Fredrick Onduso Ondieki

CST – 221

John Zupan

CST-221 Deadlock Avoidance

March 24, 2019

Git hub Link https://github.com/FREDDYSMALLZ/Operating-Systems-Concepts-CST-221.git

Introduction

The main purpose of deadlock avoidance is to merely be meant to avoid it rather than preventing it. The basic idea is to allocate resources only if the resulting global state is a safe state.

In other words, unsafe states are avoided, meaning that deadlock is avoided as well.

One famous algorithm for deadlock avoidance in the uniprocessor case is the [**Banker's Algorithm**](http://www.cs.colostate.edu/~cs551/CourseNotes/Bankers.html) **(**This algorithm allows mutual exclusion, wait and hold, and no redemption. particularly, it prevents circular wait).

Similar algorithms have been attempted for the distributed case.

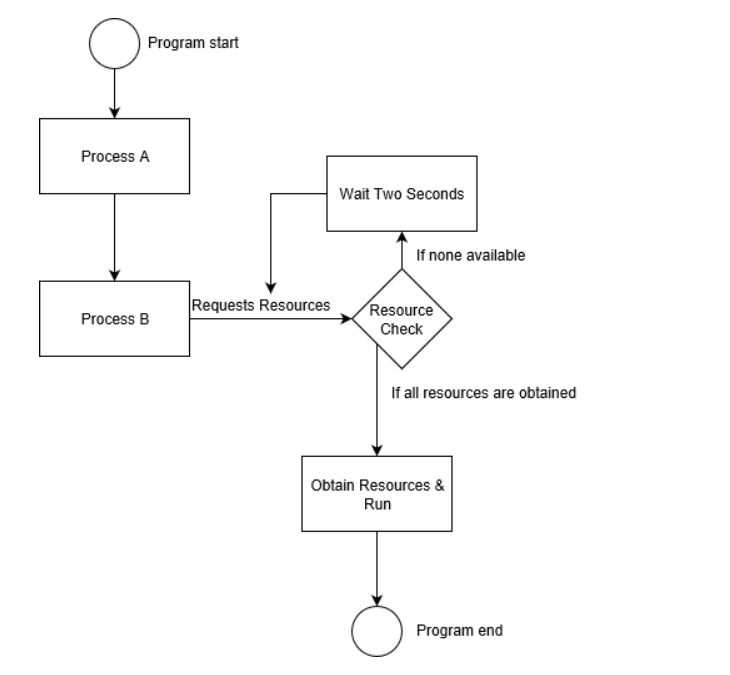
1. When a process requests a resource, even if it *is* available, it is not immediately allocated to the process. Instead the systems ***assume*** (pretends) it is so allocated.
2. With this assumption, along with advance knowledge of all the resources needed for all the processes, the system performs some analysis to decide whether granting the request is safe.
3. If the request *is* safe, the resource is granted to the requesting process.   
   Otherwise, the resource is ***not*** given to the process **at this time**.

This type of algorithm requires each site to have access to a global state *(requiring too much storage and communication)*.

Also, the checking of the involved data structures *(e.g. the Banker's Algorithm tables)* must be done in a mutually exclusive fashion. With many sites and resources, it just takes too long.

**Scenario and Approach to Implementation of the program**

**Flow chart**

****

According to the flowchart above, process A will get all the resources it needs and then start. After some time, can be seconds in our case, the second process B gathers its resources by sending a request message and then if the resources are available it beings to run. However, in our case since resources are still being utilized by Process A, there are no resources for Process B to use. Thus, in this case it causes an incidence of deadlock. To avoid this case, timers are set in place.

In our case, instead of causing a deadlock, the timer in place sends a signal to Process B to wait and it starts to count immediately and start the request for resources again later after the time limit is exceeded. During this wait time, Process A will be done with the resources and releases them making them readily available for use by Process B. On the other hand, after the time limit is exceeded, Process B wakes up from and starts the request for the process again wanting to use the resources. Since the resources are available, the second process completes successfully.

**Analysis**

Timer in Operating Systems has full control over the CPU and it is the primary duty of operating system to prevent user programs from getting stuck into an infinite loop or not calling to system services and never returning control back to the operating system. To accomplish all those services an operating system uses Timers. The main task of a timer is to interrupt the CPU after a specific period. This specific period is set by operating system and its value may vary from 1 ms to 1 second. If the operating system is interested in using variable timer then it is implemented with the help of fixed rate clock or counter and in this case, operating system sets the counter.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of text

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of text

Description automatically generatedA screenshot of a cell phone

Description automatically generated

References

Role of Timers in Operating Systems. (2008, October 19). Retrieved March 24, 2019, from http://operatingsystemconcepts4u.blogspot.com/2008/10/role-of-timer-in-operating-system.html

Dijkstra. (n.d.). CS 551: Banker's Algorithm. Retrieved March 24, 2019, from http://www.cs.colostate.edu/~cs551/CourseNotes/Bankers.html

Rehman, J. (n.d.). Deadlock avoidance in operating system. Retrieved March 24, 2019, from http://www.itrelease.com/2017/11/deadlock-avoidance-operating-system/

Program for Banker's Algorithm | Set 1 (Safety Algorithm). (2019, January 28). Retrieved March 24, 2019, from https://www.geeksforgeeks.org/program-bankers-algorithm-set-1-safety- algorithm/

Deadlock Prevention and Avoidance. (2018, September 07). Retrieved March 24, 2019, from https://www.geeksforgeeks.org/deadlock-prevention/