**Index**

|  |  |  |
| --- | --- | --- |
| **Sr.no** | **Content** | **Page no.** |
| 1. | Brief introduction | **1** |
|  | Aim of microproject | **1** |
|  | Resources required | **1** |
|  | Action plan | **2** |
| 2. | Brief description | **3** |
|  | Aim of microproject | **3** |
|  | Course outcomes | **3** |
|  | Actual procedure followed | **4** |
|  | Actual resources used | **10** |
|  | Learning outcomes | **10** |
|  | References | **10** |

**Annexure: I**

**Deadlock**

* **Brief introduction :-**

**Deadlock** is a situation that occurs in OS when any process enters a waiting state because another waiting process is holding the demanded resource. Deadlock is a common problem in multi-processing where several processes share a specific type of mutually exclusive resource known as a soft lock or software. Deadlocks are a set of blocked processes each holding a resource and waiting to acquire a resource held by another process.

* **Aim of project :-**

1. Two computer programs sharing the same resource are effectively preventing each other from accessing the resource.
2. at least one of the necessary conditions for deadlock can never hold.

* **Resources required :-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.no** | **Name of resources** | **Specification** | **Qty** | **Remark** |
|  | Computer system | Computer i3  Ram 2GB | 1 | Ok |
|  | Website | Google | 1 | Ok |
|  | Operating system | Book | 1 | Ok |

Page no:1

* **Action plan :-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.no** | **Detail of activity** | **Planned start date** | **Planned finished date** | **Name of students** |
| 1. | Raw data | 06-01-2020  1:00 to 3:00 | 13-01-2020  1:00 to 3:00 |  |
| 2. | Design | 20-01-2020  1:00 to 3:00 | 27-01-2020  1:00 to 3:00 | Awad Deepak |
| 3. | Prepare coding | 03-02-2020  1:00 to 3:00 | 10-02-2020  1:00 to 3:00 | Lokre Rohit |
| 4. | Testing of project | 17-02-2020  1:00 to 3:00 | 24-02-2020  1:00 to 3:00 | Karke Irnath |
| 5. | Display output | 02-03-2020  1:00 to 3:00 | 09-03-2020  1:00 to 3:00 |  |
| 6. | Prepare report | 16-03-2020  1:00 to 3:00 | 23-03-2020  1:00 to 3:00 |  |

* **Name of team member :-**

1. Awad Deepak Fulchand.
2. Lokre Rohit Hanmant.
3. Karke Irnath siddheshwar.

Page no:2

**Annexure: II**

**Deadlock**

* **Brief description :-**

**Deadlock** is a situation that occurs in OS when any process enters a waiting state because another waiting process is holding the demanded resource. Deadlock is a common problem in multi-processing where several processes share a specific type of mutually exclusive resource known as a soft lock or software. Deadlocks are a set of blocked processes each holding a resource and waiting to acquire a resource held by another process.

* **Aim of project :-**

1. Two computer programs sharing the same resource are effectively preventing each other from accessing the resource.
2. At least one of the necessary conditions for **deadlock** can never hold.

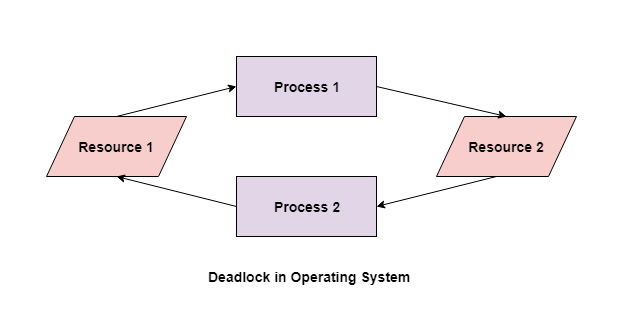
* **Course outcomes :-**

1. Install operating system and configure it.
2. Understand what deadlock is and how it can occur when giving mutually exclusive access to multiple resources.

