is present in
to synchronize
be accessing

g or as well as
set.

the semaphore
naphore Value
trol operation

should enter
or not.
ion).

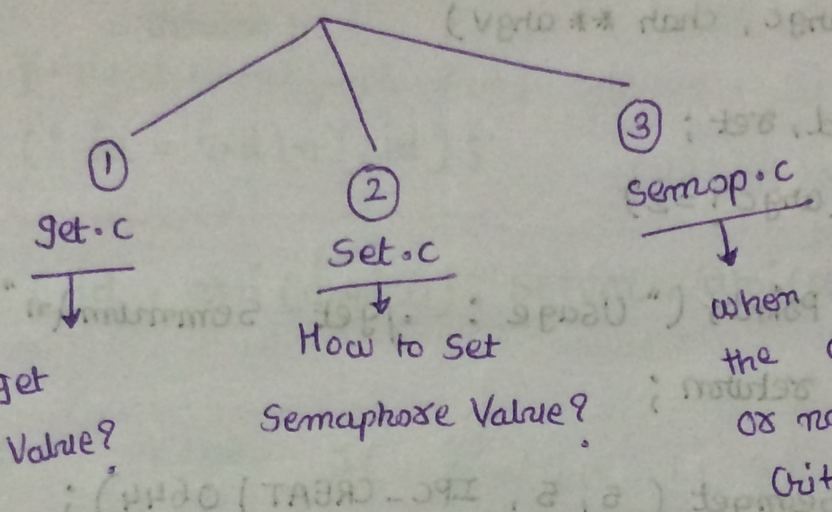
ed.
d.

⊛ $ man Semget

Semget → get a semaphore identifier.

```
#include <sys/types.h>
# include <sys/ipc.h>
# include <sys/sem.h>

int semget (key_t key, int nsems, int semflg);
```



⇒ using the semget() function creat the array of semaphore (array of integer).

ipcs -s :- You get currently How many Semaphore set.

⊛ $ man Semctl

Semctl → Semaphore control operation

```
#include <sys/types.h>
# include <sys/ipc.h>
# include <sys/sem.h>

int semctl (int semid, int semnum, int cmd, ----);
```

semnum → which you get semaphore number

GETVAL → getting the Value.

SETVAL → Setting the Value.

⇒ default we creat the semaphore set in Kernel Space all are zeros.

Program :- get.c

```
#include "header.h"
main(int argc, char **argv)
{
    int id, ret;
    if (argc!=2)
    {
        Printf("Usage : ./get Semnum\n");
        return;
    }
    id = Semget (5, 5, IPC_CREAT | 0644);
    if (id<0)
    {
        Perror("Semget");
        return;
    }
    Printf("id = %d\n", id);

    ret = semctl (id, atoi(argv[1]), GETVAL);
    Printf("ret = %d\n", ret);
}
```

Program : set.c

```
#include "header.h"
main(int argc, char **argv)
{
    int id, ret;
```

```
    if (argc != 3)
    {
        Printf ("Usage : ./set Segnum Value\n");
        return ;
    }
  id = Semget ( 5, 5, IPC_CREAT|0644);
    if (id<0)
    {
        Perror (" semget ");
        return ;
    }
 Printf (" id = %d\n", id);
```

```
 semctl (id , atoi (argv[1]) , SETVAL , atoi (argv[2]));
}
```

output :- ./set 1 2

id = 0

ret = 2

using ./get 1

(*) $ man semop

```
#include <sys/types.h>
# include <sys/ipc.h>
# include <sys/sem.h>

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

  struct sembuf
  {
      unsigned short sem_num ; // semaphore number.
      short  sem_op ; // semaphore ~~numbers~~ operation.
      short  sem_flg ; // operation flags.
  } ;
```

⇒ Sem-op is deside process should Enter or not Enter the Critical section.

Program : semop.c

```
# include "header.h"
main()
{
    Struct sembuf V;
    int id, ret;
        id = semget (5, 5, IPC-CREAT | 0644);
        if (id<0)
        {
            Perror (" semget");
            return ;
        }
    Printf (" id = %d \n", id);
```

```
V.sem_num = 2 ;
V.sem_op =                    // +ve, 0, -ve number
V.sem_flag = 0 ;

        Printf (" Before ---- \n");
        Semop (id, &V, 1);
        Printf (" After ---- \n");
}
```

(1) IF Sem_op is zero, the process must have read Permission on the Semaphore set. This is a "wait-for zero" operation ; if Semval is zero the operation can immediately proceed.

Enter or not Enter

644);

umber

read
" wait-for
operation

(2) If Sem_op is a positive integer, the operation add this value to the semaphore value (semval). The operation can always proceed - it never forces a process to wait.

```
V. sem_num = 2 ;
V. sem_op = 2 ;
V. sem_flg = SEM_UNDO ;
        Printf (" Before ---- \n") ;
        semop (id, &V, 1) ;
        Sleep(10) ;
        Printf (" Done ---- \n") ;
```

output :- positive means always enter in a critical section when enter in a critical section then ret = 4 after Sleep(10) then using ./get 2 we get ret = 2 because flag is SEM_UNDO.

