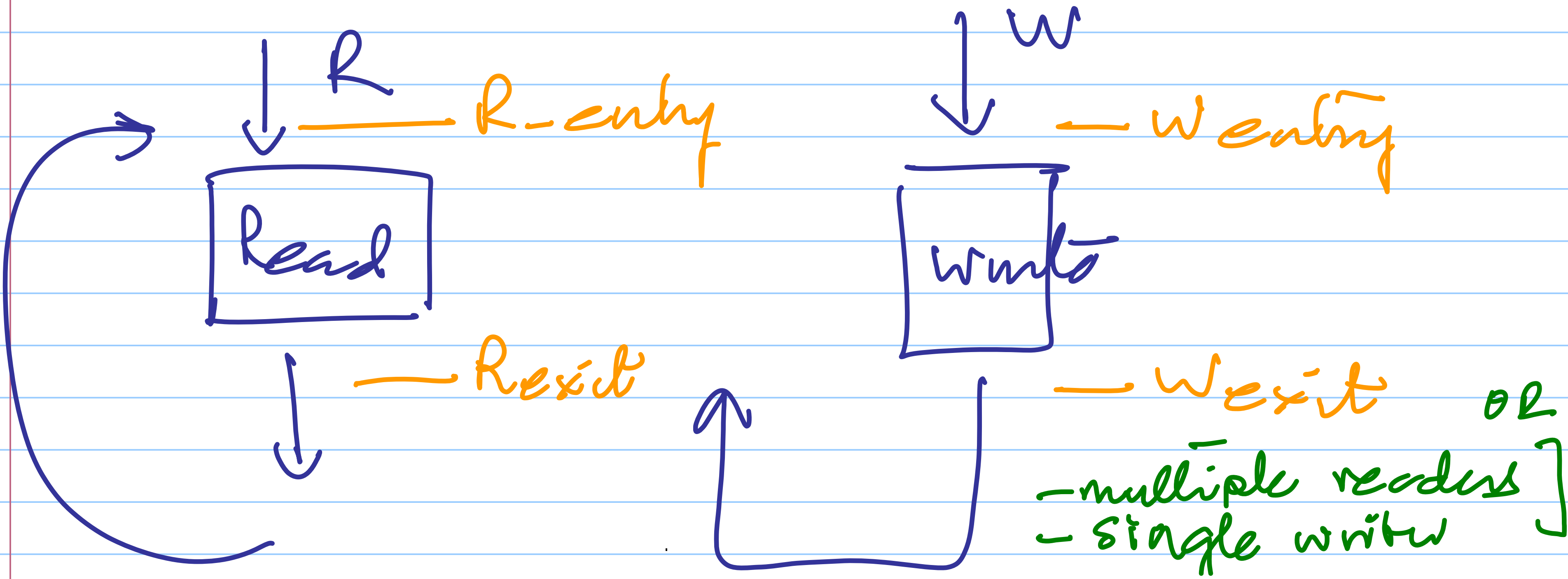
CSE 330: Operating Systems

Class: 13 **Date:** 9/29

Fall 2016

Note Title

Readers & Writers



OR

- multiple readers
- single writer

Reader Entry

```
P(mutex);
   rc++;
   if rc==1 then P(wsem);
V(mutex);
```

← other readers block here

Block → if W is active

Writer Entry

```
P(wsem)
```

w

Reader Exit

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

Writer Exit

```
V(wsem)
```

wsem = 1

mutex = 1

rc = 0

↑

reader count

— writer starvation

Reader Entry

```
P(rsem);
V(rsem)
P(rmutex);
 rc++
 if rc==1 P(wsem);
V(rmutex);
```

Reader Exit

```
P(rmutex);
 rc--;
 if rc==0 V(wsem);
V(rmutex);
```

Writer Entry

```
P(wmutex);
 wc++;
 if wc==1 P(rsem);
V(wmutex);
P(wsem);    old
```

