CS101: All in one book for Competitive Exam

Avijit Naskar

January 2025

Contents

Part I

Discrete Mathematics

Chapter 1

Mathematical Logic

Chapter 2

Linear Algebra

Chapter 3

Graph Theory

3.1 Graphs

3.1.1 Terminologies

- **Graph** A graph G=(V,E) consists of a set of objects $V = v_1, v_2, v_3, ...$ called vertices and $E = e_1, e_2, e_3, ...$ called edges.
- **Edges** An edge e_k is an unordered pair of vertices (v_i, v_j).
- **Vertex** Vertices are the nodes/points of both sides of edges.
- **Adjacent Vertices** If two vertices connected with same edge.
- **Degree or Valency** Number of edges incident on a vertex
- **Regular Graph** A graph which has all vertices of same degree
- **Isolated Vertex** A vertex of degree zero
- **Pedant Vertex** A vertex of degree one
- **Common Vertex** A vertex
- **Null Graph** A graph with no edges

3.1.2 Theorems

1. The sum of degrees of all vertices in a graph: $\sum_{i=1}^{n} d(v_i) = 2e$
2. The number of vertices of odd degree in a graph is always even

3.1.3 Planner graphs

Chapter 4

Set Theory

Chapter 5

Group Theory

Chapter 6

Optimization

Chapter 7

Probability

7.1 Probability Basics

7.2 Distributions

7.2.1 Uniform Distribution

7.2.2 Random Distribution

7.2.3 Exponential Distribution

7.2.4 Poisson Distribution

7.2.5 Binomial Distribution

Chapter 8

Statistics

Part II

Algorithms & Data Structures

Part III

Programming Languages

Chapter 9

C Programming Language

9.1 Data Types

C provides three types of data: primary, derived and enumerated.

9.1.1 Primary Data types

These data types are provided by default in C.

Chapter 10

Digital Logic

Part IV

Computer Organization and Architecture

Part V

Operating System

Chapter 11

Introduction

11.1 An Introduction to Operating System

11.1.1 OS and its functionalities

What is OS ?

- **Definition-1:** It's a program that lets you run other programs. We need to differentiate this from a command interpreter or a windowing system, which run other programs based on user requests.
- **Definition-2:** A program that provides controlled access to a computer's resources. These resources include the CPU (process scheduling), memory (memory management), display, keyboard, mouse (device drivers), persistent storage (file systems) and the network.

11.1.1.1 What are the functions of OS?

11.1.2 Different Types of OS

11.1.2.1 Batch Processing Operating System

- Many programs with similar requirements are added to the system in bunch
- Only one Job can run at a time serially
- Reduced CPU utilization

11.1.2.2 Multi-Programming Operating System

- Multiple programs resides in main memory
- if n nos. of processes are in main memory then n is called degree of multi-programming,
- degree of multi-programming doesn't depend of number of CPUs.
- have better CPU utilization and throughput

11.1.2.3 Multi-Tasking Operating System

- logical extension of multi-programming systems
- jobs are executed on the CPU in time sharing mode
- added advantage of better response time

11.1.2.4 Multi-Threading Operating System

11.1.2.5 Multi-Processing Operating System

11.1.2.6 Real-time Operating System

- Each process must finished within well defined fixed time

    1. **Hard Real-time System**
    2. **Soft Real-time System**

11.1.2.7 Distributed Operating System

-
- Also known as loosely coupled system

11.2 Kernel

11.2.1 System Calls

System calls provide an interface to the services made available by an operating system. Usually invoked by software interrupt.

Categories	WINDOWS	UNIX
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

fork()

- Mainly used for creating child process
- return 0 to child and a non zero value to parent on success
- return -ve value on failure
- If the program contains 'n' fork () system calls, then Number of child processes created = 2n–1.

Chapter 12

Process Management

12.1 Process Concept

12.1.1 Process Lifecycle

12.1.1.1 Process States

- **Created** Process is newly created by system call, is not ready to run.
- **Running** Only One process can be in running state.
- **User running** Process is running in user mode which means it is a user process.
- **Kernel Running** Indicates process is a kernel process running in kernel mode.
- **Zombie** Process does not exist/ is terminated.
- **Preempted** When process runs from kernel to user mode, it is said to be preempted.
- **Ready to run in memory** It indicated that process has reached a state where it is ready to run in memory and is waiting for kernel to schedule it.
- **Ready to run, swapped**– Process is ready to run but no empty main memory is present
- **Sleep, swapped** Process has been swapped to secondary storage and is at a blocked state.
- **Asleep in memory** Process is in memory(not swapped to secondary storage) but is in blocked state.

