



POORNIMA

COLLEGE OF ENGINEERING

Course File

Name of faculty	Manish Choubisa
Class	B. Tech – V SEM -B
Branch	Computer Engineering
Course Code	5CS4-03
Course Name	Operating System
Session	2021-2022

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

Vision & Mission of Poornima College of Engineering

Vision

To create knowledge based society with scientific temper, team spirit and dignity of labor to face the global competitive challenges.

Mission

To evolve and develop skill based systems for effective delivery of knowledge so as to equip young professionals with dedication and commitment to excellence in all spheres of life

Vision & Mission of Department of Computer Engineering

Vision

Evolve as a centre of excellence with wider recognition and to adapt the rapid innovation in Computer Engineering.

Mission

- 1) To provide a learning-centered environment that will enable students and faculty members to achieve their goals empowering them to compete globally for the most desirable careers in academia and industry.
- 2) To contribute significantly to the research and the discovery of new arenas of knowledge and methods in the rapid developing field of Computer Engineering.
- 3) To support society through participation and transfer of advanced technology from one sector to another.

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES (PEO'S)

PEO1: Graduates will work productively as skillful engineers playing the leading roles in multifaceted teams

PEO2: Graduates will identify the solutions for challenging issues inspiring the upcoming generations leading them towards innovative, creative, and sophisticated technologies.

PEO3: Graduates will implement their pioneering ideas practically to create products and the feasible solutions of research oriented problems

PROGRAM OUTCOMES (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: The ability to understand and apply knowledge of mathematics, system analysis & design, Data Modelling, Cloud Technology, and latest tools to develop computer based solutions in the areas of system software, Multimedia, Web Applications, Big data analytics, IOT, Business Intelligence and Networking systems.

PSO2: The ability to understand the evolutionary changes in computing, apply standards and ethical practices in project development using latest tools & Technologies to solve societal problems and meet the challenges of the future.

PSO3: The ability to employ modern computing tools and platforms to be an entrepreneur, lifelong learning and higher studies.

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

MAPPING OF KEY PHRASES OF THE INSTITUTES MISSION STATEMENT WITH THE KEY PHRASES OF INSTITUTES VISION STATEMENT

(Institution Mission Vs Institute Vision)

Key Phrases of the Mission Statement of the Institute	Key Phrases of the Vision Statement of the Institute		
	To create knowledge based society with scientific temper	Team spirit	To face the global competitive challenges
Skill based systems for effective delivery of knowledge	3		3
To equip young professionals with dedication		1	3
Excellence in all spheres of life	2		2

MAPPING OF KEY PHRASES OF THE DEPARTMENTS VISION STATEMENT WITH THE KEY PHRASES OF INSTITUTES MISSION STATEMENT

(Department Vision Vs Institution Mission)

Key Phrases of the Vision Statement of the Department	Key Phrases of the Mission Statement of the Institute		
	Skill Based Systems	Delivery of Knowledge	Excellence in all spheres of life
Centre of Excellence	3	2	1
Wider recognition	2	2	1
Rapid innovation.	2	1	

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

**MAPPING OF KEY PHRASES OF THE DEPARTMENTS MISSION STATEMENT WITH THE
KEY PHRASES OF DEPARTMENTS VISION STATEMENT**
(Department Mission Vs Department Vision)

Key Phrases of the Mission Statement of the Department	Key Phrases of the Vision Statement of the Department		
	Centre of Excellence	Wider recognition	Rapid innovation.
Learning-centered environment	3		2
Research and Discovery	2		2
Social Responsibility	1	2	1

**MAPPING OF KEY PHRASES PEOS WITH KEY PHRASES OF DEPARTMENTS MISSION
STATEMENT**
(PEO Vs Department Mission)

Key Phrases of PEO Statements	Key Phrases of the Mission of the Department		
	Learning-centered environment	Research and Discovery	Social Responsibility
Skillful engineers	3	2	
Innovative, Creative, and Sophisticated Technologies	3		1
Pioneering Ideas	2	2	

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

**MAPPING OF KEY PHRASES OF PSO WITH KEY PHRASES OF DEPARTMENTS MISSION
STATEMENT
(PSO Vs Department Mission)**

Key Phrases of PSO Statement	Key Phrases of the Mission Department		
	Learning-centred environment	Research and Discovery	Social Responsibility
Technical Knowledge	3	2	
Standards, Ethic, Tools, Challenges Societal Problems	1	2	2
Entrepreneur, Lifelong Learning and Higher Studies.	2		

