CS202 - 001 : OS

Review Session 3

- [x] 0. Record
- [x] 1. Process vs Thread
- [x] 2. What is concurrency?
- [x] 3. Concurrency Commandment
- [x] 4. Sequential Consistency
- [x] 5. C++ Primer
- [ ] 6. Lab 3 overview

---

1. Process vs Thread

process: an instance of a program.

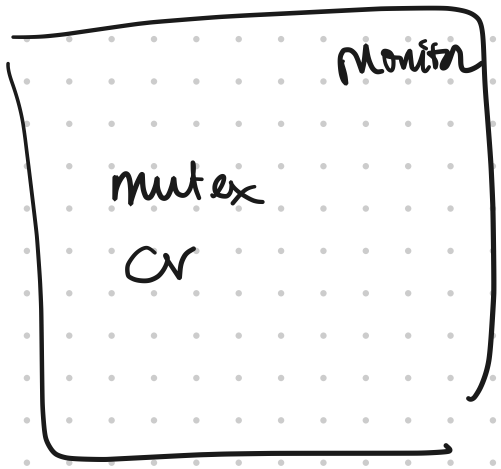
thread: A path of execution. A process can have multiple threads.

- threads:
  - their own registers
  - stack.

2. What is concurrency?

"It's when you have multiple threads"

"It's when you use mutex and CV"

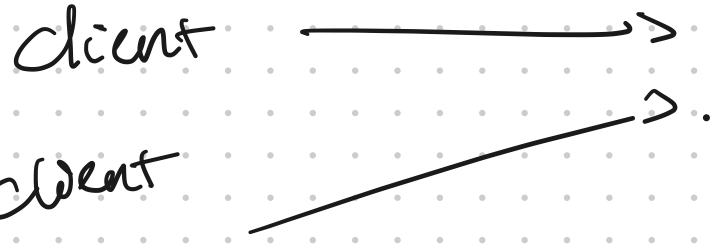


Programming Languages.
by Edward Yang
Thomas Ukis.

Concurrency:

- Things happening at the same time.

Eg:



Parallelism:

- Speed up of tasks by splitting workload into multiple cores.

Event loop:

- single threaded
- mimicking multiple things happening at the same time.

---

3. Concurrency Commandments

Rule 1: acquire / release at beginning/end.

Rule 2: a thread in wait() must be prepared to restart

```
while (cond):
    wait (&cv, &mutex).
```

Rule 3: Hold lock when doing CV operations.

