Inter Thread Communication

→ used to allow Synchronized threads to communicate.

one thread can access the monitor/CPU at a time. - while other threads will be in suspended state.

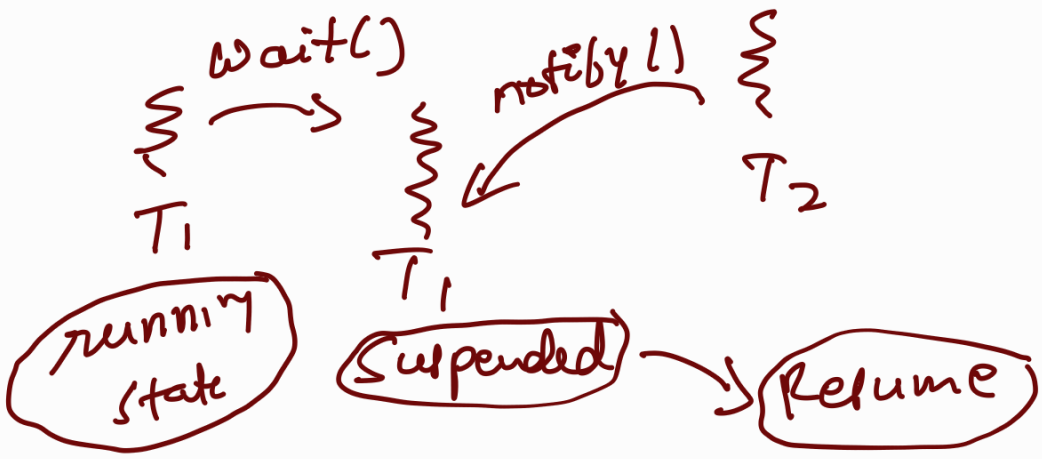
Three methods

wait()
causes current thread to release lock & wait until

- Specific time in milisec
- other thread calls notify() notifyAll()

notify()
wakes up single thread that is waiting on this object's monitor

notifyAll()
wakes up all the threads who are in wait() state.



Syntax

public void wait() throws InterruptedException

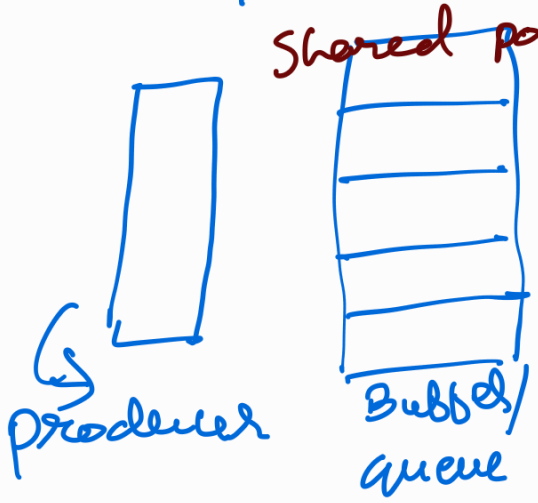
advised to call inside try-catch Block

public void notify()

public void notifyAll()

Ex: "Producer - Consumer problem"

↳ problem is multi-process Synchronize



producer generates items and saves in queue.

