LINUX → open source operating system

PROGRAMMING

① Virtualisation
② Concurrency
③ Persistance.

~~GNU/Linux~~

*may refer to Linux as GNU/Linux.

→ OS - Resource Manger. (manges hardware & all software).

→ Linux is an operating system similar to UNIX Windows and MacOS.

→ Work on linux started in 1991 by Linus Torvalds.

→ GNU Project started in 1983 by Richard M. Stallman.

↳ Goal:- to develop an OS that looked like UNIX-like OS.

↳ allowed users to copy, modify, update and redistribute

LINUX DISTRIBUTIONS — debian, ubuntu, centOS, fedora, linux mint, redhat → freware.

*all distributions use same kernel but the software may vary.

→ Advantages of Linux

↳ Open Source ↳ Flexible.

↳ Free ↳ Multiplatform.

↳ Secu

→ GNU General Public License (GPL) is used to govern modification & distribution of kernel.

→ GPL is often refered to as Copyleft

→ The GPL license allows users to the freedom to copy, modify & redistribute software.

→ Types of Kernel.

① Monolithic Kernel. ③ Hybrid Kernel

② Micro Kernel.

MacOS and Windows use hybrid kernel

monolithic kernel.

*GNU G
gover
of th

↳ Hardw

→ LINUX ARCHITECTURE.

*heart of Linux

↳ Kernel ↳ Utilities.

↳ Shell ↳ Bootloader.

↳ Applications ↳ Graphical user Interface (GUI)

HARDWARE LAYER	KERNEL
- includes physical and tangible parts.	- manages computer resources.
- external carpanet (mouse/keyboard)	- loads application, closes app, clears
- internal carpanet (HDD/GPU/RAM...)	memory etc.
	* drivers for all devices are present within KERNEL.
	- can run multiple tasks

* GNU General Public License (GPL) is refered to as Copyleft

PC comprises of procesor, memory (RAM), graphics cardd, hard drives, sound cards & much more. — all of these controlled by 'OS'.

* "Nano" is most commonly used text editor for programig in Linux.
* Linux Kernel is influenced by MINIX operating system & designed to be modular from start.

Date:_____

SHELL

- ways to cummicate with caputer
- translates instruction into language understandable to OS"

BOOTLOADER.

- system turned ON/restarted the BIOS - basic/(input/output) system conducts necesary integrity checks. over HDD and then loads bootloader.

* Device Drivers is responsible for managing hardware abstraction.

→ Graphical User Interface (GUI)

→ interface that allows users to communicate with OS through icons, windows or graphic called GUI

SHELL → the shell layer in Linux architecture is responsible for file management & execution.

→ Shell is basically command interpreter that receives the commands, translates them to binary instructions and then sends them to OS's kernel.

→ Command line Interface (CLI) is text based computer interface where users inputs a command and system executes it.

→ Shell Prompt

[username] @ [hostname] : [current_directory] $

TYPES OF SHELL — Bourne
- Z
- Bash
- C
- Korn

~~→ if a superuser (administrator)~~

* BASH is available on all Linux distributions.

Advantages of Linux.

① Open source.
→ anyone can modify update and redistribute.
→ to have full control of OS as "power users"

② Free cost
→ downloadable for free.
→ free updates/upgrades.
→ source code is available for free.

③ Security
→ has UNIX based security model.

④ Multiplatform.
→ can be adjusted to operate from any machine.

* Bootloader displays a screen where, if the system has more than one kernel, allows user to choose which he/she wants to be executed.
→ Daemons are applications that provide additional functionality to system, however are run in background.

Date:_____

→ TYPES OF KERNEL

① Monolithic Kernel.
→ all basic services interrupt handling, memory mangement computation, I/O are run in kernel space.
→ since everything is in same kernel, it is huge in size.
→ any addition of new features or fixing some bugs.

② Micro Kernel.
→ process reside in user-space in the form of servess
→ for eg one server resolves memory issues, another manges processes, while another is responsible for drivers.
→ breaks down code into smaller chunks. — reduces code size.
→ stability of os because we have minimal code running the kernel.
→ process execution is slower as compared to monolithic kernel.

③ Hybrid Kernel.
→ programed in a similar way to microkernel; however it's structure is similar to monolithic kernel.

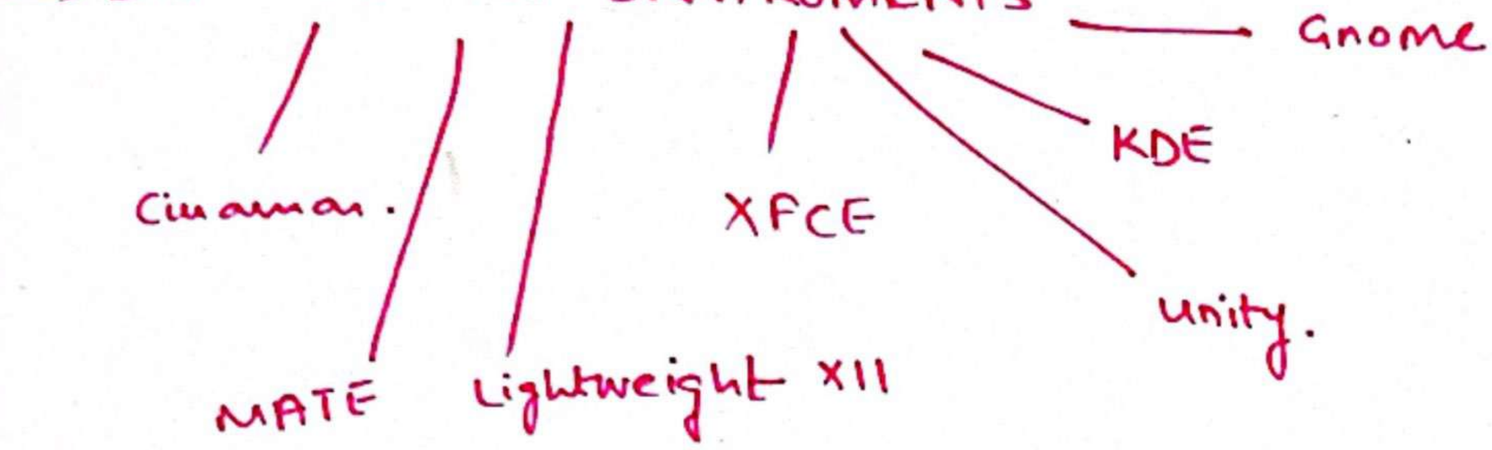
→ Utility software is fundamentally, system software designed to maintain a computer by analyzing, configuring, optimising basic tasks.
→ Application software are additional software that help a computer program. perform useful tasks.
→ whenever system is turned on/restarted the BIOS - basic input/output system conducts necessary integrity checks over HDD and then loads bootloader.

→ Windows Manager controls the way your desktop works: how it looks and acts. The windaw manger decides what kind of decoarations to put araund the windaws.

→ allaws user to open, close, resize, mave & track running windaws.

→ Desktop Enviearnet caprises of seveal caparnents like a windaw manager, file manger, graphical themes

→ LINUX DESKTOP ENVIROMENTS



→ if there is '#' as last character prompt then it means that terminal session has superuser (administrative) rights.