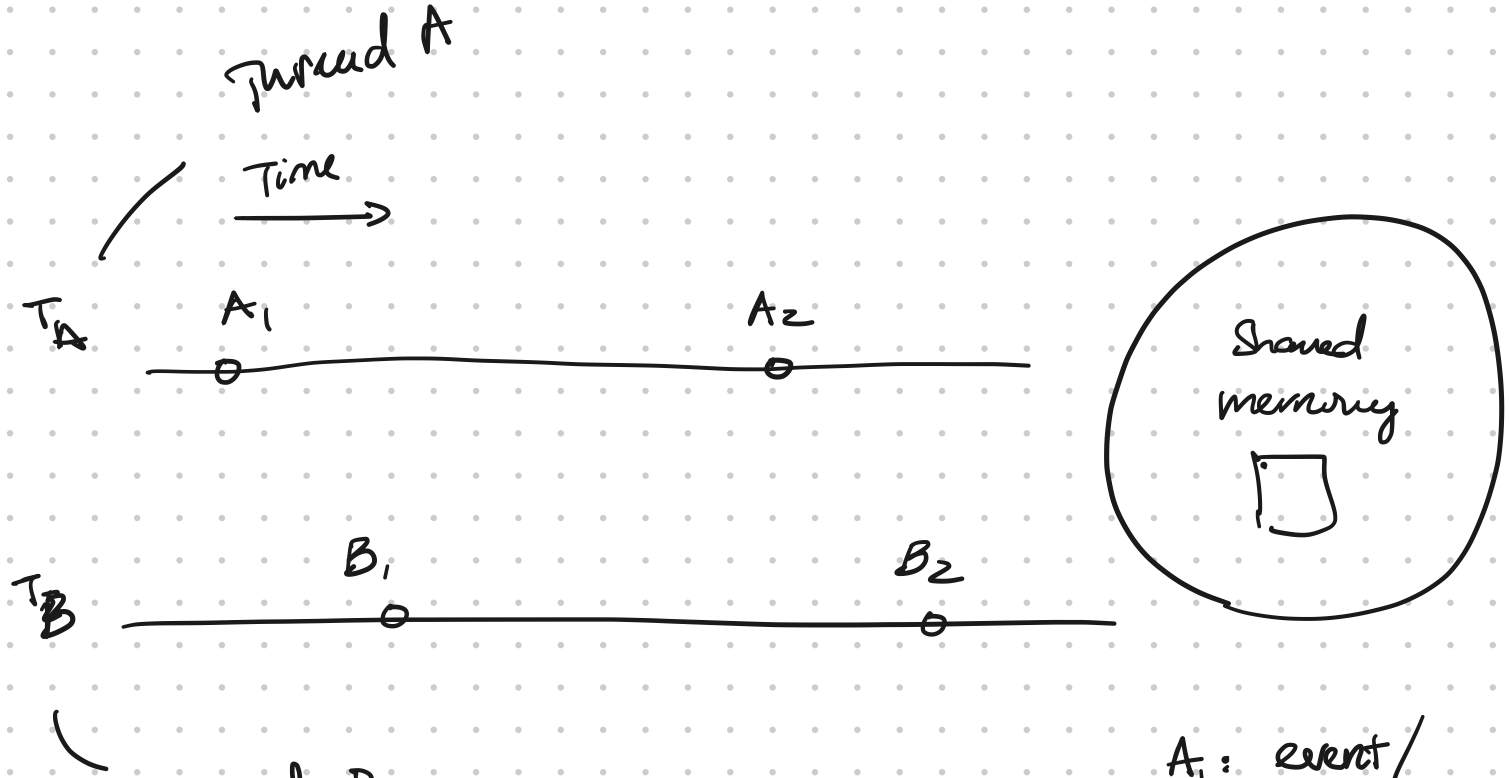
signal() $\xrightarrow{\text{Replace}}$ broadcast()

broadcast() $\not\longrightarrow$ signal().

4. Sequential Consistency

"Informally, sequential consistency implies that the operations appear to take place in some total order, & that this order is consistent with the order of operations on each individual process."

- Jepsen -



A_1: event / command

Order:

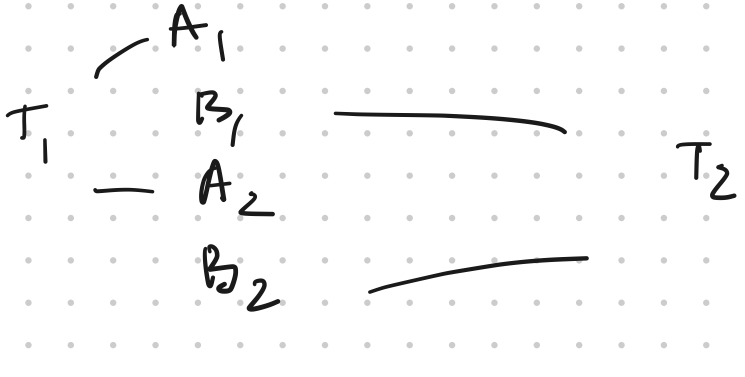
A_1 | A_2 } -> invalid.

