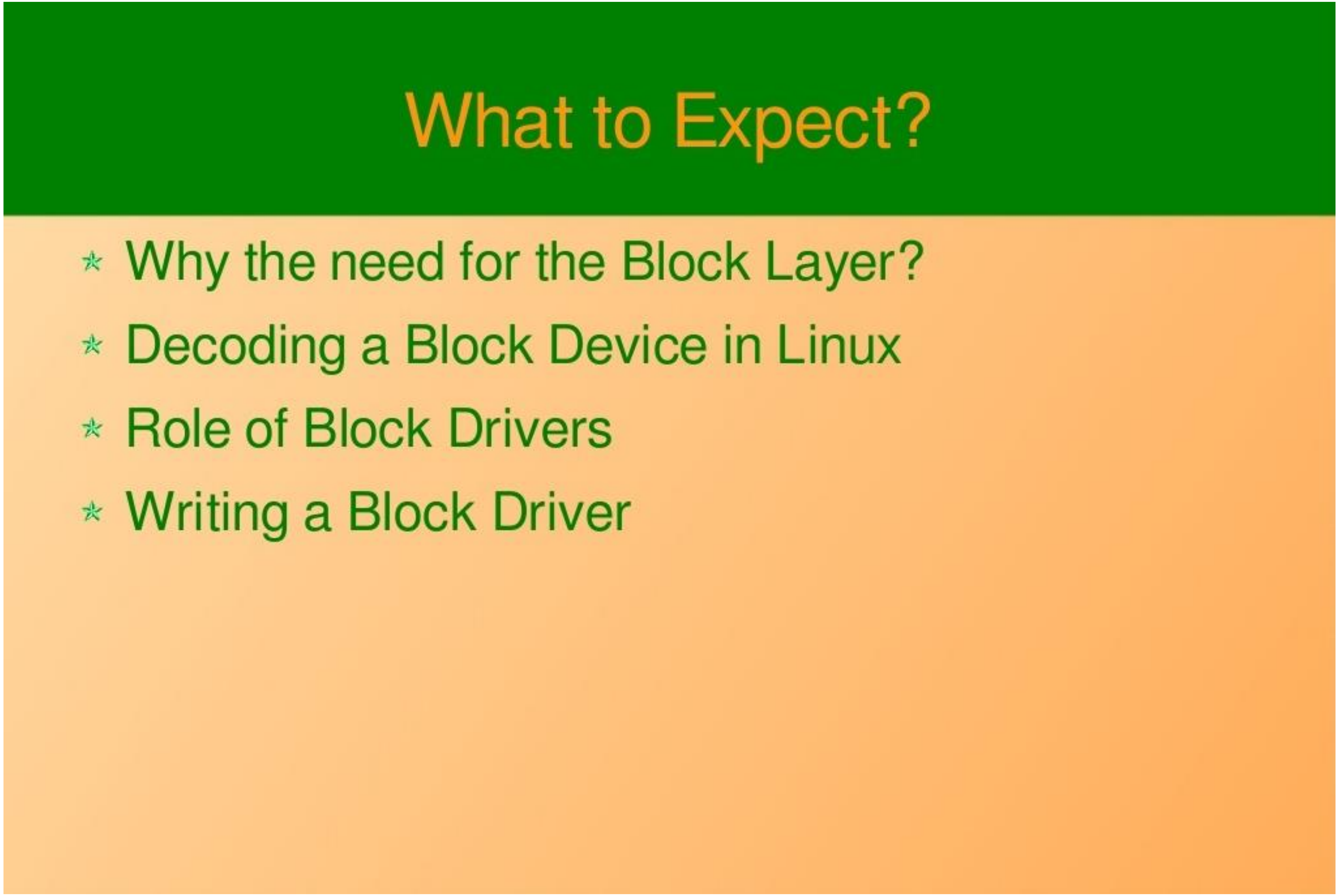
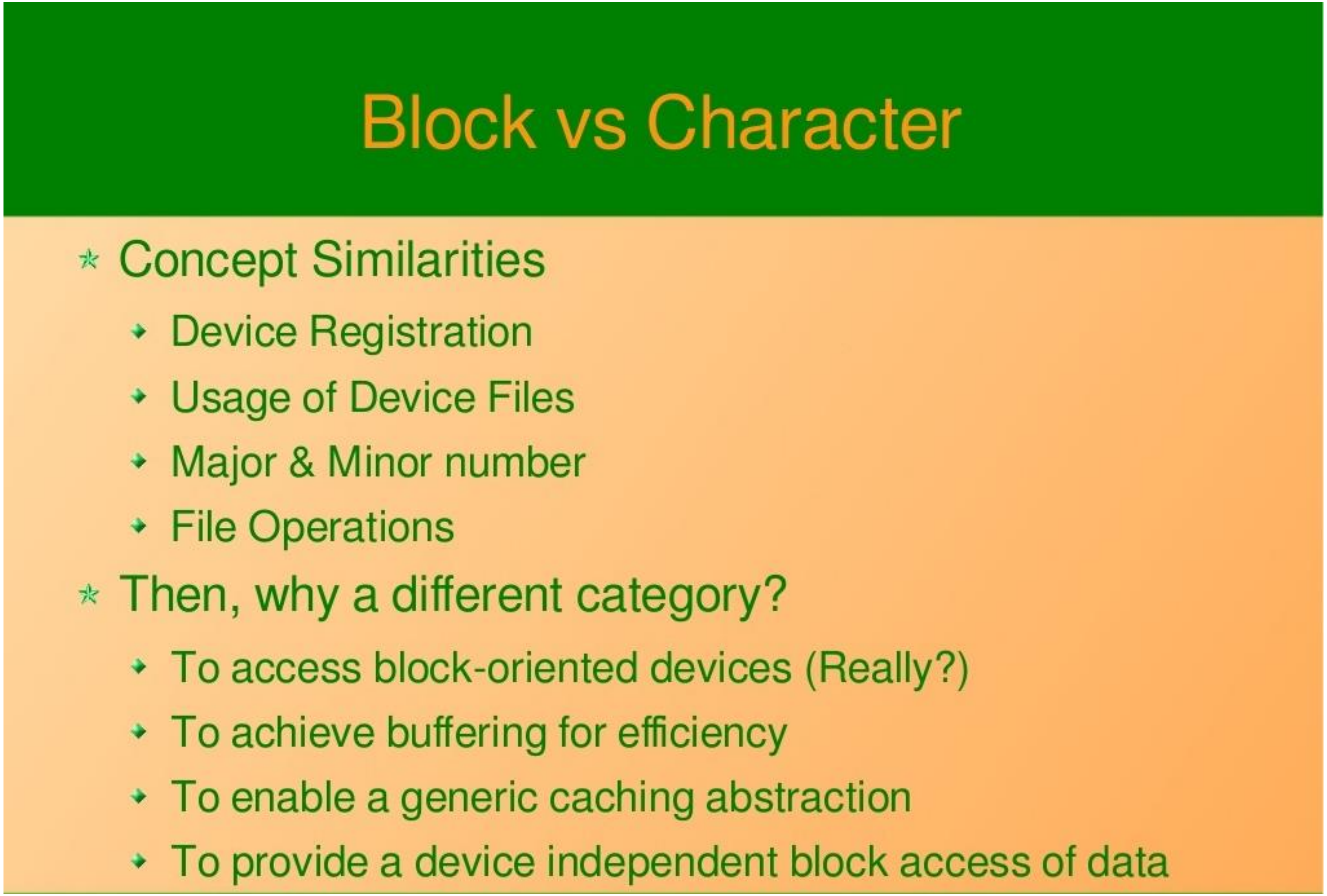
Block Driver



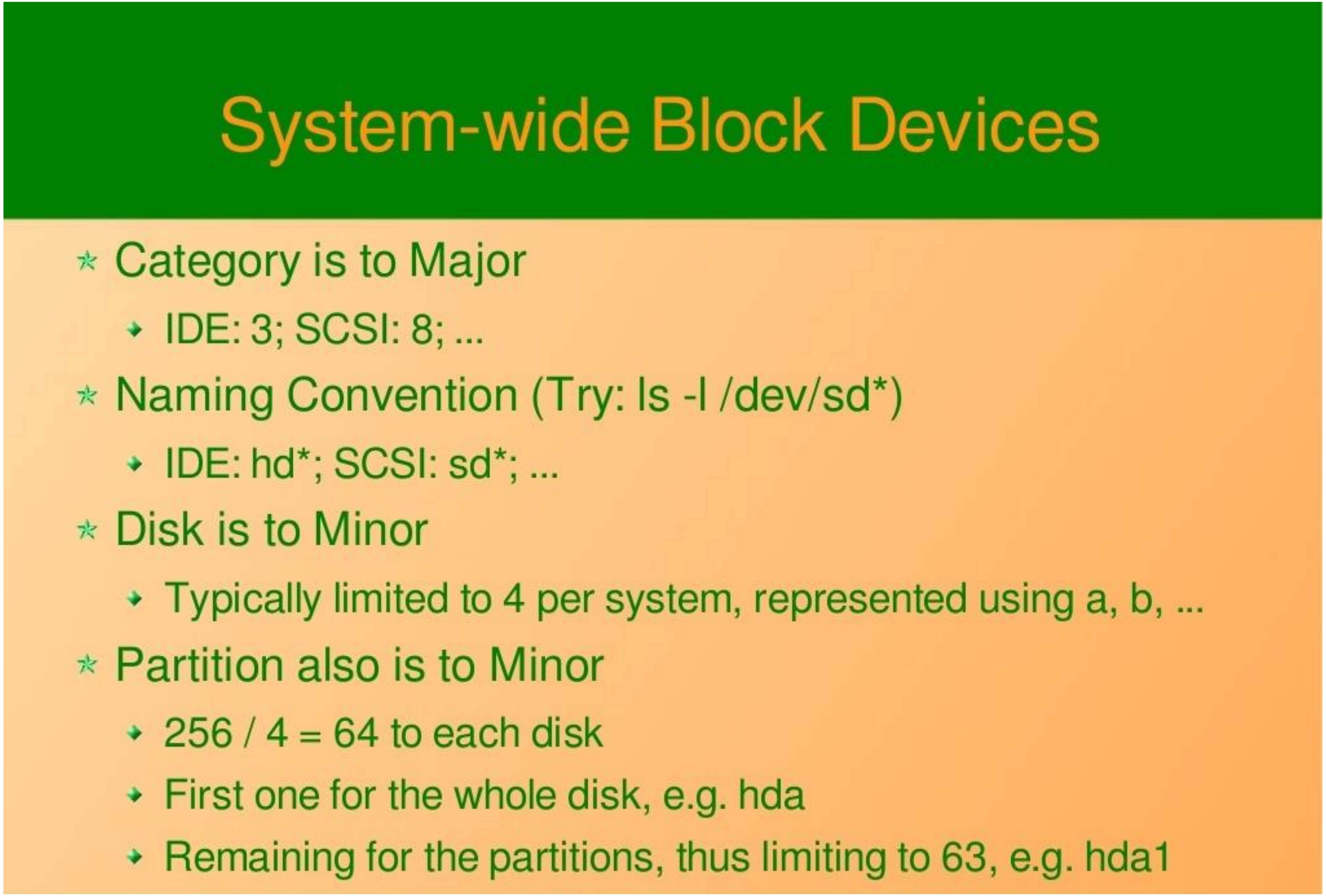
What to Expect?

