Jiyong Kwag

Lab4: Deadlock Detection and Recovery

Deadlock is a problem that process is waiting for the resource that is held by other process which is also waiting for the resource from other waiting process. Solving the deadlock is important for the system because if system stays in deadlock cycle, system cannot proceed the process any more. Thus, I used to deadlock detection and recovery to solve the problem. Deadlock detection algorithm find the cycle among the process and if there is deadlock, deadlock recovery algorithm resolve the cycle.

There are a number of solutions for the deadlock: deadlock avoidance, deadlock prevention, deadlock ignorance, and deadlock detection and recovery. Generally, most OS uses the deadlock ignorance. Deadlcok detection and recovery is the least useful way. In deadlock detection and recovery, system alwasy checks for the deadlock and runs the recovery. Thus, calling algorithms too often cause the overhead. However, it is useful because system does not have to de-allocate all the resources. Detection is similar to banker’s algorithm in which loops through the all process and resources to check the cycle. Then, recovery algorithm will search the victim process to de-allocate its process to resolve the cycle. In lab4, one should run the program with the makefile and run the run file. Run file will show allocated, request, and available resources for the system. Then, system shows the current deadlock states of each process by -1,1,0. 1 represents finished process, -1 represents cannot be finished process, and 0 for unfinished process. If system has -1, then it represents there is deadlock. Then, recovery algorithm runs and shows the victim process and result until only all processes have 1 state.

In the Operating System concept, Silberschatz explained the method to detect the deadlock. At first, we assume that all processes arrive at same time and they are in non-preemptive. In the system, there are three main vectors: available, allocation, request. Each vector is assigned with assignResource method in line 350. Then system runs the recovery method in line 306. In the recovery method, algorithm searchs for the deadlock using detection method in line 157. In the detection method, finish vector is initialized with 0. System loops through finish vector comparing the allocated, request, and available resources until all of processes are assigned with -1 or 1. Detection method will return true if there is deadlock and false if there is no deadlock. In recovery method, there are two vectors are implemented: need and yied. Need vector represents, how many resource that process need: request resource – available resource. Yield vector represents how many other process are requiring the process’ allocated resource. In line 319, if detection method returns true, recovery method will use need method with initializeNeed method in line 284 to find how many resource that each process,. Then, use yieldProcess method in line 217 to initialize yield vector. As a result, recovery method find the most seeked process using yield vector and de-allocate the process’ resource. Then, mehod runs until detection method return false.

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

Figure 1.2: Recovery

Figure 1.1: Detection

Figure 1.1 represents the result after running the program. User can see the all assigned resources in to the vectors and states of each process by -1 and 1. At the end of figure1.1, it shows victim process and figure 1.2 shows the results from de-allocating the process 0’s resources. As a result, all the processes have 1 state and deadlock is solved. There was no big difference between expected outcome and actual outcome. In the lab4, main problem was assigning resource for simulation. For assigning the resource, assigning too many resources to request cause the deadlock everytime while assigning too less is not causing the deadlock. I tried to find real-time simulation for assigning the resource, however, it simulate possibility of assigning resource for request to find suitable amount of resources that balance deadlock.

In conclusion, I was implementing the deadlock prevention and recovery, and assigning the resources. The assigning the resources is not real-time assignment but random assignment according to the maximum number of resouces in the program. Thus, it was not clear that my implementation will work toward every real-time situation. In future, it will be better with the simulation that has real-time OS resouces assignments.

Bibliography

“Recovery from Distributed Deadlock.” CS 551: Distributed Deadlock, recovery, 2001. https://www.cs.colostate.edu/~cs551/CourseNotes/Deadlock/DDRecovery.html.

Silberschatz, Abraham, Greg Gagne, and Peter B. Galvin. *Operating System Concepts, 8th Edition*. John Wiley & Sons, 2008.