→ this means that either we signed in as root user (administrator) or terminal emulator providing superuser (administrative) priwileges has been selected.

int main (int argc, char *argv[])

command line argument.

*umask command. sets default permissions for newly created files & directories.

* /dev directory stores device files that represent hardware components & other devices.

*stderr used to display error messages

Date:

$history - displays all previas cannands that were executed.

$ history -c - clears cannand history.

$rwx - stands for Read, Write & Execute.

$ asdf — referenced by number $!1

$ zxcv — ~~referenced using b~~

$ qwerty - referenced using beginig characters $!qw

$ lmnop. - offset refers Lmnop $!-3.

→ we can refer by a number, beginig characters or an offset.

$1! — returns last executed cannand.

$ whatis ls — list directory caterts

$date. — display current date & time.

$cal — display current month's calendar.

$ whoami — view username.

$ hostname. — view hostname.

* $ df — view amount space in disk drive

$ free — free memory.

$ exit — to end terminal session.

$ whatis ls cp rm mv ____ move files

/ \ ____ rename files/directories.

list directory contents / copy files & directories.

*shell ignores extra blank ie one blank is equal to Wndred blanks.

* to pass all arguments provided to shell script to function we do $@ or $*

* appending an output to file
echo "Hello" >> file.txt

* software are installed an program files in C drive

* files in accesible waywhere.

x86 are

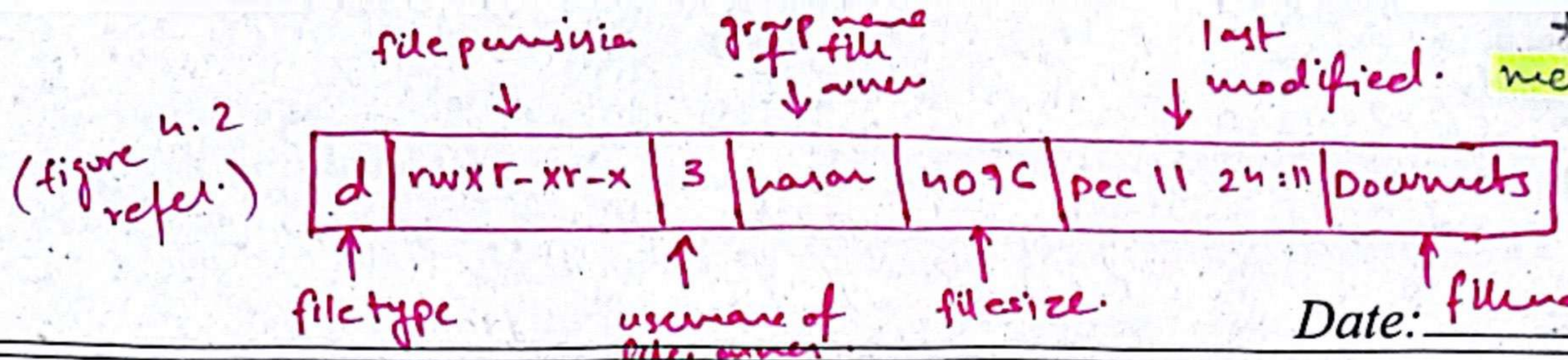
* "sudo" command (super user administrator)

$ date -u
$ date --utc
gives same output

Q) Which command is used to view the content of file one page at a time?
→ less.

Q) Which command would you use to change owner of file?

→ chown.

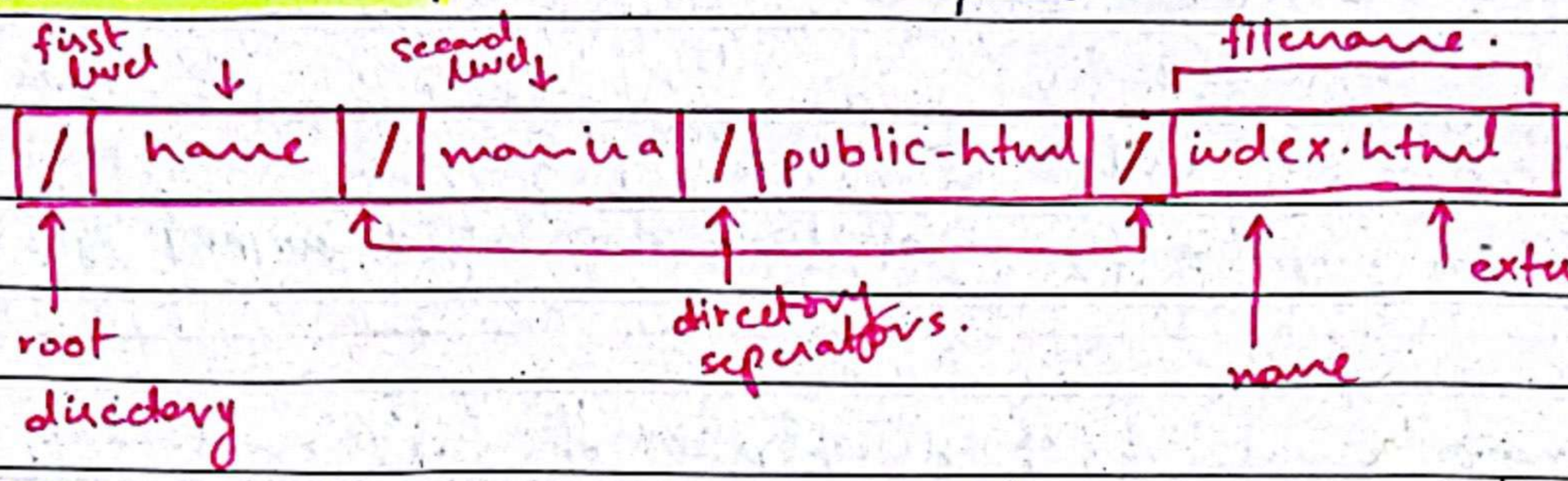


Date:

→ Linux files are organized in hierarchical directory structure

→ the very first directory in filesystem is called ROOT directory.

→ a forward slash / is used in b/w directories.



* file & directory name can have dashes, underscores & dots.
→ names are also case sensitive.

→ Current working directory can be checked using pwd (print working directory) command. $pwd.

→ Current working directory can be changed using cd (change directory) command. ~~$cd ..~~ $cd ..] if we want to change working directory to parent
$pwd

cd .. → parent directory

current working directory

$cd	— working directory changed to home directory.
$cd -	— " " changed to previous working directory.
$ cd ~<user>	— " " to home of specified user.
$ cd "<dir name>"	— quotes are used with directory names containing spaces.

→ ls (list) command is used to enlist all items (files & directory)

$ ls	— list content of current working.
$ ls /	— list content of specific directory.
$ ls -a	— list hidden files.
$ ls -l	— display properties of files.

* pathname relative to root directory is known as ABSOLUTE PATHNAME
* pathname relative to current working directory as RELATIVE PATHNAME.

single quotes preserve literal value of all characters
allows variable & comand substitution.

* 'set -e'
  ↳ to exit sciipt if comand returns non zero status
* ext4 is designed to provide high performance journalling for Linux.

