Kohl Johnson

CST-321

12-22-2024

Deadlock Avoidance

# Scenario

My scenario was to have the shared resource be the text file use for printing the activity log. Using the semaphore, each process would take turns writing what they are doing an so forth. Once a process takes control of the semaphore, a timer would be started which acts as a process hang checker. If said timer detects a hung process, it will prompt the user to decide whether that process should continue or be terminated. However, with using this approach (more specifically the shared file) I encountered segmentation faults and/or the file would never be used.

# Flowchart

A diagram of a process

Description automatically generated

# Assessment

Overall, I think the timer function is a decent deadlock avoidance technique, but a terrible prevention method since it does not stop them from occurring. One drawback is that a timer involves quite a bit of busy waiting and the possibility of interrupting a process that is working and not “hung”. Another level that I think it beneficial is having the user decide when to terminate a process or let it continue. While in my program, the prompt occurs after the timer is triggered, but having this user input as a “keypress” event would also be beneficial.

**Manually Killed Process**A screenshot of a computer

Description automatically generated

**Full Process Part 1**

A screenshot of a computer

Description automatically generated

**Full Process Part B**

**A screenshot of a computer

Description automatically generated**