COMP 530 Introduction to Operating Systems

Fall 2017  
Kevin Jeffay

Worksheet 10, October 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Your Name: |  | You worked with: |  | +1/blank/-1: |  |
|  | Aaron Zhang |  | Victor Cui |  | +1 |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. In the lecture on Monitors I showed how priority inversions were possible when using a Hoare monitor given the semantics of *signal* and *wait* in Hoare monitors. The root cause of the problem is that condition synchronization in a Hoare monitor requires a signaler to “guarantee” that an awaited condition will be true when a waiting process resumes execution and this in turn effectively requires a context switch from a signaling process to a waiting process. Mesa monitors eliminate priority inversions by adopting a different semantic model wherein the “signal” (the *notify* operation) is a “hint” and not a guarantee that the awaited condition is true.

Explain exactly how (1) priority inversions were possible in the original formulation of a Hoare monitor, and (2) explain how the different semantic model of synchronization in Mesa monitors eliminates the priority inversions that can occur in Hoare monitors. Male reference to the code for the implementation of monitors as appropriate.

2. It turns out that it is possible to eliminate priority inversions in Hoare monitors by integrating the condition synchronization and mutual exclusion mechanisms in monitors with the operating system scheduler. Describe (in prose and code if you can) how a priority scheduler could be adapted to dynamically adjust the priority of processes to ensure priority inversions do not occur when processes perform condition synchronization inside a Hoare monitor.