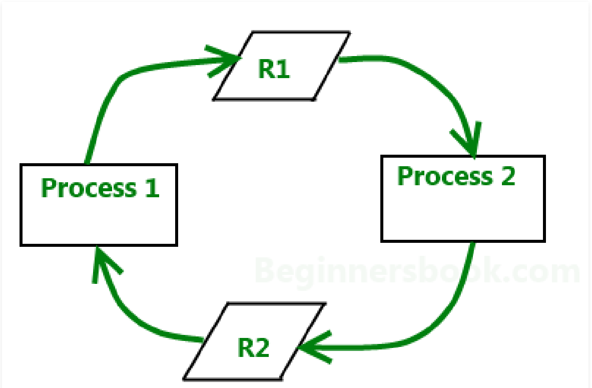
Functions of Operating System
Introduction to memory and memory units
Round Robin Scheduling with different arrival times
Preemptive and Non-Preemptive Scheduling
Memory Hierarchy Design and its Characteristics
Operating System | Kernel I/O Subsystem (I/O System)
Difference between Spooling and Buffering
Computer Organization | RAM vs ROM
Operating System | Boot Block
Fixed (or static) Partitioning in Operating System
Program for SSTF disk scheduling algorithm
Logical vs Physical Address in Operating System
Program for Deadlock free condition in Operating System

Deadlock, Starvation, and Livelock

Prerequisite – Deadlock and Starvation

Livelock occurs when two or more processes continually repeat the same interaction in response to changes in the other processes without doing any useful work. These processes are not in the waiting state, and they are running concurrently. This is different from a deadlock because in a deadlock all processes are in the waiting state.



Process P1 holds resource R2 and requires R1 while Process P2 holds resource R1 and requires R2.

Example:

Imagine a pair of processes using two resources, as shown:

```
void process_A(void)
{
    enter_reg(& resource_1);
    enter_reg(& resource_2);
    use_both_resources();
    leave_reg(& resource_2);
    leave_reg(& resource_1);
}
void process_B(void)
{
    enter_reg(& resource_1);
    enter_reg(& resource_2);
    use_both_resources();
    leave_reg(& resource_2);
    leave_reg(& resource_1);
```

AdChoices

Most popular in Operating Systems

Implementing Directory Management using Shell Script
Program for buddy memory allocation scheme in Operating Systems | Set 2 (Deallocation)
Operating System | The Linux Kernel

Paytm Interview experience for FTE (On-Campus)

Variable (or dynamic) Partitioning in Operating System

Message based Communication in IPC (inter process communication)

Non-Contiguous Allocation in Operating System

Techniques used in centralized approach of deadlock detection in distributed systems

Operating System | File Access Methods

Operating System | Privileged and Non-Privileged Instructions

CPU Scheduling | Longest Remaining Time First (LRTF) Program

Sandvine Interview Experience for FTE 2018

Get/Set process resource limits in C

Operating System | Memory Interleaving

CPU Scheduling | Longest Remaining Time First (LRTF) algorithm

Operating System | Structures of Directory

```
    }
```

Each of the two processes needs the two resources and they use the polling primitive enter_reg to try to acquire the locks necessary for them. In case the attempt fails, the process just tries again.

If process A runs first and acquires resource 1 and then process B runs and acquires resource 2, no matter which one runs next, it will make no further progress, but neither of the two processes blocks. What actually happens is that it uses up its CPU quantum over and over again without any progress being made but also without any sort of blocking. Thus this situation is not that of a deadlock(as no process is being blocked) but we have something functionally equivalent to deadlock: LIVELOCK.

What leads to Livelocks?
Occurrence of livelocks can occur in the most surprising of ways. The total number of allowed processes in some systems, is determined by the number of entries in the process table. Thus process table slots can be referred to as Finite Resources. If a fork fails because of the table being full, waiting a random time and trying again would be a reasonable approach for the program doing the fork.

Consider a UNIX system having 100 process slots. Ten programs are running, each of which having to create 12 (sub)processes. After each process has created 9 processes, the 10 original processes and the 90 new processes have exhausted the table. Each of the 10 original processes now sits in an endless loop forking and failing – which is aptly the situation of a deadlock. The probability of this happening is very little but it could happen.

Difference between Deadlock, Starvation, and Livelock:
A livelock is similar to a deadlock, except that the states of the processes involved in the livelock constantly change with regard to one another, none progressing. Livelock is a special case of resource starvation; the general definition only states that a specific process is not progressing.

Livelock:

```
var l1 = .... // lock object like semaphore or mutex etc
var l2 = .... // lock object like semaphore or mutex etc

        // Thread1
        Thread.Start( ()=> {

    while (true) {

        if (!l1.Lock(1000)) {
            continue;
        }
```

Operating System | Translation Lookaside Buffer (TLB)

Computer Organization | Random Access Memory (RAM) vs Hard Disk Drive (HDD)

Most Visited Articles

StringBuilder Class in Java with Examples

Implementing a Linked List in Java using Class

HTML Course - Starting the Project | Creating Directories

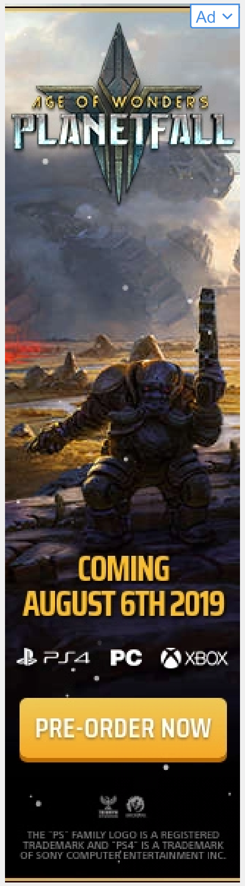
Make all numbers of an array equal

Amazon Interview Experience SDE (On Campus 2019)

