

Reader Entry

```
P(mutex);  
if (wwc>0) or (wc>0) {  
    rwc++;  
    V(mutex);  
    P(rsem);  
    P(mutex);  
    rwc--; };
```

```
rc++;  
V(mutex);
```

Reader Exit

```
P(mutex);  
rc--;  
if (rc=0) && (wwc>0) V(wsem);
```

Writer is writing

Block

why?

starvation

Writer Entry

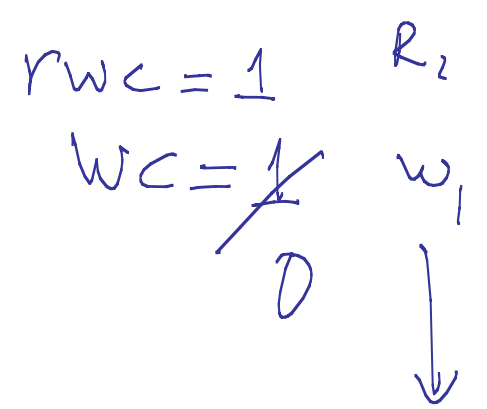
```
P(mutex);  
if (rc>0) || (wc>0) || (rwc>0) || (wvc >0) {  
    wvc++;  
    V(mutex);  
    P(wsem);  
    P(mutex);  
    wvc--; }  
  
wc++;  
V(mutex);
```

reader present
writer present

why?
Block

Writer Exit

```
P(mutex);  
wc-- ;  
if (rwc>0) then  
    for (i=1; i<=rwc; i++)V(rsem)  
else if (wvc>0) V(wsem);  
V(mutex)
```



Reader Entry

```
P(mutex);  
  if (wwc>0) or (wc>0) then begin  
    rwc++;  
    V(mutex);  
    P(rsem);  
    rwc--;  
  end;
```

```
rc++;
```

```
if rwc>0 then V(rsem)  
else V(mutex);
```

new code?

if no waiting writer
do V(mutex)

Reader Exit

```
P(mutex);  
rc--;  
if (rc=0) and (wwc>0) then V(wsem);
```



Writer Entry

P(mutex);

if (rc>0) or (wc>0)

then begin

 wvc++;

 V(mutex);

 P(wsem);

 wvc--;

end;

 wc++;

 V(mutex);

P(mutex) missing.

Writer Exit

P(mutex);

 wc-- ;

 if (rwc>0) then V(rsem)

 else

 if (wvc>0) then V(wsem);

 else V(mutex)

$P(\text{mutex})$

—
—
—

$V(\text{X})$

$V(\text{mutex})$

delete →

$P(X)$

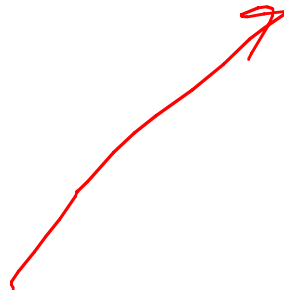
$P(\text{mutex})$

$V(\text{mutex})$

P_3

$P(m)$

$V(m)$



$P(\text{mutex})$

—
—
—

$V(X)$

$P(X)$

$V(\text{mutex})$

Implementing semaphores

Definition

$P(S) \rightarrow$ while $(S == 0);$ atomic
 $\{ S - - \}$

$V(S) \rightarrow S + +$ ← atomic

$P(s) \rightarrow s--$
 $\text{while}(s < 0);$

$V(s) \rightarrow s++$

Semaphore = an integer & a queue

Count
&] struct.

→ queue of PCB

// UNIPROC semaphore

P(S)

[S.count
S.Q.

Disable interrupts

S.count --

if (S.count < 0)

{ save context

move current process
to S.Q // addQ(cp, S.Q)

move a PCB from readyQ
to curProc

ENABLE
return

}

"block"

routine
provided by scheduler

$P(S) \rightarrow$ Disable
S. count --
if S-count < 0
 block(S.Q)
else
 ENABLE

has enable inside

Q to
block the process
into

$V(S)$:-

Disable

$S.count ++$

if ($S.count <= 0$)

move one PCB

from $S.Q$ to $readyQ$

ENABLE

$temp = delQ(S.Q)$

$addQ(readyQ, temp)$

unblock
a
process

multiprocessor implementations

- Disable does not work

- use test&set.

Spinlock(x) \rightarrow while test&set(x);

unlock(x) \rightarrow x = 1;

Semaphore

lock $\leftarrow 1$

count \leftarrow initial value

Q \leftarrow null

init Sem(S, v) \rightarrow S.count = ~~v~~;

P(s) : spinlock (s.lock)

s.count --

if s.count < 0

{ block (s.g) }

else unlock (s.lock)

V(S) : spinlock(s.lock)

s.count ++

if s.count <= 0

unblock(s.g);

unlock(s.lock)

lock → test & set