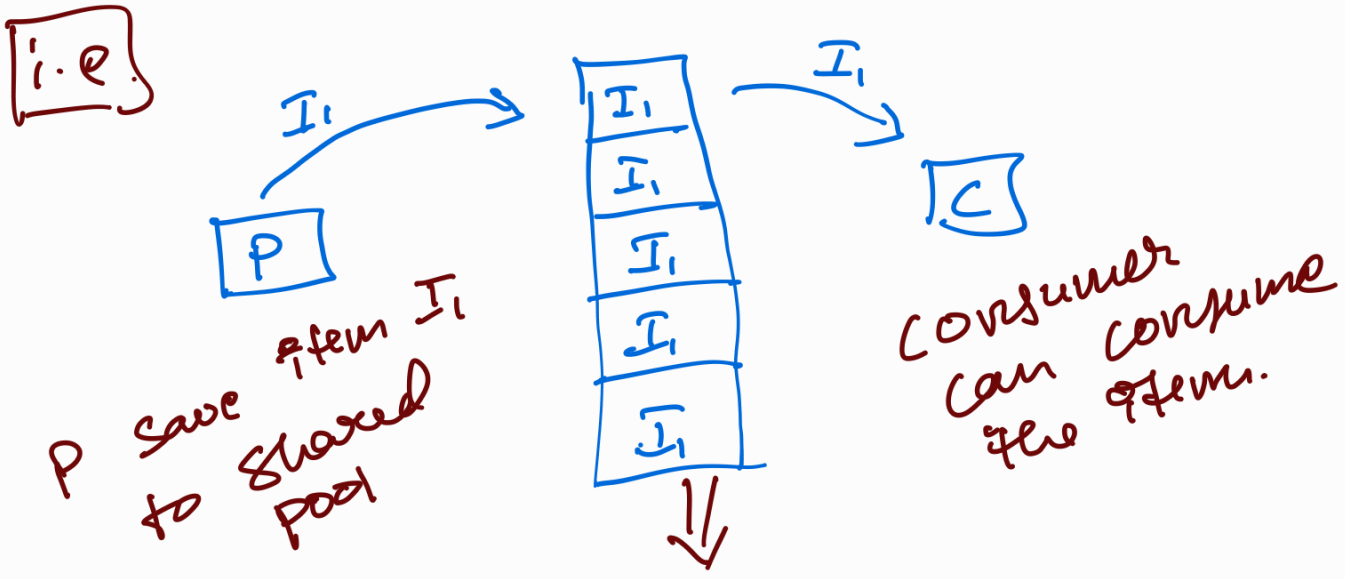
Consumer can access these items

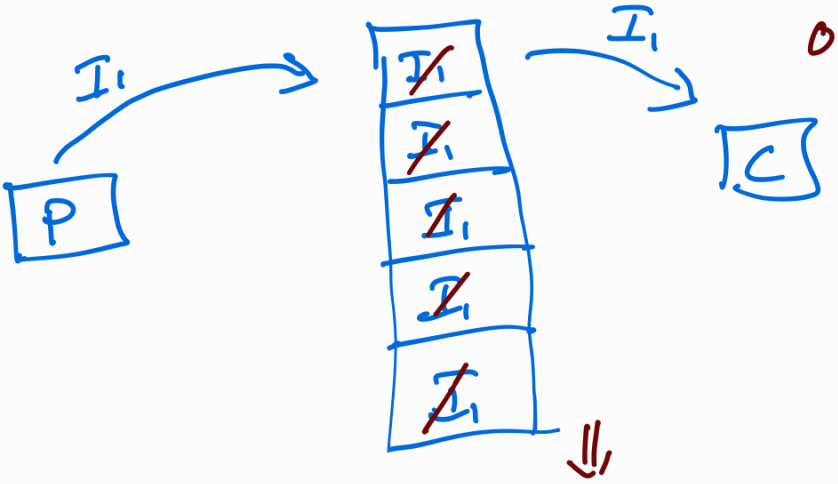
What is the problem?

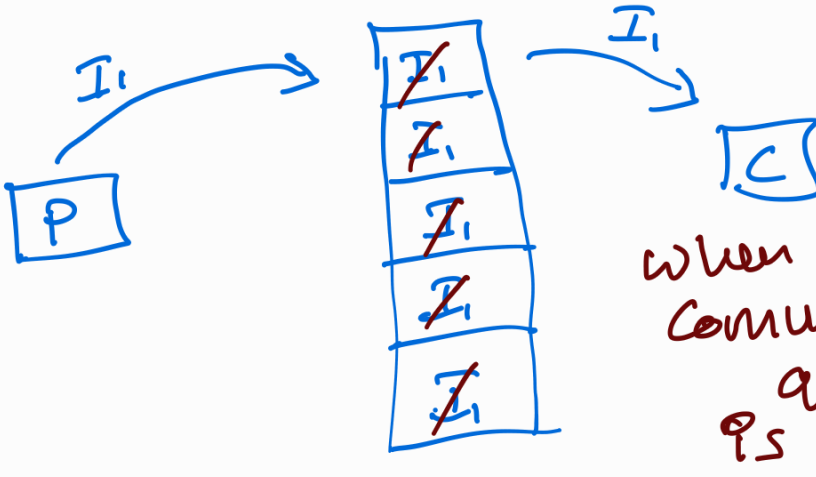
1) Producer can't add data into queue (Buffer) if it is full.