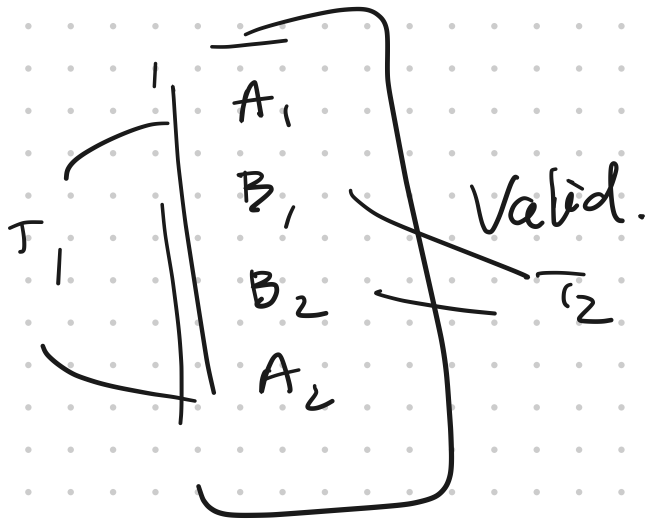
A_1

T_1 A_2 | Valid.

T_2 B_1, B_2

B_1, B_2 $\}$ T_2 → valid





A_1
B_2 → Invalid
A_2
B_1

---

5. C++ Primer

a, Destructor

TaskQueue : ~TaskQueue () opposite of constructor

- free resources
- destroy mutex

b, free : free (void * ptr).

- equivalent is delete.

delete pointer_to_queue

c, printing:

printf (...)

