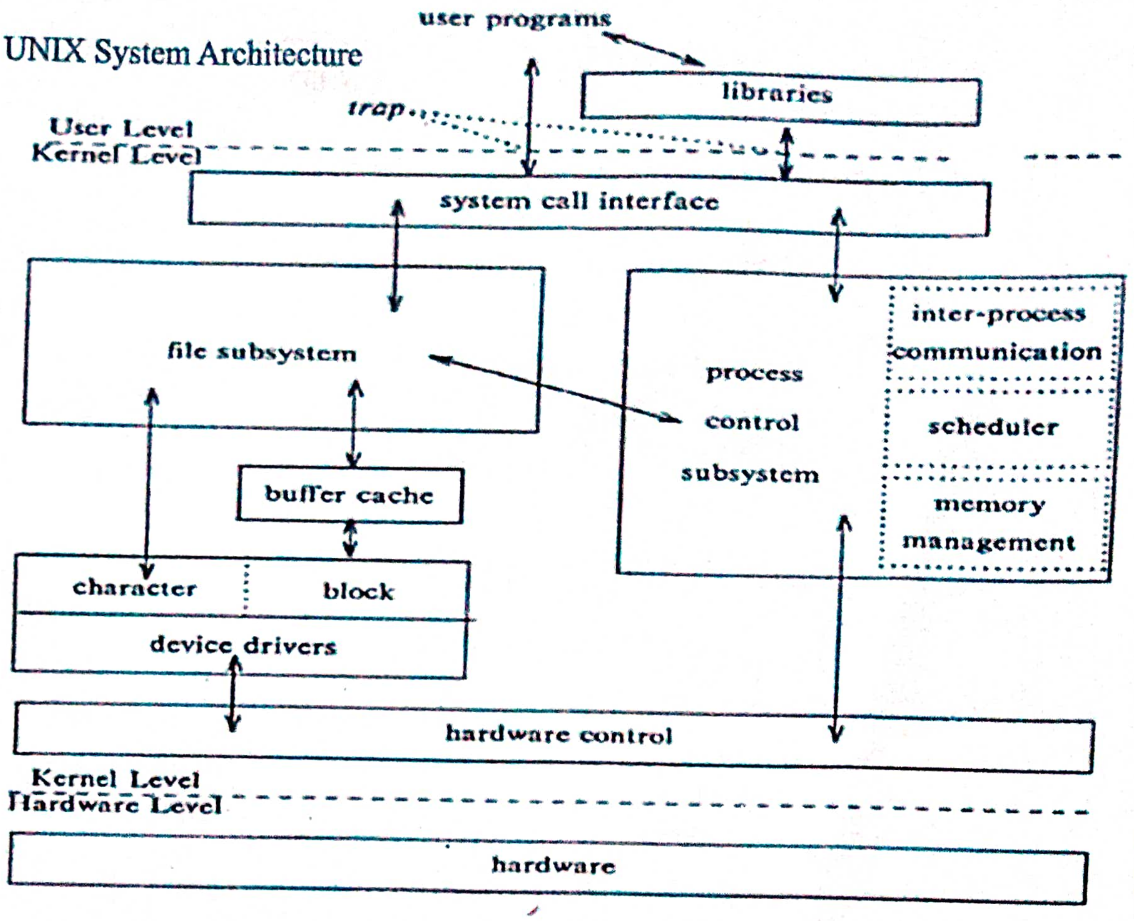
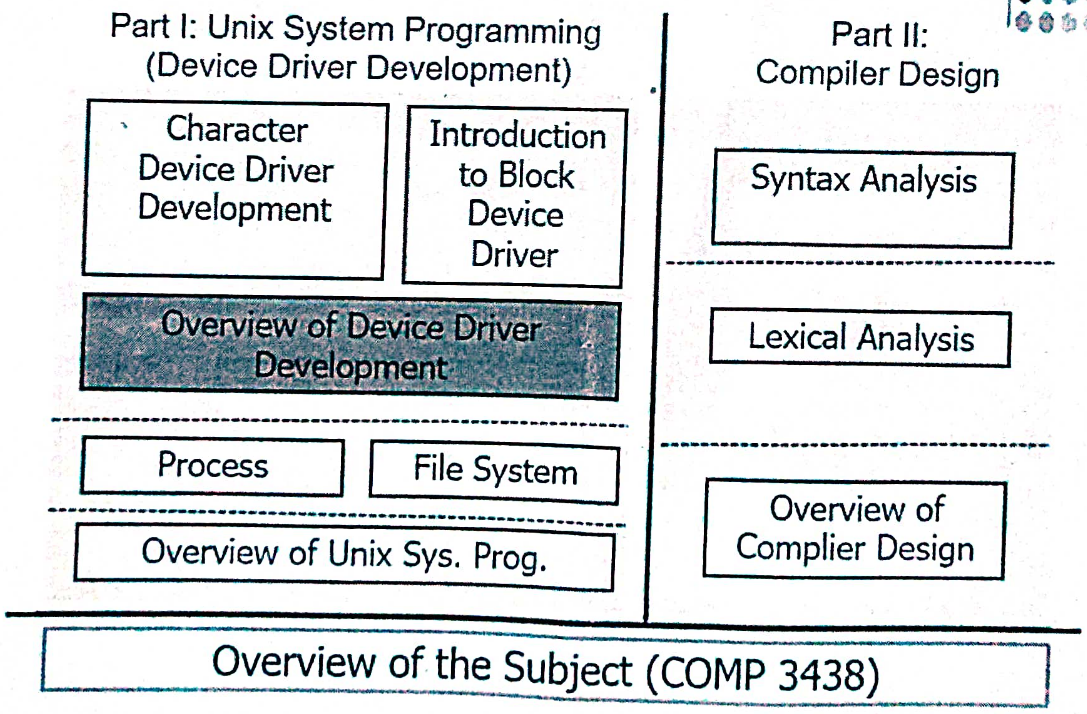
UNIX System Architecture