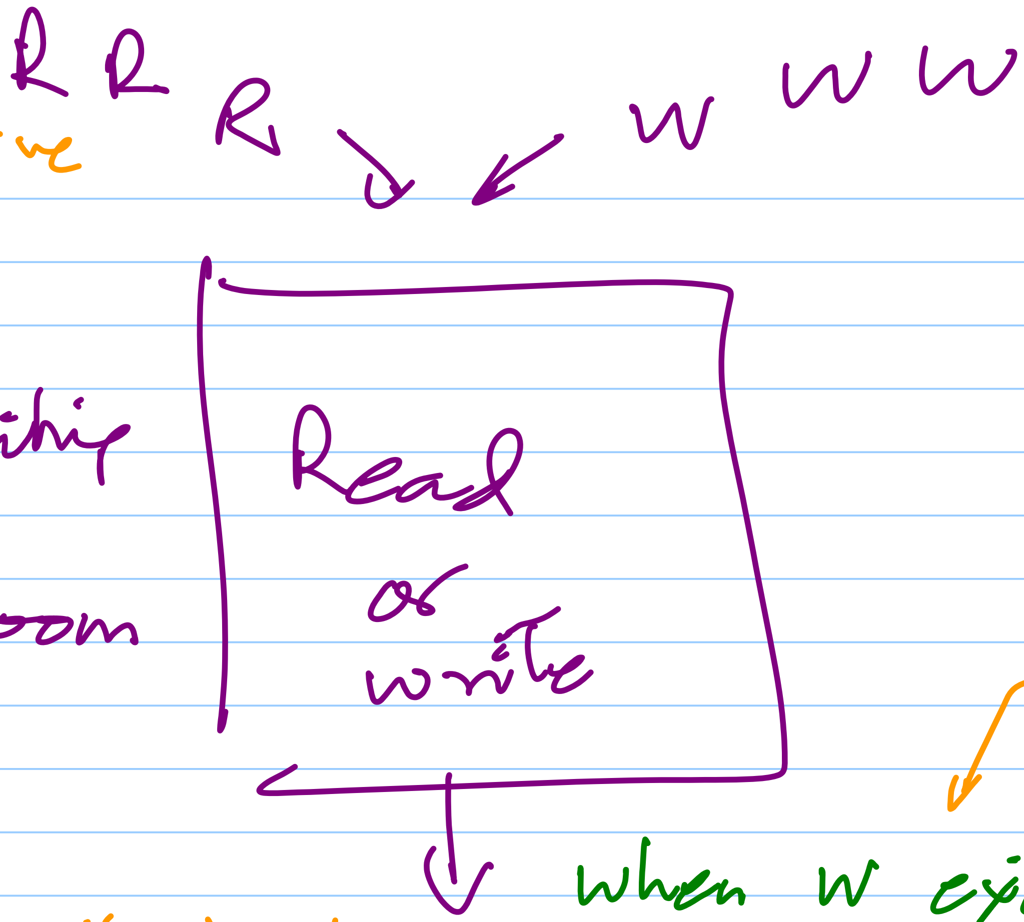
if
w active
block
R

Writer Exit

```
V(wsem);    od
P(wmutex);
  wc--;
  if wc==0 V(rsem);
V(wmutex);
```

