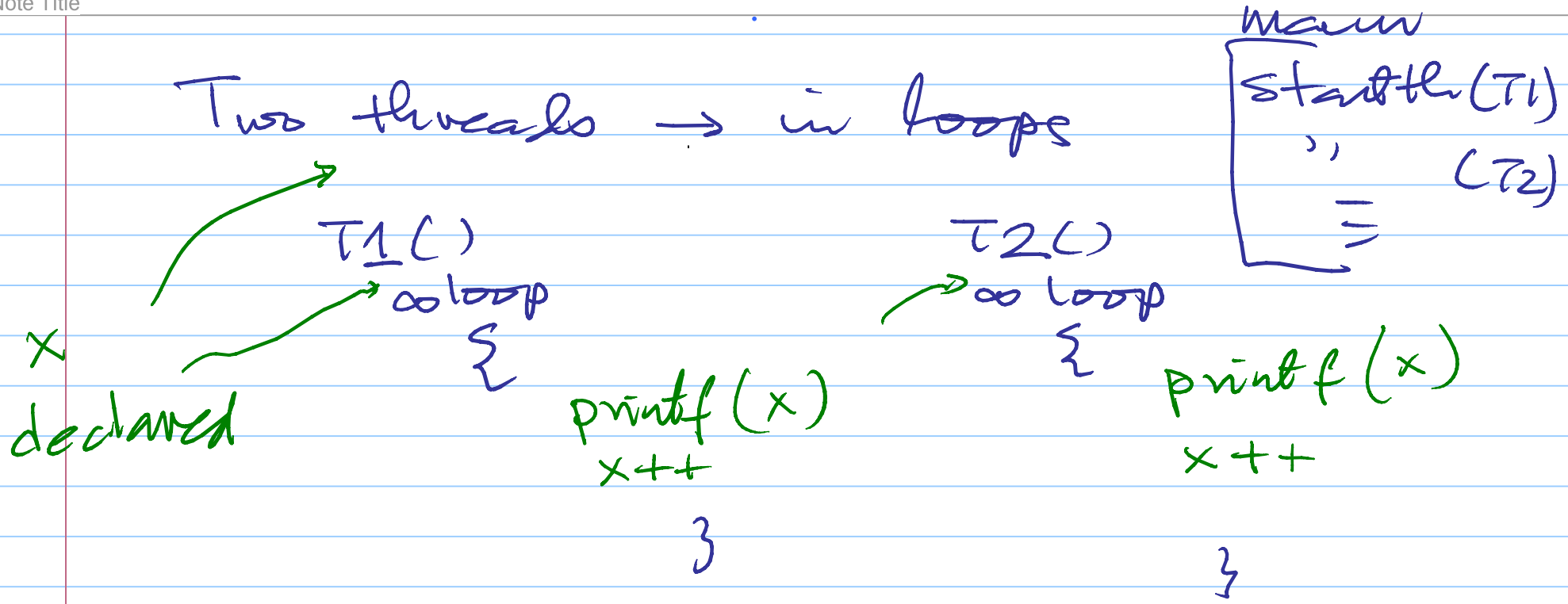
CSE 531: Distributed and Multiprocessor Operating Systems

Fall 2017

Class: 09 Date: 9/14

Note Title



x → global

```
P(S)
print("T1", x)
x++
V(S)
```

```
P(S)
print("T2", x)
x++
V(S)
```

