Indian Institute of Technology Tirupati

Renigunta Road, Tirupati 517506, AP

October 17, 2019

CS3500 Operating Systems Lab - Deadlock Detection

1. (25 marks) rollnumber_lab7a.c. Assuming all other 3 conditions for deadlock to occur are met, write a program to detect fourth requirement of the existence of cycle. Use following two constants namely, NUMRESOURCE and NUMPROCESS. Create an RAgraph matrix consisting of resources and processes namely, ragraph[][], that shows both resource requests, and resource allocations. Create and use a randomRequest() function, that randomly assigns one or zero resource to each of the processes. Create a randomAllocation() function, that randomly allocates each resource to exactly zero or one process. Create a detectDeadlock() that first simplifies ragraph to a simplified equivalent 'work-for' graph that has only processes as nodes. Use this 'work-for' graph to detect the existence of deadlock, and print the result.

2. (25 marks) rollnumber_lab7b.c. Implement Banker's safety algorithm (first part), to verify the safeness of a given state. Take 'n' processes, and 'm' resources, and produce random maximum integer counts, for each of the processes to their respective resource-types. Allocate to each process, a random number (instance count) generated between 0 and 'respective allowed maximum count' (inclusive). Generate a suitable 'available' random pool of resource instances. Use this data, to verify the safeness of a randomly generated state. For output do the following: In steps (from left to right in the sequence), show the state and available resource pool count, after the considered process in sequence is assumed to have completed. In the end of the output, mention if safe state exists or not.

Note:

- In the main folder of rollnumber_lab7, attach all the .c, .txt files (if any), and screenshot images (if any).
- **Important**: Use your last two digits of rollnumber as the random number seed