CSE 330: Operating Systems

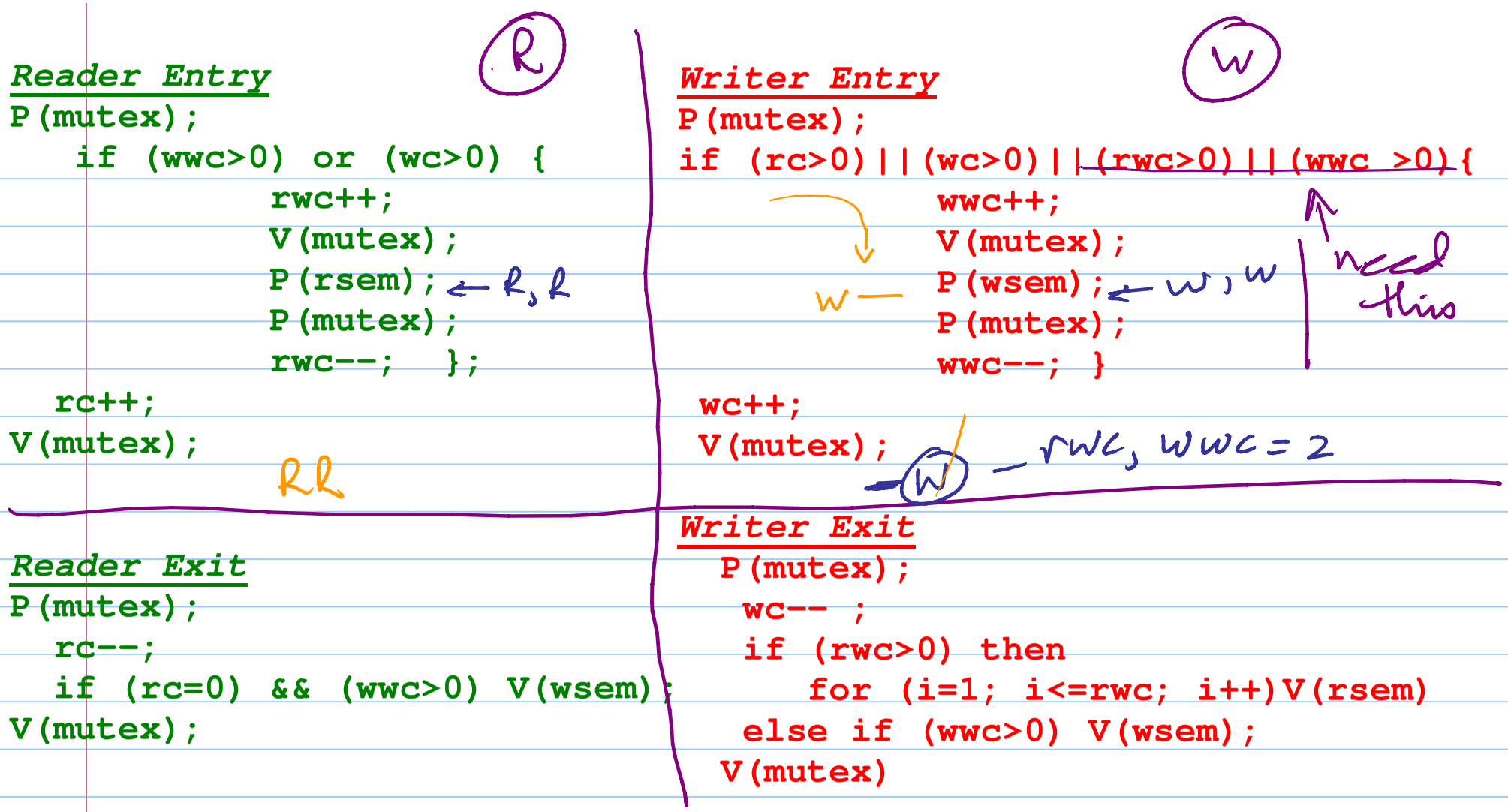
Class: 14 **Date:** 2/22

Spring 2018

Note Title

Readers & writers

1. R starvation
2. W starvation
3. No starvation → R exit gives W preference
   → W exit gives R preference


Reader Entry
P(mutex);
if (wwc>0) or (wc>0) {
rwc++;
V(mutex);
P(rsem);
P(mutex);
rwc--; };
rc++;
V(mutex);
R
R, R
RR
Writer Entry
P(mutex);
if (rc>0)||(wc>0)||(rwc>0)||(wwc >0){
wwc++;
V(mutex);
P(wsem);
P(mutex);
wwc--; }
wc++;
V(mutex);
W
W, W
need this
W
rwc, wwc = 2
Reader Exit
P(mutex);
rc--;
if (rc=0) && (wwc>0) V(wsem);
V(mutex);
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) {
              rwc++;
              V(mutex);
              P(rsem);
              rwc--;
       end;
   rc++;
   if rwc>0 then V(rsem)
   else V(mutex);
```

missing P(mutex)

creates a reader cascade

Reader Exit

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

missing V

Writer Entry

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

missing P

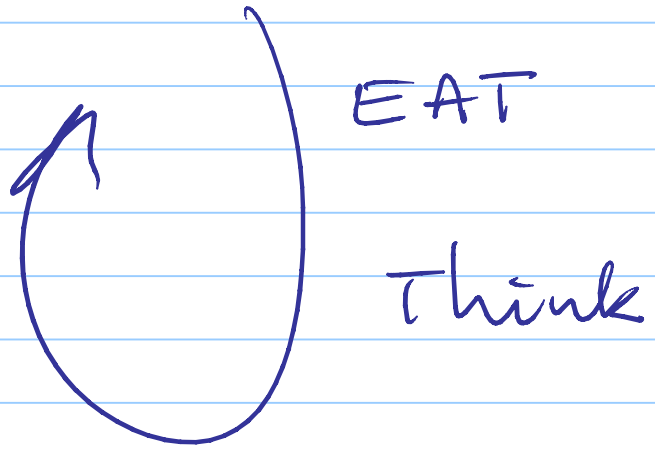
Writer Exit

