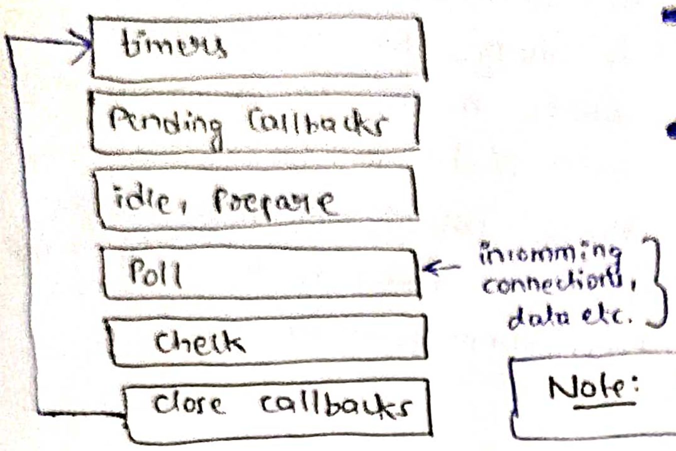
Episode-10 - Thread Pool

Tick → one cycle of the event loop is known as one Tick.

There are two more phases in the event loop →

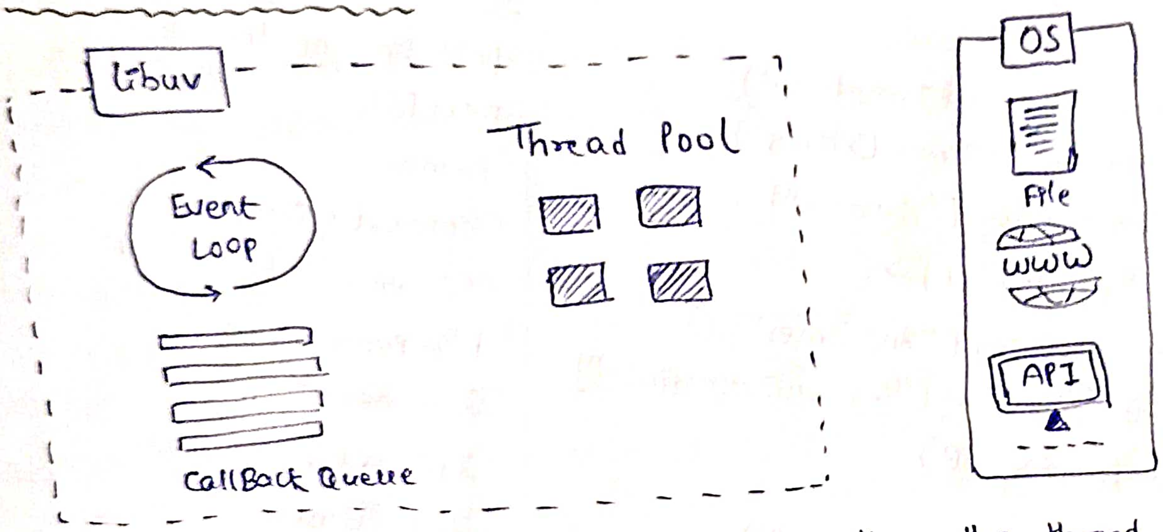


most important phase → Poll

- Pending Callbacks : executes I/O callbacks deferred to the Next loop iteration.
- idle, prepare : only used internally.

Note: location of Event loop in github → Libuv/src/unix/core.

Thread Pool



- The tasks which are offloaded to Libuv, then the thread pool will be checked and any available thread will be occupied for this task until the task is completed.
- many no. of same task can occupy more than one threads that is threads are not reserved for a particular type of Task only.
- in NodeJS, size of Threadpool is 4 threads by default.

$\boxed{\text{uv\_ThreadPool\_Size = 4}}$

- When a task is there, it occupies the thread, use it and the vacant the thread for another tasks which are pending.

Example of some functions which are offloaded to Libuv.

- fs
- dns.lookup
- crypto
- user specified input.