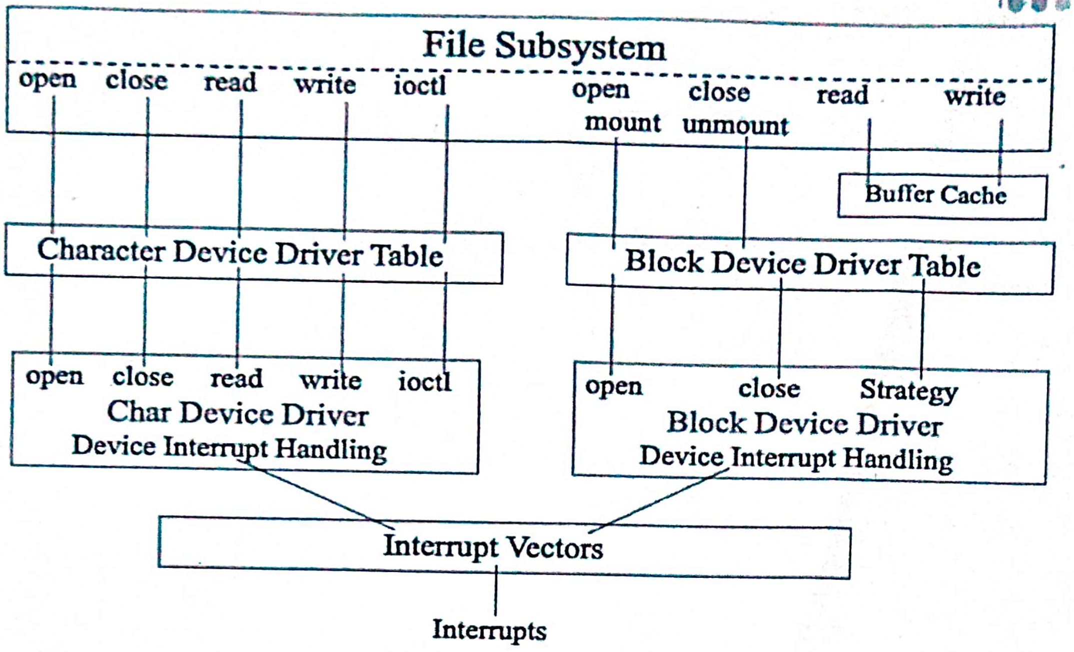


Course Organization (This lecture is in red)