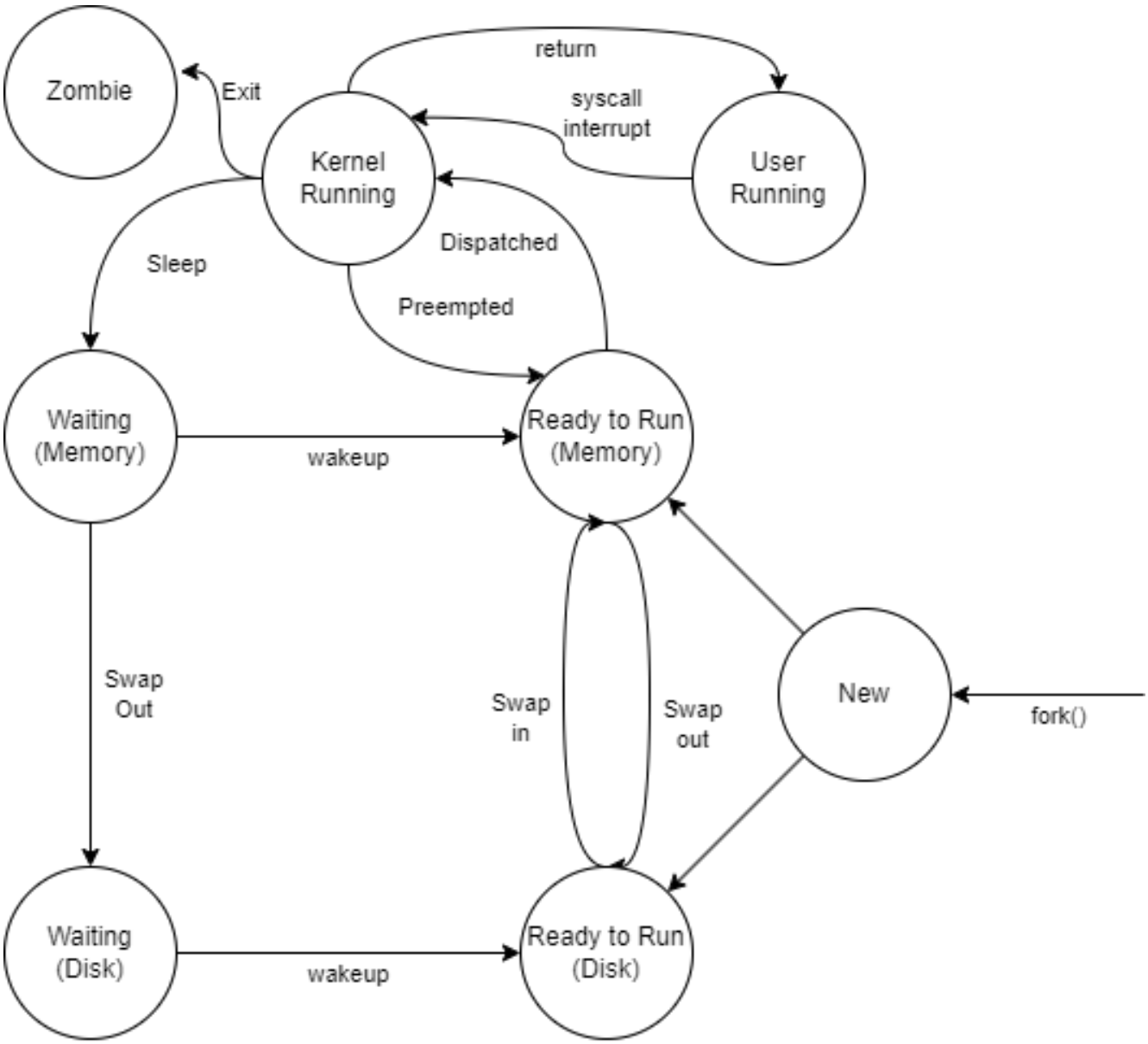

Figure 12.1: Process State Transition

12.1.2 Process Control

12.1.2.1 Process Control Block

Structure of Process Control Block

1. Process State: created, ready to run, run, asleep, zombie, preempted, orphan etc.
2. PID: Unique ID of the process
3. Program Counter: Address of the next instruction
4. Registers: Stack Pointers, Index Registers etc.
5. Memory Management Information: Paging, Segmentation etc.
6. Scheduling Info: Priority of processes, the implemented algorithms etc.
7. Open file lists: The resources that are in use devices, files etc.
8. Timing Info: Running time, Remaining CPU burst, CPU utilization.

Some notable points

- implemented using **double linked list**
- each process has it's own PCB
- PCB of all processes stored in **main memory**

12.1.3 Process Modes

The primary goal of the dual mode operation is to provide protection and security to the user application programs and the operating system from the unauthorized users. The mode bit is used to determine the particular mode in which an instruction is executing.

12.1.3.1 User Mode

- direct access to the hardware
- full access to the machine instruction set
- I/O operations, context-switching, clearing the memory map

12.1.3.2 Kernel Mode

- direct access to the hardware
- full access to the machine instruction set
- I/O operations, context-switching, clearing the memory map

12.1.3.3 Mode Switching

Steps involving switching between kernel mode and user mode

Mode bit: 0	Kernel mode
Mode bit: 1	User mode

12.1.4 Context

12.1.4.1 Context Switching

What is Context Switching? A context switch is storing the state or context of a process so that it can be reloaded whenever required for execution from the same point it was switched as earlier. Context switches can occur only in kernel mode.

The steps of context switching A typical thread context switch on a CPU happens like this:

1. Save the context of the process that is currently running on the CPU.
2. Update the process control block and other important fields.
3. Move the process control block of the above process into the relevant queue such as the ready queue, I/O queue etc.
4. Select a new process for execution.
5. Update the process control block of the selected process. This includes updating the process state to running.
6. Update the memory management data structures as required.
7. Restore the context of the previous process by loading the old values of the process control block and registers loaded again on the processor

12.2 Process Scheduling

12.2.1 Process Schedulers

Short Term Schedulers Select a job from ready queue based on its policy and gives the control of the CPU to that process with the help of "Dispatcher". Also known as CPU scheduler.

- $Ready \rightarrow Run$

Mid Term Schedulers If a process requests an I/O in the middle of execution, then the process removed from the main memory is loaded into the waiting queue of secondary memory (**swap-out**). When the I/O operation is completed, then the job moved from waiting queue of secondary memory to ready queue (**swap-in**). Also known as swapper.

- $Ready \leftrightarrow SuspendReady$
- $Block \leftrightarrow SuspendBlock$

Long Term Schedulers Select a job from disk and decides whether it should load it into main memory or not. Choosing a good mix of CPU/IO bound processes is main task. Also known as job Scheduler.