Date:_____

Black/white	- regular file.
Red	- archive or package.
Dark Blue	- directory.
Pink	- image file
Light Blue.	- soft link
Green.	- executable file. (on linux are called BINARY FILES)

→ file comand prints brief description of file's contents. $file.

→ $less - provides convenient way to view files

→ creating Directories.

→ mkdir (make directory) comand can be used.
to create directories.

$ mkdir <directory-name> ...

→ Creating files.

→ touch comand can be used to create files

$ touch <filename.extension> ...

→ ~~cat~~ nano comand allows to enter text after creation of file. OR nano allows to edit the file.

$ ~~cat~~ nano > <filename.extension>

→ Copying Files/Directories.

→ $cp <item1> <item2> - copy item1 to item2.

→ $cp <item1><item2> ... <directory> (copying to directory)

→ cat comand prints content of file.

* System Call Intiface is responsible for maping system calls b/w user space & kernel space.
* Linux processes have unique identifier called PID (Processor Identifier)
* Linux ensure memory protection b/w processes by allocating seperate virtual memory space for each process.

Date: _____

→ Move/Rename files.

$ mv item1 item2 - move item1 to item2

$ mv <item1> <item2> ... <directory>

$ rm <item1> ... Remove files ←

→ Creating Links. (a special file that references another file by creating a pointer is known as LINK)

→ can be created using ln command. (hard link).

$ln <file> <linkname>.

→ to create softlink -s flag.

$ ln -s <item> <linkname>

CLASS NOTES

* Virtualisation and Isolation.

→ to run process simultaneously & store seperately.

→ isolating each process through virtualisation.

→ each process is supposed to go through cycles use fixed cycles alloted. allocated to it that's how you isolate it.

→ using "u &" operator we can tell how many application are to be runned.

→ the last application will run on current terminal while the rest will run in backgrnd.

→ cpu "Breeha"

argv[0] argv[1]

→ even if two process are same they will have different memory allocation because they are ISOLATED.

Date: _____

→ Low-level language.
  → assembly language are regarded as low-level language
  → machine/canig.
  → 1's and 0's BINARY NUMBERS.
→ High level language
  → human readable
  → is known as source code & is saved to source file.
→ caupiler. → translates code to executable file. or binary. file.
→ Scripting language execute by using special program "interpreter"
→ vim is a text editor
    $ vim <filename>
    $ nano <filename>
→ This line is known as interpreter directive or shebang.
  /bin/bash.
→ When non-binary file is executed by kernel the file line of file is read by kernel and if line begins with #! kernel uses that line to determine interpreter.
→ In Bash, comments begin with # symbol.
→ Executing Shell Script
  ① bash <script>
  ② chmod +x <script>
  ③ ls -l <script>.
  ④ ./script
→ A shell script is a simple text file containg sequence of shell commands.

→ After writing the shell script ensure that name of file is unique and it should not conflict with default Linux commands.
→ Typically shell scripts end with .sh file extension.
→ The list of directories is stored in shell environment variable, PATH.

Date:_____

→ The script name is explicitly preceded with it's path in order for script to execute.

$ echo $PATH.

→ To move our shell script into .directory.

$ mv hello.sh /usr/local/bin.

~~$ mv hello~~.

→ To include current directory ~~in~~ within PATH

$ export PATH = $PATH

→ To view current process type ps:

$ ps

→ Syntax to declare variable.

$ VAR = <val>

→ Global variables are called environmental variables (can also be exported to other shells.) while the variable create available only in shell process which it was created is called shell variables.
→ echo command can be used to print variable's content

$ echo $sherlock.

→ to print name of variable only

$ echo sherlock.

→ ${VAR} are useful for concatenating a variable value with string.
→ the set shows list of all variables & their values.
→ grep extracts line that shows value.
→ To remove value of variable we assign " " empty strings.

* BASH has only one type 'string'. The value is therefore string.

* VAR = val is an equality operation
* VAR=val is an assignment " "

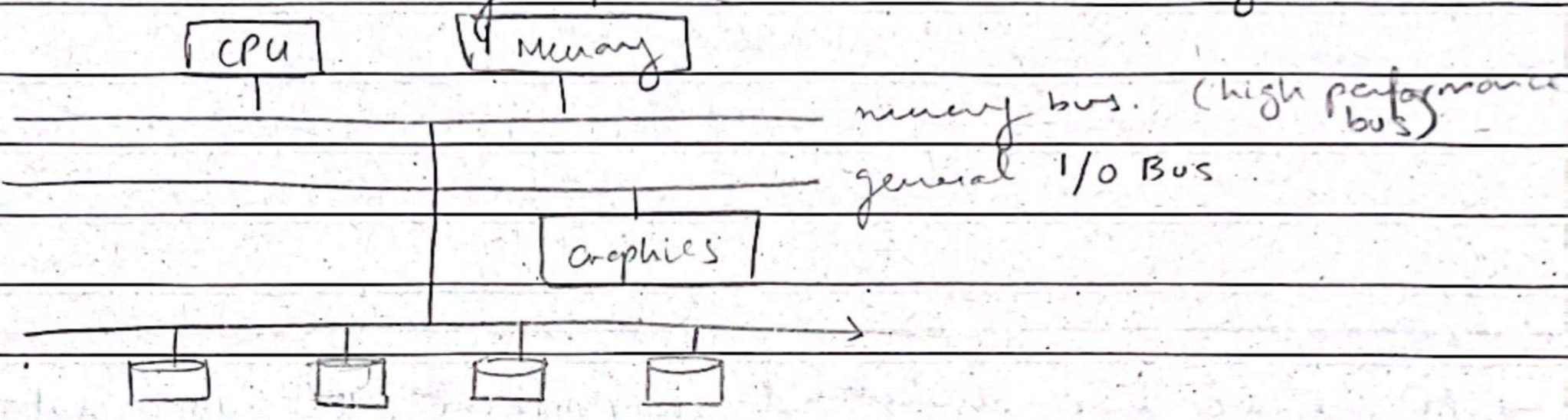
* a program is a blob of 0's and 1's (binary data) which consist of set of instructions
* " " can also contain other resources.
* A copy of program is called process.

Date:_____

→ To remove variable itself we will use unset keyword..
→ $printenv - prints all ^variables or $env. environment
→ $ set - displays all shell & environment variables. (sorted order)
→ $ echo $PATH - to show an environment variable's content.

CLASS NOTES

* Program doesn't have any output (patting / counting / watching a drama)
*



P_1	P_2	→ P_1 is faster even though it has lower clock rate.
2 GHz	2.4 GHz	
16MB	8MB	→ if you go by process, P_2 is cheaper.

* Why gold is used in cache?
→ higher conductivity
→ semi-conductors are made up of silicon
→ heat sync is on top of metal chip
→ fins increase surface area therefore heat dissipation occurs.

→ Cache is present in
① Processor
② Hard Disk

Date:_____

→ I/O chip special chip to connect peripheral devices and it also works with network devices.

→ DMI (Direct Media Interface)

* 70% of code are found in device drivers.
  → most device drivers written by "amateurs"
  → tend to have many more bugs

* eSATA

* Peripheral components are faster than eSATA

* Canonical device helps understand how device interact.