Page no:3

* **Actual procedure followed :-**

A deadlock happens in operating system when two or more processes need some resource to complete their execution that is held by the other process. In deadlock, processes get blocked because each process is holding some resource and they are waiting for other resource, which is held by another process.



In the above diagram, the process 1 has resource 1 and needs to acquire resource 2. Similarly process 2 has resource 2 and needs to acquire resource 1. Process 1 and process 2 are in deadlock as each of them needs the other’s resource to complete their execution but neither of them is willing to relinquish their resources

* **Coffman Conditons :-**

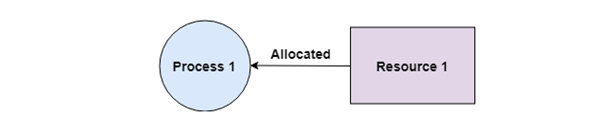
A deadlock occurs if the four Coffman conditions hold true. But these conditions are not mutually exclusive.

The Coffman conditions are given as follows –

Page no:4

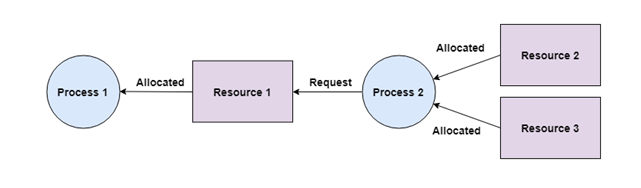
* **Mutual Exclusion**

There should be a resource that can only be held by one process at a time. In the diagram below, there is a single instance of Resource 1 and it is held by Process 1 only.



**Hold and Wait**

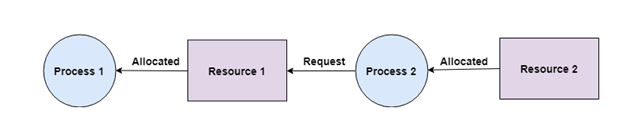
A process can hold multiple resources and still request more resources from other processes which are holding them. In the diagram given below, Process 2 holds Resource 2 and Resource 3 and is requesting the Resource 1 which is held by Process 1.



**No Preemption**

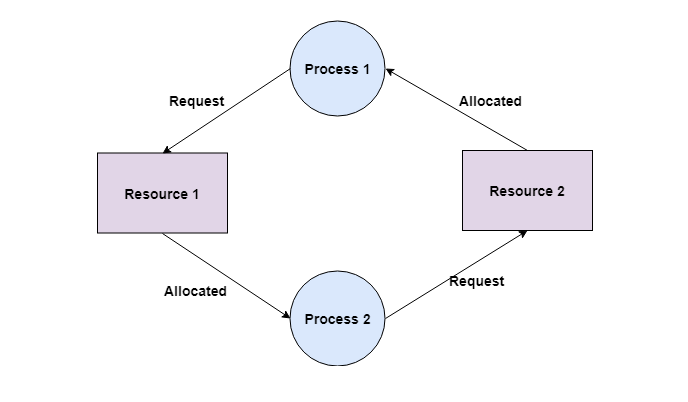
A resource cannot be preempted from a process by force. A process can only release a resource voluntarily. In the diagram below, Process 2 cannot preempt Resource 1 from Process 1. It will only be released when Process 1 relinquishes it voluntarily after its execution is complete.

Page no:5



**Circular Wait**

A process is waiting for the resource held by the second process, which is waiting for the resource held by the third process and so on, till the last process is waiting for a resource held by the first process. This forms a circular chain. For example: Process 1 is allocated Resource2 and it is requesting Resource 1. Similarly, Process 2 is allocated Resource 1 and it is requesting Resource 2. This forms a circular wait loop.



* **Deadlock Detection**

A deadlock can be detected by a resource scheduler as it keeps track of all the resources that are allocated to different processes. After a deadlock is detected, it can be resolved using the following methods –

* All the processes that are involved in the deadlock are terminated. This is not a good approach as all the progress made by the processes is destroyed.

