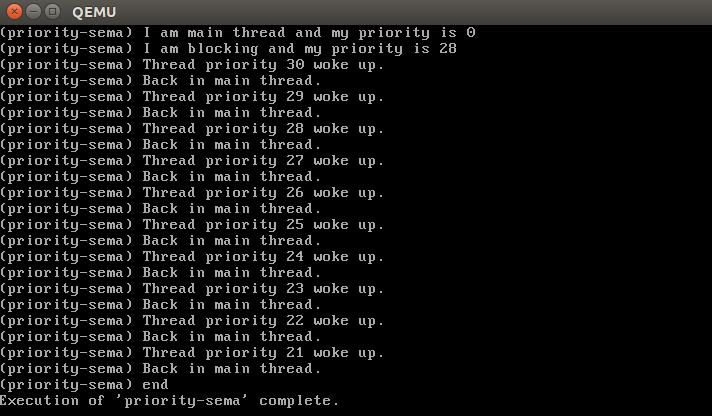
6 

Analyzing Scheduling Algorithms, Deeply studying Pintos to implement Priority Scheduling



 Pintos Priority-Sema

Tests that the highest-priority thread waiting on a semaphore is the first to wake up. The main thread first change its priority to PRIORITY\_MIN. The main thread creates 10 new threads with different priorities. Each created thread will execute the function named priority\_sema\_thread in which sema\_down will block all the threads and insert in waiting list in decreasing order of priority so that whenever sema\_up is called by the main thread, the thread with highest priority is popped out for execution.



7 

Analyzing Scheduling Algorithms, Deeply studying Pintos to implement Priority Scheduling



Summary



Pintos already implements thread creation and thread completion, a simple scheduler to switch between threads, and synchronization primitives.

We as team implemented three Scheduling Algorithms on threads of PintOS and analysed how each one fares amongst each other. We analysed the graphs of Waiting time and Turnaround time of all three algorithms.

We found that Shortest Job Found (SJF) is the optimal algorithm but considering how difficult it is to know the burst time of the processes at boot time, we have implemented Priority Based Scheduling and hence replacing the default First Come First Serve (FCFS) algorithm in PintOS.

In this process we started by creating a hello world program and proceeding to implement Semaphores, Priority Donation and Pre-emption of Processes.

References



1. [http://web.stanford.edu/class/cs140/projects/pintos/pintos\_2.html#SEC2](http://web.stanford.edu/class/cs140/projects/pintos/pintos_2.html#SEC26)6
2. [https://pintosiiith.wordpress.com/2012/10/15/assignment-priority-scheduling](https://pintosiiith.wordpress.com/2012/10/15/assignment-priority-scheduling-pintos/)- [pintos](https://pintosiiith.wordpress.com/2012/10/15/assignment-priority-scheduling-pintos/)/
3. [http://www.scs.stanford.edu/15au-cs140/pintos/pintos\_3.html#SEC3](http://www.scs.stanford.edu/15au-cs140/pintos/pintos_3.html#SEC32)2
4. [https://cs162.eecs.berkeley.edu/static/projects/project1.pd](https://cs162.eecs.berkeley.edu/static/projects/project1.pdf)f
5. [http://infoman.teikav.edu.gr/~stpapad/OS\_8th\_Edition.pd](http://infoman.teikav.edu.gr/~stpapad/OS_8th_Edition.pdf)f

8 