CSE 330: Operating Systems

Class: 14 **Date:** 10/4

Fall 2016

Note Title

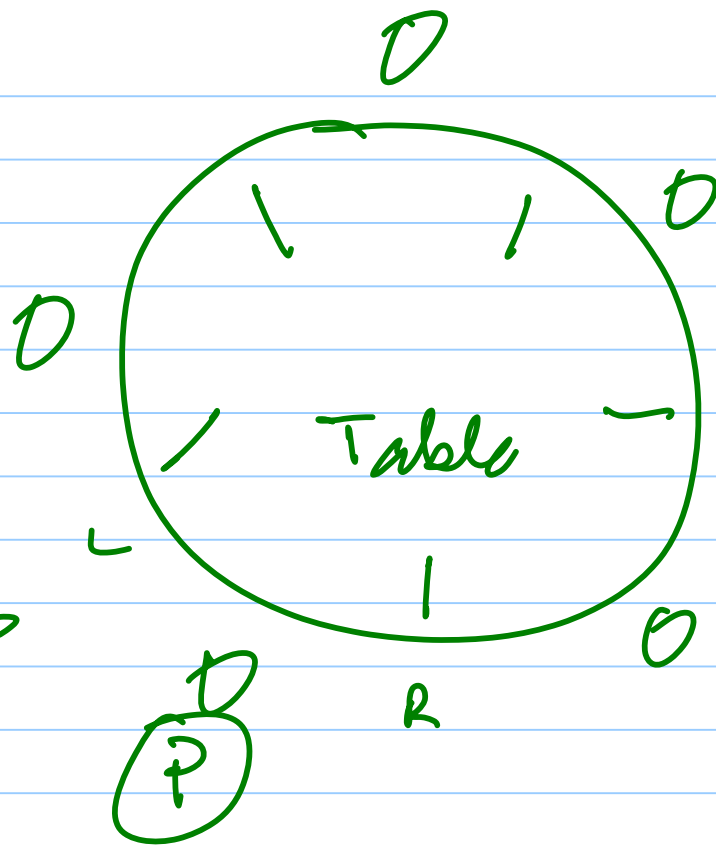
→ Readers & Writers

→

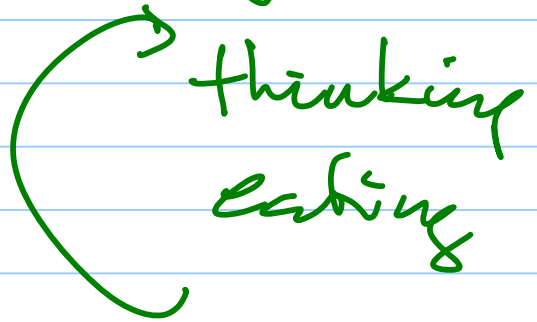
→ Dining Philosophers

→ Resource Allocation Deadlocks ...

5 chopsticks
→ needs two
to eat



5 philosophers



philo [i] $i \rightarrow 0 \ldots 4$

{

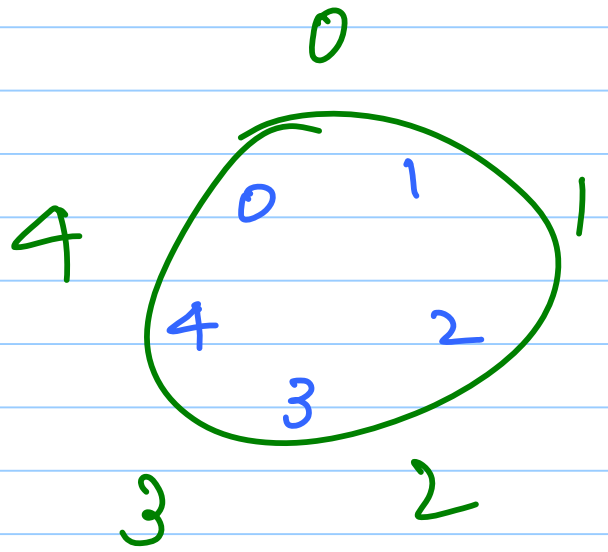
think

→ get chopstics

eat

→ putdown chopstics

}



chopstix → array of 5 semaphores

↑ init to 1

```
Semaphore chp[5]
init chp[0..4] to 1
```

philo[i]

think

P(chp[i])

P(chp[(i+1) % 5])

eat

V(chp[i])

V(chp[(i+1) % 5])

P_0 picks up ch_0 → waiting for ch_1

P_1 " " ch_1 → ch_2

⋮

P_4 " " ch_4 → ch_0

→ deadlock prone

int chp[4] = 1

1 → avail
0 → not avail

X [while(ch[i] & ch[i+1] are not available) ;
chp[i] = 0 ; chp[i+1] = 0 ;]