* Firmware — device itself

* firmware is housed in harddisk while driver is physically located in OS (SSD/etc)
  → Status is called polling.

→ Aliases are are element of environment that does not show either set or printenv.

→
```
$ vegetable = "carrot"
$ export vegetable.
$ printenv | grep vegetable
  vegetable=carrot.
```

→ Positional parameters.

→
```
#!/bin/bash
echo "Filename = $0"
echo "1st Argument = $1"
echo "2nd Argument = $2"
```

Date:_____

$posit-param.sh "Breeha" "Namel"

Filename = /home/namel/bin/posit-param.sh

1st Argument = Breeha

2nd Argument = Namel.

→ (*) willdcard which can be used to match lists of files in any directory

$ echo *

→ $ $ echo D*

Desktop Downloads Documents.

→ $ echo *s

Documents Downloods Pictures Templates Videos.

→ (~) tilde, when used at staut of a phrase, it expands into usei's home directory

$echo ~

/home/namel.

→ Multiple text patterns can be creatid using brace expasion from patter cantaing braces.

$echo Top-{A,B,C}-Bottom.

→ Syntax of Arithmetic Expresion.

$ ((expresion))

→ Parameter expasian is used to replace variable by it's cauents

${parometei}

→ Length of string stored in a paiameter can be expanded as follows.

${#parametei}

→ for extracting a part of ~~osiing~~ string stored in parameter

${parameter:offset}

${parametei:offset:length}

Date:

→ Exit Status

→ they given a value known as exit status when they terminate

→ integer value ranging between 0 to 255 where 0 indicates success and a non-zero exit status indicates failure.

→ [[expression]]

→ the above evaluates an expression either to be true or false

→ string =~ expression.

→ if returns true then it means they matched & returns false otherwise.

→ ((expression)) – Arithmetic expression.

→ returns true '0' if arithmetic evaluation is non-zero

→ '&&' for logical AND.

→ '||' for logical OR

→ '!' for logical NOT

IF CONDITION

→ if <condition>
then
<command>
fi

→ if <condition>
then
<command>
else
<command>
fi

→ if <condition>
then
<commands>
else
if <conditions>
then
<commands>
fi
fi

Date:_____

→ CASE CONDITION

```
case <word> in
    pattern1)  <commands> ;;
    pattern2)  <commands> ;;
    ...
    patternN)  <commands> ;;
    *)         <commands> ;;
esac.
```

LOOP.

→ WHILE ~~CONDITION~~

```
while <condition>
do
    <commands>
done.
```

→ FOR LOOP

```
for <var> in <words>.
do
    <commands>
done.
```

→ C Language FOR LOOP

```
for ((expression1; expression2; expression3))
do
    <commands>
done.
```

* Eclipse & Netbeans are popular IDEs for developing software on Linux.

* early time-sharing system was developed to maximise efficiency of CPU usage by allowing multiple users to interact with system.

Date: _____

→ NESTED LOOPS

```
while <condition>
do
        for (exp1 ; exp2 ; exp3)
        do
            ...
        done
done
```

Outer Loop

Inner Loop

BOOK 2

CHAPTER 2

Von Neumann Model

→ When a program runs, it executes instructions. Many millions of times every second, the processor fetches instruction from memory, decodes it & executes it.

→ But when a program runs, alot of other things are also going on

→ There is a body of software, that is responsible for making it ~~easy~~ easy to run programs, allowing programs to share memory, enabling programs to interact with devices. That body of software is called OS.

In Short, the Operating System simplifies system use by managing the execution of programs & other resources.

* Real-Time Operating System is designed to provide quick responses to input and is often used in embedded system.

What mechanism does an OS use to virtualize memory?
→ using combination of hardware and software to map virtual addresses to physical addresses

*malloc( ) will allocate memory

Date: _____

→ **VIRTUALIZATION** (OS also called virtual machine).

→ the OS virtualizes resources (CPU, memory, disks) to create an easier and more powerful virtual form of these resources.

→ It allows multiple programs to run concurrently ~~by creating~~ the illusion of many (at same time) even if there's only one CPU.

→ the process whereby OS makes use of hardware support to map virtual addresses to physical is known as Address Translation

→ **CONCURRENCY**

→ It involves handling multiple tasks at same time, both within OS and in multi-threaded programs.

→ The OS must manage concurrency issues, ensuring that processes or threads do not interfere with each other inappropriately.

→ **PERSISTENCE**

→ Data in memory is volatile, thus the OS must manage persistent storage on devices like hard drive or SSDs. (data are managed)

→ The file system, a part of OS, is responsible for reliably & efficiently storing data on disks.

→ The OS provides simple tools, known as APIs that let users & programs interact with system, like running programs or accessing memory & files.

Date:_____

→ GOAL OF AN OS

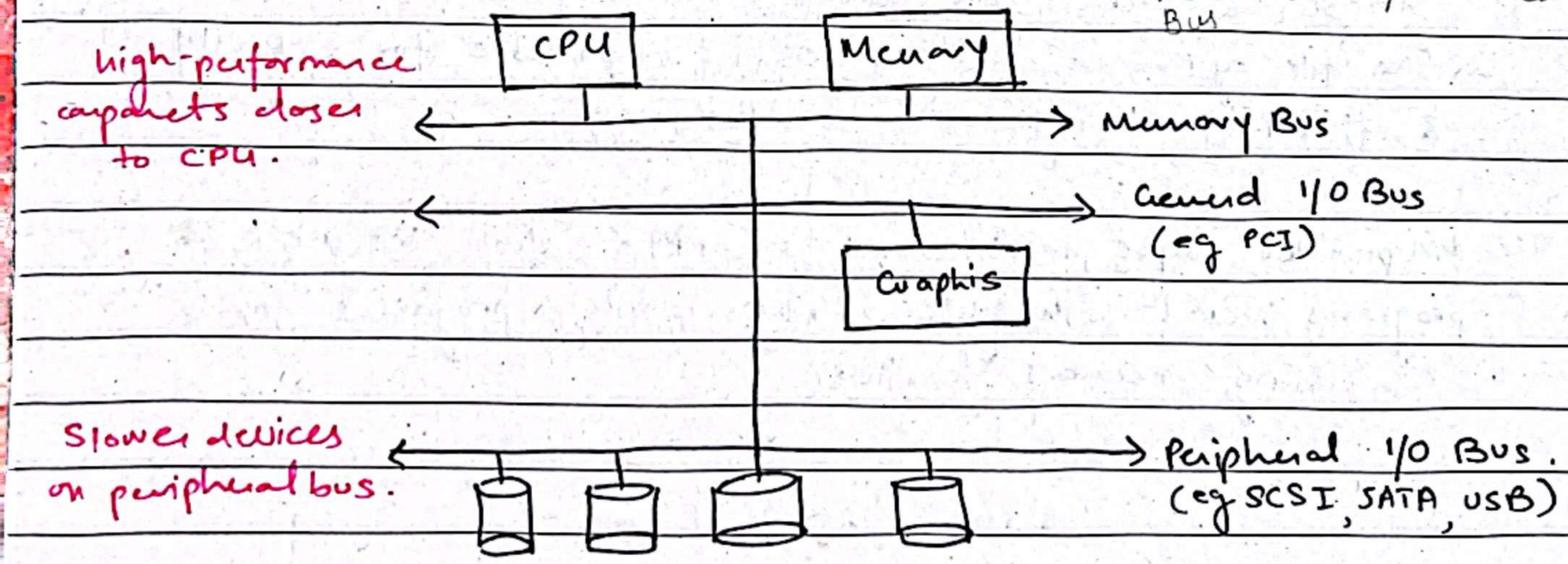
(1) Making things simple by hiding complex details.
(2) Efficiency, the OS tries to do it's job without slowing things down too much.
(3) Prevents programs from harming each other.
(4) Needs to be stable & run without crashing.
(5) Protects system from attacks & keeps data safe.