- Why the need for the Block Layer?
- Decoding a Block Device in Linux
- Role of Block Drivers
- Writing a Block Driver



Block vs Character

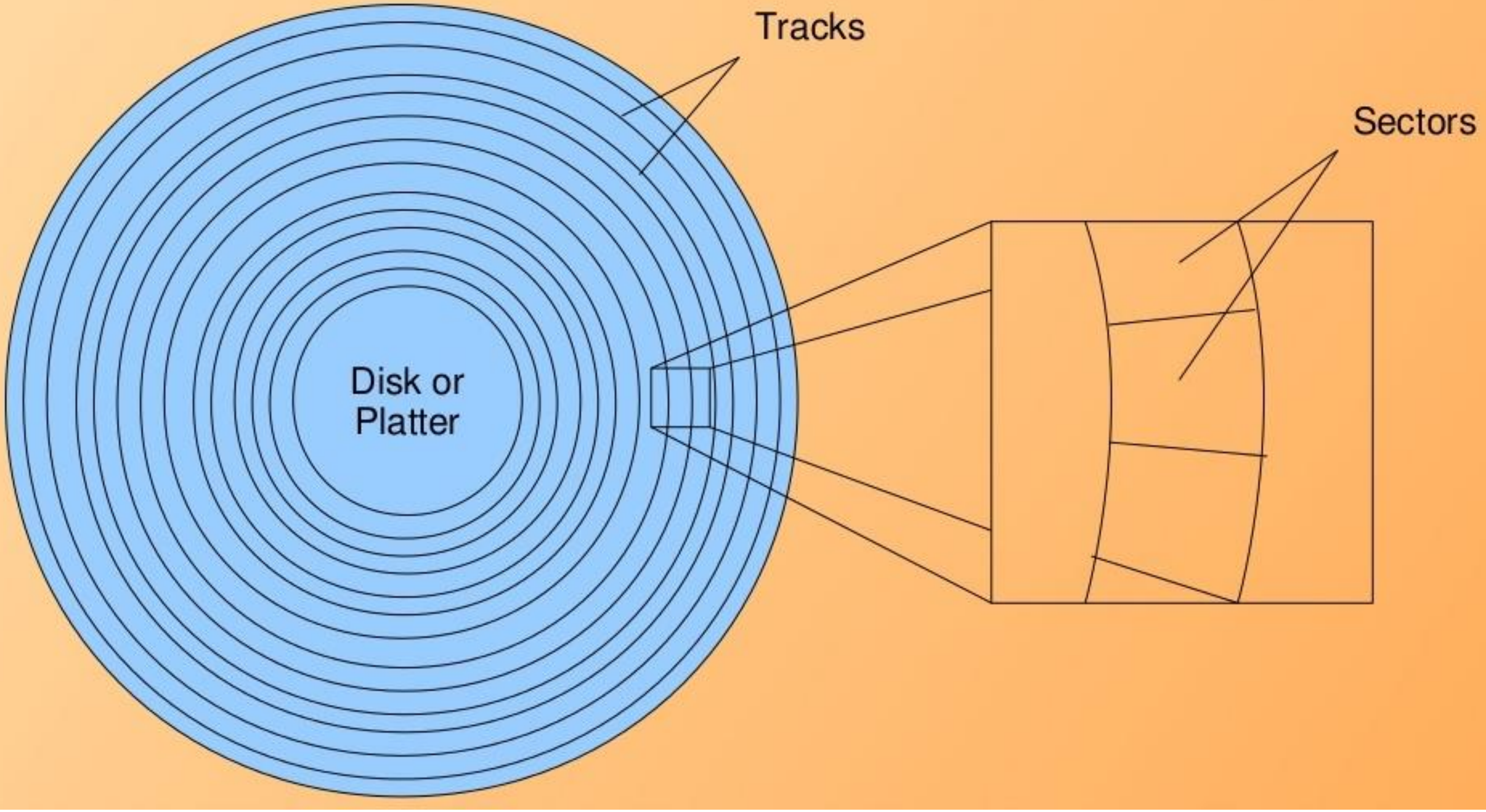
- Concept Similarities
  - Device Registration
  - Usage of Device Files
  - Major & Minor number
  - File Operations
- Then, why a different category?
  - To access block-oriented devices (Really?)
  - To achieve buffering for efficiency
  - To enable a generic caching abstraction
  - To provide a device independent block access of data

