

**Quick Links
for Operating
Systems**[Recent Articles](#)[MCQ / Quizzes](#)[Practice
Problems](#)[Last Minute
Notes \(LMNs\)](#)**Basics**[What happens
when we turn on
computer?](#)[Explore More...](#)**Processes &
Threads**[Process -
Introduction](#)[Thread](#)[User Level thread
vs. Kernel Level
thread](#)[Zombie
Processes and
their Prevention](#)[Maximum
number of
Zombie process a
system can
handle](#)[Maximum
number of
threads that can
be created within
a process in C](#)

What exactly Spooling is all about?
Multi threading models
Explore More...
Process Synchronization
Introduction & Critical Section
Inter Process Communication
Mutex vs Semaphore & Monitors
Peterson's Algorithm for Mutual Exclusion Set 1 & Set 2
Readers-Writers Problem
Priority Inversion : What the heck !
Banker's Algorithm & Program
Priority Inversion vs. Priority Inheritance
Explore More...
CPU Scheduling
Process Management - Introduction
CPU Scheduling & Process Scheduler
FCFS Scheduling Set 1 & Set 2
SJF scheduling
Round Robin scheduling
Priority Scheduling



Starvation and Aging
Explore More...
Deadlocks
Introduction
Detection And Recovery
Prevention And Avoidance
Explore More...
Memory Management
Partition Allocation Method
Virtual Memory
Paging
Segmentation
Page Replacement Algorithms
Static and Dynamic Libraries
Working with Shared Libraries Set 1 & Set 2
Explore More...
File & Disk Management
File System
File Allocation Methods
Disk Scheduling Algorithms
Explore More...
Linux
Linux File Hierarchy Structure



Initializing and
Cache
Mechanism in
Linux Kernel

Some useful
Linux Hacks

Explore More...

Operating System | Banker's Algorithm

3.3

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue.

Following **Data structures** are used to implement the Banker's Algorithm:

Let '**n**' be the number of processes in the system and '**m**' be the number of resources types.

Available :

- It is a 1-d array of size '**m**' indicating the number of available resources of each type.
- $\text{Available}[j] = k$ means there are '**k**' instances of resource type **R_j**

Max :

- It is a 2-d array of size '**n*m**' that defines the maximum demand of each process in a system.
- $\text{Max}[i, j] = k$ means process **P_i** may request at most '**k**' instances of resource type **R_j**.

Allocation :

- It is a 2-d array of size '**n*m**' that defines the number of resources of each type currently allocated to each process.
- $\text{Allocation}[i, j] = k$ means process **P_i** is currently allocated '**k**' instances of resource type **R_j**

Need :

- It is a 2-d array of size '**n*m**' that indicates the remaining resource need of each process.
- $\text{Need}[i, j] = k$ means process **P_i** currently allocated '**k**' instances of resource type **R_j**
- $\text{Need}[i, j] = \text{Max}[i, j] - \text{Allocation}[i, j]$

Allocation_i specifies the resources currently allocated to process **P_i** and Need_i specifies the additional resources that process **P_i** may still request to complete its task.

Banker's algorithm consist of Safety algorithm and Resource request algorithm

Safety Algorithm



The algorithm for finding out whether or not a system is in a safe state can be described as follows:

- 1) Let Work and Finish be vectors of length ' m ' and ' n ' respectively.

Initialize: Work = Available

Finish [i] = false; for $i=1, 2, \dots, n$

- 2) Find an i such that both

a) Finish [i] = false

b) $Need_i \leq work$

If no such i exists goto step (4)

- 3) Work = Work + Allocation _{i}

Finish [i] = true

goto step (2)

- 4) If Finish [i] = true for all i ,

then the system is in safe state.

Resource-Request Algorithm

Let Request _{i} be the request array for process P_i . Request _{i} [j] = k means process P_i wants k instances of resource type R_j . When a request for resources is made by process P_i , the following actions are taken:

- 1) If Request _{i} \leq Need _{i}

Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.

- 2) If Request _{i} \leq Available

Goto step (3); otherwise, P_i must wait, since the resources are not available.