CHAPTER 36

→ Input/Output (I/O) is critical for computer systems. Without I/O, a program would either produce same result each time or have no output.

→ A typical system architecture includes a CPU connected to main memory via a memory bus, with devices connected through general I/O bus (e.g PC) and a peripheral bus (e.g SCSI, SATA, USB).

*CPU attached to main memory via Memory Bus



PROTOTYPICAL SYSTEM ARCHITECTURE

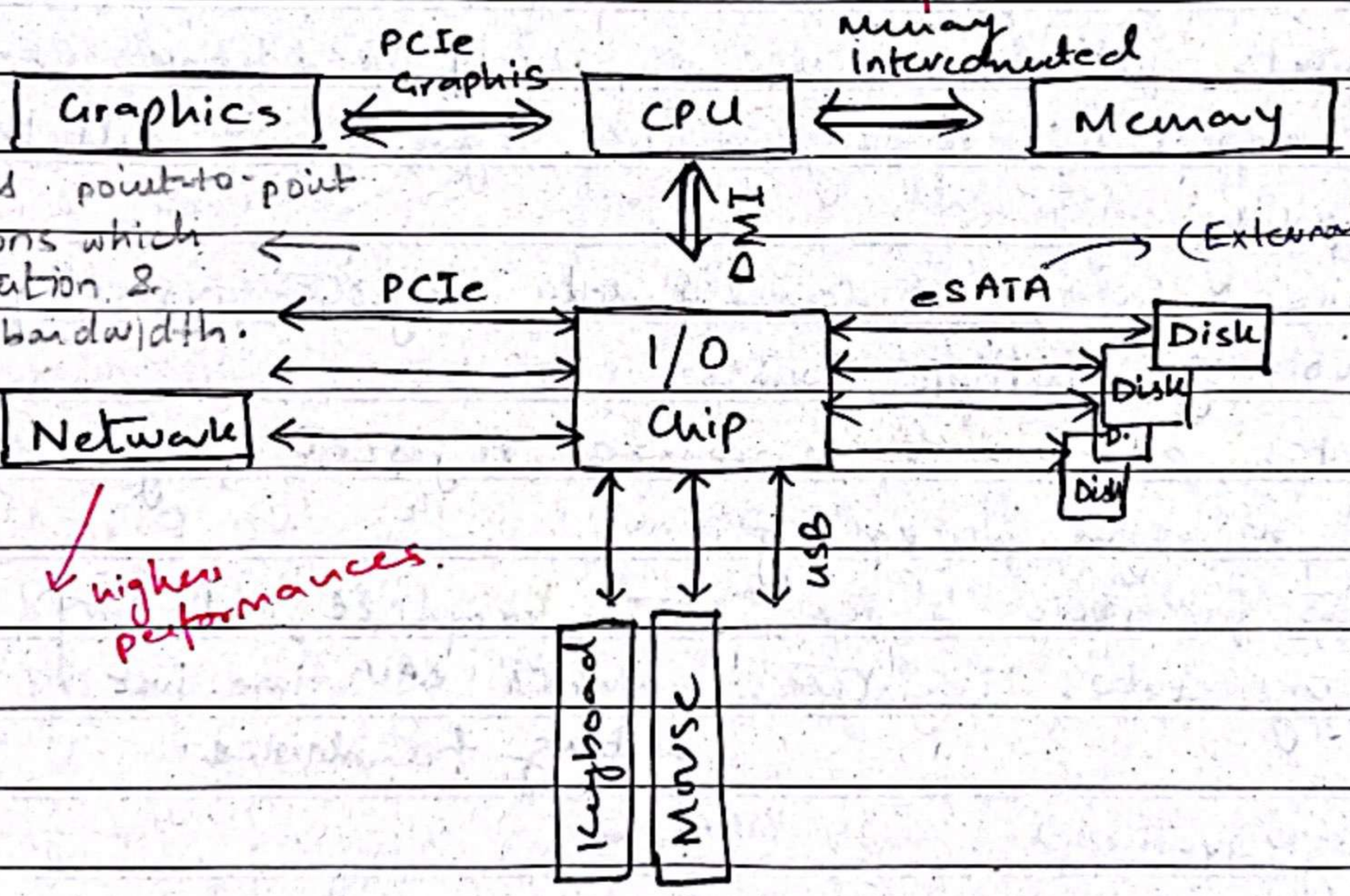
Date: _____

→ The CPU connects to an I/O chip via Intel's proprietary DMI (Direct Media Interface). and the rest of devices are connected to this chip via a number of different interconnects

→ On the right are or more hard drives connect to system via the eSTATA interface; ATA then SATA and now eSATA represent evolution of storage interfaces over past decades.

→ Below the I/O chips are a number of USB (Universal Serial BUS) connections which in this depiction enable a keyboard and mouse to be attached to computer.

→ On left, other higher performances devices can be connected to system via PCIe (Peripheral Component Interconnect Express)



MODERN SYSTEM ARCHITECTURE.

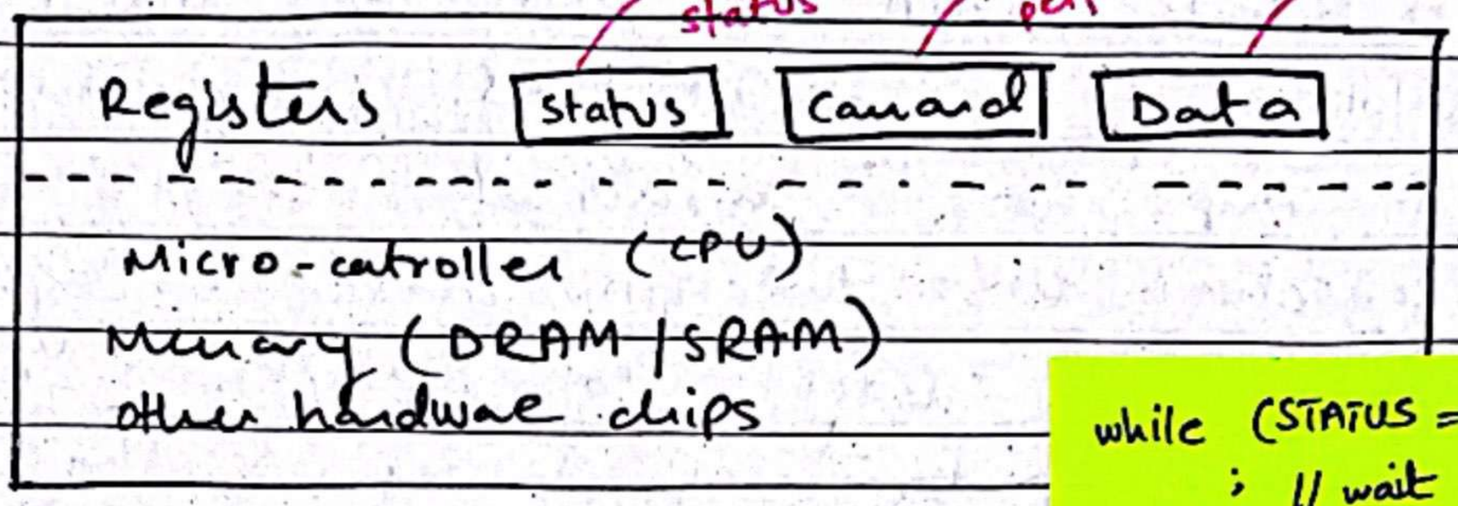
*the introduction of multiprogramming influenced modern OS