Que:- Is NodeJS single Threaded or Multi-Threaded?
In case of synchronous code → JS uses single thread only i.e main thread.
In case of Asynchronous code → Then JS uses the libuv's Thread pool where 4 (default) threads are available.
Hence it depends on the code,

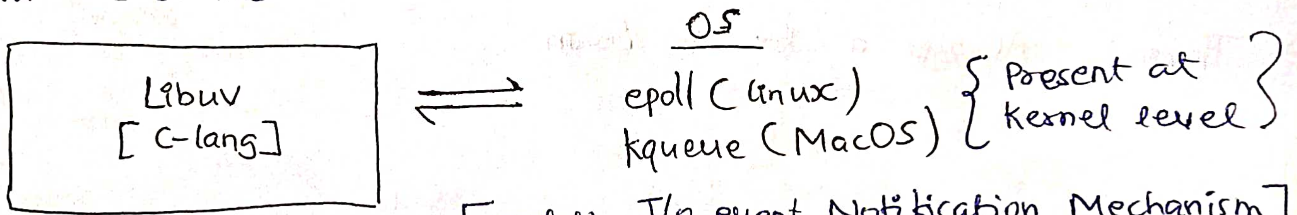
Que:- Can we change the size of ThreadPool?
Yes - By using the variable → $\boxed{\text{process.env.UV\_THREADPOOL\_SIZE}}$ = N
Hence no. of threads can be increased when congestion is happening that is in case of large No. of asynchronous Tasks. Maximum size of Threadpool is 1024 Threads.

Thread Per Connection Model :→

All the Networking happens on <u>sockets</u>. There are different sockets.
A socket is needed for connection in case of Incoming Requests.
Each socket has a socket descriptor (Also called file descriptor)(fds)

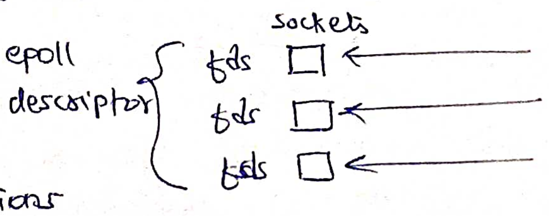
- In case of writing operation (when connection is made via socket), the thread is occupied and blocked (writing is blocking operation), hence we cannot do anything else on this thread.
- Hence multiple threads will be needed for multiple connections according to no. of users incoming.
- For each concurrent requests, each of them are assigned their own Threads.
- This Threads Per Connection Model is not preferred as so many threads will be Blocked

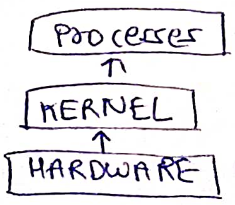
<u>Solution</u> →



[scalable I/o event Notification Mechanism]

- Both epoll and kqueue are same type of Algo. just the platform differs.
- in case of multiple connections, each of sockets has a fds (file descriptor)
- epoll descriptor is a collection of fds and one epoll descriptor handle multiple connections
- epoll → Notification management system, as soon as any activity happens on any of connection, it notifies libuv Then Libuv takes cares of it.
- Libuv interacts with epoll in OS.





- When we create a web-server, we open a socket and listens onto it. Now anybody can make a connection with this server.
- While Accepting the connection, we need to deal with socket descriptors, epoll and all related mechanisms.

Homework → Read about → ① fds, socket descriptor ② Event Emitter ③ Pipes
④ Stream And Buffer

- The epoll and kqueue handles Asynchronous I/O at operating system level.

Important Learnings / Tips

⇒ "Dont Block the Main Thread"
- do not use sync methods
- do not use complex regex
- do not use Heavy json objects
- do not use complex calculation / loops.

⇒ Data Structures is Important
- epoll uses Red Black Tree data structure.
- Timers queue uses MinHeap. data structure

⇒ Naming is Very Important

<u>example</u> process.nextTick() v/s setImmediate → doesn't happen immediately

↳ executes more immediately
↳ happens not in next Tick / cycle

- The names should be swapped but breakages may occur as they are artifacts of past and are present in many packages on npm.

⇒ There's Always a lot to Learn.