- $New \rightarrow Ready$

Dispatcher

- loading the job onto the CPU
- perform context-switching

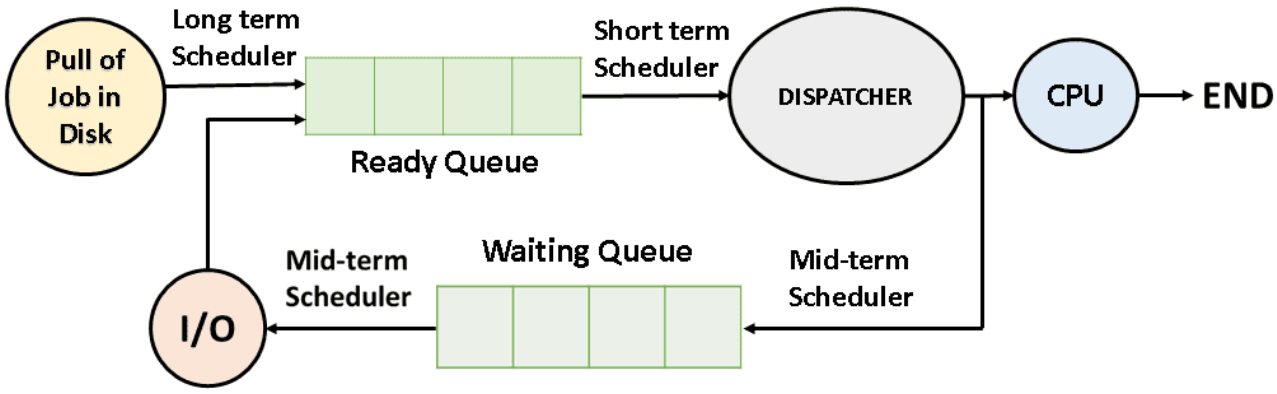

Figure 12.2: Process Schedulers

12.2.2 CPU Scheduling Criteria

CPU Scheduling Criteria in OS

CPU Utilization (↑) This is the percentage of time that the processor is busy, CPU utilization may range from 0-100%.

Throughput (↑) It means how many jobs are completed by the CPU.

Waiting Time (↓) Waiting time is the sum of the periods spent waiting by a process in the ready queue.

Response Time (↓) Response time is the time duration between the submission and the first response.

Turnaround Time The time interval between the submission of the process and the time of the completion is the turnaround time. This is the turnaround time formula in OS [Turnaround time = Waiting time in the ready queue + executing time + waiting time in waiting-queue for I/O].

Arrival Time time required from birth to the Ready state

Burst / Service Time time required by the process to complete execution

12.2.3 Process Scheduling Algorithms

First Come First Serve (FCFS)

- This is a non-preemptive algorithm
- Use queue (FIFO) for implementation
- Free from Starvation
- Suffers from *convoy effect*

Last Come First Serve (LCFS)

- This is a non-preemptive algorithm
- Use stack (LIFO) for implementation

Priority Based

- May be preemptive or non-preemptive
- Suitable for realtime systems where critical task to be finished first and deadline to be met

Shortest Job First (SJF)

- non-pre-emptive algorithm.
- Extension of priority scheduler, here highest priority is given to the process with smallest CPU burst time
- Large processes may starve
- CPU burst time is unknown at the time of arrival
- Minimizes average response time of processes

Shortest Remaining Task First (SRTF)

- preemptive algorithm
- shortest job at this instant is assigned to the CPU
- Large processes may starve
- Minimizes the average turnaround time of the processes

Round-Robin (RR)

- preemptive algorithm.
- CPU has a fixed time quantum i.e. each Process is given a limited time to complete its task. If it gets completed within that fixed time or the time is over then the control of CPU is given to another process waiting in the queue. Creates a illusion that all processes is running at the same time.
- Used mainly in time-sharing systems.
- $ns + (n-1)q = t$ where n is number of jobs, s is process switching time, q is time quantum

Highest Response Ratio Scheduling (HRRS)

- non-preemptive algorithm.
- No starvation
- Processes are assigned to the CPU based on their highest response ratio
- $ResponseRatio = \frac{WT+BT}{BT}$, Favors jobs that are waiting longer

Multilevel Queue

- Used both priority and round-robin scheduling
- Processes are put into multiple levels based on priority
- multiple processes in the highest-priority queue, are executed in round-robin order
- Common division Foreground Process (Realtime process, System process, interactive process), Background Process (batch process)

Multilevel Feedback Queue

- Processes moves between queue
- I/O-bound & interactive processes which are typically characterized by short CPU bursts are put in the higher-priority queues if a process uses too much CPU time that demoted to low priority queue

Scheduling Algorithm	CPU Overhead	Throughput	TAT	RT
FIFO	LOW	LOW	HIGH	LOW
SJF	MEDIUM	HIGH	MEDIUM	MEDIUM
PRIORITY	MEDIUM	LOW	HIGH	HIGH
ROUND ROBIN	HIGH	MEDIUM	MEDIUM	HIGH

Table 12.1: Comparison between different scheduling

12.3 Threading

12.3.1 Threading Concept

12.3.2 Thread Control Block

- **Thread ID** Unique ID for each thread.
- **Thread states** Current state of the thread
- **CPU information** Program Counters and Registers
- **Thread Priority** Priority of that thread
- A pointer which points to the process which triggered the creation of this thread.
- A pointer which points to the thread(s) created by this thread.

12.3.3 Thread Types

12.3.3.1 Based on level

Kernel level threads

- Kernel level threads are supported and managed directly by the operating system.
- The kernel knows about and manages all threads.
- One process control block (PCP) per process.
- One thread control block (TCB) per thread in the system.
- Provide system calls to create and manage threads from user space.

