OS-III

Thread Synchronisation

1. Producer - Consumer problem
   - Semaphores
   - Basic mutex
2. Concurrent DS
   - Atomic Integer
   - AS atomic integer

- Concurrent Hash maps

Thread synchronisation

- lock - mutex
- Critical section

- only one thread can access the critical section

T1
T2 } waiting
T3

Producer - Consumer

- Dijkstra

Producer — unit of work
Tshirts ip-d.

Consumer →

Con only pick up items
if they are produced

- Bounded - buffer problem

Consumer

if cur_size > 0:

get item

current_size -= 1

Producer

if Cur_size < buffer

Create item

cur_size += 1

2 < 3

P2

P1

rewind

add to the queue

P1

cur = 2

if 2 < 3

P2

cur = 2

push

cur = 3+1

Cur_s = 2+1

2nd

p1

cur_size = 2

if 2 < 3

push

p2

cur_size = 2

if 2 < 3

push

2 + 2 = 4

Consumer

C1 ①

if cus_size > 0 ⇒ T

C1.remove()

C2 ①

if cus_size > 0 ⇒ T

C2.remove()

C1

C2

C3

Кофва

Роббит ма

Task

p

- lock.lock()

lock.lock()

if (size < MAX){

if (size > 0)

if (sig

Add to the q

y

lock.unlock()

Remove

lock.unlock()

→ Synchronisation problem ✓

→ Sequential

Mutex in Java

— synchronized

— Lock - Reentrant lock

① PC problem

② Mutex

5:56 - 6:01

10:26 - 10:31

< 11:50

PC problem → 20 | exceptions

→ sync. issues

→ Thread sync

→ Mutex - synchronized

→ Sequential process

Ideal

1. Producer → Max size

→ 0

Dynamic - on the bus

of size no. of
produces should change

(2) Consumers → 0 - 10
→ # of filled slots

P ⇒ # of empty slots
C ⇒ # of filled slots

(1)

what do we need?

Multiple threads entering the CS

Semaphores

→ ways to synchronize multiple threads

Semaphore(10) → No um my threads

Semaphore(1) = mutex

→ semaphore.acquire()

→ semaphore.release()

Semaphore (0) → 3 processes can be active at the same time

Producer | Consumer

0 objects - 20 - 0

→ 19 ← 1

→ 10 ← 10

0 - 20 - 1

if produce λ

→ signal to the C

Produce

C → produce

Consume

→ Consume

→ signal to consumer

→ send a signal to producer.

P [Semaphore]

while (True) {

① acquire semaphore

② create unit & add to P

③ signal consumer.

}

C [Semaphore]

while (True)

① acquire

② consume

③ send a signal to product.

Semaphore 2 0

(1) Semaphore(19) 0 → 1

↳ SemaphoreConsume(1)

	Sp	Sc
0	20	0
1	19	1

20	0
19	1
18	2

→ decrease own count

→ increase consume count.

How many instances can run at a moment?

$S_p(20)$ — 20 more threads

can be created