std :: cout <<

---

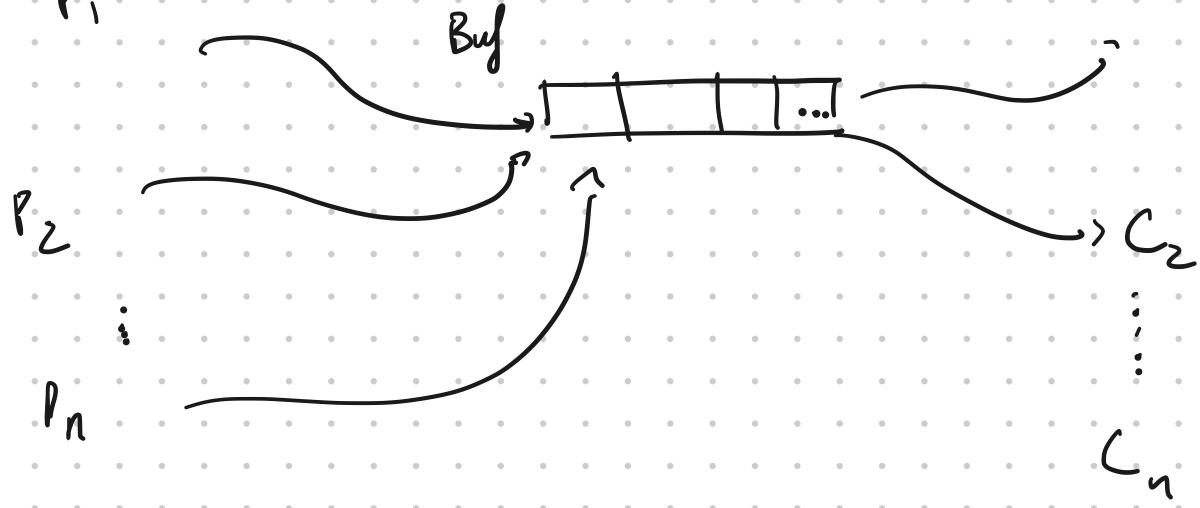
6. Lab 3 Overview

P : Producer

C : Consumer

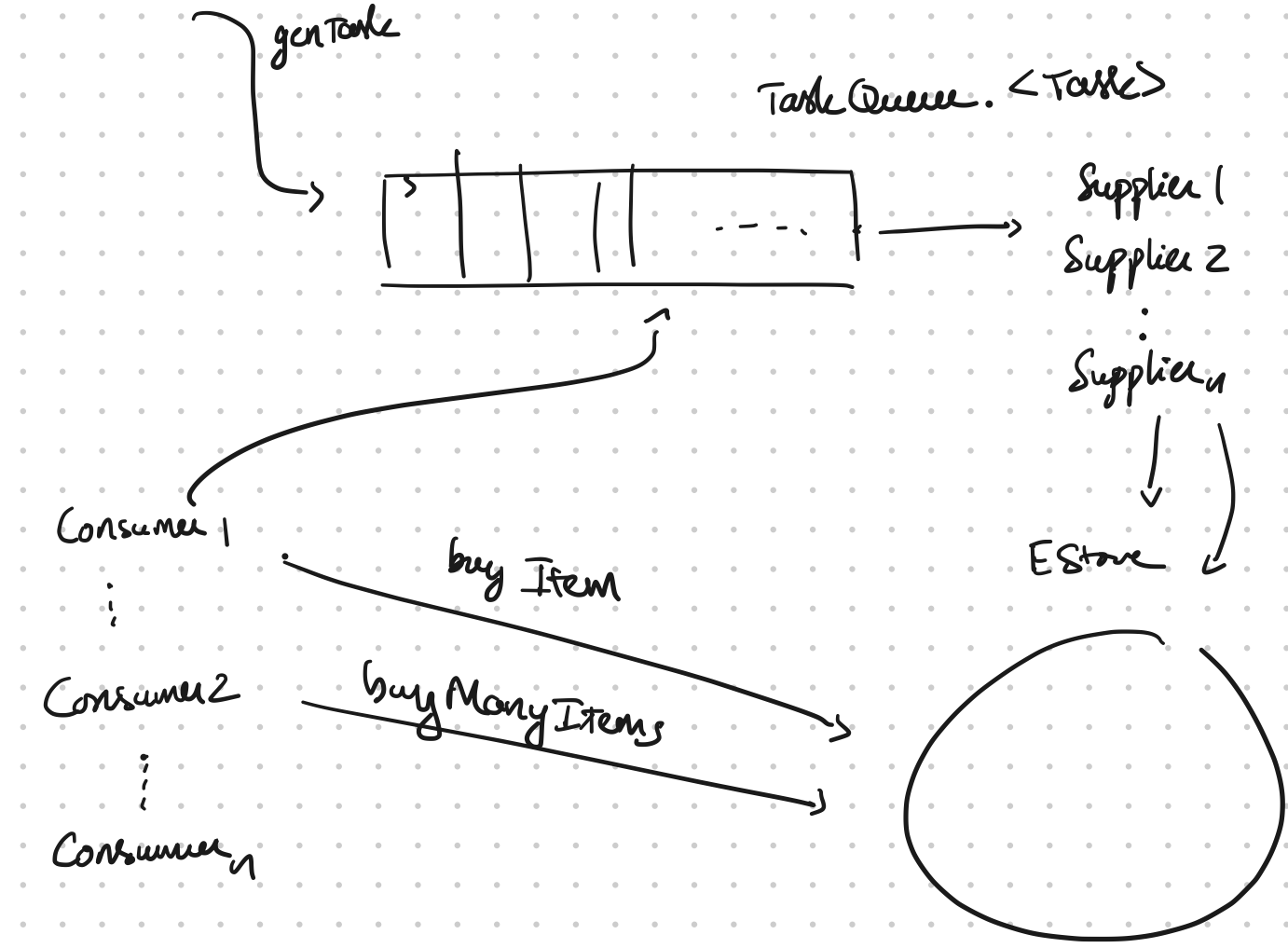
Producer - Consumer Arch

C1

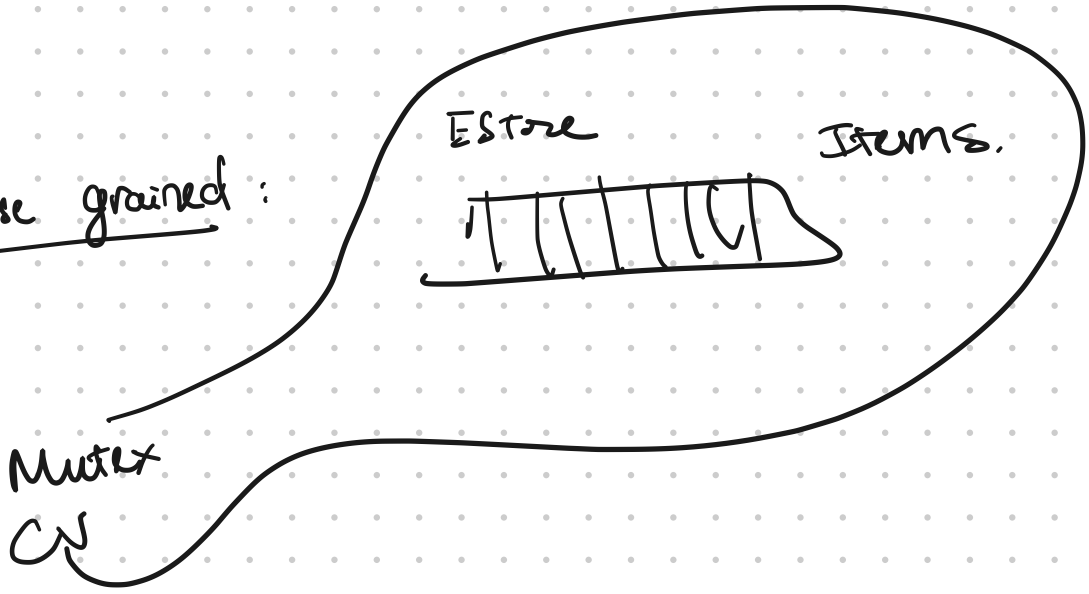


Lab 3 :

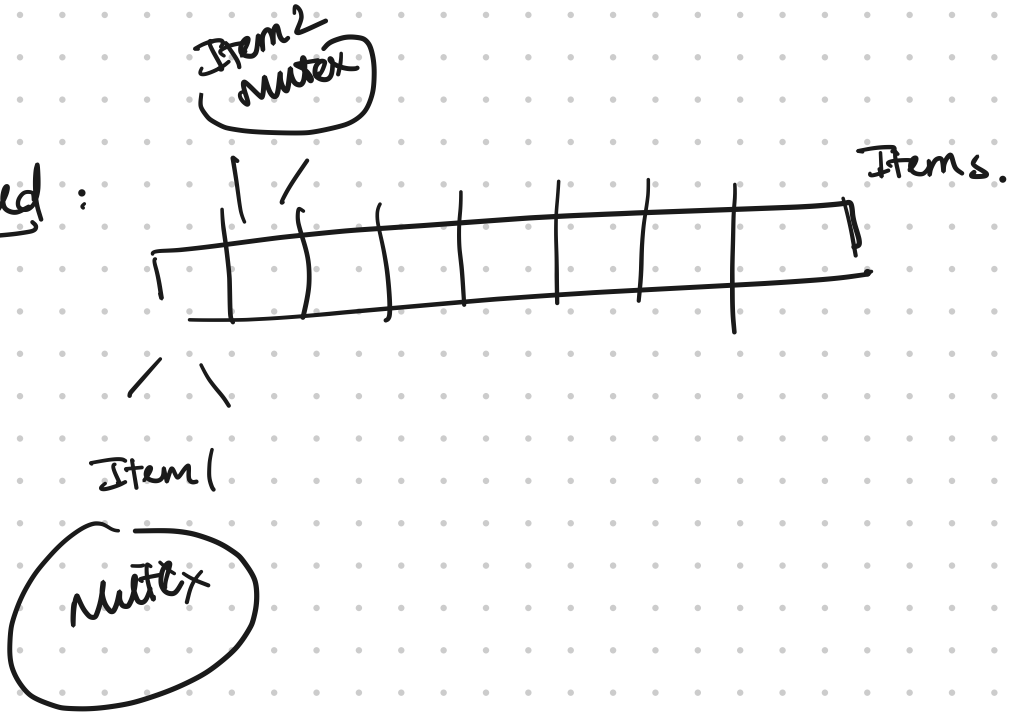
* Generator



Coarse grained :



Fine grained :



Pit Fall :

- Forgot to free resources — destroy mutex destroy cv.
- Forgot to release the mutex — never call release()
- Forgot to provide destructor
- Store value in variable w/o recomputing

```
int temporary = a+b
while (temporary == 15)
    wait(---)
;
```

Shared memory:

a = 5;

b = 10;