T1 1
T2 2
T1 3
T2 4

T1 1
T1 2
T2 3
T1 4

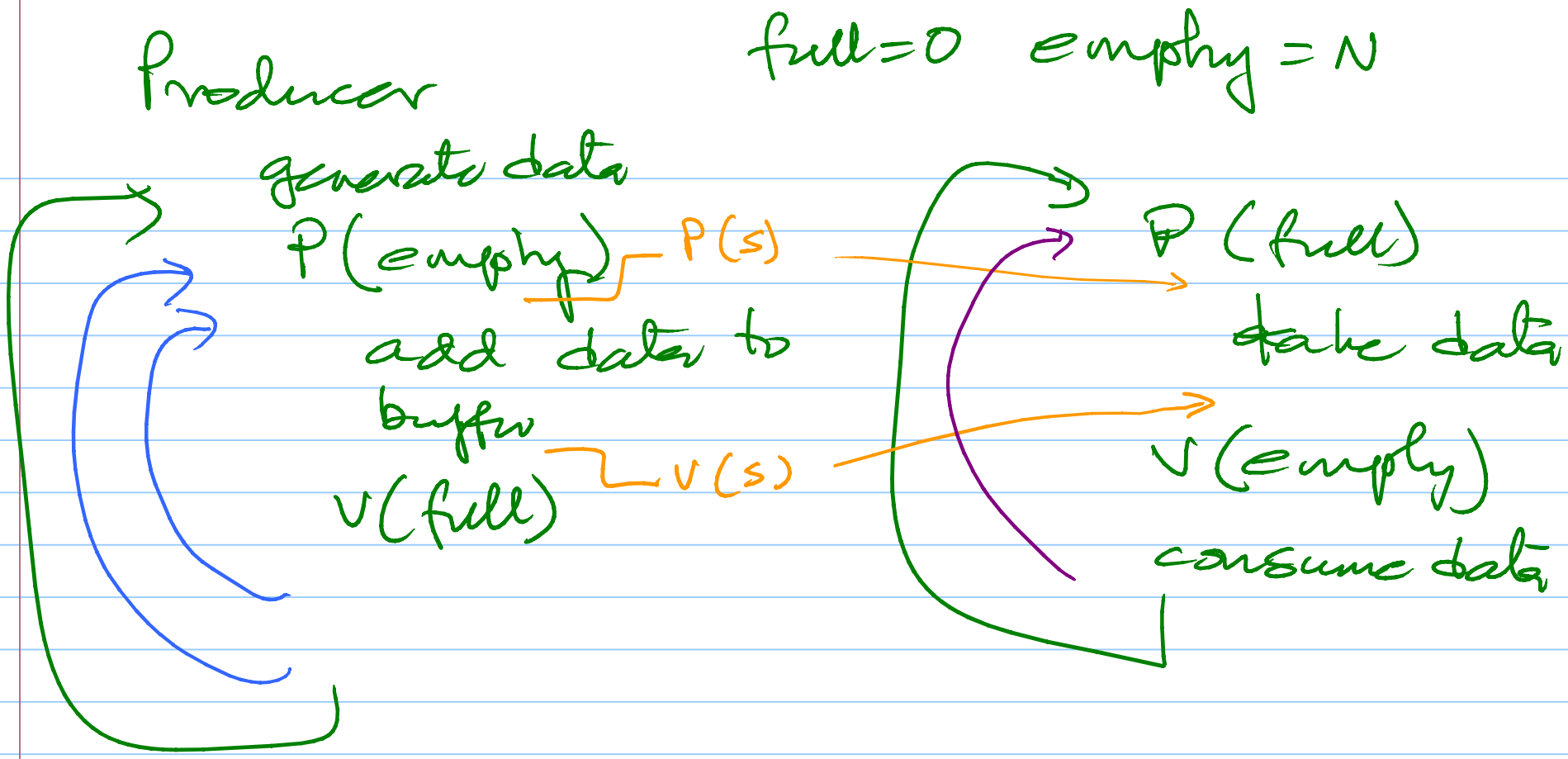
S2 $S \leftarrow 1$ $S_2 \leftarrow 0$

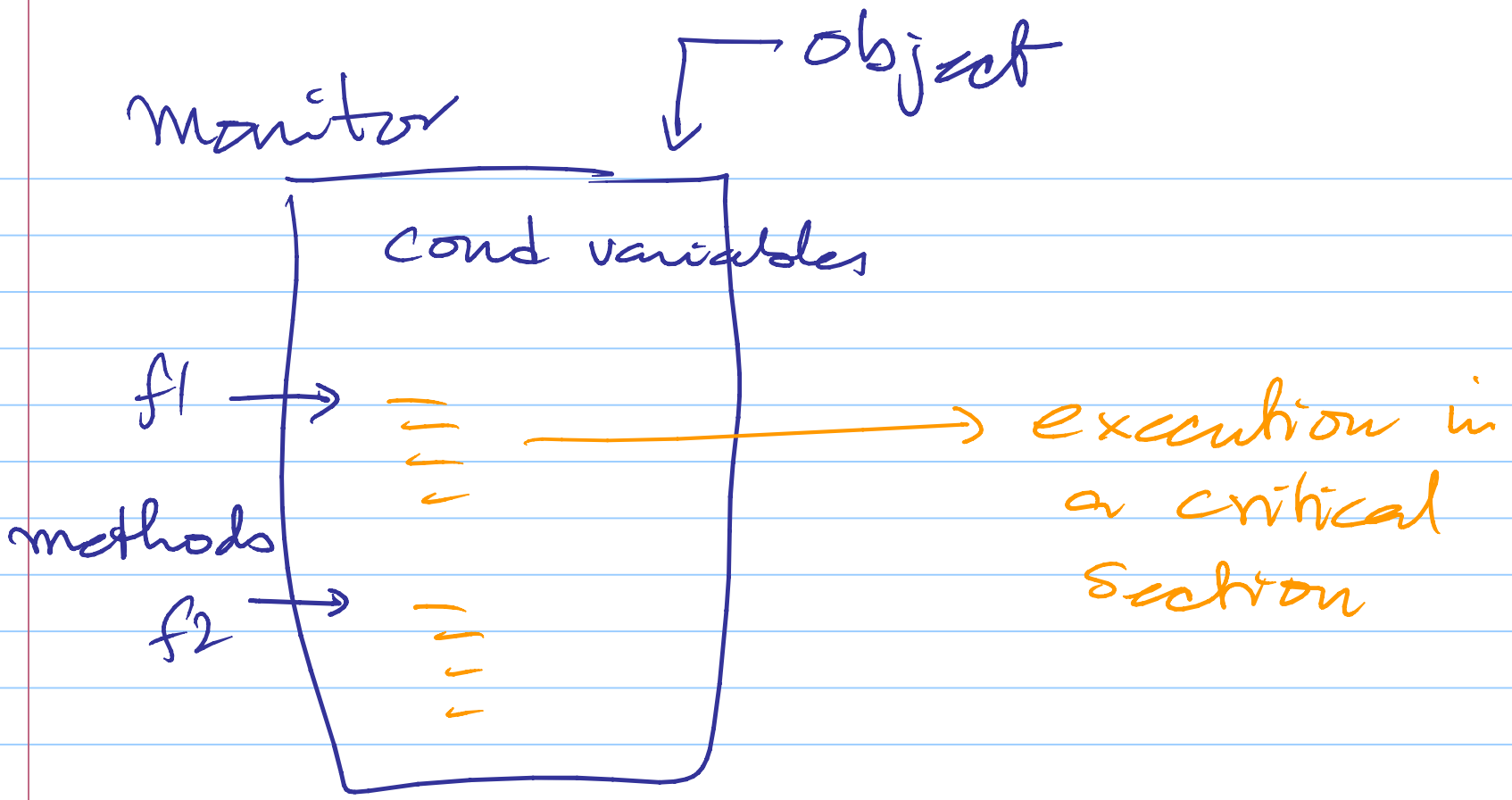
P(S)
print (T1, x)
x++
V(S)
2

P(S)
2
print (T2, x)
x++
V(S)

Producer, Consumer

aka Bounded Buffer


full=0 empty=N
Producer
generate data
P(empty)
P(s)
add data to buffer
V(full)
V(s)
P(full)
take data
V(empty)
consume data

