Concurrency in Rust

- Race condition: thread access data

Race Conditions

```
     T1                          T2
      ----------------           ------------------------
1.    Read Acct ($100)
2.                                   Read Acct ($100)
3.    Write NewAmt($90)[take 10]
4.                                   Write NewAmt($50)[take 50]
5.                                   End
6.    End
```

- The 10$ transaction is lost forever
- T2 should have read $90 but since they are sharing the resource and have no idea what other thread is doing.
- To avoid this , introduce locking

DeadLocks

```
      T1            T2
     -------       --------
  1.  Lock(x)             - ok , x is locked and only T1 can use it.
  2.                Lock(y)        - ok, y is locked and only T2 can use it
  3.  Write x=1           - ok , T1 mutated x
  4.                Write y=19     - ok, T2 mutated y
  5.  Lock(y)                 - T2 is currently locking Y so waiting for T2 to unlock it.
  6.  Write y=x+1             - T1 is halted because , it's still waiting
  7.                Lock(x)           - T1 is currently locking X so waiting for T1 to unl
  8.                Write x=2      - Alas T1 is halted and T2 is halted
  9.  Unlock(x)
  10.               Unlock(x)
  11. Unlock(y)
  12.               Unlock(y)
```

- T1 will keep on waiting on 5th step
- T2 will keep start to wait on 7th step as it can't acquire lock for X as T1 has the access for X

What is Runtime

By runtime it is understood that the code that is included by the language in every binary. Every language that is not assembly will have some runtime code.

Green Threads: Virtual threads that are scheduled by a runtime library or virtual machine instead of natively by the underlying operating system. Managed in

User Space and not on kernel space. Also known as M:N threading(hybrid threading)

1:1 threading(Kernel level threading): Kernel handles it after program maps to a scheduled thread.

N:1 : all threads that are created by a user in a application are usually mapped to a single scheduled kernel thread.