Process states and Transitions in a UNIX Process

Operating System | Second Chance (or Clock) Page Replacement Policy

Program for buddy memory allocation scheme in Operating Systems | Set 1 (Allocation)



```
            if (!l2.Lock(1000)) {
                continue;
            }

            /// do some work
        });

            // Thread2
            Thread.Start( ()=> {

            while (true) {

                if (!l2.Lock(1000)) {
                    continue;
                }

                if (!l1.Lock(1000)) {
                    continue;
                }

                // do some work
            });
```

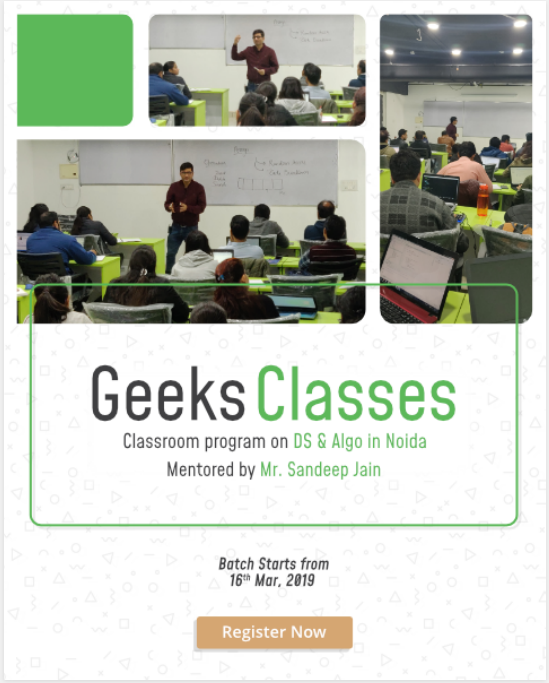
Deadlock:

```
var p = new object();
lock(p)
{
    lock(p)
    {
        // deadlock. Since p is previously locked
        // we will never reach here...
    }
```

A *deadlock* is a state in which each member of a group of actions, is waiting for some other member to release a lock. A *livelock* on the other hand is almost similar to a deadlock, except that the states of the processes involved in a livelock constantly keep on changing with regard to one another, none progressing. Thus Livelock is a special case of resource starvation, as stated from the general definition, the process is not progressing.

Starvation:
Starvation is a problem which is closely related to both, Livelock and Deadlock. In a dynamic system, requests for resources keep on happening. Thereby, some policy is needed to make a decision about who gets the resource when. This process, being reasonable, may lead to a some processes never getting serviced even though they are not deadlocked.

```
Queue q = .....
```


Geeks Classes
Classroom program on DS & Algo in Noida
Mentored by Mr. Sandeep Jain
Batch Starts from 16th Mar, 2019
Register Now

Most Trending Articles

Implement a stack using singly linked list

Amazon Interview Experience SDE 1 (2.5 Years Experienced)

HTML | Introduction

SYSTEM DESIGN
STARTING FROM
30TH MARCH, 2019
CLASSROOM PROGRAM IN NOIDA
MENTORED BY INDUSTRY EXPERTS
REGISTER NOW
BEST SUITED FOR EXPERIENCED DEVELOPERS

```
        while (q.Count & gt; 0)
{
    var c = q.Dequeue();
    .........

        // Some method in different thread accidentally
        // puts c back in queue twice within same time frame
        q.Enqueue(c);
    q.Enqueue(c);

    // leading to growth of queue twice then it
    // can consume, thus starving of computing
}
```

Starvation happens when “greedy” threads make shared resources unavailable for long periods. For instance, suppose an object provides a synchronized method that often takes a long time to return. If one thread invokes this method frequently, other threads that also need frequent synchronized access to the same object will often be blocked.

Recommended Posts:

- Operating System | Starvation and Aging in Operating Systems
- Deadlock Prevention And Avoidance
- Deadlock Detection And Recovery
- Operating Systems | Deadlock | Question 2
- Operating Systems | Deadlock | Question 1
- Operating System | Deadlock detection algorithm
- Program for Deadlock free condition in Operating System
- DBMS | Introduction to TimeStamp and Deadlock Prevention Schemes
- Operating System | Deadlock detection in Distributed systems
- Operating System | Process Management | Deadlock Introduction
- Techniques used in centralized approach of deadlock detection in distributed systems
- Operating System | Translation Lookaside Buffer (TLB)
- Operating System | The Linux Kernel
- Computer Organization | Random Access Memory (RAM) vs Hard Disk Drive (HDD)

Taking input from console in Python
Python list sort()
Functions in Python
Understanding "static" in "public static void main" in Java
How do I become a good Java programmer?
How to prepare for GATE CSE - 2019
GATE CSE Test Series - 2019 | Sudo GATE
Rotate a matrix by 90 degree in clockwise direction without using any extra space
Program for K Most Recently Used (MRU) Apps
Coin game of two corners (Greedy Approach)
Hashing in Java
OOPs | Object Oriented Design

Advertise Here

RashiBhardwaj
Check out this Author's contributed articles.

Article Tags : Operating Systems Deadlocks Operating Systems-Deadlock Process Synchronization

Practice Tags : Operating Systems

Be the First to upvote.

☐ To-do ☐ Done

0

No votes yet.

Feedback/ Suggest Improvement Add Notes Improve Article

Previous
Functions of Operating System

Next
Memory Hierarchy Design and its Characteristics