**Date: 10-Apr-2013**

**Key Note:**

***Experiment No: 1***

Write a C program to Develop Applications to illustrate Inter Process communication using Semaphores

### Semaphores

A *semaphore* is an integer count that is accessed atomically using two operations that are conventionally called P and V:

* The P operation (mnemonic de*p*lete) decrements the count. If the result is not negative, the operation succeeds and returns. If the result is negative, the P operation suspends the calling process until the count has been made nonnegative by another process doing a V operation.
* The V operation (mnemonic re*v*i*v*e) increments the count. If this changes the value from negative to nonnegative, one process that is waiting in a P operation is unblocked.

### Managing Unnamed Semaphores

An unnamed semaphore is a semaphore object that exists in memory only. An unnamed semaphore can be identified only by its memory address, so it can be shared only by processes or threads that share that memory location.

The functions for creating and freeing unnamed semaphores are summarized below:-

**Functions to Manage Unnamed Semaphores**

| **Function Name** | **Purpose and Operation** |
| --- | --- |
| sem\_init | Initialize a semaphore object, setting its value and preparing it for use. |
| sem\_destroy | Make a semaphore unusable. |

***Header File:-***

#include <semaphore.h>

***Creation of Semaphore Variable:-***

sem\_t mutex;

***Syntax:-***

int sem\_init(sem\_t \*sem, int pshared, unsigned int value);

**For Eg:-**

**sem\_init(mutex,0,1);**

***Syntax:-***

int sem\_destroy(sem\_t \*sem);

**For Eg:-**

**sem\_destroy(&mutex);**

**Functions to Operate on Semaphores**

| **Function Name** | **Purpose and Operation** |
| --- | --- |
| sem\_getvalue | Return a snapshot of the current value of a semaphore. |
| sem\_post | Perform the V operation, incrementing a semaphore and possibly unblocking a waiting process. |
| sem\_trywait | Perform the P operation only if the value of the semaphore is 1 or more. |
| sem\_wait | Perform the P operation, decrementing a semaphore and blocking if it becomes negative. |

***Syntax:-***

**int sem\_post(sem\_t \*sem);**

**For Eg:-**

**sem\_post(&mutex);**

***Syntax:-***

**int sem\_getvalue(sem\_t \*sem, int \*sval);**

**For Eg:-**

***int x=sem\_getvalue(&mutex,3);***

***Syntax:-***

**int sem\_wait(sem\_t \*sem);**

**For Eg:-**

**sem\_wait(&mutex);**

***Syntax:-***

**int sem\_trywait(sem\_t \*sem);**

Functions handling Named Semaphore:-

To create semaphore:-

semget - get a semaphore set identifier

Header Files:-

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

Syntax:-

int semget(key\_t key, int nsems, int semflg);

The semget() system call returns the semaphore set identifier associated with the argument key. A new set of nsems semaphores is created.

Example:- int semid = semget((key\_t)10,5,IPC\_CREAT | 0666);

To Control Semaphore:-

int semctl(int semid, int semnum, int cmd, ...);

DESCRIPTION:-

semctl() performs the control operation specified by cmd on the semaphore set identified by semid, or on the semnum-th semaphore of that set. This function has three or four arguments, depending on cmd.

This function has three or four arguments, depending on cmd. When there are four, the fourth has the type union semun. The calling program must define this union as follows:

union semun

{

int val; /\* Value for SETVAL \*/

struct semid\_ds \*buf; /\* Buffer for IPC\_STAT, IPC\_SET \*/

unsigned short \*array; /\* Array for GETALL, SETALL \*/

struct seminfo \*\_\_buf; /\* Buffer for IPC\_INFO (Linux specific) \*/

};

Example:-

int retval = semtcl(semid,0,SETVAL,1);

int retval = semctl(semid,1,GETVAL,0);

int retval = semctl(semid,IPC\_RMID,0);

int retval = semctl(semid,0,GETALL,array);

int retval = semctl(semid,0,SETALL,array);

Operations to be performed on Semaphore:-

int semop(int semid, struct sembuf \*sops, unsigned nsops);

semop() performs operations on selected semaphores in the set indicated by semid. Each of the snops elements in the array pointed to by sops specifies an operation to be performed on single semaphore.

The elements of sembuf structure is

unsigned short sem\_num; // Semaphore number

short sem\_op; // Operation to be performed on semaphore +ve or -ve

short sem\_flg; // Flag value is IPC\_NOWAIT or SEM\_UNDO

Each operation is performed on the sem\_num-th semaphore of the semaphore set, where the first semaphore of the set is numbers 0.

There are three type of operations distinguished by the value of sem\_op:-

1. Positive integer:- adds this value to the semaphore value
2. Zero:- Wait for zero operation, operation can immediately proceed
3. Less than zero:- Specified value is subtracted from semaphore value.

Example:-

int retval = semop(semid,&sops,0);