State University of New York at New Paltz

Student Name: Andrew McDonald

OS Lab section: 01 (Friday,2:00-4:40)

Semester: Fall 2021

REPORT for LAB # 08

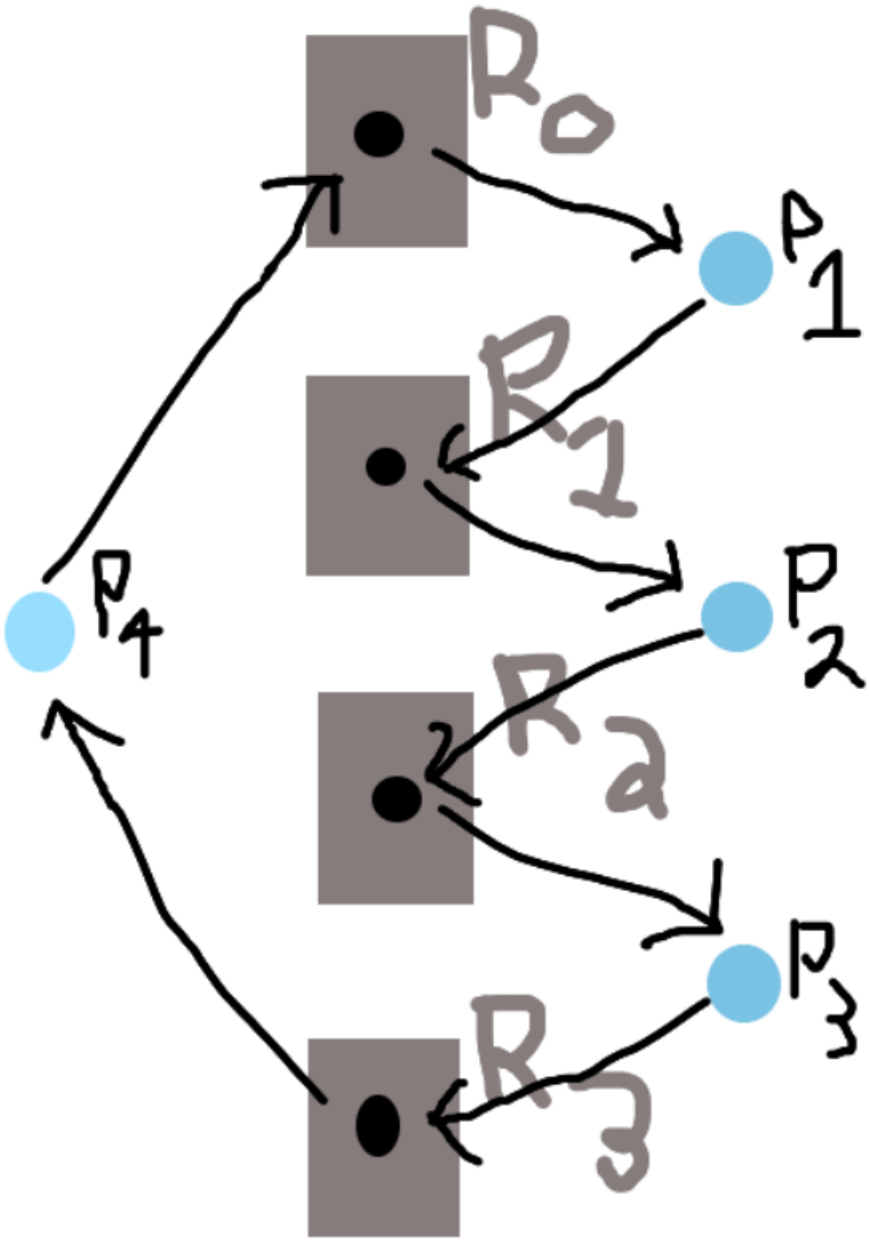
(Investigating Deadlocks)

"Operating Systems" Fall 2021

(Professor Hanh Pham)