NO
W-starve

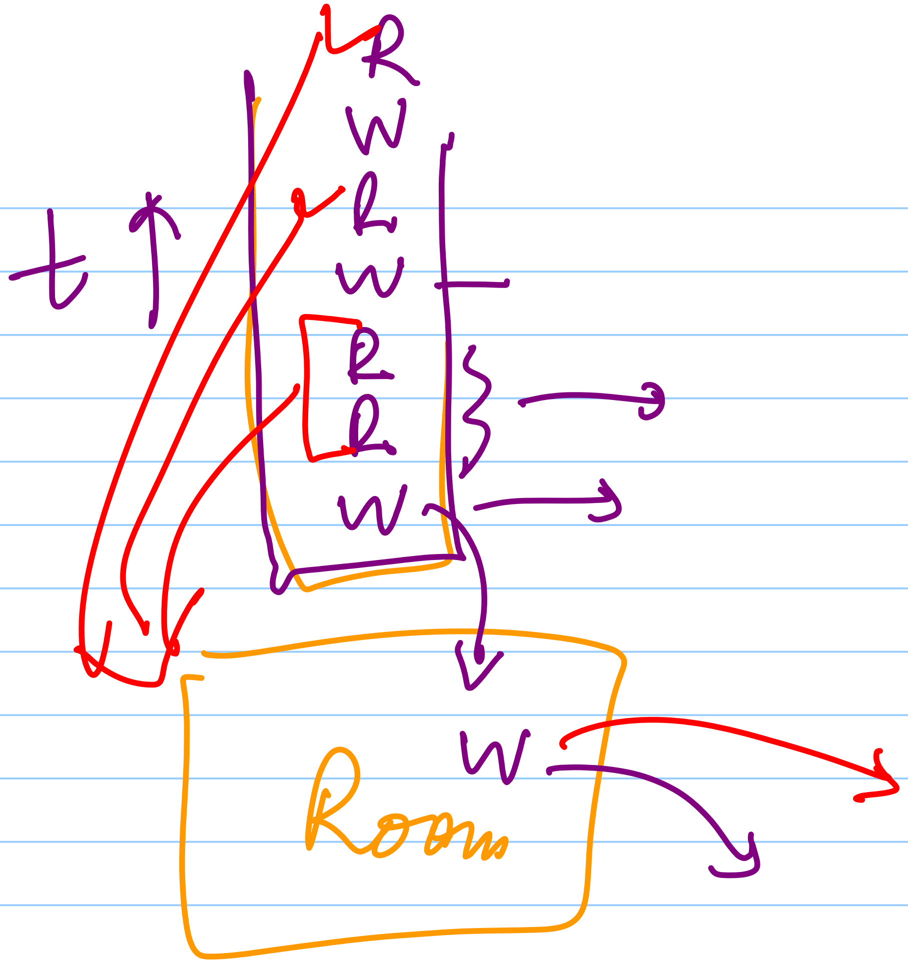
R not
allowed
if W waiting
or
W in room

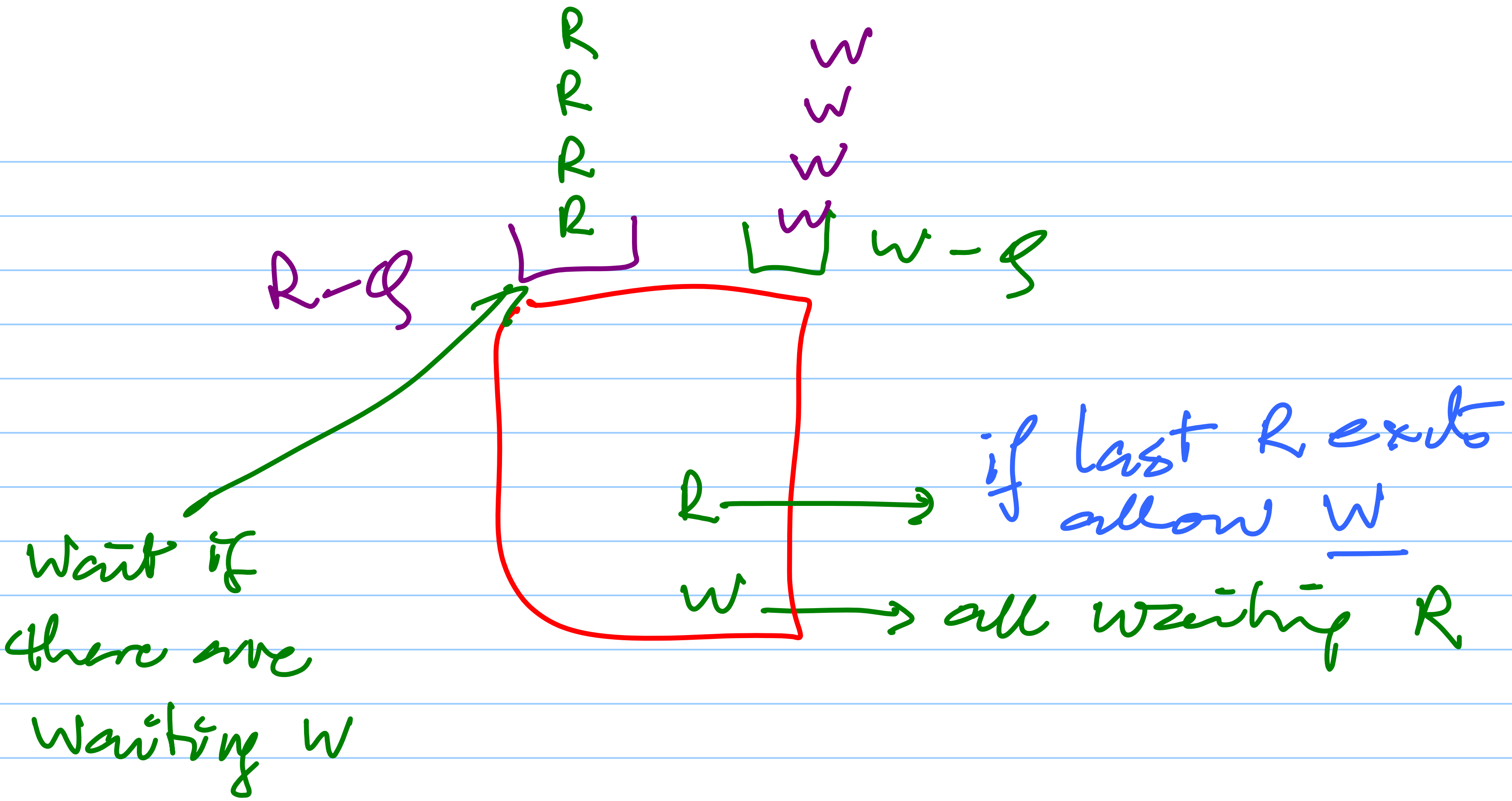


No R
starvation

when W exits
allow R to enter

Result
give priority to W


t
R
W
R
W
R
R
W
W
Room



Wait if there are waiting W

R → if last R exits allow W

W → all waiting R

rsem = 0
wsem = 0 → to block R or W

↳ 2 queues

mutex = 1 → to control race condition

↳ NO blocking inside mutex

readercount ↓ rc, wc ↓ writer count

rc, wc
rwc, wwc } = 0

wwc → w_wait_c

rwc ← reader_waiting_count

Reader Entry

W-waiting

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

sleep

writer inside

Writer Entry

no R or w

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

needed to prevent a race cond

Reader Exit

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

✓

WAKE

Writer Exit

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

missing P(mutex)

wake up other waiting readers if any

Reader Exit

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

missing V(mutex)

Writer Entry

```
P(mutex);
if (rc>0)or(wc>0) [no extra]
        then begin
           wwc++;
           V(mutex);
           P(wsem); ← no P(mutex)
           wwc--;
        end;
   wc++;
V(mutex);
```

Writer Exit

```
  P(mutex);
   wc-- ;                ← wake up 1 R
   if (rwc>0) then V(rsem)
   else
      if (wwc>0) then V(wsem); ← wake up 1 W
      else V(mutex) ← V if no one woken up
```