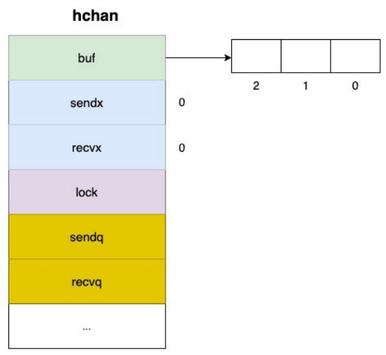
GO Channels

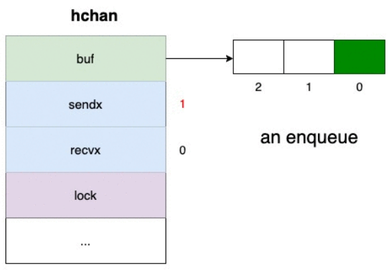
1. Init Channel with make

ch := make(chan int, 3)

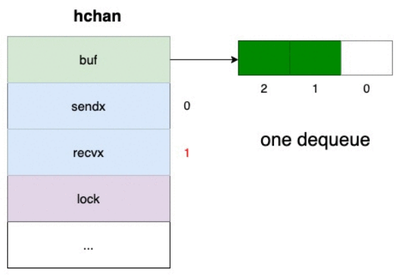


- `buf` is a pointer to an array, which maintains a **circular queue**
- `sendx` is the index of the sent element in the array
- `recvx` is the index of the received element in the array
- `lock` ensures that the reading and writing of the channel is an atomic operation
- **recvq** stores the blocked goroutines while trying to read data on the channel.
- **sendq** stores the blocked goroutines while trying to send data from the channel.

SEND DATA

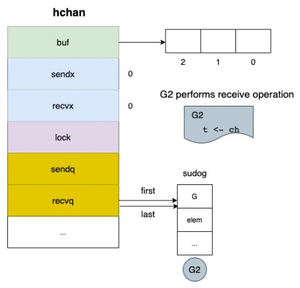


RECEIVING DATA



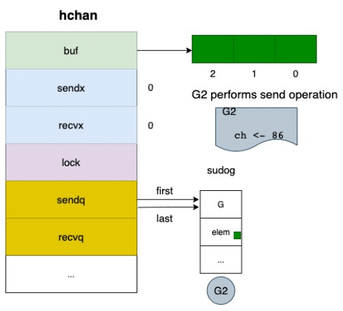
Receives From An Empty Channel

When the channel is empty, a receive operation leads to the blocking of the current goroutine. All the blocked goroutines are stored inside the `recvq` queue.



Sends on A Full Channel

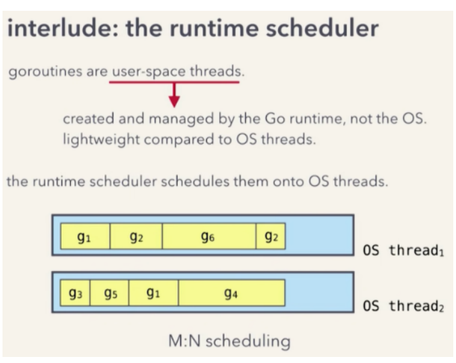
When the channel is full, the next send operations block their respective goroutines. All the blocked goroutines are stored inside `sendq` queue.

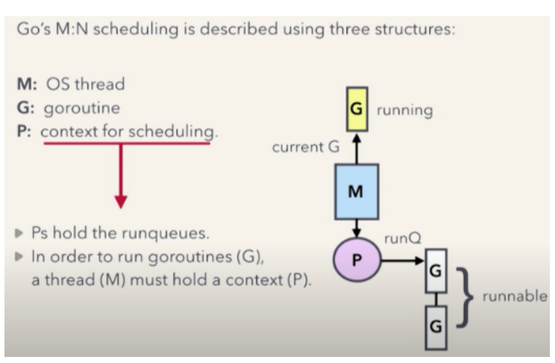


When Any blocking call trigger, how the go-routines are paused/resumed with the help of go-scheduler.

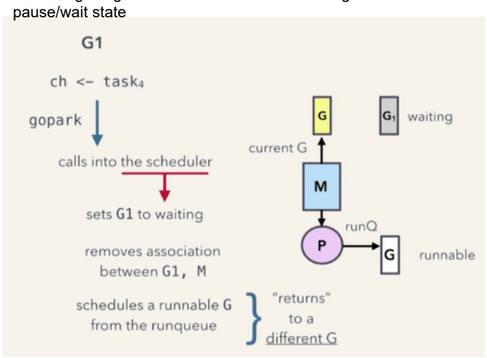
With M:N Scheduling patterns helps to achieve this.

Actions like gopark & goready are main building blocks here.





Removing G1 go-routine from active and making as pause/wait state



Resuming G1 go-routine from in-active and making as current state(including/scheduling in **runQueue**)