★ Polling is repeatedly reading status register.

Date: _____

helps understand how devices interact

→ Canonical Devices typically have two components: a hardware interface & internal structure. The hardware interface allows the system to control device while internal structure implements device's functions.




```
while (STATUS == BUSY)
    ; // wait until the device
      is not busy.
```
POLLING

→ The protocol has four steps.

(1) OS waits until the device is ready to receive command by repeatedly reading status register, we call this polling.

POLLING

(2) OS writes data to device's data register, typically involving multiple writes.

(3) OS writes a command to command register, signalling device to begin processing.

(4) OS waits for the device to complete task by polling status register. - wastes CPU time just waiting for device.

→ Reducing Polling overhead.

→ Instead of polling, the OS can issue a request & context to switch to another task. When device completes task it sends an INTERRUPT to CPU which handles completion of request. - more efficient use of CPU & disk.

If device is fast → POLL is best
If device is slow → INTERRUPTS is better.

* The method where the operating system performs operation by writing to or reading from device registers is known as MEMORY-MAPPED I/O. Date:_____

→ Interrupts allows CPU to perform other tasks while waiting for I/O operations to complete.

→ Polling vs Interrupts

→ When to Poll

→ Polling is more efficient for fast devices where the time spent polling is minimal compared to overhead of handling interrupts.

→ ~~Interrupts are bet~~

→ When to Interrupt

→ Interrupts are beneficial for slow, as they free the CPU to perform other tasks instead of waiting.

→ Hybrid Approach.

→ A two-phased approach where the OS initially polls & switches to interrupts if the device is slow, balancing the benefits of both methods.

→ DIRECT MEMORY ACCESS (DMA)

→ In PIO, the CPU is burdened with the task of moving data b/w memory & data device, which is inefficient.

→ DMA allows data transfer b/w devices & memory with minimal CPU involvement.

→ The OS programs the DMA engine with the location and size of data and destination device. The DMA controller handles the transfer & raises an interrupt when complete.

→ During DMA transfers, CPU is free to execute other processes, significantly improving system efficiency.

Date:_____

→ Early systems used explicit instructions (eg 'in' and 'out' on x86) to communicate with devices by reading and writing to device-specific registers.

→ Only the OS can issue these instructions to maintain control and security.

→ Device registers are mapped to specific memory addresses, allowing OS to interact with devices.

→ Device drivers encapsulates the specifics of device interactions allowing OS to remain device-neutral.

→ over 70% of OS code is found in device drivers.

→ drivers are often written by "amateurs" they tend to have many more bugs and thus are primary contributor to kernel crashes.

→ Four types of registers

(1) Control.

Address 0x3F6 = 0x80 (0000 1RE0): R = reset, E=0 means "enables interrupt"

(2) Command Block Register

Address 0x1F0 = Data Port

Address 0x1F1 = Error

...

CLASS NOTES

* Cache is found on processor & hard disk.
  → it is fastest for data retrieval.

* Describe atomic operation concerning hard disk updates.
  → Atomic operation with respect to hard disk updates refers to process where drive manfacturers can only guarantee the succesful completion of single 512-byte write operation at a time.
  → either will fully write or not at all.

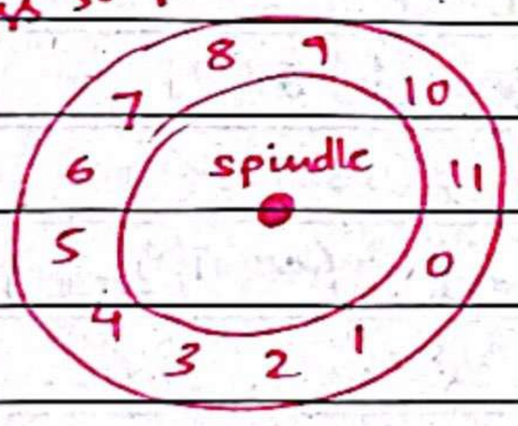
Date: _____

CHAPTER 37: Hard Disk Drives.

→ Hard disk drives are the main form of persistent data storage in computers.
  → the drive consists of sectors (512-byte blocks)
  → Atomic: When updating the disk, drive manfacturess can only guarantee making a single 512-byte write.
  → It will either write or won't write at all (called torn write)
  → Accessing blocks near to each other is faster (sequential read/write) than accessing blocks far away from each other (called random acess)
  → Address Space: We can view disk with n sectors as an array of sectors; 0 → (N-1)

updates to disk are atomic ←

due to reduced need for moving read/write heads extensively across surface. ↙



* a disk with single track and 12 sectors.

* eg 10,000 RPM: a single rotation takes about $\frac{60}{10000}$ = 6ms

① Platter (Aluminum coated with thin magnetic layer)
② A circular hard surface. Data is stored by inducing magnetic changes to it.
③ Each platter has 2 sides, each of which is called a surface
④ The platters are all bound together around spindle.
⑤ Spindle is connected to a motor that spins the platter around.
⑥ The rate of rotations is measured in RPM (Rotations per Minute)

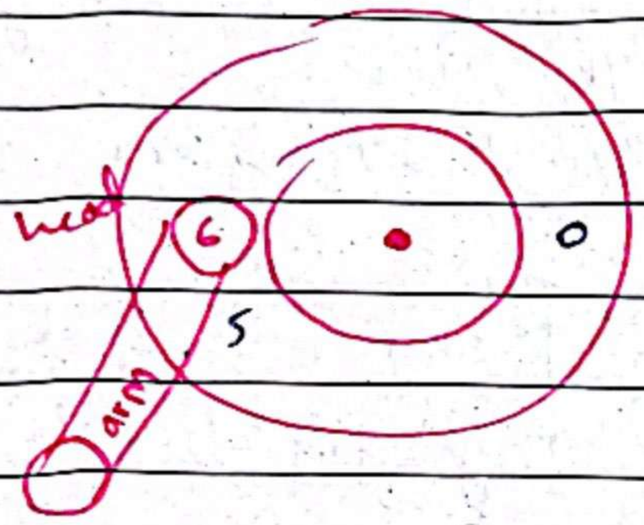
Rotational Delay

(BEST CASE) R/2
(WORST CASE) R-1

Date:_____

⑦ Data is encoded on each surface in concentric circles of sectors; we call one such concentric circle a track.