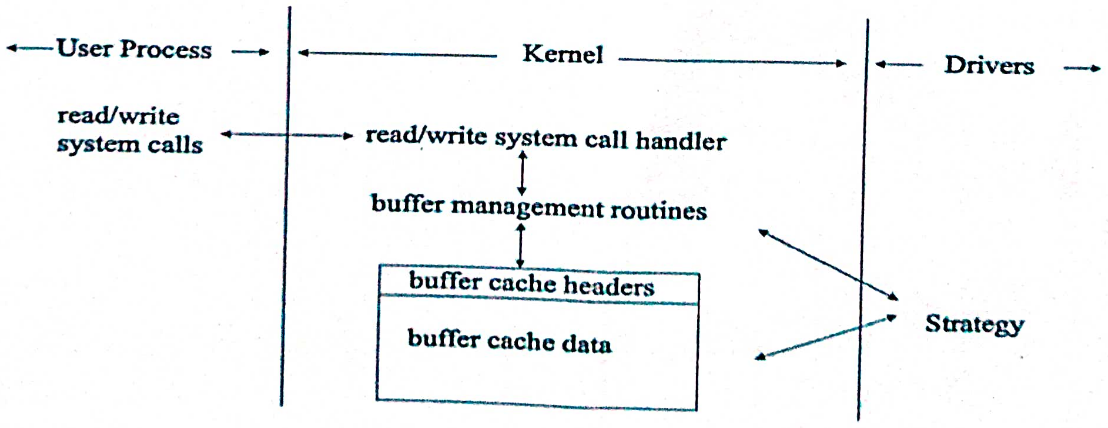
Part I: Unix System Programming
(Device Driver Development)
Character Device Driver Development
Introduction to Block Device Driver
Overview of Device Driver Development
Process
File System
Overview of Unix Sys. Prog.
Part II:
Compiler Design
Syntax Analysis
Lexical Analysis
Overview of Complier Design
Overview of the Subject (COMP 3438)

File Subsystem & Char/Block Device Driver Tables


File Subsystem
open close read write ioctl
open mount
close unmount
read write
Buffer Cache
Character Device Driver Table
Block Device Driver Table
open close read write ioctl
Char Device Driver
Device Interrupt Handling
open close Strategy
Block Device Driver
Device Interrupt Handling
Interrupt Vectors
Interrupts

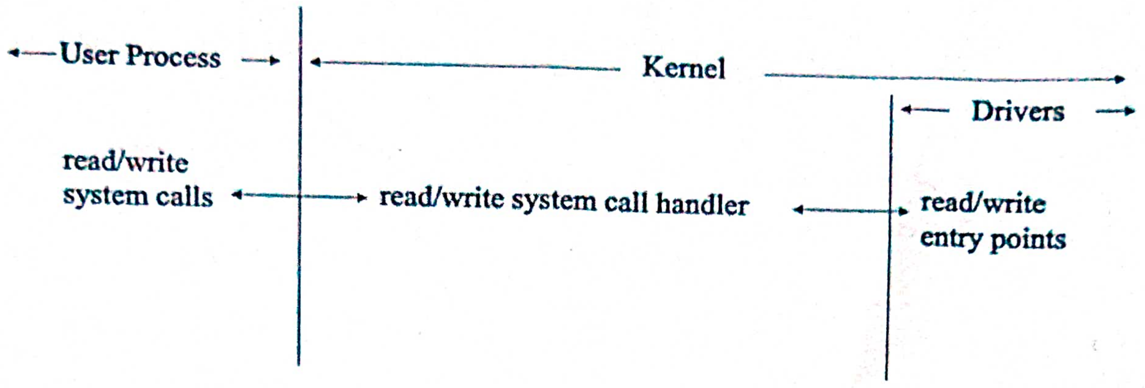
Block Device Drivers

- Block drivers – Communicate with O.S. through a collections of fixed-sized buffers.