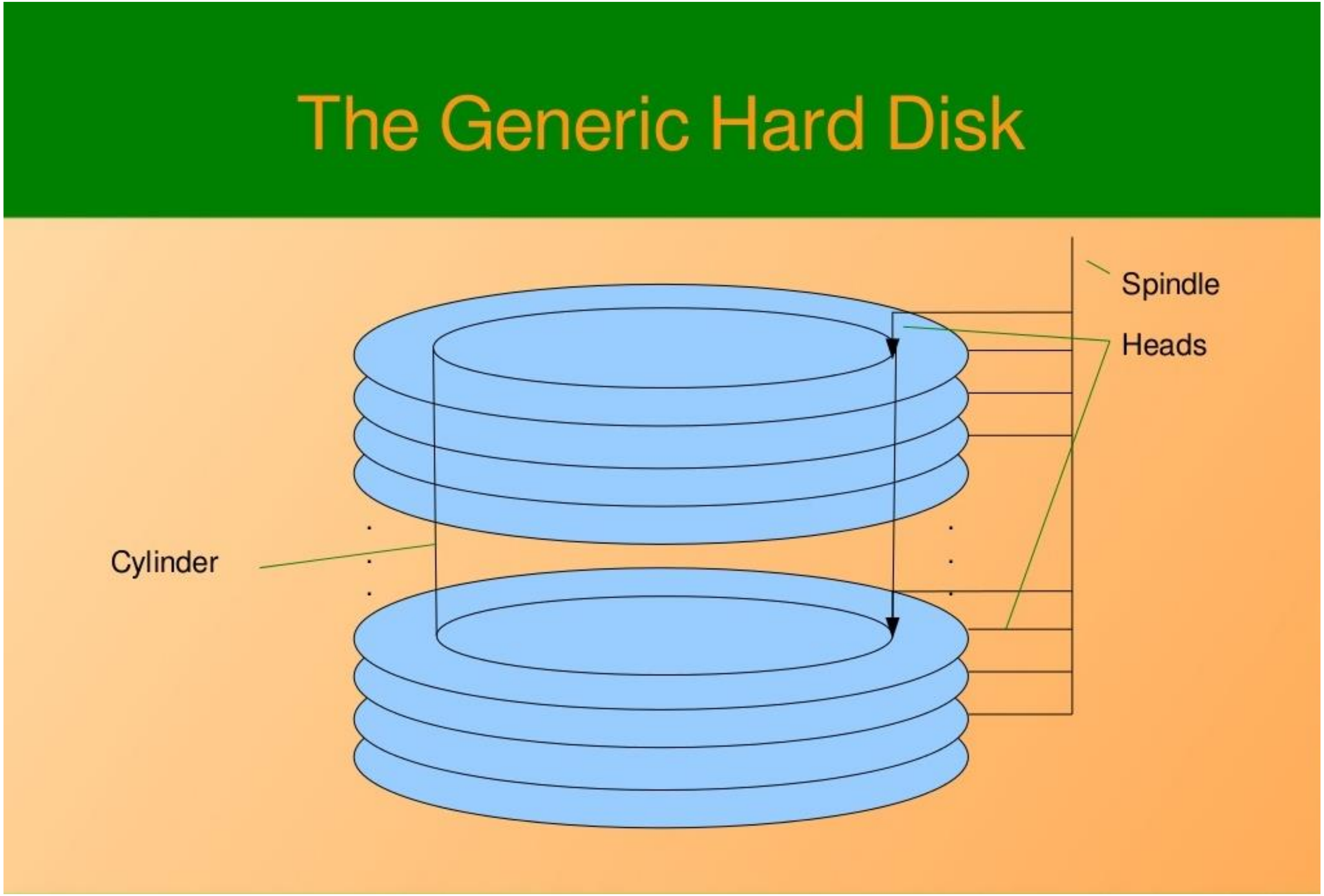


System-wide Block Devices

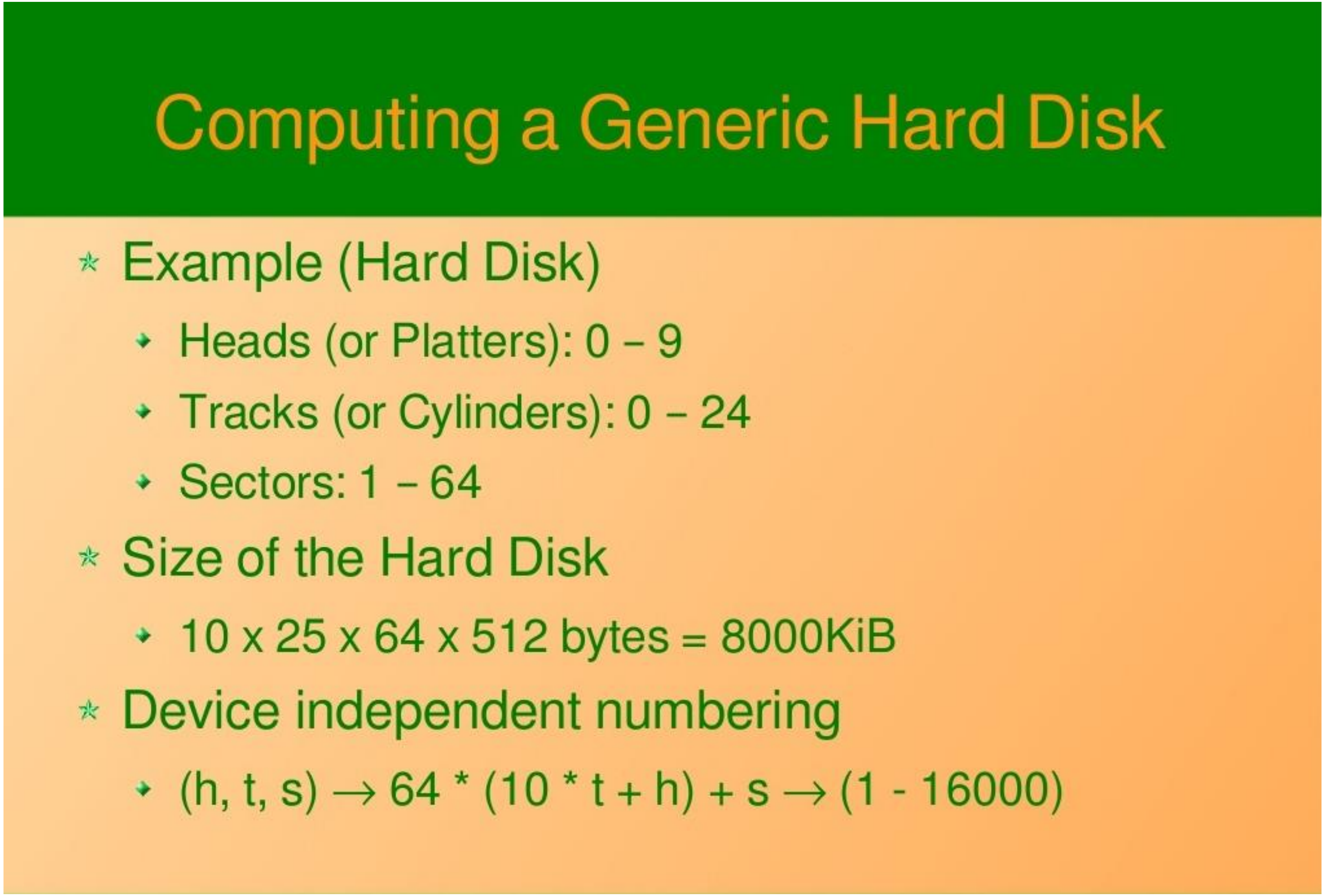
- Category is to Major
  - IDE: 3; SCSI: 8; ...
- Naming Convention (Try: ls -l /dev/sd*)
  - IDE: hd*; SCSI: sd*; ...
- Disk is to Minor
  - Typically limited to 4 per system, represented using a, b, ...
- Partition also is to Minor
  - 256 / 4 = 64 to each disk
  - First one for the whole disk, e.g. hda
  - Remaining for the partitions, thus limiting to 63, e.g. hda1

The Generic Hard Disk



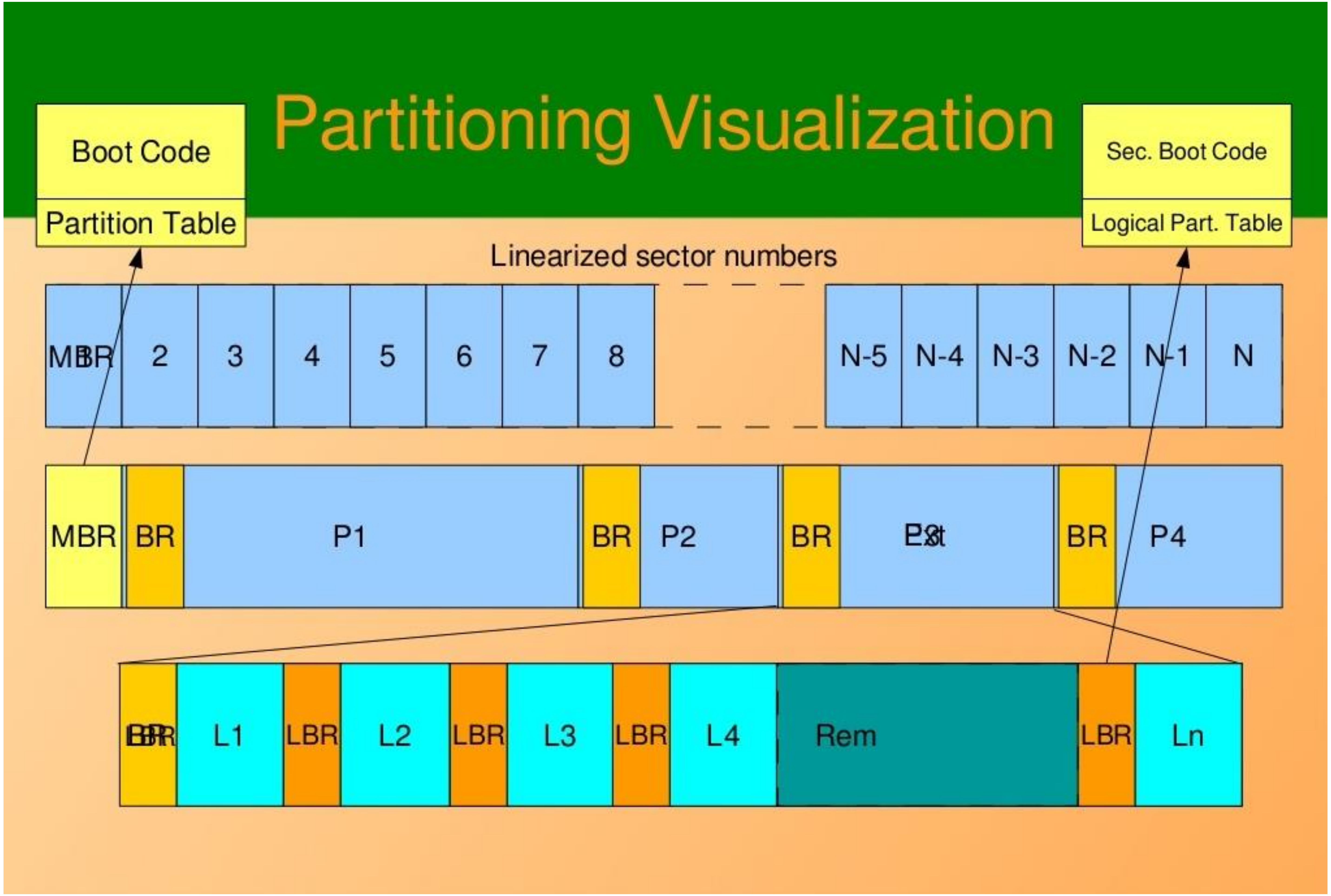


The Generic Hard Disk

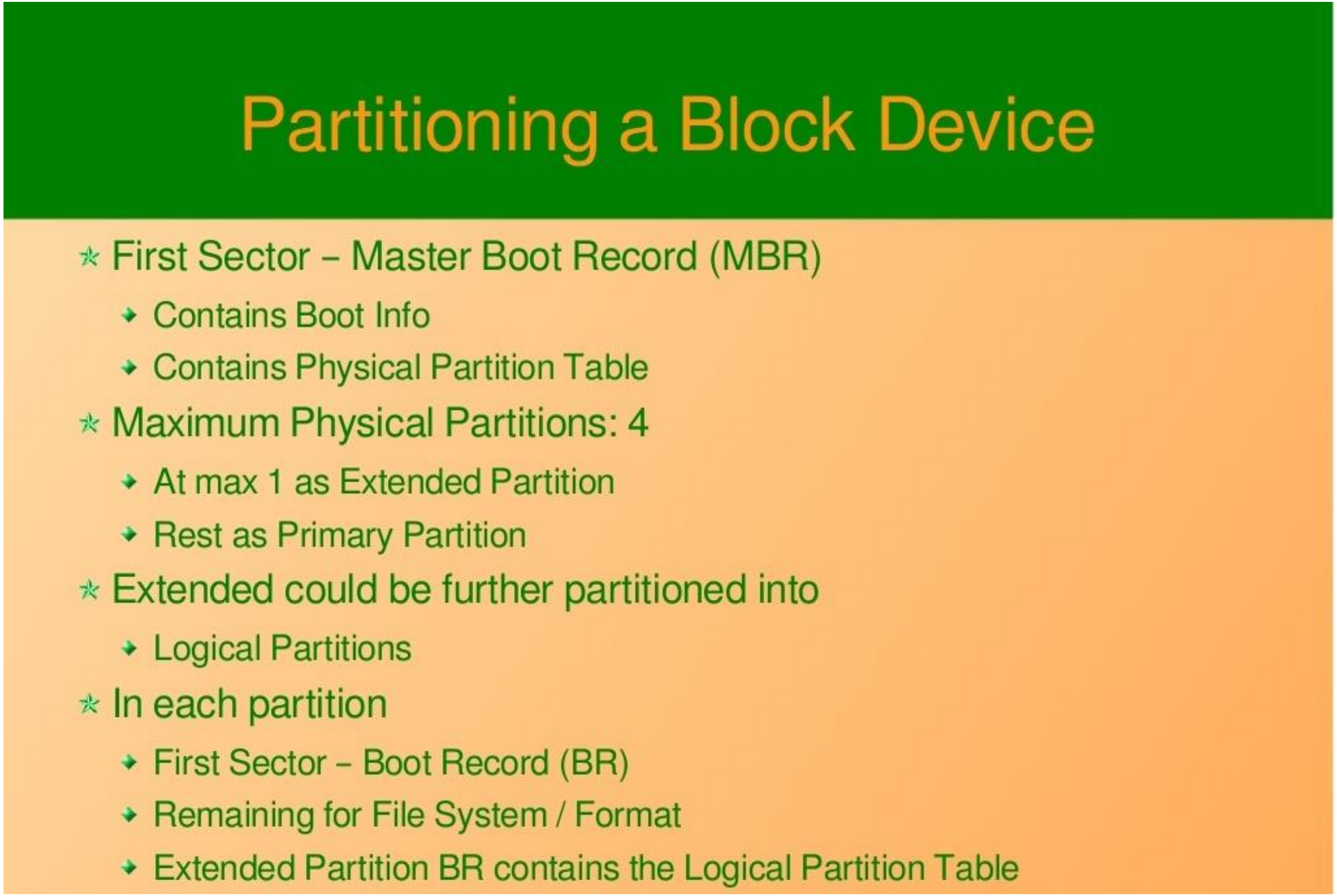


Computing a Generic Hard Disk

- Example (Hard Disk)
  - Heads (or Platters): 0 – 9
  - Tracks (or Cylinders): 0 – 24
  - Sectors: 1 – 64
- Size of the Hard Disk
  - 10 x 25 x 64 x 512 bytes = 8000KiB
- Device independent numbering
  - (h, t, s) → 64 * (10 * t + h) + s → (1 - 16000)



