Kohl Johnson

Deadlock Avoidance

06-23-2024

## Scenario

My scenario was to have the shared resource be the text file used for printing the activity log. Using the semaphore, each process would take turns writing what they are doing and so forth. Once a process takes control of the semaphore, a timer would be started which acts as are process hang checker. If that timer detects a hang, it will prompt the user to decide whether to let that process continue, or to kill it. However, with using this approach (more specially the shared file) I encountered segmentation faults and/or the file would simply not be used. (which I just now discovered was due to fopen deleting the contents of the file prior to the new writes).

## Flowchart

A diagram of a process

Description automatically generated

## Assessment

Overall, I think the timer function is a great deadlock avoidance technique, but not necessarily a prevention since it doesn’t stop them from occurring. The drawback to using a timer is that it included quite a bit of busy waiting. 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 has triggered, but having this user input as a “keypress” event would be beneficial.

## Screencast

Link: <https://www.loom.com/share/89f908e3f0054306a7cac43af1f2f8a9?sid=e2abe6e4-4de2-4247-a23a-b8cdef231a1c>