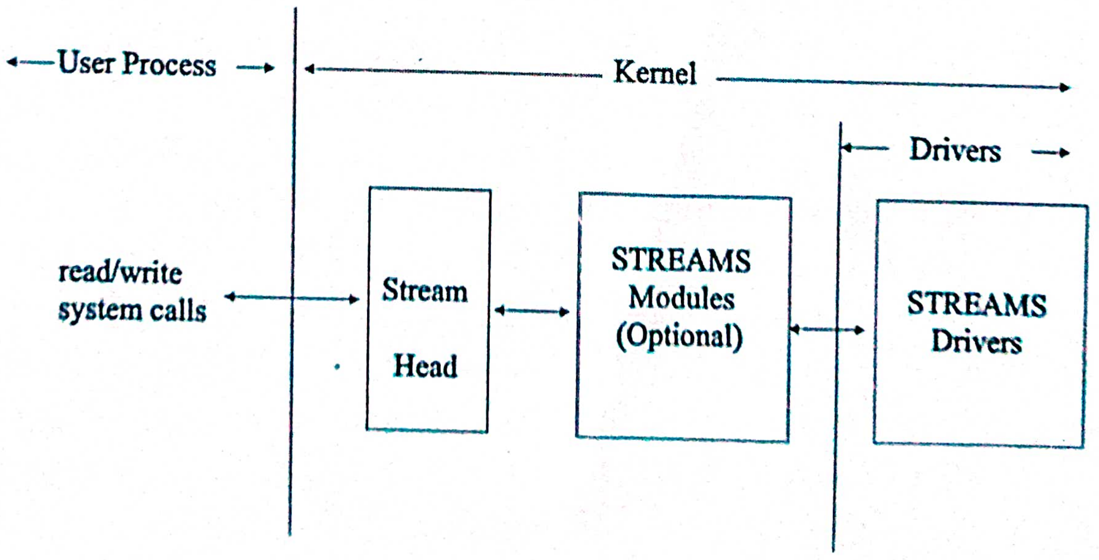
Character Device Drivers

- The communication structure of character device driver



STREAM Drivers

- The communication structure of Terminal Drivers



Char/Block Device Driver Tables

Character Device Driver Table

	open	close	read	write	ioctl
0	conopen	conclose	conread	conwrite	conioctl
1	testopen	testclose	testread	testwrite	nodev
3	dzbopen	dzbclose	dzbread	dzbwrite	dzbioctl
4	syopen	nulldev	syread	sywrite	syioctl
5	nulldev	nulldev	mmread	mmwrite	nodev