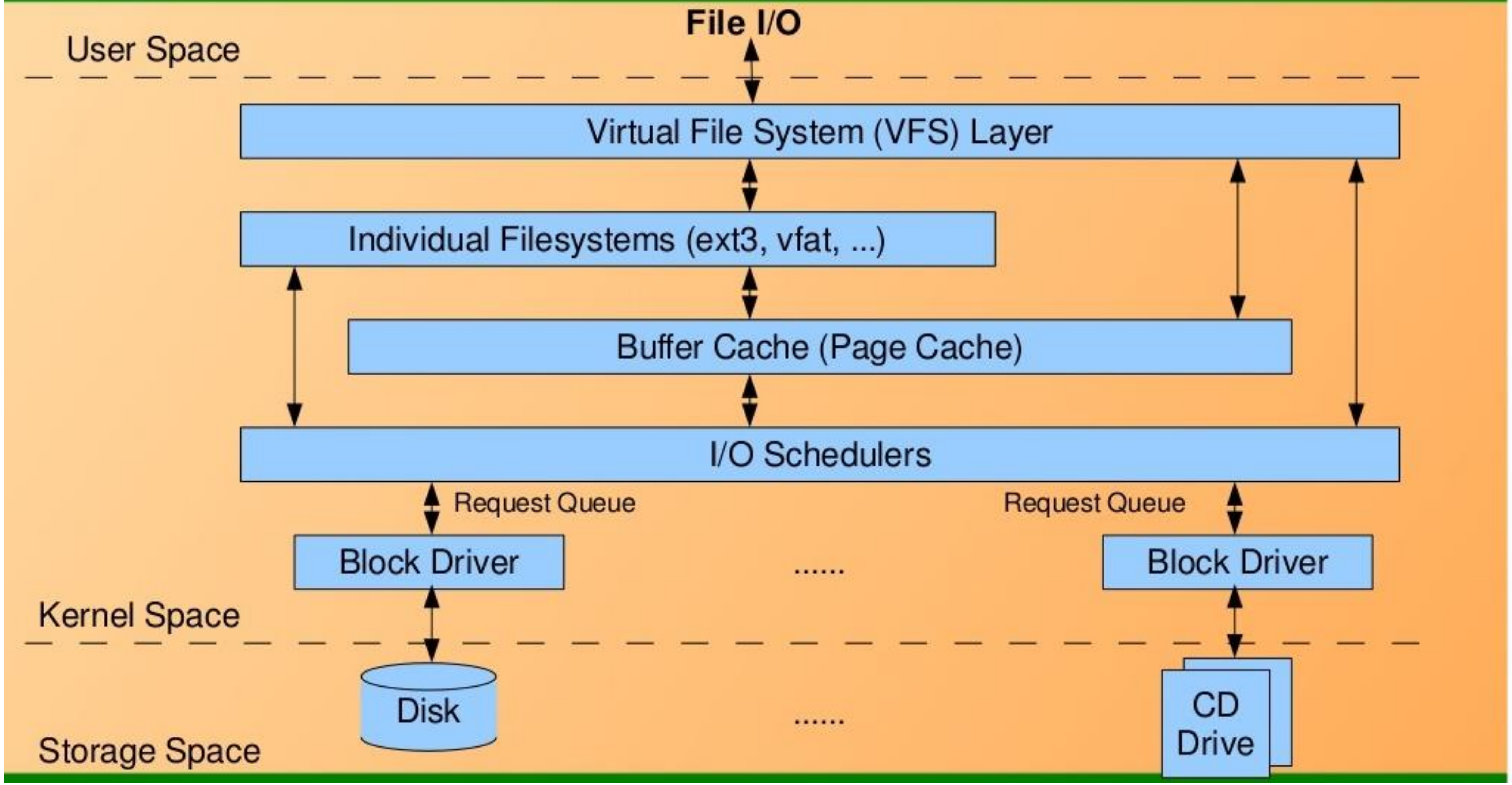
Partitioning Visualization



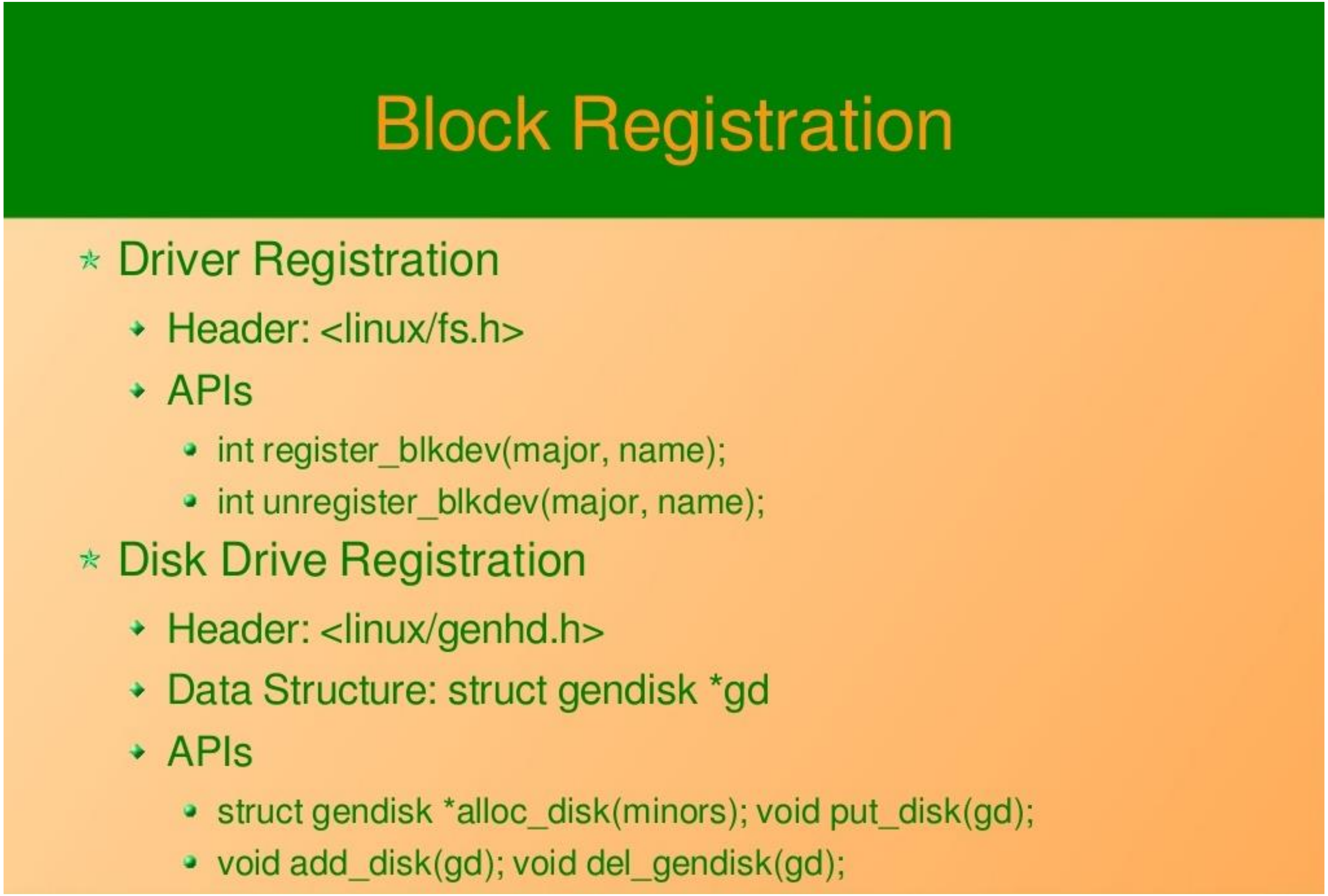
Partitioning a Block Device

- First Sector – Master Boot Record (MBR)
  - Contains Boot Info
  - Contains Physical Partition Table
- Maximum Physical Partitions: 4
  - At max 1 as Extended Partition
  - Rest as Primary Partition
- Extended could be further partitioned into
  - Logical Partitions
- In each partition
  - First Sector – Boot Record (BR)
  - Remaining for File System / Format
  - Extended Partition BR contains the Logical Partition Table