```
  P(mutex);
   wc-- ;
   if (rwc>0) then V(rsem)
   else
       if (wwc>0) then V(wsem)
       else V(mutex)
```

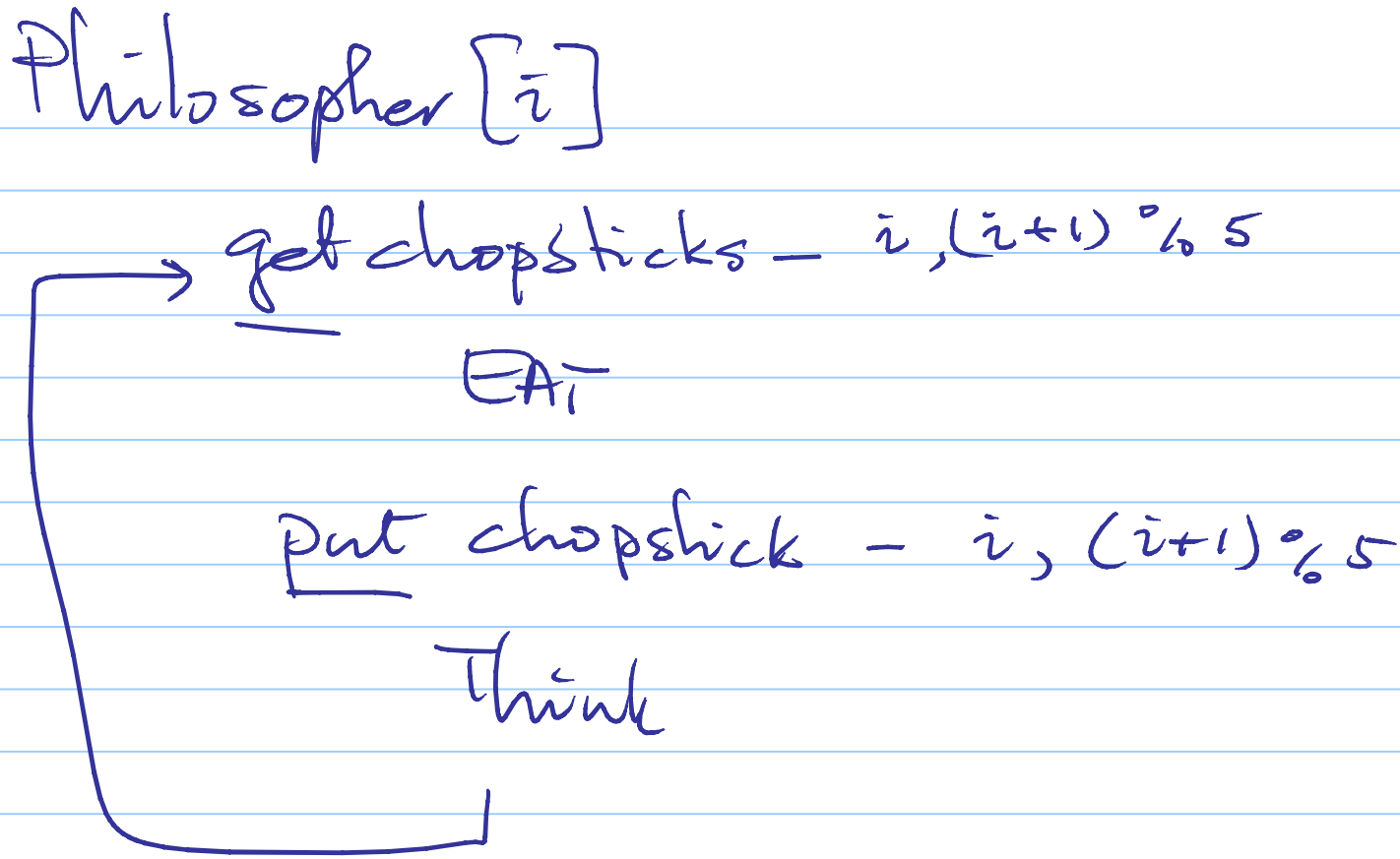
missing V(mutex)

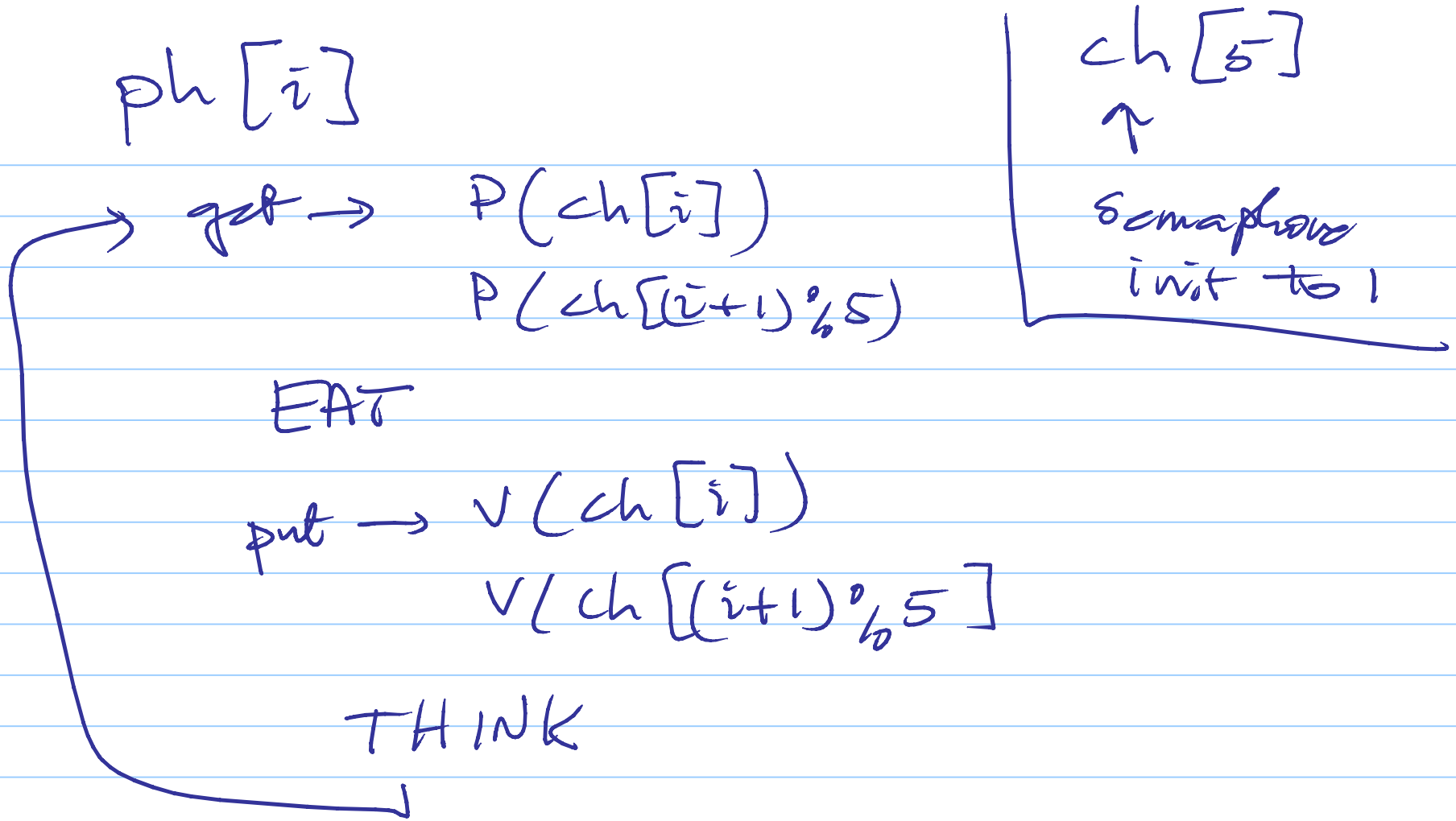
Dining Philosopher – 5 philosophers

Philosopher



4
0
3
4
0
0
3
1
2
2
1


Philosopher [i]
get chopsticks – i, (i+1) % 5
EAT
put chopstick – i, (i+1) % 5
Think


ph[i]
get →
P(ch[i])
P(ch[(i+1)%5)
EAT
put →
V(ch[i])
V(ch[(i+1)%5]
THINK
ch[5]
semaphore
init to 1

No deadlock

→ Philosophers pick right & then left
except ph[4] → L then R

- Before taking any chopstick make sure both are available
  - Do not grab 1 & wait for another
    - if not available, sleep.

1 mutex sem → init 1

5 sleep sem → init 0

5 chopstick variables (integer)

0 → not avail

1 → avail.

philosopher [i]

pickup

P(mutex)

while (ch[i] == 0) OR (ch[i+1] == 0)

V(mutex)

P(sleep[i])

P(mutex)

ch[i] = 0

ch[j+1] = 0

V(mutex)

putdown

P(mutex)

ch[L], ch[R] = 1

V(sleep[left])

V(sleep[right])

V(mutex)

(i+1) % 5

[i-1]

[i+1] mod 5