- 3) Have the system pretend to have allocated the requested resources to process P_i by modifying the state as follows:

Available = Available – Request _{i}

Allocation _{i} = Allocation _{i} + Request _{i}

Need _{i} = Need _{i} - Request _{i}

Example:

Considering a system with five processes P_0 through P_4 and three resources types A, B, C. Resource type A has 10 instances, B has 5 instances and type C has 7 instances. Suppose at time t_0 following snapshot of the system has been taken:



Process	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P ₀	0	1	0	7	5	3	3	3	2
P ₁	2	0	0	3	2	2			
P ₂	3	0	2	9	0	2			
P ₃	2	1	1	2	2	2			
P ₄	0	0	2	4	3	3			

Question1. What will be the content of the Need matrix?

$\text{Need}[i, j] = \text{Max}[i, j] - \text{Allocation}[i, j]$

So, the content of Need Matrix is:

Process	Need		
	A	B	C
P ₀	7	4	3
P ₁	1	2	2
P ₂	6	0	0
P ₃	0	1	1
P ₄	4	3	1

Question2. Is the system in safe state? If Yes, then what is the safe sequence?

Applying the Safety algorithm on the given system,



Step 1 of Safety Algo
 $m=3, n=5$
 Work = Available
 Work =

3	3	2
0	1	2

 Finish =

false	false	false	false	false
-------	-------	-------	-------	-------

Step 2
 For $i=0$
 $Need_0 = 7, 4, 3$
 $Finish[0]$ is false and $Need_0 > Work$
 So P_0 must wait
 But $Need \leq Work$

Step 2
 For $i=1$
 $Need_1 = 1, 2, 2$
 $Finish[1]$ is false and $Need_1 < Work$
 So P_1 must be kept in safe sequence

Step 3
 Work = Work + Allocation₁
 Work =

5	3	2
0	1	2

 Finish =

false	true	false	false	false
-------	------	-------	-------	-------

Step 2
 For $i=2$
 $Need_2 = 6, 0, 0$
 $Finish[2]$ is false and $Need_2 > Work$
 So P_2 must wait

Step 2
 For $i=3$
 $Need_3 = 0, 1, 1$
 $Finish[3]$ is false and $Need_3 < Work$
 So P_3 must be kept in safe sequence

Step 3
 Work = Work + Allocation₃
 Work =

7	4	3
0	1	2

 Finish =

false	true	false	true	false
-------	------	-------	------	-------

Step 2
 For $i=4$
 $Need_4 = 4, 3, 1$
 $Finish[4]$ is false and $Need_4 < Work$
 So P_4 must be kept in safe sequence

Step 3
 Work = Work + Allocation₄
 Work =

7	4	5
0	1	2

 Finish =

false	true	false	true	true
-------	------	-------	------	------

Step 3
 Work = Work + Allocation₀
 Work =

7	5	5
0	1	2

 Finish =

true	true	false	true	true
------	------	-------	------	------

Step 2
 For $i=2$
 $Need_2 = 6, 0, 0$
 $Finish[2]$ is false and $Need_2 < Work$
 So P_2 must be kept in safe sequence

Step 3
 Work = Work + Allocation₂
 Work =

10	5	7
0	1	2

 Finish =

true	true	true	true	true
------	------	------	------	------

Step 4
 $Finish[i] = true$ for $0 \leq i \leq n$
 Hence the system is in Safe state

The safe sequence is P_1, P_3, P_4, P_0, P_2

Question3. What will happen if process P_1 requests one additional instance of resource type A and two instances of resource type C?