Advantages

- The kernel has full knowledge of all threads.
- Scheduler may decide to give more CPU time to a process having a large number of threads.
- Good for applications that frequently block.
- The context size is more
- Hardware support is required

Disadvantages

- Kernel manage and schedule all threads.
- Significant overhead and increase in kernel complexity.
- Kernel level threads are slow and inefficient compared to user level threads.
- Thread operations are hundreds of times slower compared to user-level threads.

User level threads

- User level threads are supported above the kernel in user space and are managed without kernel support.
- Threads managed entirely by the run-time system (user-level library).
- Ideally, thread operations should be as fast as a function call.
- The kernel knows nothing about user-level threads and manage them as if they where single-threaded processes.
- The context size is less
- No hardware support required

Advantages

- Can be implemented on an OS that does not support kernel-level threads.
- Does not require modifications of the OS.
- Simple representation: PC, registers, stack and small thread control block all stored in the user-level process address space.
- Simple management: Creating, switching and synchronizing threads done in user-space without kernel intervention.
- Fast and efficient: switching threads not much more expensive than a function call.

Disadvantages

- Not a perfect solution (a trade off).
- Lack of coordination between the user-level thread manager and the kernel.
- OS may make poor decisions like:
  - scheduling a process with idle threads
  - blocking a process due to a blocking thread even though the process has other threads that can run
  - giving a process as a whole one time slice irrespective of whether the process has 1 or 1000 threads
  - unschedule a process with a thread holding a lock.
- May require communication between the kernel and the user-level thread manager (scheduler activation) to overcome the above problems.

12.3.3.2 Based on Number

Single-threading

Multi-threading

Multi-Threading Model

- **One-to-One** Each user level thread is mapped to single kernel-level thread
- **Many-to-One** Many user level threads are mapped to single kernel-level thread.
- **Many-to-Many** Many user level threads are mapped to multiple kernel-level threads

Disadvantages

- Blocking one user-level thread of a process blocks the entire process
- TCB is considered as overhead for the system

12.4 Process Coordination

12.4.1 Process Communication

12.4.1.1 Types of Communication

- **Cooperative** Execution of one process affects or affected by other process
- **Independent** No communication between processes

12.4.2 Problems arises for not having synchronization

12.4.2.1 Producer Consumer

12.4.2.2 Dining Philosopher Problem

12.4.3 Synchronization

12.4.3.1 Critical Section

- **Critical Section** The portion of program text where shared variables or shared resources will be placed.
- **Non-Critical Section** The portion of program text where the independent code of the processes will be placed.
- **Race condition** The final value of any variable depends on execution sequence of the processes in which access takes place.
- To avoid the Race condition, only one process is allowed to enter into critical section

Conditions for handling critical Section To prove that this solution is correct. All the following conditions to be satisfied.

- **Mutual Exclusion** Only one process is allowed to enter into critical section at any point of time
- **Progress** No process that are running in remainder section can choose who can run in the critical section. Selection cannot be postponed indefinitely
- **Bounded Waiting** No process should have to wait forever to enter into critical section, there should be a bound/limit or it will lead to starvation.

12.4.3.2 Solutions for Critical Section Problem

Peterson's solution

- Classic Software based solution
- May or may not work on RISC systems
- every processes given a number i
- the turn indicates which process's turn to enter into critical section
- flag indicates if a process want to enter in critical section
- always gives way to the other process

```
boolean flag[2];
while (true) {
    flag[i] = true;
    turn = j;
    while (flag[j] && turn == j);
    /* critical section */
    flag[i] = false;
    /* remainder section */
}
```

Algorithm Name	Mutual Exclusion	Progress	Bounded Waiting
Lock Variable	×	√	×
Decker's Algorithm	√	×	√
Peterson's Algorithm	√	√	√
TSL Instruction	√	√	×

12.4.3.3 Monitor

- Monitors is a programming language compiler support type of solution to achieve synchronization.
- Monitors is collection of variables and procedures combined together in a special kind of module or package.
- The process running outside the monitor cannot directly access the internal variables of the monitor but however they can call the procedures of the monitor.
- Monitors has an important property that only one process can be active inside the monitor at any point of time.

12.4.3.4 Semaphore

Semaphore is an integer variable which is used by the various processes in a mutual exclusive manner to achieve synchronization.

- Down(); or wait (); or p ()
- up(); or signal (); or v (); or release ();

Binary Semaphore

Counting Semaphore

12.4.4 Deadlock

12.4.4.1 Deadlock Characteristics

- **Mutual Exclusion** Each resource can be assigned to at most one process only.
- **Hold and Wait** Processes hold a resource and may seek an additional resource.
- **No Preemption** Processes that have been given a resource cannot be preempted to release their resources.
- **Circular Wait** Every process awaits release of at least one resource held by some other processes.

A deadlock can occur only when all four conditions are present simultaneously.

12.4.4.2 Deadlock Prevention

Methods for deadlock prevention Deadlock can be prevented by unsatisfying one of the deadlock characteristics

- **Mutual Exclusion** cannot be unsatisfied because of the concept of sharable and non-sharable resources
- **Hold and Wait**
  - A process should be assigned all the required resources before the start of its execution. This may lead to low device utilization

  - The process should release all the existing resources before making a new request. This may eventually lead to starvation.

- **No Preemption** A process will be put into waiting state if all the resources required by the process is not available. If a process is waiting then it's resources can be preempted, like "wound–wait"
- **Circular Wait** Similar resources will be marked uniquely, so it will behave like a single resource and when any two process requests it, then resources can be assigned based on process timestamp.

