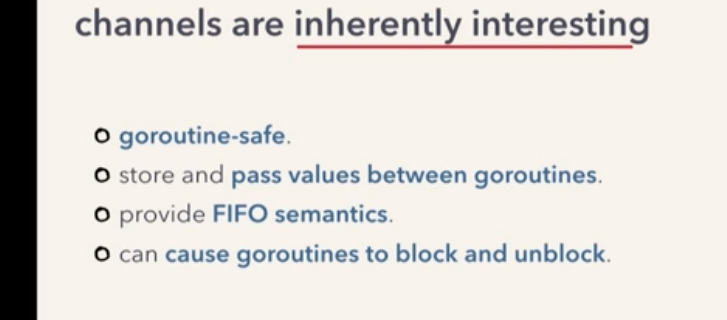
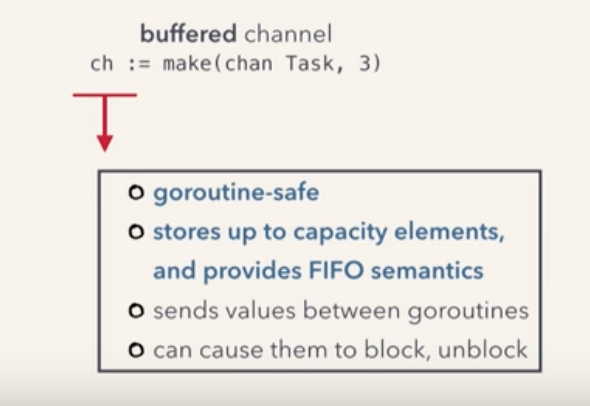
* For communication between go routines.
* Properties of goroutines:
* 

* Making channels:

-using built in make function.



* Channel Internal working:



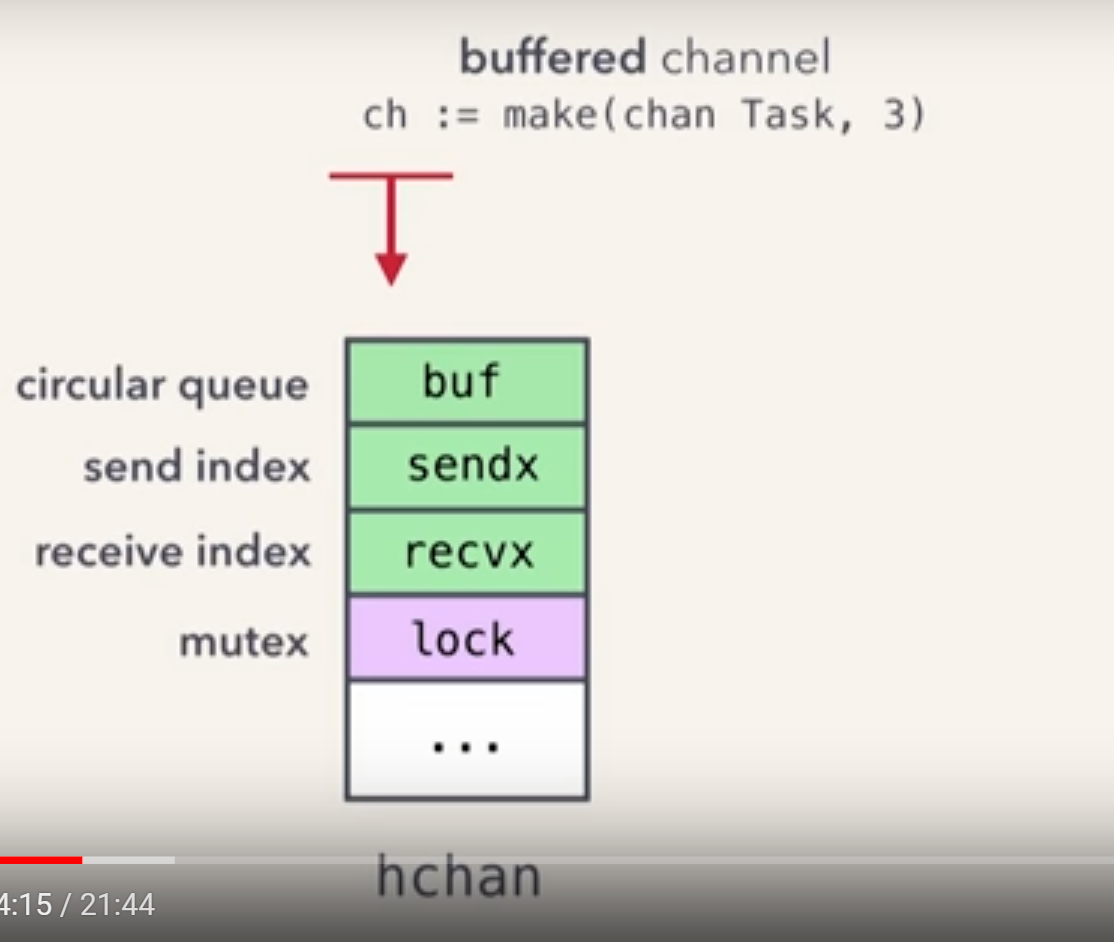
* Goroutine Safe:

- Channels internally use a queue with a lock.