Request₁ =

1	0	2
---	---	---

To decide whether the request is granted we use Resource Request algorithm

Step 1
 $Request_1 < Need_1$

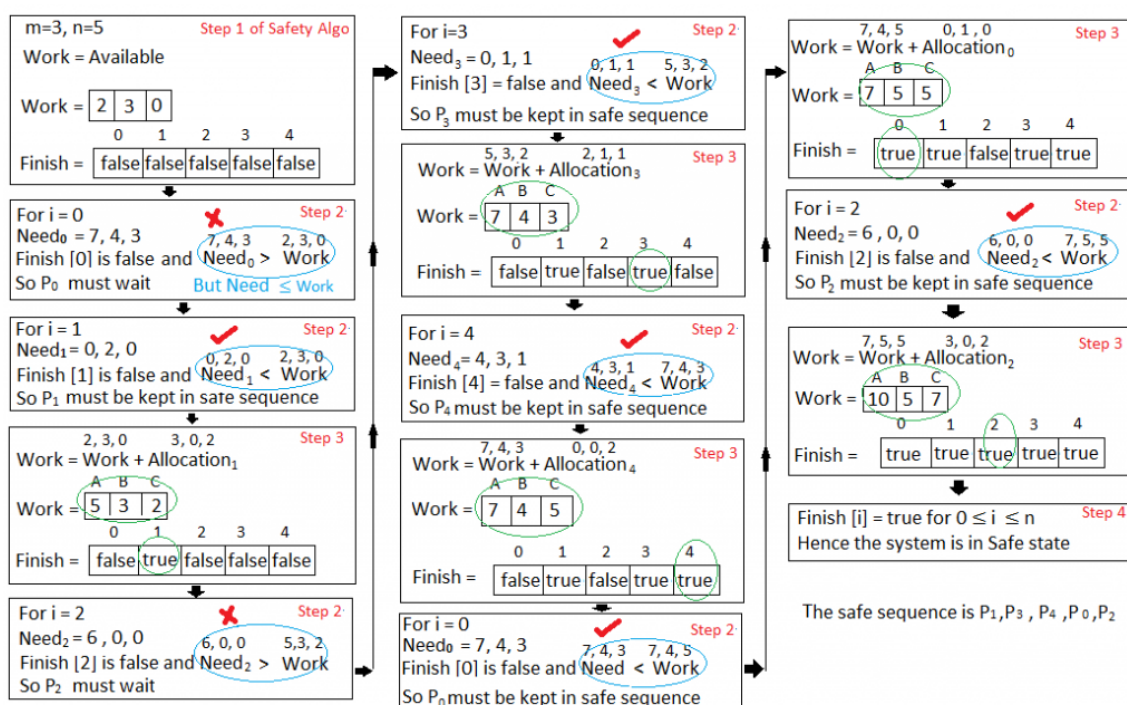
Step 2
 $Request_1 < Available$

Step 3
 Available = Available - Request₁
 Allocation₁ = Allocation₁ + Request₁
 Need₁ = Need₁ - Request₁

Process	Allocation	Need	Available
	A B C	A B C	A B C
P ₀	0 1 0	7 4 3	2 3 0
P ₁	3 0 2	0 2 0	
P ₂	3 0 2	6 0 0	
P ₃	2 1 1	0 1 1	
P ₄	0 0 2	4 3 1	

We must determine whether this new system state is safe. To do so, we again execute Safety algorithm on the above data structures.





Hence the new system state is safe, so we can immediately grant the request for process P_1 .

GATE question:

<http://quiz.geeksforgeeks.org/gate-gate-cs-2014-set-1-question-41/>

Reference:

Operating System Concepts 8th Edition by Abraham Silberschatz, Peter B. Galvin, Greg Gagne

This article has been contributed by [Vikash Kumar](#). Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above



File System Filter Driver

Controls file access and provides file access audit logging without impacting applications



ExpressVPN

VPN That Works in China

Access Anything Blocked Online. Get Fast, Secure, Free Internet.

GATE CS Corner Company Wise Coding Practice



Operating Systems

Process Synchronization

[Login to Improve this Article](#)

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

[Program for Banker's Algorithm | Set 1 \(Safety Algorithm\)](#)

[Priority Inversion : What the heck !](#)

[Operating System | Process Management | Deadlock Introduction](#)

[What's difference between Priority Inversion and Priority Inheritance ?](#)

[Readers-Writers Problem | Set 1 \(Introduction and Readers Preference Solution\)](#)

[Operating System | Introduction of Operating System – Set 1](#)

[Operating System | Process-based and Thread-based Multitasking](#)

[Operating System | Difference between multitasking, multithreading and multiprocessing](#)

[Operating System | Reader-Writers solution using Monitors](#)

[IPC through shared memory](#)

(Login to Rate)

3.3

Average Difficulty : **3.3/5.0**
Based on **21** vote(s)



Add to TODO List



Mark as DONE

Basic

Easy

Medium

Hard

Expert

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

[Load Comments](#)[Share this post!](#)

@geeksforgeeks, Some rights reserved

[Contact Us!](#)

[About Us!](#)

[Careers!](#)

[Privacy Policy](#)

