9 Chapter:

SMP-safe, preempt-safe, interrupt-safe. **Dead lock**, n to n, **Solution**: 1. Implement lock ordering 2. Prevent starvation (what is starvation) 3. Don't acquire same lock 4. Simple as possible **Same Order is Important**

Lock contention and scalability. The granularity of locking is lock protect small amount, improve contention.

10 Chapter: remember lock the data, not the code.

Atomic Integer Operation. atomic_t to protect atomicity, barrier to protect ordering. Less over-head, less cache-line. Better compared to lock mechanism.

Atomic Bitwise Operation: it's object is pointer, so it can pointer to any type of variable you want. Nonatomic is much faster

Spin lock, busy waiting, less overload, make sure lock won't be hold too long. Are not recursive, can be utilized in interrupts. Bottom half can preempt process context, should protect context and block bh. Same as interrupts with bh.

Reader and Writer spin lock (interrupt context only use this)

Semaphores, more overload, suitable for lock being held by long time. Cannot use spinlock at the same time. **Count** multiple holders, Binary single holder.

Mutex, count is 1 's semaphores. Lock should be pair with unlock. A strict but simple version of semaphores.

(sleep only use this)

Completion. BLK, can recursive, can sleep, only use in process context.

Sequential locks: lots of readers, less writers, writer is prior

to reader. Simple shared data but can't use atomic.

Preemption diable enable. Barrier read_barrier and write_barrier.

12 Chapter:

Pages, system's page. Flag represents the state of this page. Page structure is for monitoring those pages. Free pages should be really really careful. Alloc_page

Zone, pooling in place to satisfy allo- cations as needed Zone structure

Vmalloc compare with malloc . GFP_MASK

Slab layer, a generic data structure caching layer. Kmem_cache->cache, (cachep)-→slab (kmem_create) --→object(kmem_malloc)

Allocation on Stack. Single page kernel stacks. Make fair on the stack, make sure every process use less resources.

High memory mappings can not being loaded to kernel address, kmap permanent kmap_atomic temprory for context that can not sleep.

Per-CPU allocation. Is exclusive for each CPU's data. Use get_cpu to stop kernel preemption. Use CPU-data don't need to use lock. Reduces cache invalidation.(have to stop kernel preemption)

Picking allocation method. (1)need Contiguous physical pages. Use kmalloc or low memory allocators. (2)high memory , Use alloc_pages()return page structure(no mapping) (3)don't need contiguous physical pages, vmalloc() (4) create and destroy large data, allocation and deallocation frequently, use slab cache. Use GFP mask to meet kernel constraints.

13 Chapter:

Filesystem abstraction layer.
Write()->sys_write()->filesystem->physical

VFS objects: superblock, inode, dentry, file. And its corresponding objects

Superblock object, generate when runs and store in memory. Run in process context, and can be blocked. All files visited through inode objects. Dentry object is a file, special file. State: used unused negative. the Dentry cache will store the path and the inode temporarily to save time. File object not represent openfile, insead dentry represent. Namespace tell which filesystem using.

14 Chapter:

Block Device: sector smaller than block. Buffers: page to block. operation bigger more overload than bio structure. each block I/O request is represented by a bio structure.

Requests queue, store all the requests to all the IO devices. IO schdulers will do depends on the time of find position. Merging close requests and also Sorting the queues by their position.

Linus Elevators: first it perform front and back merging. If not, then it find a propriate place to insert.

Deadline IO Scheduler. Has a deadline. Read 500ms, write 5s improve read operation performance(predict)

The complete fair scheduling, which has multiple queues for every operations, use round-robin.**noop** merge

15 Chapter:

process hold all the memory. Address space, memory areas that achievable. Memory descriptor. Fork() copy father's descriptor then alloc. Drop then none use.

Kernel thread has no memory, when its run, it can use pre-processes' mm_struct to store some info.

In kernel, memory area call virtual memory areas. Kernel use vm_area_struct to manage memory.

VMA symbol can determine the behavior. Mmap and mm_rb to access the vm_area_struct. List and rb

Find_vma() , first find cache, then search rb tree. Do_mmap()

Page table, for processor to find physical address. Three level, TLB buffer store recent visit, improve performance.

16 Chapter:

Approach to caching, store data that just visited, if cache hit, then read, if not, find it in IO, then cache it

Write cache: no write, make cache's data invalid, write-through-cache, overload, Write-back can merge multiple write.. Use a dirty list, lift complexity.

Cache eviction, not dirty block, long time no use, Two lists, inactive list replace.

Address_space is actually like page_cache_entity., physical page of a file. Find use hash table,

The buffer cache, for replace page. Flusher. Avoiding congestion with multiple threads.