(mutex = 0) self = 0

sem

P(mutex)

phil[i]

if not (chp[i]==1) && (chp[(i+1)%5] ==1)

while

{ V(mutex)

P(self[i])

P(mutex)

}

chp[i] = 0 ; chp[i+1] = 0 }

V(mutex)

putdown for philosopher i

P(mutex)

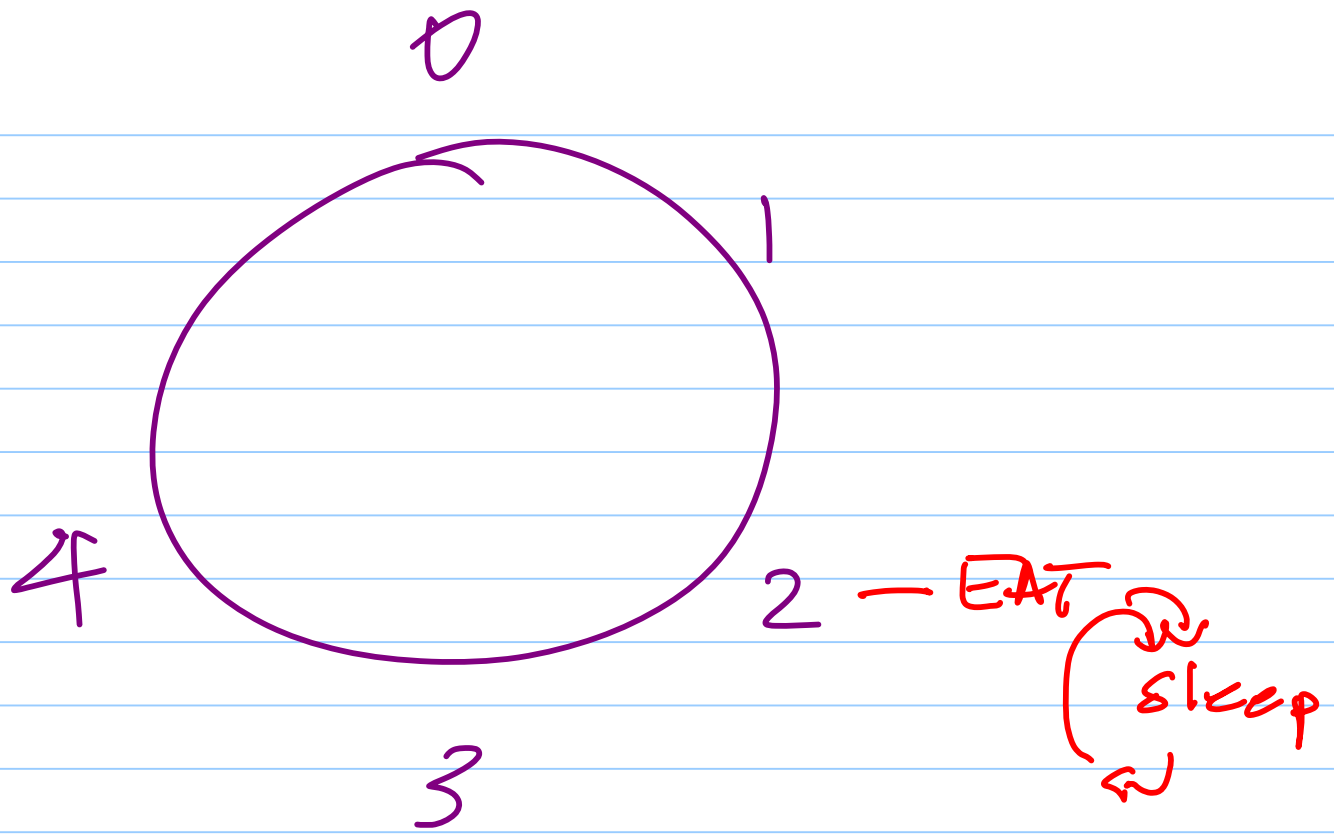
ch[i] = 1; ch[(i+1) % 5] = 1;

