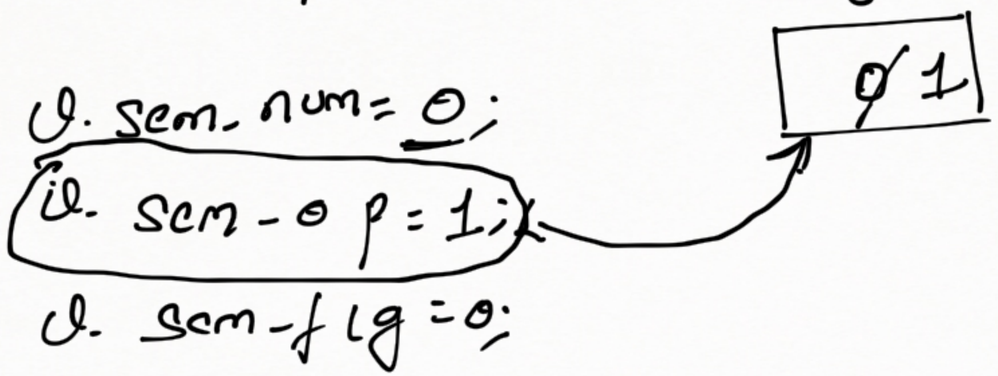
Date: 26-11-21 :

Case-2 : if sem_op value postive

if sem_op value positive number then this value added To the semaphore value, and the process allways proceed.



semop(id, &v, 1);

Semval = Semval + Sem_op

0 + 1

1

1 + 1

2

Case-3: if sem_op is negative integer (less than zero)

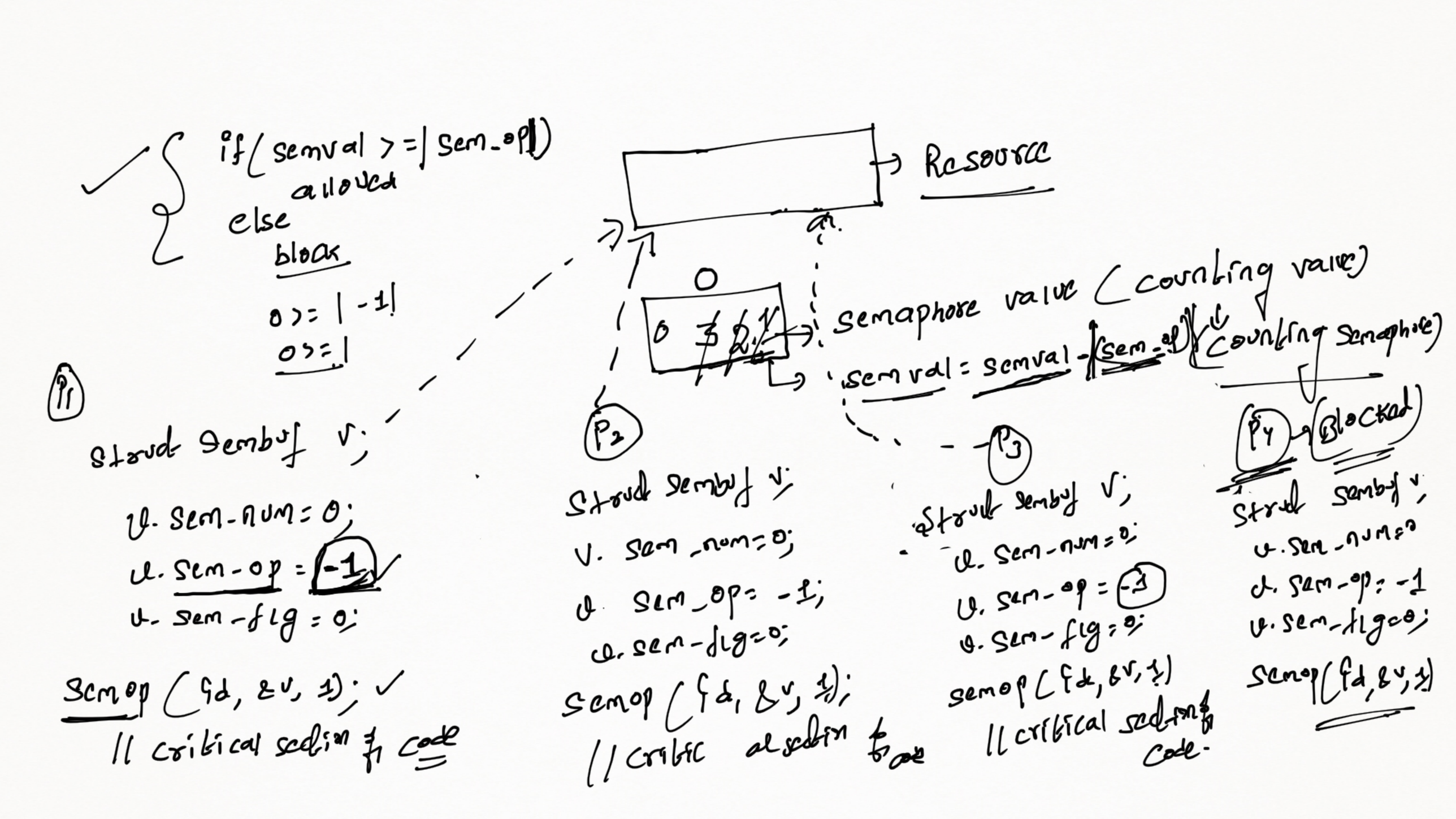
→ if sem_op is less than zero, if semval greater than (or) equal to the absolute value of sem_op, operation proceed otherwise blocked

if (semval >= |sem_op|)

process allowed ✓

else

process blocked ✓


if(semval >= | sem_op|)
allowed
else
block
0 >= | -1|
0 >= 1
Resource
0
0 1 2 V
semaphore value (counting value)
semval = semval - (sem_op)
(counting semaphore)
P1
P2
P3
P4
Blocked
struct sembuf v;
v.sem_num = 0;
v.sem_op = -1
v.sem_flg = 0;
semop (fd, &v, 1); ✓
// critical section of code
struct sembuf v;
v.sem_num = 0;
v.sem_op = -1;
v.sem_flg = 0;
semop (fd, &v, 1);
// critical section of code
struct sembuf v;
v.sem_num = 0;
v.sem_op = -1
v.sem_flg = 0;
semop (fd, &v, 1)
// critical section of code
struct sembuf v;
v.sem_num = 0
v.sem_op = -1
v.sem_flg = 0;
semop (fd, &v, 1)

counting semaphore : If the semaphore value greater than (or) equal to 1 than those semaphores are called counting semaphores

sem_undo flag:

Ex 1 :

```
v. sem_num = 0;
v. sem_op = -1
v. sem_flg = SEM_UNDO
```

since sem_op = -1 will check the condition if semval greater than or equal to absolute value of sem_op, if condition satisfied then semval = semval - | sem_op |

When process terminated the subtracted value added to semaphore value if sem_undo flag specified.

Ex2:

```
0. sem_num = 0;
0. sem_op = 1;
0. sem_flg = SEM_UNDO;
```

→ In this case sem_op is a positive integer so when process started semval = semval + sem_op (this process is allways allowed)

when process terminated added value is subtracted that means semval = semval - sem_op (Because of SEM_UNDO flag)