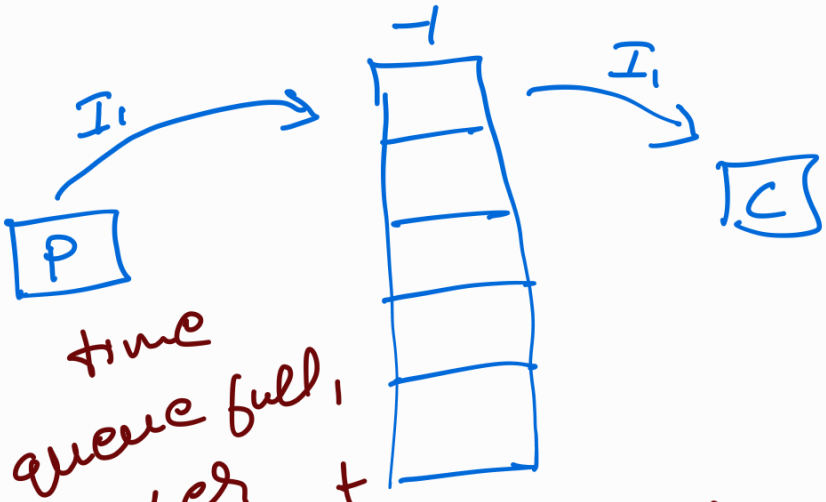
2) Consumer wait consume data if it is empty.

i.e.









But yet consumer tries to consume.

Same time on queue full, producer tries to add item by incrementing (n+1)

⇓

overflow state

So to solve these issue, we use [Inter-thread Comm]

⇓

In Producer % we add [wait()]

↙

till when Producer has to wait, unless there is a notify() sent by consumer.

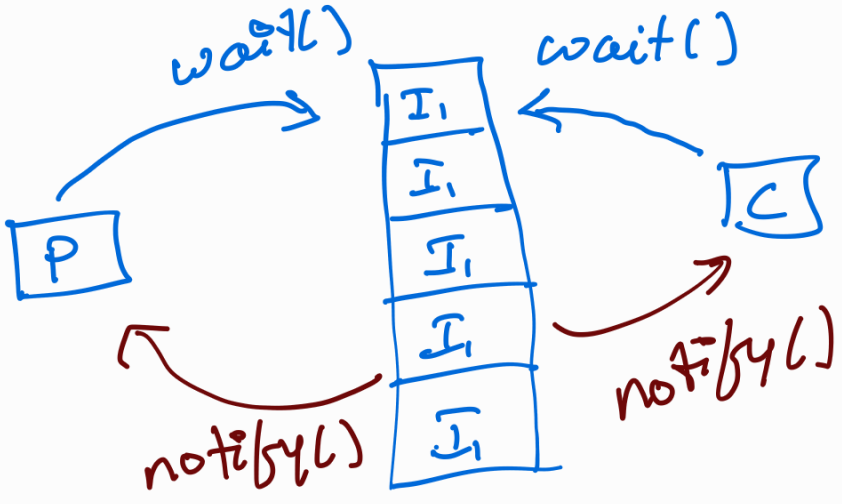
⇓

only on queue empty, [P] can start saving item to queue.

For this wait() is required with corresponding notify()

⇓

Here at C also a wait() is used to tell C to wait unless next notify() is sent by P



Ex: Pseudo code
⇓

```
Class Item
{ int i ;
  boolean produced = false ;
  public void Synchronized Produce(int x)
  { if (produced)
      { try { wait(); }
        catch { }
      }
      i = x ;
      produced = true ;
      notify();
  }

  consume( )
  { if (! produced)
      { try {wait();}
        catch { }
      }
      produced = False ;
      notify();
  }
```

Thread 1

Thread 2