12.4.4.3 Deadlock Detection

Safe & Unsafe States

- If a system is in safe state ⇒ no deadlocks.
- If a system is in unsafe state ⇒ possibility of deadlock.

Resource Allocation Graph RAG is a directed graph whose nodes are either square(Resource) or circle(Process). An arc from circle to square means process is requesting for that resource. An arc from square to circle means process is holding that resource.

12.4.4.4 Deadlock Avoidance

Banker's Algorithm

1. Let Work and Finish be vectors of length 'm' and 'n' respectively.
   - Initialize: Work = Available
   - Finish[i] = false; for i=1, 2, 3, 4...,n
2. Find an i such that both
   - Finish[i] = false
   - Need i <= Work
3. if no such i exists go to step (4)
4. Work = Work + Allocation[i]
   (a) Finish[i] = true
   (b) go to step (2)
5. if Finish[i] = true for all i then the system is in a safe state

Limitations of Banker's Algorithm

- Banker's algorithm can't eliminate an existing deadlock.
- Resource requirements must be are known beforehand.
- Number of live processes and resources are limited.
- Process synchronization is not possible because it doesn't guarantee any particular sequence.

Chapter 13

Memory Management

13.1 Basics

13.1.1 Partition

13.1.2 Fragmentation

13.1.3 Swapping

13.1.4 Thrashing

13.1.5 Demand Paging

13.2 Virtual Memory

13.2.1 Paging

13.2.2 Page Tables

13.2.3 Page Faults

13.2.4 Page Replacement

Optimal Page replacement

LRU

LFU

FIFO

Second Chance

Clock

Chapter 14

Storage Management

14.1 Disk Management

14.1.1 Disk Components

- **Platter** The surface of a magnetic disk is known as platter.
- **Track** The surface of a platter is logically divided into circular shapes known as tracks.
- **Sector** Tracks are divided into smaller sections known as sectors.
- **Cylinder** A set of tracks that are at one arm position are called cylinders.

Relations Between Above

$$SingleTrackDataCapacity = SectorsPerTrack * BytesPerSector$$

$$TransferSpeed = \frac{RPM}{60} * SingleTrackDataCapacity$$

Disk Timings

- **Seek Time**
- **Access Time**
- **Latency**

14.1.2 Disk Scheduling Algorithms

FCFS (First Come First Serve)

SSTF (Shortest Seek Time First)

SCAN & CSCAN

Algorithm	Advantages	Disadvantages
FIFO		LOW
SSTF	MEDIUM	HIGH
SCAN/LOOK	MEDIUM	LOW
C-SCAN/C-LOOK	HIGH	MEDIUM

Table 14.1: Comparison between different disk scheduling

LOOK & CLOOK

14.2 File Management

14.2.1 File Systems

Floppy Disk FS

FAT

NTFS

EXT

Chapter 15

Protection & Security

Chapter 16

Special Systems

16.1 Windows XP

16.2 Linux OS

16.3 Mac OS

16.4 Android OS

16.5 Distributed System

16.6 Real-time System

Part VI

Computer Networking

Chapter 17

Introduction to computer Networks

17.1 Problems associated with Computer Networks

17.1.1 Communication

Protocols Protocol is set of rules that govern data communications.

Syntax: The term refers to the structure or format of the data, meaning the order in which they are presented.

Semantics: The term refers to the meaning of each section of bits. How are a particular pattern to be interpreted, and what action is to be taken based on that interpretation?

Timing: The term refers to two characteristics: a) When data should be sent b) How fast they can be sent.

HTTP (**Hyper Text Transfer Protocol**) Used for browser requesting resources from a server. Available versions are: 1.0 (*RFC 1945*), 2.0 (*RFC 7540*). Methods available are: GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH etc. Port used: 80 and 443 (secure). This is a TCP protocol.

SMTP (**Simple Mail Transfer Protocol**) Used for Delivering mails to servers. Available in *(RFC 821)*. Available methods are: HELO, MAIL, RCPT, SEND, DATA, VRFY, AUTH, RSET, QUIT etc. Port used: 25. This is a TCP protocol.

FTP (**File Transfer Protocol**) Used for downloading or uploading file from or to server. Available in *(RFC 959)*. Available methods are the common file handling commands of UNIX system. Port used: 20/21. This is a TCP protocol.

POP (**Post Office Protocol**) Client connect, retrieve, store them and finally delete them from server. Available in POP *(RFC 918)*, POP3 *(RFC 1081)*. Available methods are: USER, PASS, QUIT, STAT, RETR, DELE, RSET etc. Port used: 110 and 995 (secure).

IMAP (**Internet Message Access Protocol**) Used for complete management of mailbox. User can download a message using different clients and keep a copy on the server until explicitly deleted. Available in *(RFC 3501)*. Available methods are: APPEND, CHECK, SEARCH, SELECT, STORE etc. Port used: 143 and 993 (secure).

TELNET (**TELetype NETwork**) Provides a bidirectional interactive text-oriented communication facility. Available in *(RFC 15)*. Available methods are: CLOSE, DISPLAY, OPEN, SET, SEND, STATUS, UNSET, QUIT etc. Port used: 23. This is a TCP protocol.

17.1.2 Identification

To send packets from source to destination we need to identify systems. Two types of addressing are used for this: a) Logical address (IP) and b) Physical addressing (MAC)

IP Address

Characteristics of IP address

Classification of IP address

IP Class	NID-HID	Starting bits	Range	Area of usage
A	8-24	0	1.0.0.1 to 126.255.255.255	Defense, Govt organizations etc.
B	16-16	10	128.0.0.1 to 191.255.255.255	MNC, Banks, Big organizations etc.
C	24-8	110	192.0.0.1 to 223.255.255.255	etc.
D	nil	1110	224.0.0.1 to 239.255.255.255	Used for multi-casting purposes
E	nil	11110	240.0.0.1 to 255.255.255.255	Reserved for future uses

Some special IP addresses **127.*.*.*** is reserved for loop-back.