Block Input / Output

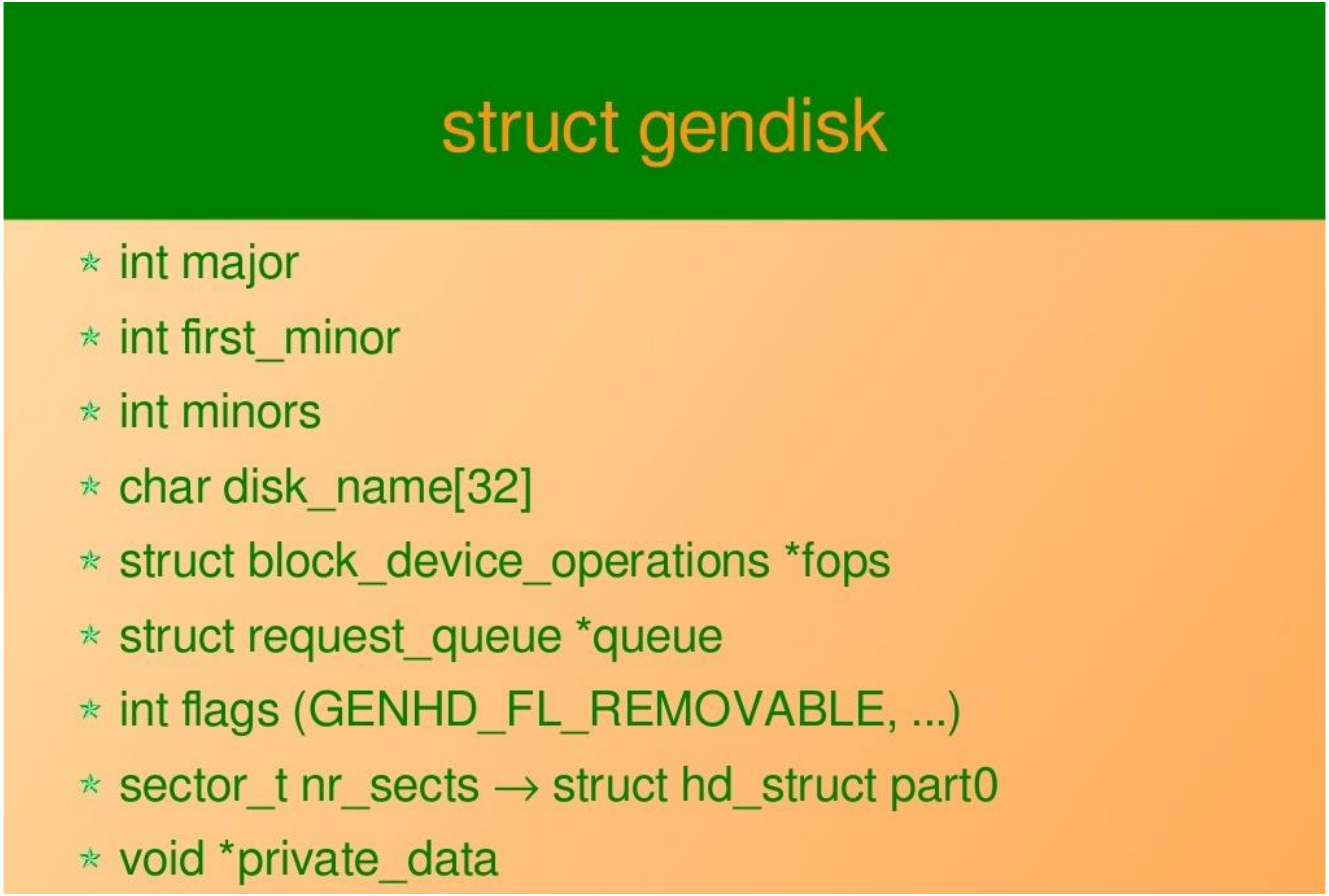


Now, let's write a Driver
to
Achieve the Purpose



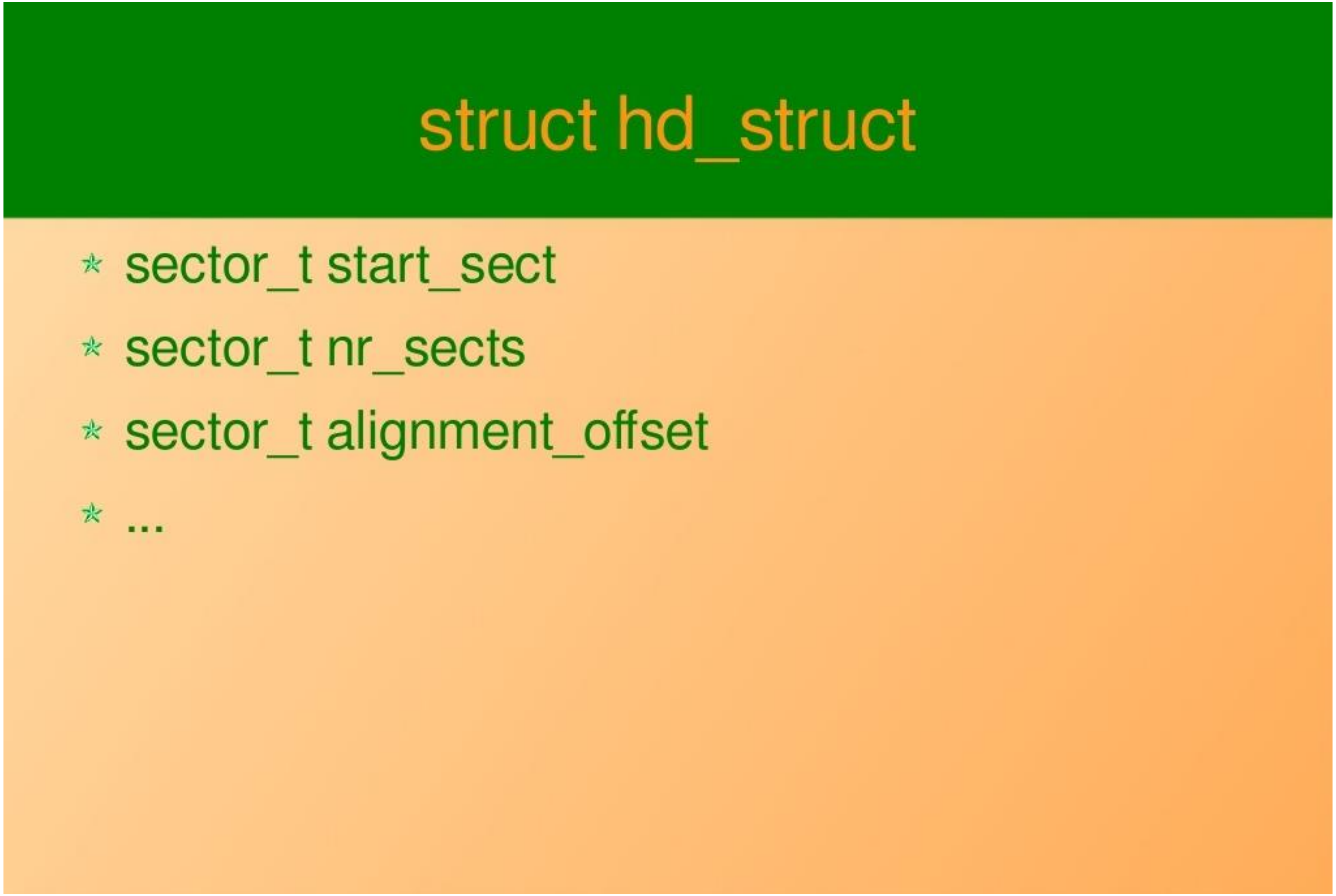
Block Registration

- Driver Registration
  - Header: <linux/fs.h>
  - APIs
    - int register_blkdev(major, name);
    - int unregister_blkdev(major, name);
- Disk Drive Registration
  - Header: <linux/genhd.h>
  - Data Structure: struct gendisk *gd
  - APIs
    - struct gendisk *alloc_disk(minors); void put_disk(gd);
    - void add_disk(gd); void del_gendisk(gd);



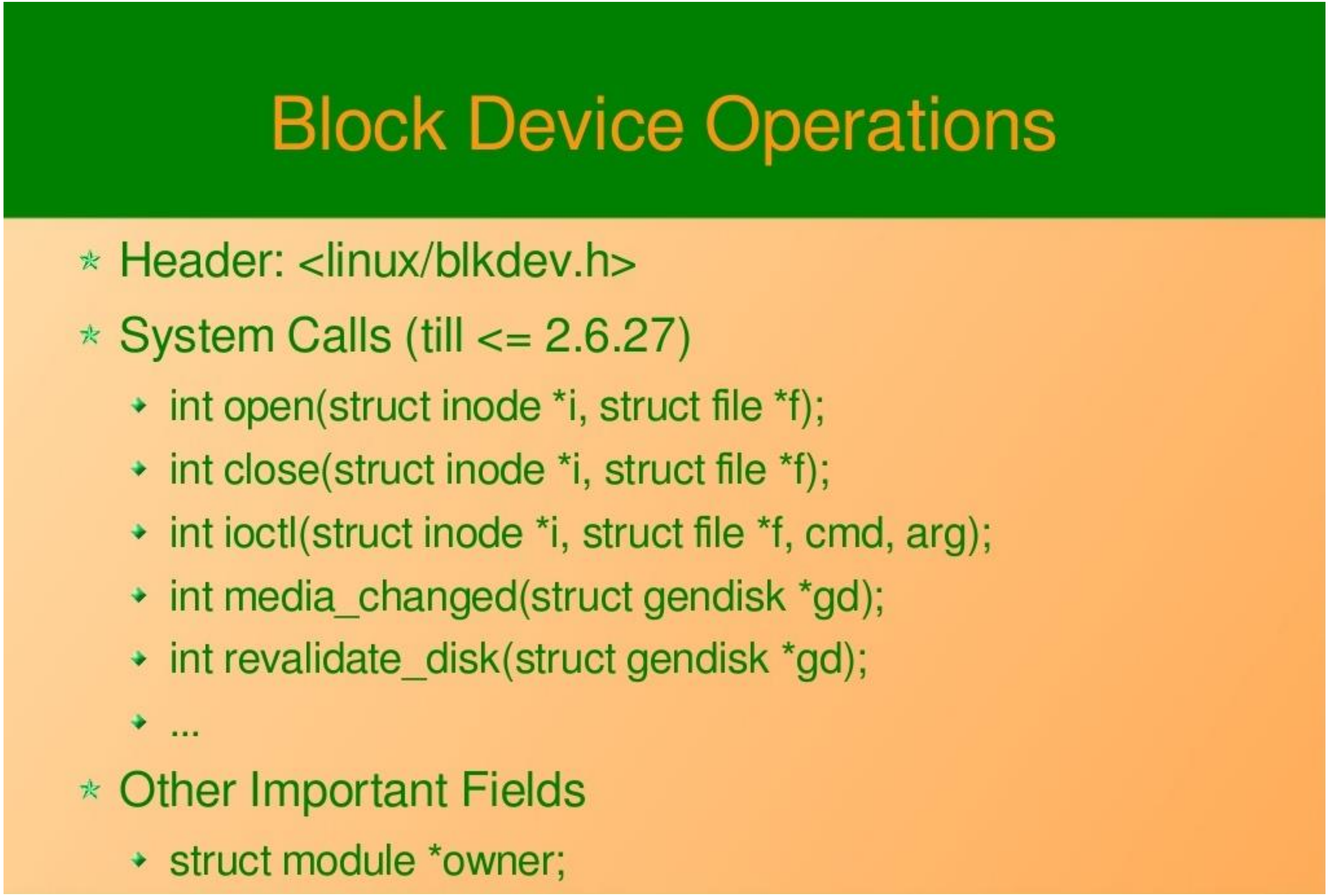
struct gendisk

- int major
- int first_minor
- int minors
- char disk_name[32]
- struct block_device_operations *fops
- struct request_queue *queue
- int flags (GENHD_FL_REMOVABLE, ...)
- sector_t nr_sects → struct hd_struct part0
- void *private_data