⑧ A single surface contains many thousands & thousands of tracks



→ Disk head (one head per surface of drive)

→ The process of reading & writing is accomplished by disk head.

→ attached to single disk arm, which moves across surface

→ Rotational Delay. Time for desired sector to rotate.

e.g full rotational delay is R & we start at 6.

→ Read sector 0: Rotational delay = R/2

→ Read sector 5: Rotational delay = R-1 (WORST CASE)

→ Seek Move the disk arm to correct track.

→ Seek Time: Time to move head to track contain disk operation.

→ MOST COSTLY OPERATION

* The number of sectors per track ~~change~~ increases across a platter from the inner to outer tracks. due to increased circumference.

PHASES OF SEEK Date:_____

Acceleration → Coasting → Deceleration → Settling.

- Acceleration → disk arm gets moving
- Coasting → arm is moving at full speed
- Deceleration → arm slows down
- Settling → head is carefully positioned over correct track.

→ The final phase of I/O
  → data is either read from or written to surface.

→ Complete I/O time
  ① Seek ② Rotational delay ③ Transfer.

→ Sequential reads can be properly serviced even when crossing track boundaries.
  → without TRACK SKEW, the head would be moved to next track but desired next block would have already rotated under head.

→ cache holds data read from or written to disk
  → small amount of memory
  → allow drive to quickly respond to requests

→ WRITEBACK: acknowledge a write has completed when it has put the data in its memory.

→ WRITE THROUGH: acknowledge a write has completed after write has actually been written to disk.

Date:_____

$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$.

→ Rate of I/O ($R_{I/O}$)

$$R_{I/O} = \frac{Size_{transfer}}{T_{I/O}}$$

CHEETAH EXAMPLE

$T_{seek} = 4ms$. $T_{transfer} = 30\mu s$.

$T_{rotation} = 2ms$ $T_{I/O} = 4 + 2 + 30\,\mu s \approx 6ms$

→ RPM = 15,000 → RPS = $\frac{15000}{60} = 250$

$T_{rotation} = \frac{1}{250} = 4ms$. On avg disk will encounter half rotation.. $4/2 = 2ms$

→ Random workload (4KB)

$125 \frac{MB}{s} \rightarrow 125 \times 1024 \frac{KB}{s} = 128{,}000 \frac{KB}{s}$

$T_{transfer} = \frac{4}{128000} = 31.25\mu s \approx 30\mu s$.

* Random workload (8KB)
* Sequential workload (100 MB)

* huge gap in drive performance b/w random & sequential.

* RAID enhances reliability allowing systems to tolerate the loss of one or more disks without data loss. Performance is enhanced through use of multiple disks in parallel increasing the throughput.

Date: _____

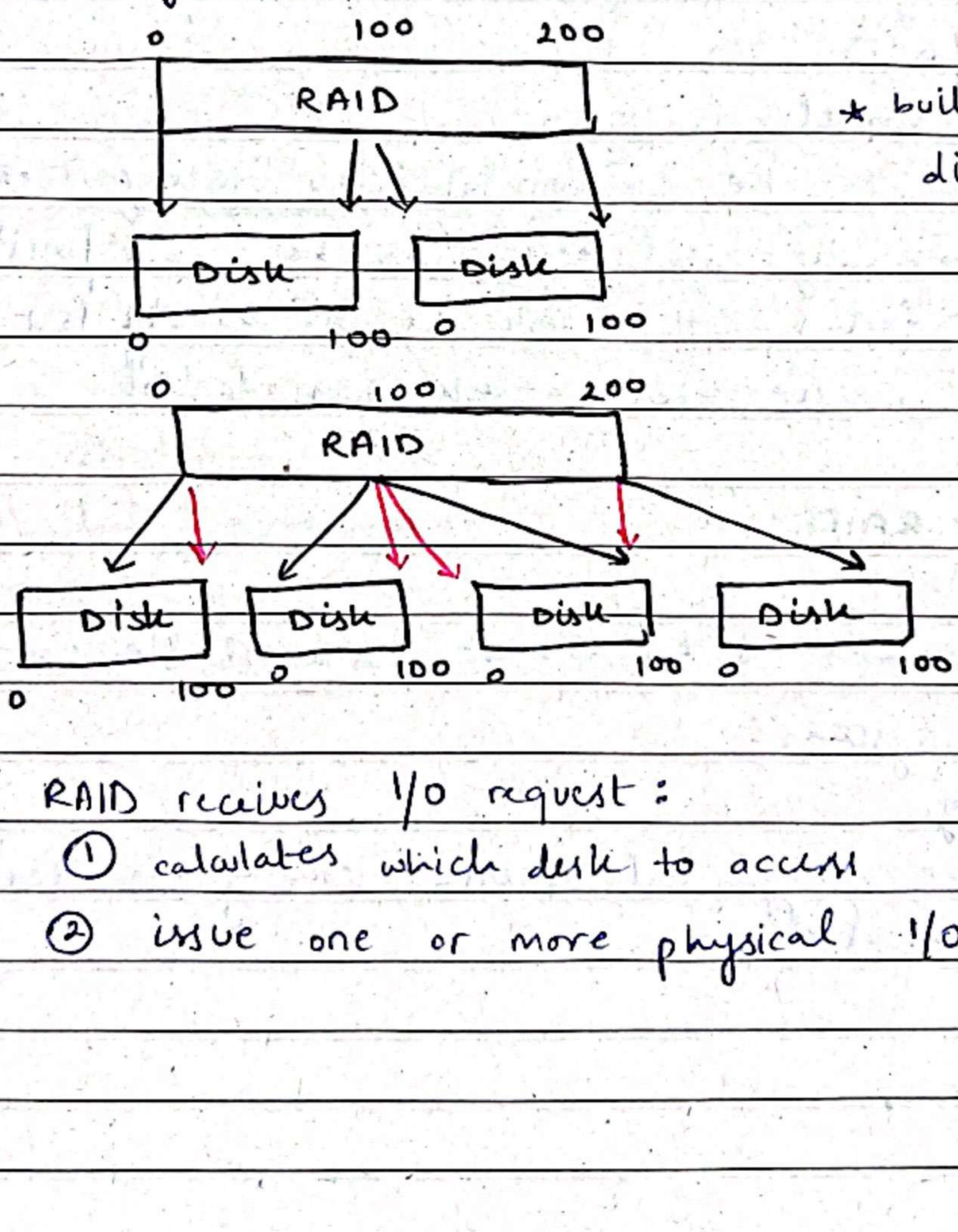
CHAPTER 38: Redundant Arrays of Inexpensive ~~Data~~ Disks (RAIDS).

RAID (Redundant Array of Inexpensive Data).

→ Use multiple disks in concert to build faster, bigger & more reliable disk system.

Advantages

① Performance & Capacity : Using multiple disks in parallel.
② Reliability : RAID can't tolerate the loss of disk.



