

Homework 8

Solution Key

Problem 8.1

Would a process ever want to acquire both a read lock and a write lock on the same resource? Why or why not? How would it/could it go about doing so safely? Does your answer change if there are multiple threads or processes that would want to be doing this?

Almost certainly not, because holding a write lock grants all the privileges of a read lock as well. Furthermore, holding the same lock as a read lock and as a write lock would be impossible, as once it acquired one, it would be unable to acquire the other (since read locks and write locks mutually exclude each other) and therefore would deadlock. In order to hold both kinds of lock on a single resource, you would have to create them as separate locks and then create a protocol for what order to acquire and release them in; that protocol would essentially end up reimplementing the readers/writers solution anyway.

The answer doesn't particularly change if more than one process wants to hold both kinds of lock, because it's already useless and complicated enough even if only one of the processes wants both. (It just makes race conditions even *more* likely and harder to debug.)

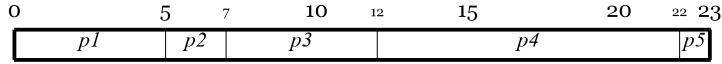
Problem 8.2

Consider the following use case...

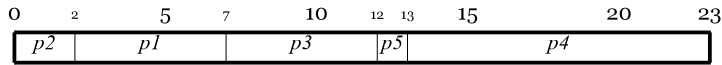
<i>Process</i>	<i>In queue at time</i>	<i>Length</i>	<i>Priority</i>
<i>p₁</i>	<i>0</i>	<i>5</i>	<i>5</i>
<i>p₂</i>	<i>0</i>	<i>2</i>	<i>8</i>
<i>p₃</i>	<i>3</i>	<i>5</i>	<i>3</i>
<i>p₄</i>	<i>6</i>	<i>10</i>	<i>4</i>
<i>p₅</i>	<i>11</i>	<i>1</i>	<i>1</i>

Draw usage charts for the different CPU scheduling algorithms:

a. *FCFS*



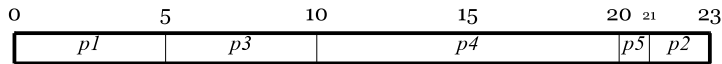
b. *SJF*



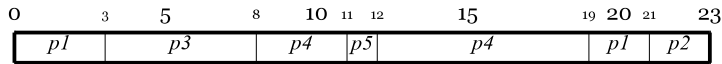
c. *SJF-preemptive (aka shortest-remaining-time-next)*

A bit of a trick question; within this use case, this policy produces precisely the same results as non-preemptive SJF!

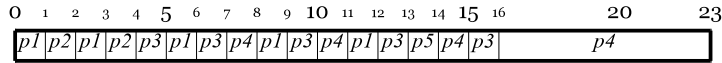
d. *non-preemptive priority scheduling*



e. *preemptive priority scheduling*

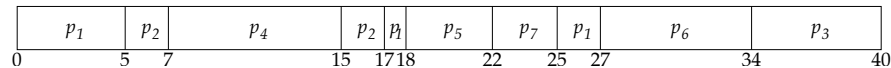


f. *round robin*



Problem 8.3

Consider the following processor-usage chart.



a. *What CPU scheduling policy(ies) might give rise to this chart? How do you know?*

Preemptive priority scheduling was the one I had in mind. It must be preemptive since some jobs are interrupted and later resume. It can't be FCFS or a derivative because the chunks occur out of order. It can't be round-robin because the chunks are of variable size (and out of order). Given that e.g. p_4 can preempt p_2 and p_1 , both of which have less time left to run, indicates it's not an oracular SJF model; it could conceivably arise from an estimated SJF scheduler that was preemptive, though.

- b. Calculate the average turnaround time for the processes according to this CPU allocation.

$$\frac{27 + 12 + 34 + 8 + 4 + 15 + 5}{7} = \frac{105}{7} = 15$$