Block Device Driver Table

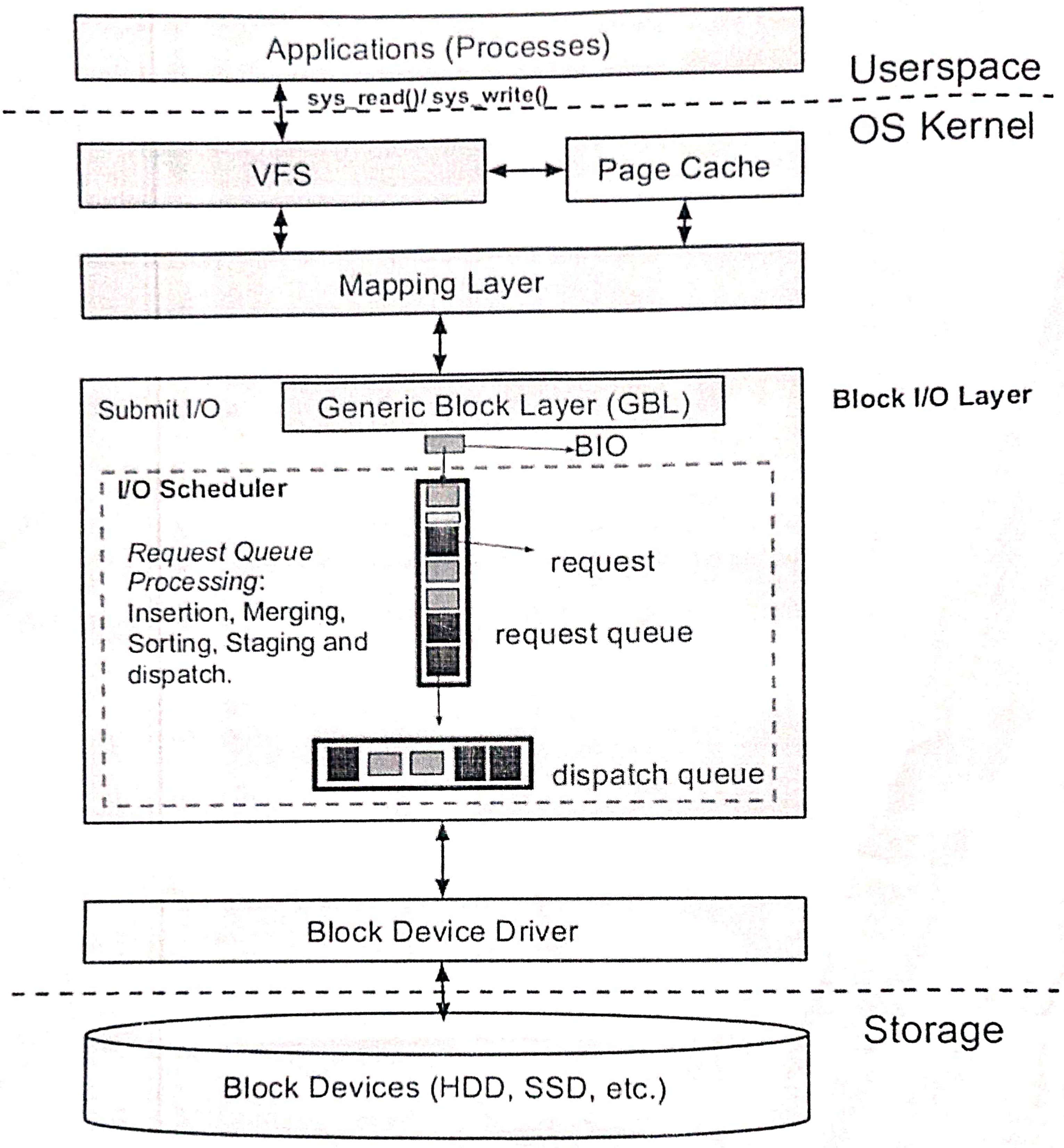
open	close	Strategy
gdopen	gdclose	gdstrategy
rdopen	rdclose	rdstrategy

Major differences between Block/Char drivers

- Major difference with block driver
  - Block driver – only interact with buffer cache
  - Char driver – direct interact with user requests from user processes
    - I/O requests are directly passed (essentially unchanged) to the drivers from processes
    - Char driver is responsible for transferring data directly to and from between kernel memory space and user memory space

General Programming Considerations

- Device drivers are parts of the kernel and not normal user processes, which means
  - We can only use the kernel routines
    - C library functions or system calls provided for users **cannot** be used
    - Some kernel routines may have the same names as C library functions, but they are totally different in implementation
- Make frugal use of stack (local arrays & recursive functions)
  - The stack space in the kernel is limited and not expandable
- Don't use floating-point arithmetic – May cause incorrect results
- Don't do busy wait that will prevent the whole system from doing nothing but responding to interrupts


Applications (Processes)
Userspace
sys_read()/ sys_write()
OS Kernel
VFS
Page Cache
Mapping Layer
Submit I/O
Generic Block Layer (GBL)
Block I/O Layer
BIO
I/O Scheduler
Request Queue Processing: Insertion, Merging, Sorting, Staging and dispatch.
request
request queue
dispatch queue
Block Device Driver
Storage
Block Devices (HDD, SSD, etc.)