* build large, fast disk from smaller ones

→ When the RAID receives I/O request :
① calculates which disk to access
② issue one or more physical I/Os to do so.

Date:

→ Ra[illegible] internals

① Microcontroller (run firmware to direct operation of RAID)
② Volatile memory (such as DRAM, buffer data blocks)
③ Non volatile memory (buffer writes safely)
④ Specialized logic to perform parity calculation.

→ RAIDS are designed to detect & recover from certain kinds of disk faults
→ Fail-Stop fault model.
→ a disk can be in two states : working or failed
→ working : all blocks can be read / written
→ failed : the disk is permanently lost
→ Can imediately observe when a disk has failed

How to evaluate RAID?
① Capacity
↳ how much useful capacity is available to systems?
② Reliability
↳ how many disk faults can given design tolerate?

Date:_____

③ Performance
- N : number of disks
- C : capacity of 1 disk
- S : sequential thrughput of 1 disk
- R : random " "
- D : Latency of are small I/O operation

④ ~~Redundancy~~ Capacity.
→ without redundancy → (N x B)
→ with redundancy → $\frac{(N \times B)}{2}$

RAID LEVELS 2,3 X
→ Raid 0, Raid 1, Raid 4, Raid 5

RAID-0 (stripping blocks)
→ spread blocks across disks in round-robin fashion.
→ no redundancy
→ excellent performance & capacity.

Disk 0	Disk 1	Disk 2	Disk 3	
0	1	2	3	→ Stripe (blocks in same row)
4	5	6	7	
8	9	10	11	

Date:_____

Chunk Sizes.

→ mostly affects performance of the array.
  → SMALL
    → increasing parallelism
    → increasing positioning time to access blocks.
  → BIG.
    → reducing intra-file parallelism
    → reducing positioning time.

MAPPING PROBLEMS.

→ Disk = A % num of disks. A: block address.

→ offset = $\frac{A}{\text{num of disks}}$

e.g request arrives for block 14.

→ given that there are 4 disks

14 % 4 = 2 (disk 2)

→ exact block is calculated as $\frac{14}{4} = 3$ (block 3)

Date:_____

RAID - 0 Advantages

(1) Capacity (Raid 0 is perfect)
(2) Performance (" " is excellent)
→ all disks often utilized in parallel.
(3) Reliability (Raid 0 is bad).
→ any disk failure will lead to data loss

RAID - 1 (tolerates disk failures) - (mirroring)

→ copy more than one of each block in system.
→ copy block places on seperate disk.

Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

RAID - 1 ANALYSIS

(1) Capacity (expensive)
(2) Reliability (does well)
→ can tolerate failure of any one disk (up to N/2 failures)

Date:

RAID-4 (have single parity block) - (saving space with parity)

→ parity block stores redundant information for that stripe of blocks.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4	* P: Parity
0	0	1	1	P0	→ XOR for all bits.
2	2	3	3	P1	
4	4	5	5	P2	
6	6	7	7	P3	

RAID - 4 ANALYSIS

① Capacity (useful capacity is N-1)

② Reliability (tolerates one disk failure & no more)

33-36.

Date:

→ The parity disk can be bottleneck

→ eg update blocks 4 and 13 (marked with *)

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4.
0	1	2	3	P0
*4	5	6	7	+P1
8	9	10	11	P2
12	*13	14	15	+P3.

→ disk 0 and disk 1 can be accessed in parallel.

→ disk 4 prevents any parallelism.

(Sensible) throughput under random small writes is (R/2) MB/s.

Slide 38

Date:

RAID-5 (Rotating Parity)

→ is a solution of small write problem.
  ↳ rotate parity blocks across drives.
  → remove parity-disk bottleneck for RAID-4

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

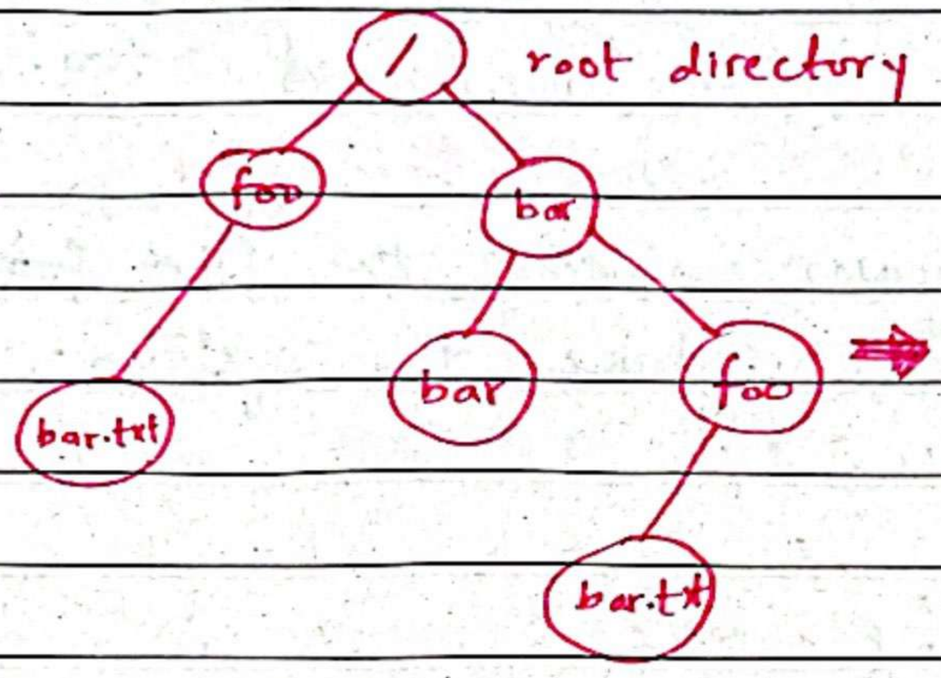
~~CHAPTER 38~~ (Chapter 39 from Book 2): Interlude: Files & Directories.

→ Keep a data intact even if there is power loss.
  ① HDD ② SSD ③ RAID storage.
→ Two key abstractions in virtualization of storage.
  → file → directory.

→ A file is a stream (linear array) of 8-bit bytes.
→ each file has a high-level name like "myself.txt"
→ each file has a low-level name (inode number). The user is not aware of this name.
→ Filesystem has responsibility to store data persistently on disk.
→ Directory is like a file, also has high-level & low-level name.

Date: _____

→ Directory is like a file
  → it contains list of (user-readable name, low-level name) pairs
    → the low-level name (inode number)
    → each entry in a directory refers to either files or other directories.



✓Valid files:
/foo/bar.txt
/bar/foo/bar.txt

✓Valid directory:
/ /bar/foo/
/foo
/bar
/bar/bar

CREATING FILE

→ Use system call & appropriate flags to create file.
  → system call : open()
  → flag: O_CREAT flag

```
int fd = open("foo", O_CREAT | O_WRONLY | O_TRUNC);
```

- fd → file descriptor
- open → system call
- "foo" → filename
- O_CREAT → create file
- O_WRONLY → write only when opened
- O_TRUNC → truncate file size to zero (remove any existing content)
- | → flags.