-Making a channel allocates a 'hchan' struct

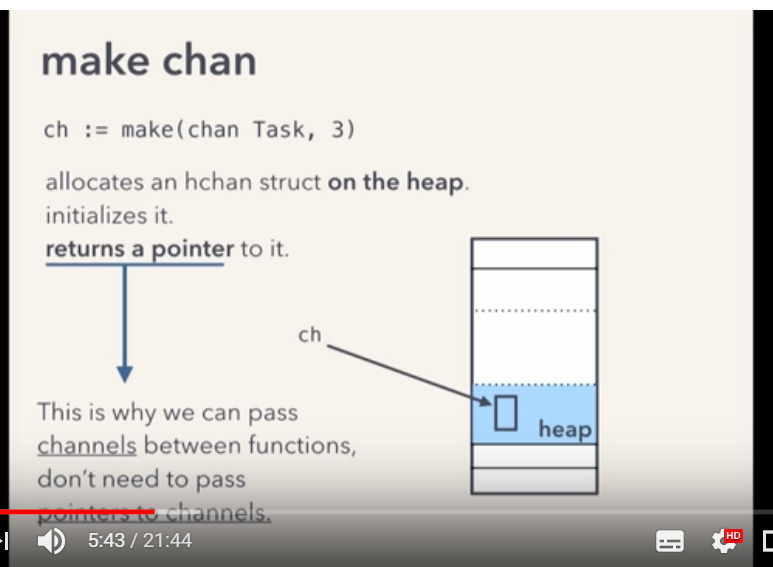
- hchan struct has some fields that implement a queue

- hchan struct:

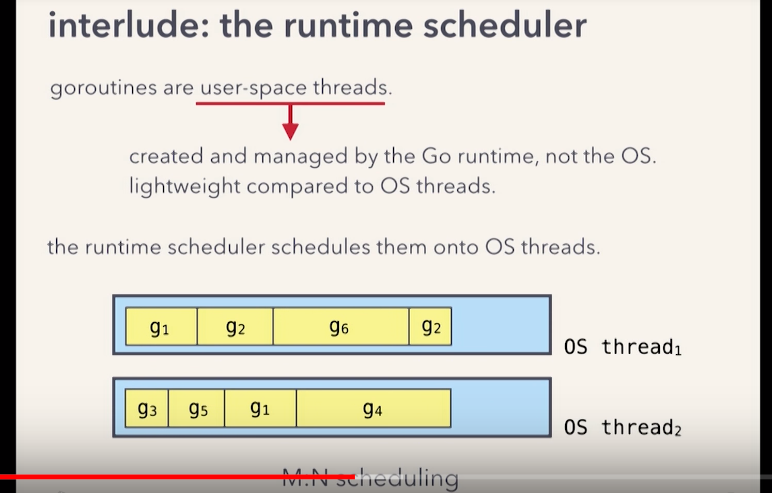


- Here, buf is a circular buffer that wraps around it self.

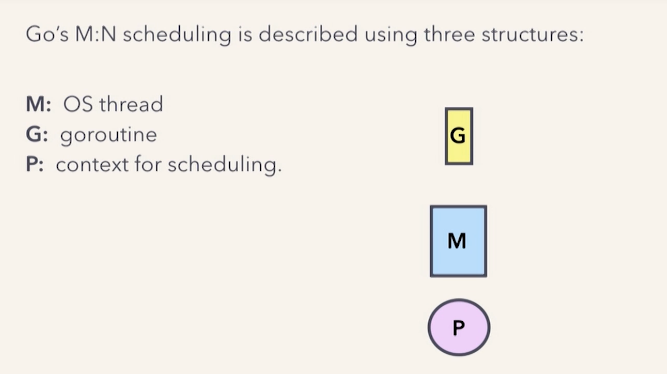
* Send index and receive index are for tracking the send and receive positions in that buffer.

* Hchan struct is allocated on the heap:
* The make function returns a pointer to it.
* Since make function already returns a pointer, we don't need to send pointers for channels across the goroutines.
* 

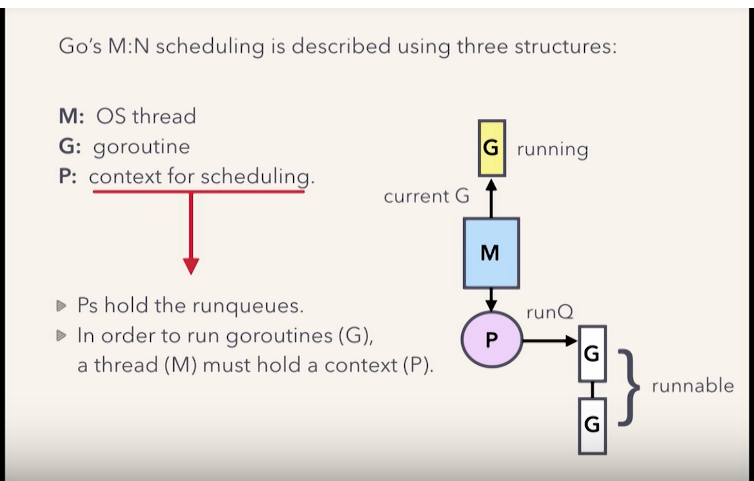
* Goroutines are user space threads.
* They are started and managed by go runtime schedular and not by os.
* They are lightweight and less expensive compared to os threads as they require less resources.
* The schedular uses an MN scheduling modules to run these user space threads on the os thread.
* The schedular takes our many goroutines and multiplexes them onto a few os threads.



* M:N scheduling:



* 'P' holds the goroutines that are ready to run that is the run queues.
* The os threads get their workload from the goroutine in the 'P'



* So our goroutine is blocked but not the underlying os thread.