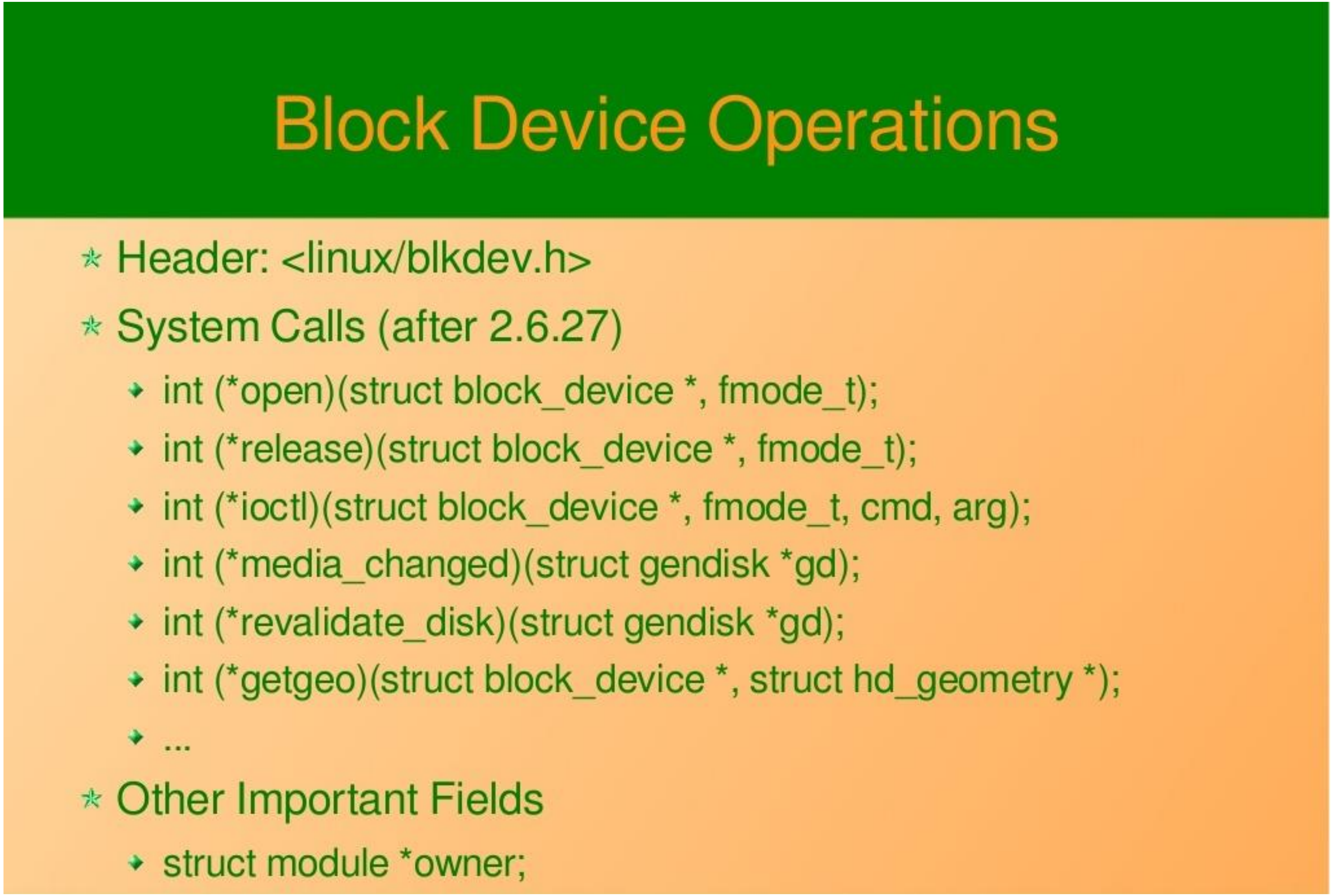
struct hd_struct

- sector_t start_sect
- sector_t nr_sects
- sector_t alignment_offset
- ...



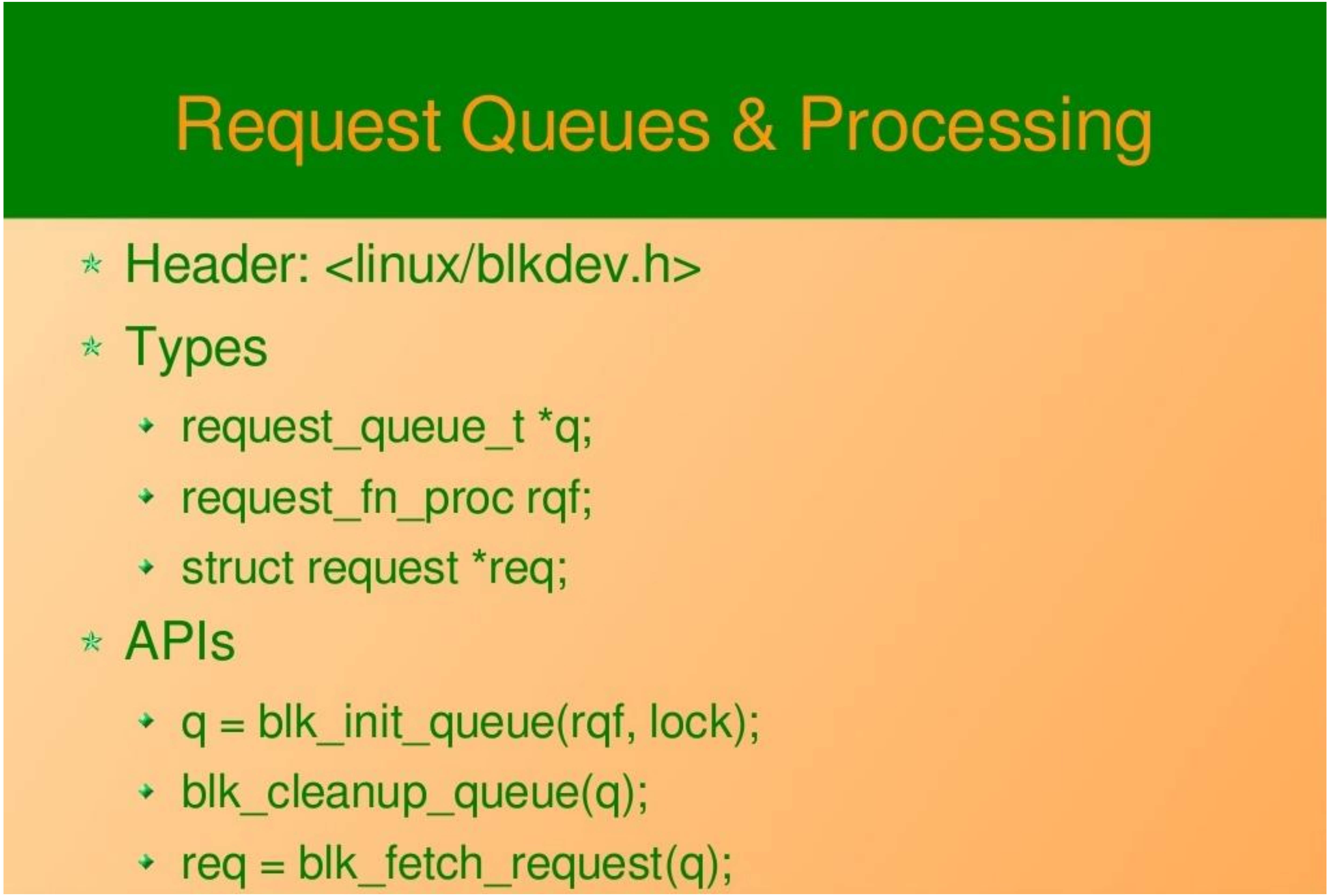
Block Device Operations

- Header: <linux/blkdev.h>
- System Calls (till <= 2.6.27)
  - int open(struct inode *i, struct file *f);
  - int close(struct inode *i, struct file *f);
  - int ioctl(struct inode *i, struct file *f, cmd, arg);
  - int media_changed(struct gendisk *gd);
  - int revalidate_disk(struct gendisk *gd);
  - ...
- Other Important Fields
  - struct module *owner;