(3) If Sem_op is less than zero, the process must have after permission on the semaphore set. If semval is greater than or equal to the absolute value of Sem_op, the operation can proceed immediately the absolute value of Sem op is substracted from semval.

(*) semval >= |sem op|

1 >= |-2|

Then block

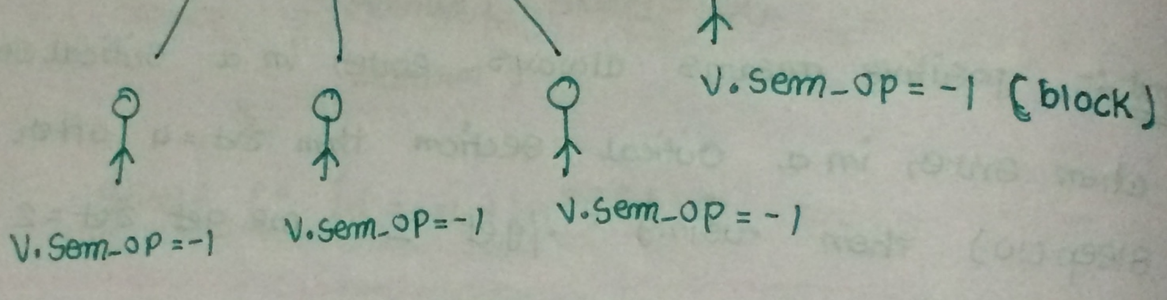
(*) semval >= |sem op|

4 >= |-2|

Enter in critical section.

```
V.sem_num = 2 ;
V.sem_op = -2 ;
V.sem_flg = 0 ;

        printf (" Before ----\n");
        Sem_op (id, &v, 1)
        Sleep (10);
        Printf (" Done ----\n");
```



Semaphore two types

binary Semaphore

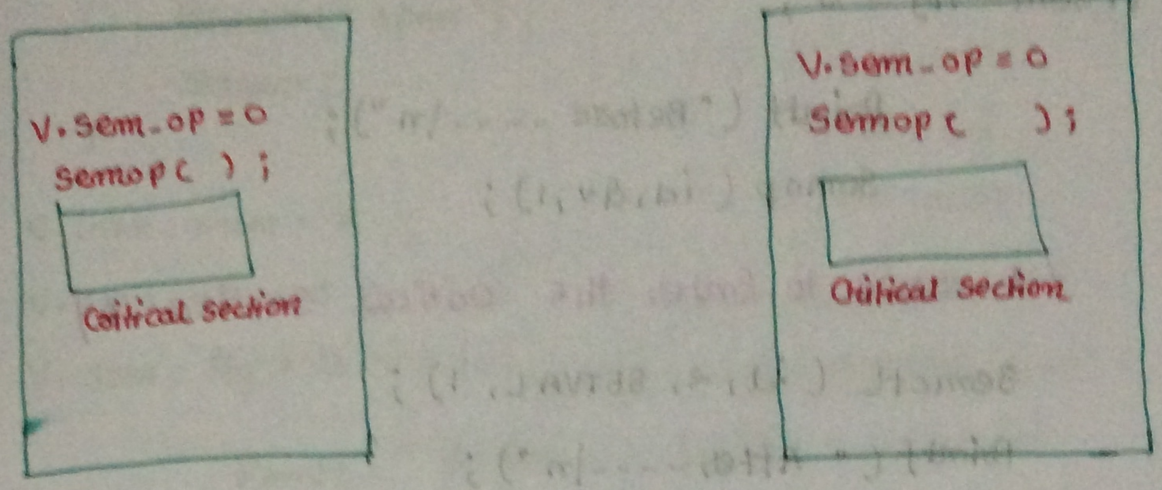
- put only 0 & 1

Counting Semaphore

apart from 0 & 1, all Positive Value allowed.

- we can use Counting Semaphore as a Binary Semaphore
- Binary semaphore can not use as a Counting Semaphore.

Program :- both process write data one-by-one in data file using Semaphore.



Program W1.c :-

```
#include "header.h"
main()
{
    struct sembuf V ;
    int id ;
    char a[] = " Savan " ;
    int fd, i ;
        id = Semget (5,5, IPC_CREAT |0644);
        if (id<0)
        {
            Perror (" Semget") ;
            return ;
        }
    fd = open ("data", O_RDWR |O_APPEND|O_CREAT.
                                        0644) ;
    if(fd<0)
    {
        Perror (" open");
        return ;
    }
```

```
V.sem_num = 2 ;
V.sem_op = 0 ;
V.sem_flg = 0 ;
        Printf ("Before ----\n");
        Semop ( id, &V, 1) ;

/* Start to Enter the Critical Section */

Semctl (id, 2, SETVAL, 1) ;
Printf (" After ----\n");
for (i=0 ; a[i] ; i++)
{
      write ( fd, &a[i], 1) ;
      Sleep(1) ;
}
Printf ("Done ----\n");
      Semctl (id, 2, SETVAL, 0);
}
```

Program W2.c

```
#include "header.h"
main()
{
    Struct sembuf V ;
    int id ;
    Char a[] = " Bhalodiya " ;
    int fd, i ;
        id = Semget (5, 5, IPC_CREAT | 0644);
        if (id < 0)
        {
            Perror (" Semget");
            return ;
        }
```