MAC Address

17.1.3 Connection

Types of Connections

Uni-casting When source is sending data to only one device.

Multi-casting When source is sending data to multiple devices.

Broadcasting :
Limited Broadcasting: A packet is sent to a specific network or series of networks.
Directed Broadcasting: A packet is sent to specific destination address.
Flooded Broadcasting: A packet is sent to every network.

Netting

Subnetting

Supernetting

Hardware Devices

Repeater (PL) It receives signals and either amplifies or regenerate it.

Hub (PL) Passive Broadcasting device no s/w needed.

Bridge (PL, DL) Connects multiple LAN segments (similar) or subnets

Switch

Router

Gateway

Chapter 18

Layers of computer Networks

Here is a Cheat-sheet for Layers of OSI model:

Layer Name	Objectives	PDUs	Protocols
Application Layer	Bla Bla	Bla	DNS, HTTP, FTP, DHCP, IMAP, LDAP, POP, RSTP, RIP, SMTP, SNMP, SSH, TelNet, SIP, TLS/SSL,
Presentation Layer	Bla Bla	Bla	DNS, HTTP, FTP, DHCP, IMAP, LDAP, POP, RSTP, RIP, SMTP, SNMP, SSH, TelNet, SIP, TLS/SSL,
Session Layer	Bla Bla	Bla	DNS, HTTP, FTP, DHCP, IMAP, LDAP, POP, RSTP, RIP, SMTP, SNMP, SSH, TelNet, SIP, TLS/SSL,
Transportation Layer	Bla Bla	BLA	TCP, UDP, DCCP, SCTP, RSVP
Network Layer	Bla Bla	BLA	IPv4, IPv6, IGMP, IPsec, ICMP
Data Link Layer	Bla Bla	BLA	ARP, NDP, L2TP. PPP, DSL, ISDN, FDDI
Physical Layer	Bla Bla	BLA	ARP, NDP, L2TP. PPP, DSL, ISDN, FDDI

18.1 Application Layer

18.2 Presentation Layer

18.3 Session Layer

18.4 Transport Layer

18.5 Network Layer

18.6 Data Link Layer

18.6.1 Error Control

Error: During transmission when bits or signal gets changed, called error.
Single-bit Error: When only one bit gets changed.
Burst Error: When 2 or more bits get changed.

Parity Checking Adding redundant bits to the end of actual data, called parity bits. During sending of data the parity bits are calculated using XOR operation for even parity and XNOR for odd parity. In the receiving end they parity is calculated again if 0 then there is no error so remove the parity bits and accept data else reject and request again.

Hamming Code In this mechanism to get the final data we need to ind the parity bit of those which have 1's in n^{th} position and add them to the 2^n position. The number of redundant bits needed are: $2^{n+1} > len(Code)$. On the receiving end recalculate the hamming code, if result is all 0's then accept the data, else alter the bit position of the result to get the error free data as it is a error detection and correction code. This can detect more than 1 bit error but cannot correct them.

Cyclic Redundancy Check To get the redundant bits, data is divided with a prime divisor. The remainder is added to the data. On the receiving end, the modified data is also divided by the same divisor. If remainder is 0 then accept the data, else reject the data.

Checksum Data are divided into same size segments with the help of padding, then summed together. The result is then inverted and send with the actual data. On the receiving end, the data received is again divided and summed up if resulted in 0 then data is correct else reject the data.

18.6.2 Data Link Control

Framing Helps to identify messages from different sources and destinations. It has three main parts: HEADER, containing control information, source and destination addresses, type of PDU, Control fields; DATA, contains content of the packets; TRAILER, contains error control bits, stop bits.

Character Oriented Framing Data are sent in byte format.
Default structure: Flag + Header + ... Data ... + Trailer + Flag
Byte Stuffed: ESC bytes are added before every FLAG and ESC in the data. e.g. *Flag + Header + ... ESC FLAG ... Data ... ESC ESC ... + Trailer + Flag*

Bit Oriented Framing **Default structure:** Flag + Header + ... Data ... + Trailer + Flag
Bit Stuffed: ESC bytes are added before every FLAG and ESC in the data. e.g. *Flag + Header + ... ESC FLAG ... Data ... ESC ESC ... + Trailer + Flag*

18.6.3 Flow Control

Noiseless Channel

Noisy Channel

Stop and wait protocol

Go-back-N protocol

Selective repeat protocol $TransmissionDelay = \frac{FrameSize}{Bandwidth}$ $OptimalSenderWindowSize = 1+2a = 1+2*\frac{PropagationDelay}{TransmissionDelay}$ $min.SequenceNumberBits = \lceil log_2(OptimalSenderWindowSize+OptimalReceiverWindowSize)\rceil$

18.6.4 Access Control

Random Access Control

Controlled Access Control

Channelization Control

18.7 Physical Layer

18.7.1 Data and Signals

Signals

Different Types of Signals :
Analog Signals It includes an infinite number of values along its path.
Digital Signals It includes limited number of defined values along its path.
Periodic Signals It repeats patterns in a common time frame.
Non-periodic Signals It doesn't repeat any pattern.
Analog Data It refers to information that is continuous.
Digital Data It refers to information that is discrete in nature.
(*) Periodic analog signals and non-periodic digital signals are commonly used.

