Concurrency ①

concurrent
- inter-leave running at one core
- look like running at the same time

Asynchronous
- Independent to each other
- I don't care when you running. do not depend upon the time

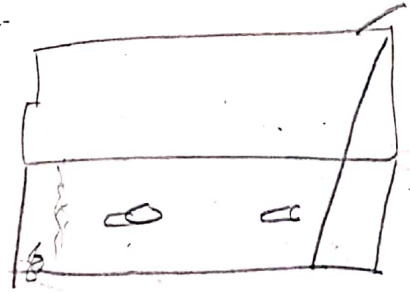
Independent
- They will have minimum depta

Synchronous
- Depend upon the time
- If will wait for you to finish something and you will wait for me to finish something.

Parallel
- process run at the same time on the separate processors

process ko span karna

what happen when things occour in parallel?
→ well, fish die!



You
Ask Mom if fish have been fed
if not:
- Get the food
- Feed the fish
- Tell Mom!

Brother
Ask Mom if fish have been fed
if not:
- get the food
- Feed the fish
- Tell Mom

Brother come

Brother case

→ If we run the process run interleave then both brother will food the fish and then fish will die

Ask Mom fish have bee fed
If no
↳ get food
↳ Feed fish
↳ Tell Mom

Ask Mom fish have been Feed
If not:
↳ get food
↳ Feed to food
↳ Tell Mom

→ If you run in parallel then fish will save phely ik Bhai sara kaam karay then Dusra start karay.

TOU
If light is red:
feed fish
turn light green

Brothers
If light is red:
↳ Feed fish
↳ turn light green

Bhai ko chapair Marni Hai

Race condition
↳ is a bug
↳ work when specific sequence of events..

Example:
↳ current Bank Balance = $1,000
↳ withdraw $500 from ATM while $5000 direct deposit is coming In

withdrawal
① ↳ Read Account Balance
↳ subtract 500
↳ write Account Balance.

only withdrawal = 500

Deposit
② ↳ Read Account Balance
↳ Add 5000
↳ write Account Balance

only Deposit = 5000

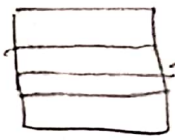
Synchronization

↳ To get expected Result
↳ Overcome to un-expected Results

$x = x + 1$

May have have race condition

```
mov eax, [esi]
inc eax,
mov [esi], eax
```



This is not thread-safe... because it has race condition

Mutual Exclusion

Critical Section
↳ where race condition can be rise.

Mutual Exclusion
↳ only one thread enter/work in Critical section

Lock
↳ When you enter in critical section then you need to key (Door lock) when till you have key of lock then neither another person ~~of~~ not enter there

For understanding lock
↳

Acquire(fish lock)

You
fish case

Release(lock)

Brother
fish case

You need to do
↳ Acquire the lock
↳ Release the lock

~~Now we pro~~
Now we solved the problem but within have more problems
↳ What is acquire lock / Release lock??
↳ is it function
↳ is it code

The critical section problem
↳ Condition for solution:
↳ Mutual exclusion
↳ progress (kuch nahi kuch to raha ho)
↳ Bounding waiting
↳ No threads running outside the critical section.

If preemption is yes then face problem
↳ If not Preemption then no will face problem.

Solution #01
↳ Disable Interupts
If you have one core But not Solution in for Multiple Cores.
↳ You are to much control on the Application
If thread is dead After Disable the Interrupts

Solution #02

Software Test and Set locks

```
While (locked) ;
locked= 1
% Critical section %
locked = 0
```

→ Busy Waiting
→ Here this not Solved the problem because there running race Condition outside the Critical region.

Busy Waiting
↓
Busy Asking

```
While(locked)
"busy waiting"
```

Solution #3 : lockstep Synchronization

lets turn the Hardware

Atomic Instruction
↳ granted to Indivisible
↳ not coming preemption
↳ race condition can be come
↳ CPU instruction

```
int test and set (int *x) {
  last value = *x ;
  *x = 1;
  return last value
}
```

↳ Test and set
↳ compare and Swap
↳ Fetch and Increment

```
While (test-and-set(&lo
  % do critical
    Section %
  lock = 0;
```

Atomic ← only ← differ
↳ But have ↳ Busy waiting → spin lock

Test-and-set ✓✓

Fetch and increment → X

Compare and swap → X

→ because relay on the busy waiting that's why it's not Important.
↳ waste cpu cycles

spins are not greed

Semaphores → IUP
↳ Just variable can do Two operation
↳ down, up operations → Atomic
↳ if =0
↳ Already zero
↳ go to sleep

```
down (sem s)
    if (s > 0)
        s = s - 1
    else
        sleep on event q
    }

up (sem s) {
    if (someone is waiting on s):
        wake up one of the threads
    else
        s = s + 1
```

T1 → down(s) CR → up(s)

T2 → down(s) CR → up(s)

S [0]

Binary semaphor ⟹ (Mutex)

↓ store the binary value

And problem
Polling:
↳ it waste cpu cycle just asking the data to producer.

empty [5] producer:
full [0] up(full).