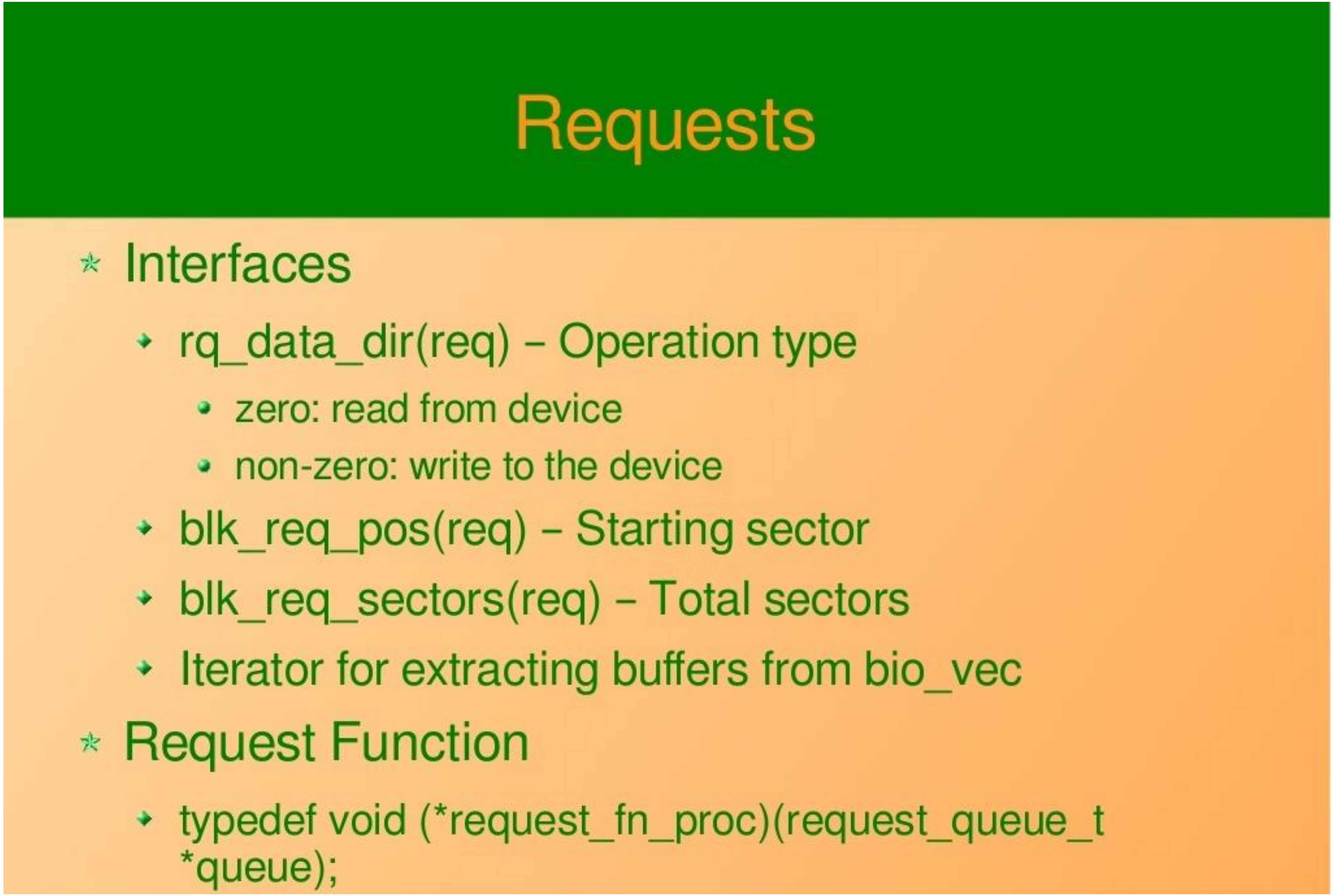
Block Device Operations

- Header: <linux/blkdev.h>
- System Calls (after 2.6.27)
  - int (*open)(struct block_device *, fmode_t);
  - int (*release)(struct block_device *, fmode_t);
  - int (*ioctl)(struct block_device *, fmode_t, cmd, arg);
  - int (*media_changed)(struct gendisk *gd);
  - int (*revalidate_disk)(struct gendisk *gd);
  - int (*getgeo)(struct block_device *, struct hd_geometry *);
  - ...
- Other Important Fields
  - struct module *owner;



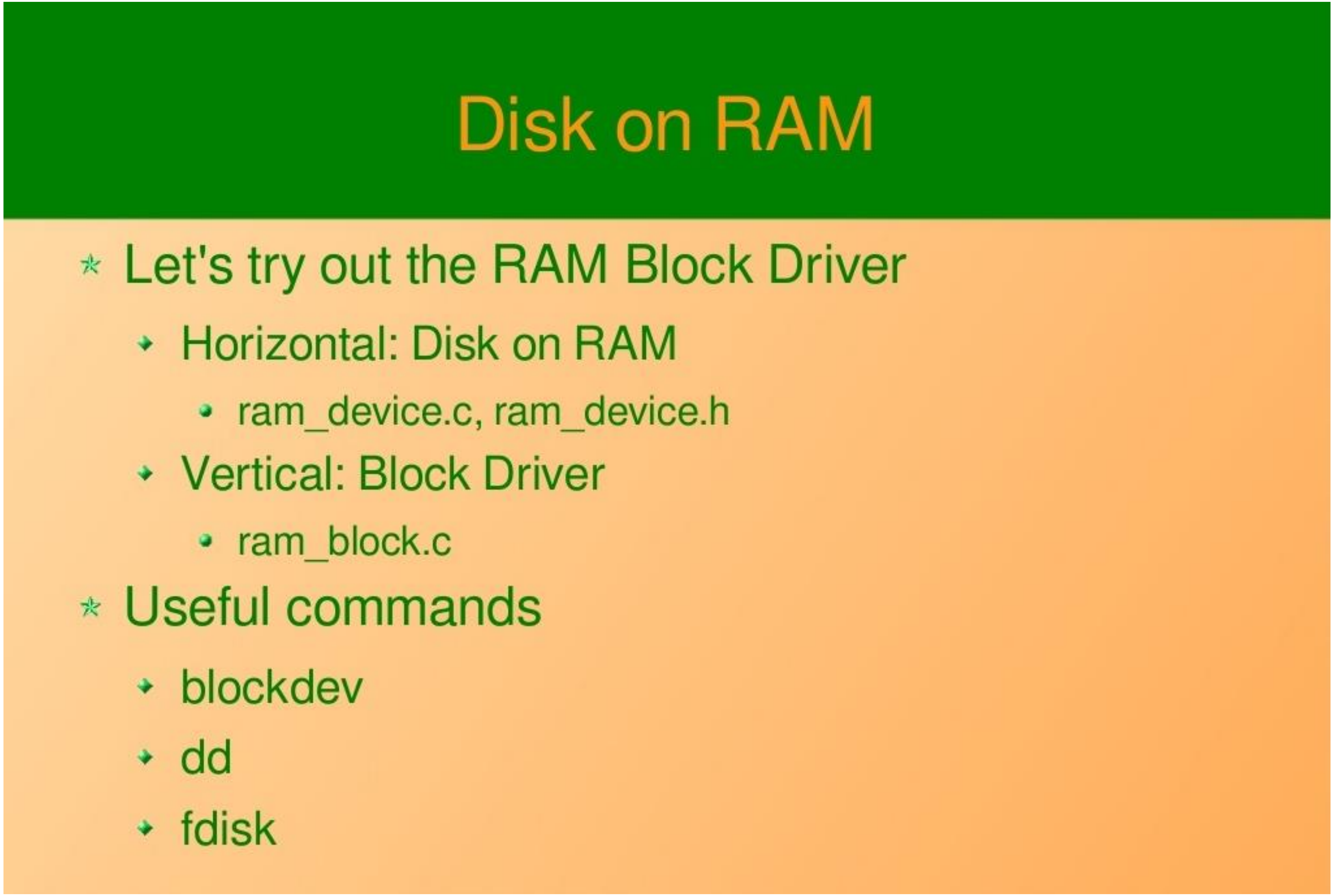
Request Queues & Processing

- Header: <linux/blkdev.h>
- Types
  - request_queue_t *q;
  - request_fn_proc rqf;
  - struct request *req;
- APIs
  - q = blk_init_queue(rqf, lock);
  - blk_cleanup_queue(q);
  - req = blk_fetch_request(q);