Different Properties of Analog Signals :
Period and Frequency: The time needed to complete 1 cycle is called period (T). Frequency (f) means to the number of periods in 1s. i.e. $f = \frac{1}{T}$.
Phase: It denotes position of the signal relative to time 0. i.e. if $\frac{1}{C}$ is offset related to $time_0$ $d = 360 * \frac{1}{C}$.
Wavelength: The distance a signal travels before completing a cycle is called wave lengths. $wavelength = PropagationSpeed * period = \frac{PropagationSpeed}{frequency}$.
Composite Signal: Combination of different analog sine waves is called composite signal.
Bandwidth: The difference between highest and lowest frequency of a composite signal.

Different Properties of Digital Signals :
Bit rate: The number of bits send per second is called bit rate.
Bit Length: Distance 1 bit occupies in the medium. i.e. $BitLength = PropagationSpeed * BitDuration$.
Throughput: How fast we can send data though network. If a channel carries F frames per second and B bits per frame then throughput T = F * B.
Latency: Total time taken to send first bit to entire message. Calculation: $Latency = PropagationTime + TransmissionTime+QueuingTime+ProcessingDelay$; $PropagationTime = \frac{Distance}{PropagationSpeed}$; $TransmissionTime = \frac{MessageSize}{Bandwidth}$.

Anomalies during transmission :
Attenuation: During traveling through medium signal loses its energy is called attenuation. Calculation: $dB = 10log_{10}\frac{P_e}{P_s}$
Distortion: When signals changes its form or shape.
Noise: When impurities from outside of medium corrupts the signal is called noise. $SNR_{dB} = 10log_{10}SNR = 10log_{10}\frac{AvgSignalPower}{AvgNoisePower}$

Transmission of Signals

Conversions

Transmission Modes :
Parallel Transmission: n-wires send n-bits of data at same time mostly binary data 0 and 1. Normally used when distance is too short as it costs more.
Serial Transmission: Data are transmitted serially
Asynchronous: Units of bits are send begin with start bit(0) and end with stop bits(1). Data bits are asynchronous at byte level.
Synchronous: Data bits are combined and send as frames one after another and receiver has the job to separate those bits into groups.

Multiplexing Bandwidth is shared if a medium got higher bandwidth than the needs of devices this technique is called multiplexing.
Frequency Division Multiplexing: An analog method is used to combine signals. Different frequencies are separated by the guard-bands to prevent overlapping.
Time Division Multiplexing: Data from different slow-rate devices are given a fixed time slots in round robin manner to achieve high-rate.
Wave Division Multiplexing: Designed for Optic Fiber, uses the technique of refraction.

18.7.2 Transmission Medium

Guided Media

Twisted Pair Cable A twisted pair of conductors each with own plastic insulation.

Co-axial Cable

Optic Fiber Cable

Unguided Media

Radio wave

Micro wave

Infrared wave

18.7.3 Switching

Circuit Switching Switches connected through actual links and circuits. Reserve all the resources during transmission. Implemented in physical layer.

Packet Switching Data are arranged into packets and then send over network. Resources are allocated as they needed. Each switch has routing table.

Datagram Networks Packets are sent independent of each other. Normally switching done in network layer and called connection-less. Each have a routing table contains destination address and output port.

Virtual Circuit Networks During setup phase the path is selected, all packets follow same path to reach destination. Normally implemented in data-link layer and called connection-oriented. Routing table consists of incoming/outgoing ports/VCI.

Chapter 19

Network Security

Chapter 20

Mobile Technologies

Chapter 21

Cloud Computing

Chapter 22

Not Sorted

22.1 IP

To check an IP belongs to same network, do AND(.) operation between subnet mask and IP.
CIDR representation is IP/C. To find the range do $N = 32 - C$. Truncate last N bits to 0s for lowest IP and to 1s for highest IP and total IPs will be 2^N.
CIDR aggregation: 1) All IPs must be contiguous. 2) Calculate total IP addresses. 3) Aggregated CIDR will be IP/X where, $X = 32 - log_2 T$.
IP Fragmentation: 1) Calculating $ActualTransmissionSize = MaximumTransmissionUnit - IPHeaderSize$ 2) Calculating $NumberOfSegments = \frac{TotalPacketSize}{ActualTransmissionSize}$ 3) Calculating $Offsets_i = \frac{ActualTransmissionSize}{8} * i$; Initial offset is always 0. **Hamming Code:** 1) Calculating $Min.HammingDistance = min_1(XORofEachCodewords)$ 2) Calculating $Max.ErrorBits = \frac{Min.HammingDistance-1}{2}$ **CSMA/CD:** 1) Calculating $RoundedTurnaroundTime = 2 * PropagationDelay$ 2) Calculating $min.FrameSize = RoundedTurnaroundTime * Bandwidth$
DHCP (Dynamic Host Configuration Protocol): Assign IP addresses automatically, Centralized, Works on port UDP 67 and 68, Application Layer protocol, Supports both IPv4 and IPv6, Sometimes behaves as router.
SLIP(SeriaL IP): IP Encapsulation over Serial Ports, Replaced later by PPP, preferred for micricontrollers due to small overhead but not good for PCs.
SNMP (Simple Network Management Protocol): Management and Monitoring of networks, Application layer protocol, Uses ports UDP 161 and 162.
IGMP(Internet Group Multi-casting Protocol): Uses IPv4, manage by IP Encapsulation.
IGMP(Internet Control Message Protocol): Uses IPv6, manage by Multi-cast Listener Discovery
ARP(Address Resolution Protocol): Find out MAC address from IPv4, works on Layer 2 to 3, replaced by NDP in IPv6
RARP(Reverse Address Resolution Protocol): Find out IP address from MAC, works on Layer 3 to 2, replaced by BOOTP7 then DHCP in IPv6
TCP congestion control: Slow start size of the congestion window increased exponentially until threshold. **Congestion avoidance** size of the congestion window increased additively until threshold. **Congestion detection** size of the threshold dropped to half (multiplicative decrease).