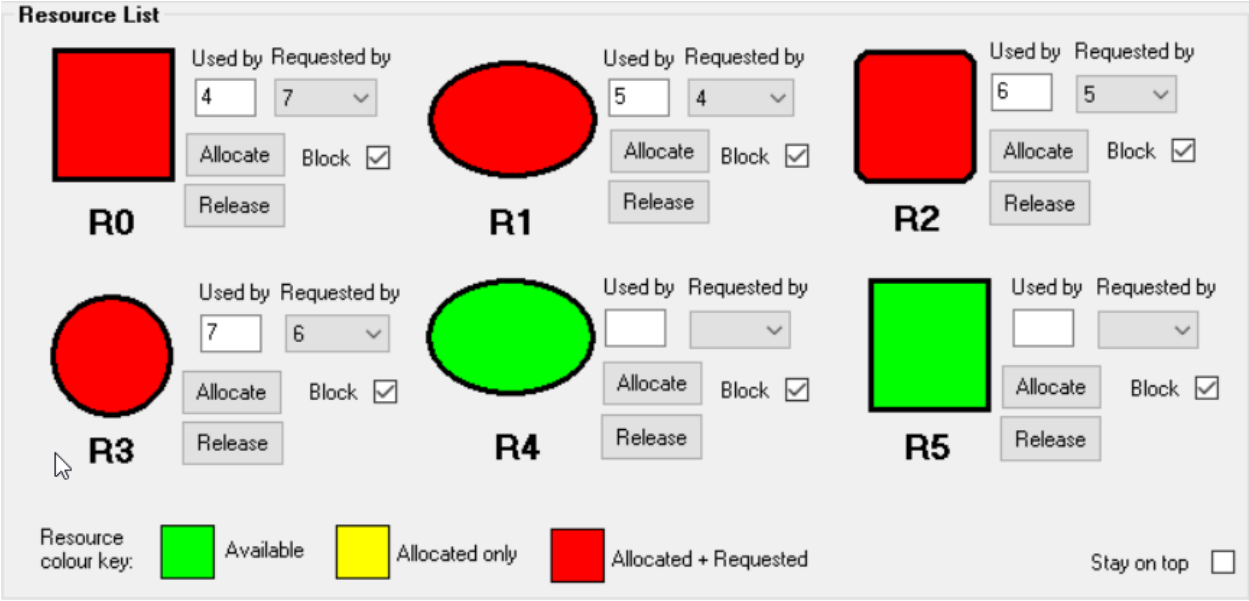
Lab Instructor: BINDHUPRIYA THRIPURANENI

1. **Four processes are running. They are called P1 to P4.There are also four resources available (only one instance of each). They are named R0 to R3. At some point of their existence each process allocates a different resource for use and holds it for itself forever. Later each of the processes request another one of the four resources. Draw the resource allocation graph for a four process deadlock condition. Do not continue until you do this and get it verified by the tutor.**



100%

2. **In the compiler window, enter the following source code in the compiler source editor area (under PROGRAM SOURCE frame title).**



Yes I got a deadlock and the deadlock situation as I drew in question 1

3. **Is the deadlock situation resolved? Explain briefly why this helped resolve the deadlock. Has this managed to resolve the deadlock? Explain briefly why this helped resolve the deadlock.**

   After using the release button the chain of deadlock was released and all of the subsequent processes were executed. This happens because 1 process being removed allows the one waiting for the resource it is utilizing to utilize the resource and so on for all the other processes. After removing one of the processes the next process was able to execute a bit more and then the remaining three got into another deadlock. That is because they are all focused on the same three resources.

4. **Try to re-create the same deadlock condition as before. Have you been successful? What happened?**

   I was unable to reproduce the same results with selecting the “Disable hold and wait” checkbox and the processes that were supposed to be the start of the deadlock were given priority for execution and continued as normal. When I selected “Disable circular wait” I got the results of the deadlock. This did not fix the problem and I am not really sure what it does.

5. **Looking at your resource allocation graph can you see how this ordering can prevent a deadlock? Try to re-create the same deadlock condition as before. Have you been successful? What happened? Click on the SHOW DEADLOCKED PROCESSES… button and observe the displayed information in the text window for potential clues. What happened? Comment.**

   I can see how the total ordering method works in preventing deadlock as it disallows a chain of processes being Allocated and Requested. After attempting to recreate the same deadlock condition I was successful as it only allowed one process to go to the waiting queue and cycled through the other three processes allowing the whole set of processes to eventually execute. There were no deadlocks throughout the execution so looking at the Show Deadlock Processes section is not useful.