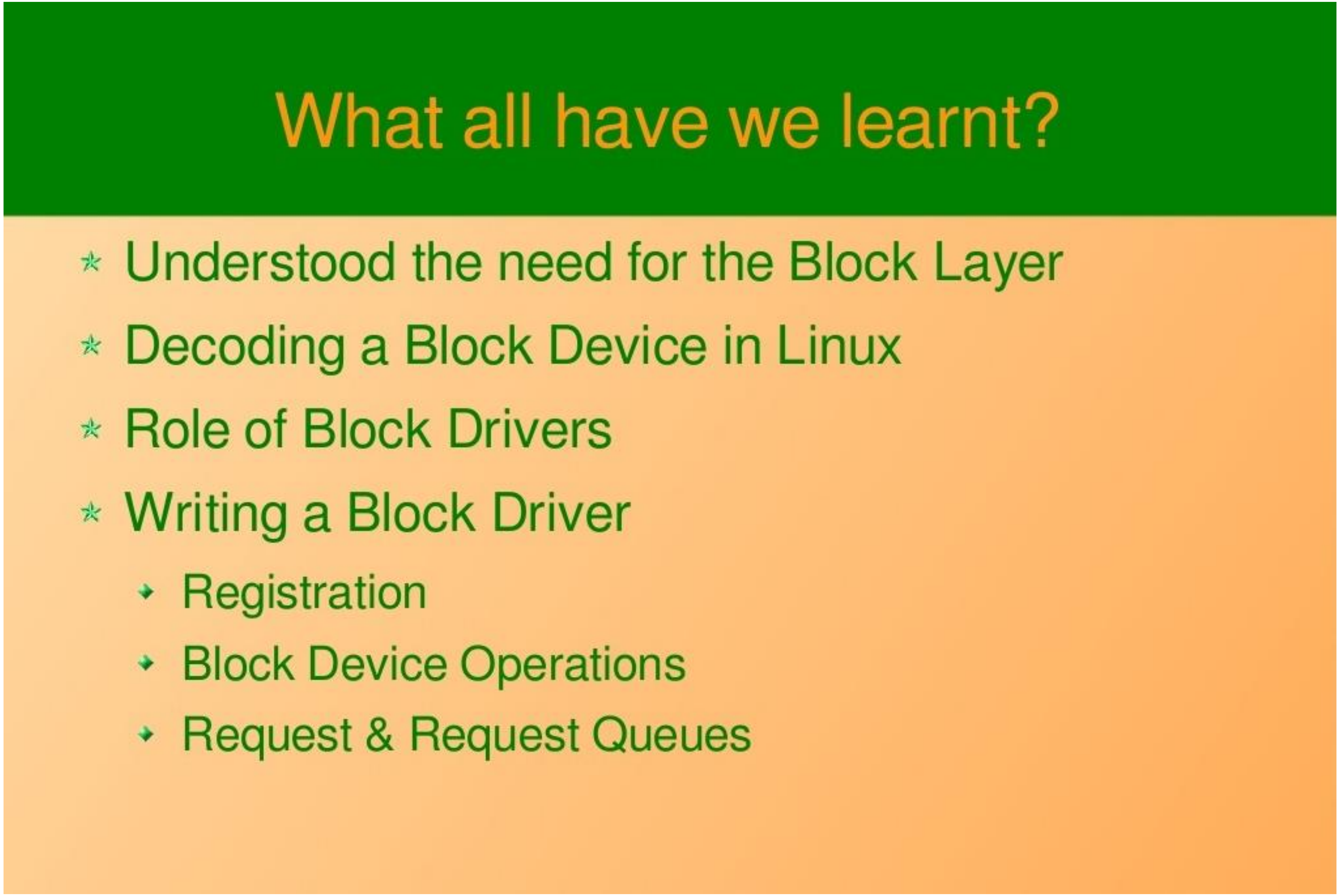
Requests

- Interfaces
  - rq_data_dir(req) – Operation type
    - zero: read from device
    - non-zero: write to the device
  - blk_req_pos(req) – Starting sector
  - blk_req_sectors(req) – Total sectors
  - Iterator for extracting buffers from bio_vec
- Request Function
  - typedef void (*request_fn_proc)(request_queue_t *queue);



Disk on RAM

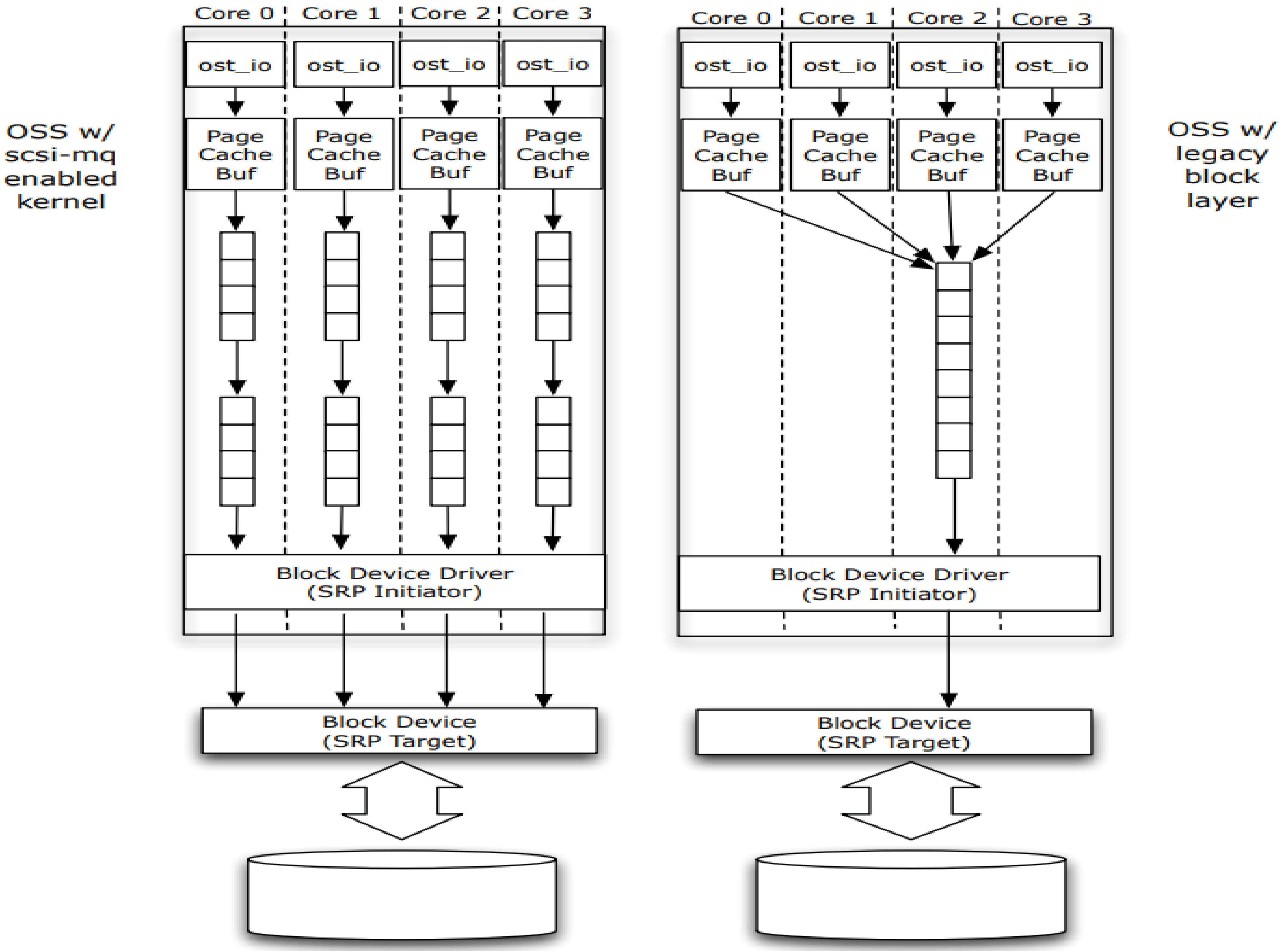
- Let's try out the RAM Block Driver
  - Horizontal: Disk on RAM
    - ram_device.c, ram_device.h
  - Vertical: Block Driver
    - ram_block.c
- Useful commands
  - blockdev
  - dd
  - fdisk



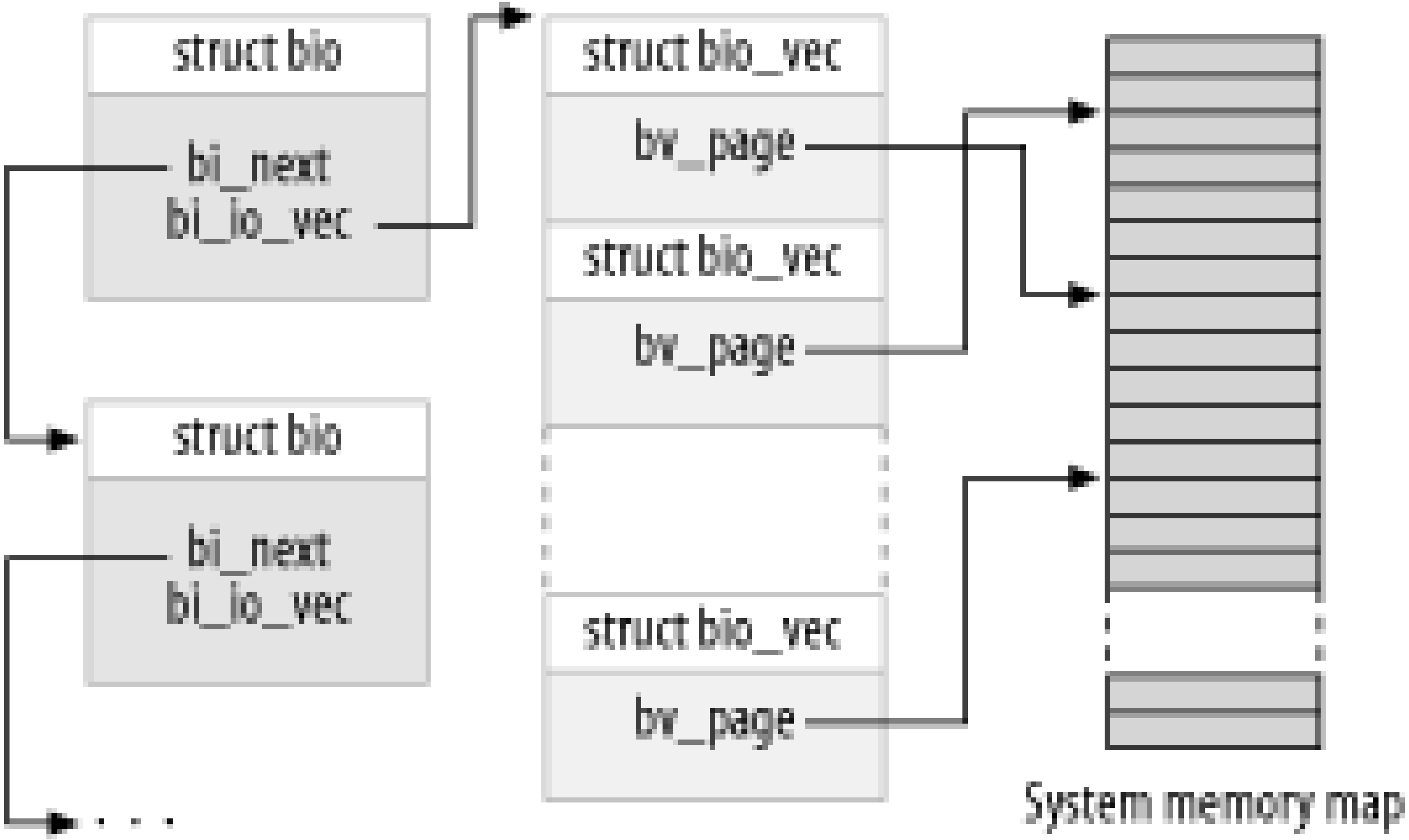
What all have we learnt?

- Understood the need for the Block Layer
- Decoding a Block Device in Linux
- Role of Block Drivers
- Writing a Block Driver
  - Registration
  - Block Device Operations
  - Request & Request Queues

Single & Multi-queue Block I/O



Block I/O Structure


struct bio
bi_next
bi_io_vec
struct bio
bi_next
bi_io_vec
struct bio_vec
bv_page
struct bio_vec
bv_page
struct bio_vec
bv_page
System memory map