Part VII

Database Design

Chapter 23

Introduction

23.1 Background

Data are the known facts that can be recorded and that have implicit meaning

Information

Database

23.2 Database Architecture

23.2.1 Terminologies

- **X**

23.3 Database

Chapter 24

Relation Model

24.1 Relation

24.1.1 Terminologies

- **Relation** A relation is a table with columns and rows. (File)
- **Tuple** A tuple is a row of a relation. (Record)
- **Attribute** An attribute is a named column of a relation. (Field)
- **Arity/Degree** Number of attributes of database table.
- **Cardinality** Number of tuple of database table.
- **Domain** A domain is the set of allowable values for one or more attributes.
- **Base Relation** A named relation corresponding to an entity in the conceptual schema, whose tuples are physically stored in the database.
- **View** The dynamic result that is provides a virtual relation that may or may not exists and produced upon users' request
- **Relational database** A collection of normalized relations with distinct relation names
- **Relational Schema** A named relation defined by a set of attribute and domain name pairs.
- **Relational Database Schema** A set of relation schemas, each with a distinct name.
- **Key** Attribute or set of attribute is called key.
- **Super key** An attribute, or set of attributes, that uniquely identifies a tuple within a relation
- **Candidate Key** A superkey such that no proper subset is a superkey within the relation. It has two properties *Uniqueness*, *Irreducibility*, For any RDBMS table there must be atleast one candidate key, whose field value can not be null..
- **Overlapped Candidate Key**: Two or more candidate key with some common attribute.
- **Compound Candidate Key**: A candidate key with atleast two attributes.
- **Prime Attribute** the attributes that are part of a candidate key, a relation can have multiple candidate key but all may not contain same prime attribute.
- **Non-prime attribute** the attribute that does not belongs to any of the candidate keys of the relation.
- **Primary Key** A primary key is the candidate key chosen for use in identification of tuples
- **Null** Represents a value for an attribute that is currently unknown or is not applicable for this tuple.
- **Entity Integrity** In a base relation, no attribute of a primary key can be null.
- **Referential integrity** If a foreign key exists in a relation, either the foreign key value must match a candidate key value of some tuple in its home relation or the foreign key value must be wholly null.
- **Constrains** Additional rules specified by the users or database administrators of a database that define or constrain some aspect of the enterprise.

24.2 E-R Model

24.3

24.4 Query Language

Chapter 25

Database Analysis & Design

Chapter 26

Storage and Indexing

Chapter 27

Transaction Management

Chapter 28

Special Databases

28.1 Distributed Database

28.2 Semi-structured Database

28.3 Blockchain Database

Part VIII

Emerging Technologies

Chapter 29

Artificial Intelligence

Chapter 30

Web Technologies

Chapter 31

Data Mining and Warehousing

Chapter 32

Blockchain

Chapter 33

Cloud Computing

Chapter 34

Internet of Things

34.1 Device

34.1.1 Bluetooth

34.1.1.1 Introduction to Bluetooth

What is Bluetooth?

- Wireless technology standard for short-range communication
- Operates in the 2.4 GHz ISM band
- Uses radio waves for data transmission

Purpose and Uses of Bluetooth

- **Wireless Data Transfer:**
  - Share files, photos, contacts between devices
  - Bluetooth Object Push Profile (OPP) for file sharing
- **Audio Streaming:**
  - Stream audio to headphones, speakers, car systems
  - A2DP (Advanced Audio Distribution Profile) for high-quality audio
- **Hands-Free Communication:**
  - Bluetooth headsets, car kits, earphones for hands-free calls
  - HFP (Hands-Free Profile) for call control
- **Peripheral Device Connection:**
  - Connect keyboards, mice, printers, game controllers wirelessly
- **IoT Devices:**
  - Used in smart home devices (e.g., lights, locks, thermostats)
  - Bluetooth Low Energy (BLE) for low-power IoT communication
- **Health Monitoring:**
  - Sync fitness trackers, heart rate monitors, smartwatches
- **Location-Based Services:**
  - BLE beacons for indoor navigation, proximity marketing
- **Automotive Applications:**
  - Hands-free calls, music streaming in cars
- **Gaming:**
  - Wireless controllers for gaming consoles, smartphones

34.1.1.2 How Bluetooth Works?

Key Concepts

- **Pairing Process:**
  - Devices need to pair by discovering and authenticating each other (PIN or passkey)
- **Frequency Hopping:**
  - Bluetooth uses **79 channels** in the 2.4 GHz band - Avoids interference by hopping between channels at 1600 times per second
- **Profiles and Protocols:**
  - Defines data types and communication methods (e.g., A2DP for audio, SPP for data transfer) - **L2CAP** protocol handles basic data transmission
- **Security:**
  - Uses **encryption** and **authentication** to secure data - Security modes: Just Works, Numeric Comparison, Passkey Entry

34.1.1.3 Bluetooth Versions and Evolution

34.1.1.4 Bluetooth in Modern Devices

34.1.1.5 Bluetooth Security

34.1.1.6 Advantages of Bluetooth

34.1.1.7 Challenges and Limitations

34.1.1.8 Future of Bluetooth

Chapter 35

UI/UX Design

Appendix A

Abbreviations

A.1 Random

SPOOL Simultaneous Peripheral Operations On-Line **PCS** Process Contention Scope **SCS** System Contention Scope **AFH** Adaptive frequency-hopping spread spectrum **RSSI** Received signal strength indicator