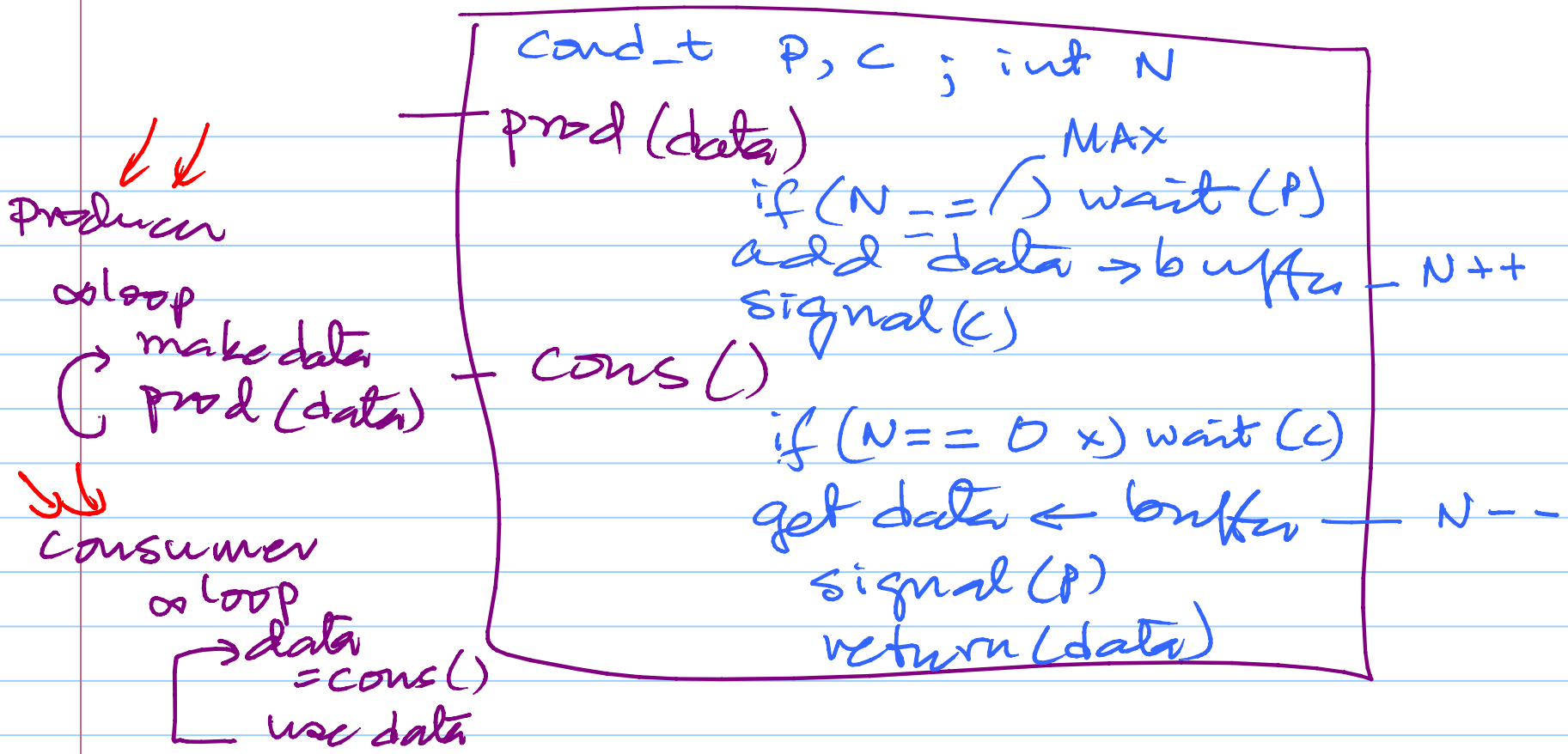

Monitor
object
cond variables
f1
methods
f2
execution in a critical section

in a monitor method

```
f()
  -
  - wait(c)
  -
  -
  - signal(c)
  -
  .
  .
```

wait(c) → block, always
- ↳ unlock monitor
- ↳ lock it when woken

signal(c) → wake up 1 process if any (else, NO_OP)
- ↳ woken up thread waits for me to unlock


cond_t P, C ; int N
prod (data)
MAX
if (N ==) wait (P)
add data → buffer
N++
signal (C)
cons ()
if (N == 0 x) wait (C)
get data ← buffer
N--
signal (P)
return (data)
Producer
∞ loop
make data
prod (data)
Consumer
∞ loop
data
= cons ()
use data

p-threads

types [pthread_mutex_t
" _cond_t

[pthread_mutex_lock (x)
" " unlock (x)

[pthread_cond_wait (c)
" " _ signal (c)

A monitor is a lock variable

A condition is a cond variable

• pthread_mutex_t pcMon

prod (data)

{

monitor

pthread_mutex_lock(pc_mon)

if(N == MAX) pthread_cond_wait (P, pc_mon)

add data → buffer

N++

monitor

conditions

lock

pthread_cond_signal (C)

pthread_mutex_unlock (pc_mon)

}

```
pthread_cond_wait (φ, pc_mon)
{ unlock (pc_mon)
  block on φ
  lock (pc_mon)
}
```

Define S as a struct with a mutex, condition & counter

P(S)

```
mutexlock(S.lock)
S.count--
if S.count == 0 wait(S.cond)
mutex unlock(S.lock)
```

V(S) lock(S.lock), count++, signal(cond)

Readers & writers.

P1 → Read

P2 → "

P3 → waiting to write