Page no:6

* Resources can be preempted from some processes and given to others till the deadlock is resolved.
* **Deadlock Prevention**

It is very important to prevent a deadlock before it can occur. So, the system checks each transaction before it is executed to make sure it does not lead to deadlock. If there is even a slight chance that a transaction may lead to deadlock in the future, it is never allowed to execute.

* **Deadlock Avoidance**

It is better to avoid a deadlock rather than take measures after the deadlock has occurred. The wait for graph can be used for deadlock avoidance. This is however only useful for smaller databases as it can get quite complex in larger databases.

* **Deadlock Recovery**

A traditional operating system such as Windows doesn’t deal with deadlock recovery as it is time and space consuming process. Real-time operating systems use Deadlock recovery.

* **Recovery method:-**

**Killing the process:** killing all the process involved in the deadlock. Killing process one by one. After killing each process check for deadlock again keep repeating the process till system recover from deadlock.

**Resource Preemption:** Resources are preempted from the processes involved in the deadlock, preempted resources are allocated to other processes so that there is a possibility of recovering the system from deadlock. In this case, the system goes into starvation.

Page no:7

* **Bankers algorithm:-**

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.

* **Code:-**

// Banker's Algorithm

#include <iostream>

using namespace std;

int main()

{

// P0, P1, P2, P3, P4 are the Process names here

int n, m, i, j, k;

n = 5; // Number of processes

m = 3; // Number of resources

int alloc[5][3] = { { 0, 1, 0 }, // P0 // Allocation Matrix

{ 2, 0, 0 }, // P1

{ 3, 0, 2 }, // P2

{ 2, 1, 1 }, // P3

{ 0, 0, 2 } }; // P4

int max[5][3] = { { 7, 5, 3 }, // P0 // MAX Matrix

{ 3, 2, 2 }, // P1

{ 9, 0, 2 }, // P2

{ 2, 2, 2 }, // P3

{ 4, 3, 3 } }; // P4

int avail[3] = { 3, 3, 2 }; // Available Resources

int f[n], ans[n], ind = 0;

for (k = 0; k < n; k++) {

f[k] = 0;

}

int need[n][m];

for (i = 0; i < n; i++) {

for (j = 0; j < m; j++)

need[i][j] = max[i][j] - alloc[i][j];

}

int y = 0;

for (k = 0; k < 5; k++) {

Page no:9

for (i = 0; i < n; i++) {

if (f[i] == 0) {

int flag = 0;

for (j = 0; j < m; j++) {

if (need[i][j] > avail[j]){

flag = 1;

break;

}

}

if (flag == 0) {

ans[ind++] = i;

for (y = 0; y < m; y++)

avail[y] += alloc[i][y];

f[i] = 1;

}

}

}

}

cout << "Following is the SAFE Sequence" << endl;

for (i = 0; i < n - 1; i++)

cout << " P" << ans[i] << " ->";

cout << " P" << ans[n - 1] <<endl;

return (0);

}

* **Output:-**

Following is the SAFE Sequence

P1 -> P3 -> P4 -> P0 -> P2

Page no:10

* **Resources required :-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.no** | **Name of resources** | **Specification** | **Qty** | **Remark** |
|  | Computer system | Computer i3  Ram 2GB | 1 | Ok |
|  | Website | Google | 1 | Ok |
|  | Operating system | Book | 1 | Ok |

* **Learning Outcomes:-**
  1. I will learn the concept of deadlock.
  2. I will studied about deadlock detection and prevention.
  3. I will learn use of deadlock in operating system.
  4. I will studied the algorithm for bankers which are present in deadlock which are present in operating system.
* **References:-**

1. [www.google.com](http://www.google.com)
2. [www.TutorialsPoint.com](http://www.TutorialsPoint.com)
3. [www.guru99.com](http://www.guru99.com)

Page no:10