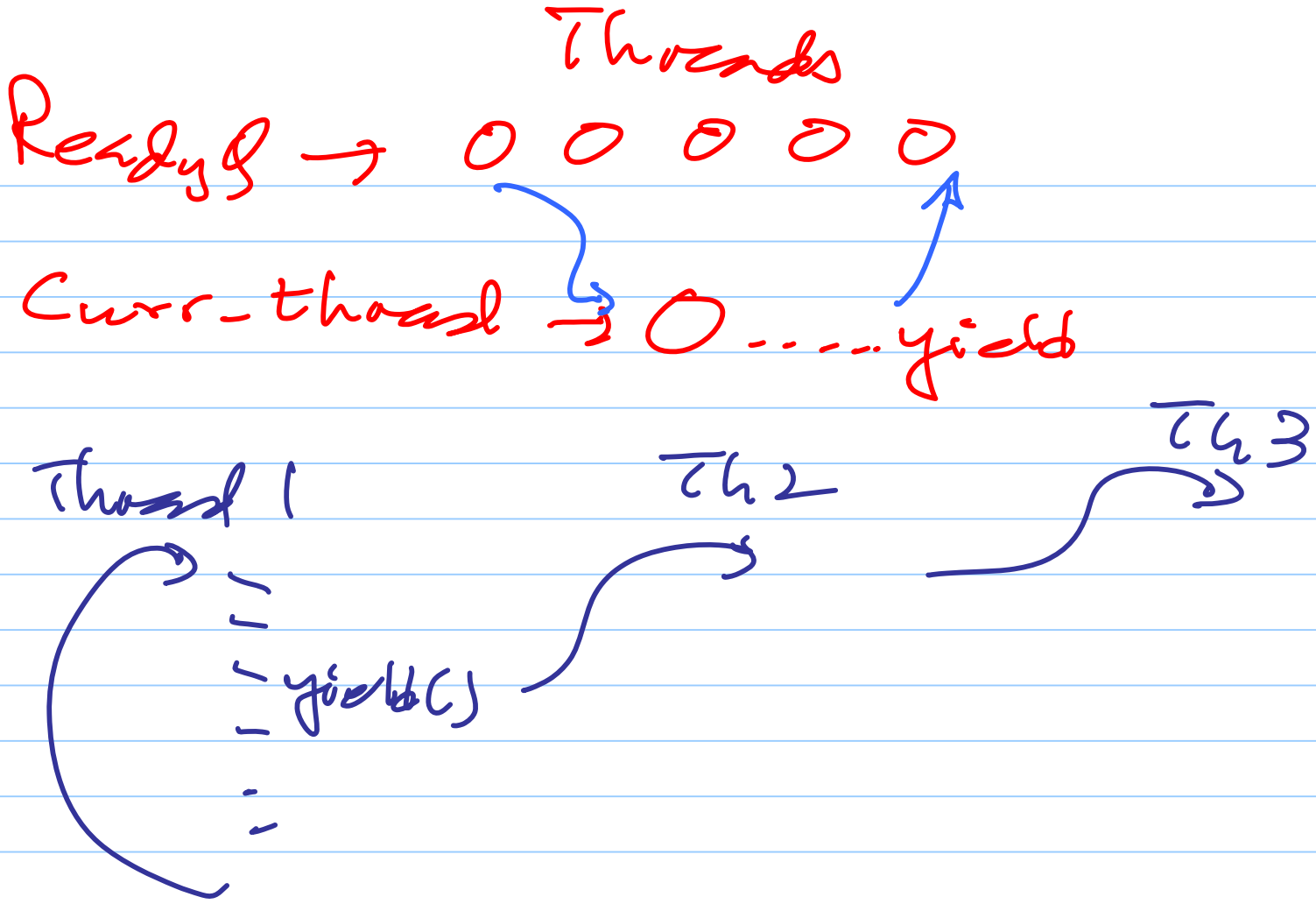
V(self @); V(self[(i+1) % 5])

V(mutex)

~~self[(i-1) % 5]~~

self[(i+4) % 5]


0
1
2 — EAT
sleep
3
4


Threads
Ready Q
Curr_thread
yield
Thread 1
Th 2
Th 3
yield()

Sem_t → struct { int val;
TCB_t *q; }

void InitSem (Sem, value)
int

Sem_t

void P (Sem)

void V (Sem)

```
InitScan(Scan, { Scan.val =
          value)                value; }
```

P(Sem)

```
TCB_t *from;
{ Sem.val--;
  if(Sem.val <= 0) { // block
                    AddQ(Sem.Q, Curr_thread)
from = Curr_thread.
to→ Curr_thread = DelQ(ReadyQ)

  swapcontext(from->context, Curr_thread->
                                context)
}
```

```
V(Sem)
{  Sem.val++;
   if(Sem.val <= 0) {
        AddQ(ReadyQ, DelQ(Sem.Q));
        yield()
   }
}
```

why?

why?

MONITOR → higher level sync / mutex tool

↳ mutex conditions

] structured as a "class"