**MAPPING OF KEY PHRASES OF PEO WITH KEY PHRASES OF PO
(PEO Vs PO)**

	Key Phrases of PO Statement											
Key Phrases of PEO Statement	Engine ering knowle dge:	Pro ble m analysis:	Desig n/ devel opme nt of solutions:	Con duct investi gation s of compl ex proble ms:	Mo der n tool usa ge:	The engi nee r and soci ety:	Envir onmen t and sustai nabilit y:	Et hic s:	Indi vidu al and team wor k:	Com muni catio n:	Project manage ment and finance :	Life-long learn ing:
Skillful engineers	3	3	3	3	2		1	1	2	2	1	2
Innovative, Creative, and Sophisticated Technologies		3	2	2		1	1	1				
Pioneering Ideas	2	1	1	2	2			1	1	2	1	2

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

MAPPING OF KEY PHRASES OF PSO WITH KEY PHRASES PEO
(PSO Vs PEO)

Key Phrases of the PEO Department	Key Phrases of the PSO Department		
	Technical Knowledge	Standards, Ethic, Tools, Challenges Societal Problems	Entrepreneur, Lifelong Learning and Higher Studies.
Skillful engineers	3	2	2
Innovative, Creative, and Sophisticated Technologies	2	2	2
Pioneering Ideas		1	2

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

RAJASTHAN TECHNICAL UNIVERSITY, KOTA
SYLLABUS



RAJASTHAN TECHNICAL UNIVERSITY, KOTA

Syllabus

III Year-V Semester: B.Tech. Computer Science and Engineering

5CS4-03: Operating System

Credit: 3

Max. Marks: 150(IA:30, ETE:120)

3L+0T+0P

End Term Exam: 3 Hours

SN	Contents	Hours
1	Introduction: Objective, scope and outcome of the course.	01
2	Introduction and History of Operating systems: Structure and operations; processes and files Processor management: inter process communication, mutual exclusion, semaphores, wait and signal procedures, process scheduling and algorithms, critical sections, threads, multithreading	04
3	Memory management: contiguous memory allocation, virtual memory, paging, page table structure, demand paging, page replacement policies, thrashing, segmentation, case study	05
4	Deadlock: Shared resources, resource allocation and scheduling, resource graph models, deadlock detection, deadlock avoidance, deadlock prevention algorithms Device management: devices and their characteristics, device drivers, device handling, disk scheduling algorithms and policies	15
5	File management: file concept, types and structures, directory structure, cases studies, access methods and matrices, file security, user authentication	07
6	UNIX and Linux operating systems as case studies; Time OS and case studies of Mobile OS	08
	Total	40

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

Campus: Poornima College of Engineering

Year/Section: 3rd

Date: 18 Sept 2021

Course: B.Tech.

Semester/ Section – 5B

Name of Faculty: Mr. Manish Choubisa

Name of Subject: Operating System

Code: 5CS4-03

ABC Analysis (RGB method)

Unit No.	A (Hard Topics)	B (Topics with average hardness level)	C (Easy to understand topics)	Preparedness for 'A' topics
I	Semaphores, wait and signal procedures, process scheduling and Algorithms, critical sections, threads, multithreading	Processor management: inter process communication, mutual exclusion,	Structure and Operations; processes and files	Revision of the topic
II	Paging, page table structure, demand paging, page replacement policies	Contiguous memory allocation, virtual memory	Thrashing, segmentation, case study	Revision & taking test
III	Deadlock, deadlock detection, deadlock avoidance, deadlock Prevention algorithms, disk scheduling algorithms and policies	Shared resources, resource allocation and scheduling, resource graph models	Devices and their characteristics, device drivers, Device handling	Revision of the topic
IV	Directory structure, user Authentication	Access methods and matrices, file security	File concept, types and structures, cases studies	Revision with the students
V	UNIX and Linux operating systems as case studies	Time OS and case Studies of Mobile OS	-	Power point Presentation & Revision

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

COURSE BLOWN UP

Campus: Poornima College of Engineering
Course: B.Tech.
Name of Faculty: Mr. Manish Choubisa

Year/Section: 3rd
Semester/ Section – 5B

Date: 18 Sept 2021

Name of Subject: Operating System **Code:** 5CS4-03

SNo.	TOPIC AS PER SYLLABUS	BLOWN UP TOPICS (up to 10 Times Syllabus)
1.	Zero Lecture	Objective, scope and outcome of the course.
2.	Introduction of Operating systems and Processor management	Structure and operations; processes and files inter process communication, mutual exclusion, semaphores, wait and signal procedures, process scheduling and algorithms, critical sections, threads, multithreading
3.	Memory management	Contiguous memory allocation, virtual memory, Paging, page table structure, demand paging, page replacement policies, thrashing, segmentation, case study
4.	Deadlock and Device management	Shared resources, resource allocation and scheduling, Resource graph models, deadlock detection, deadlock avoidance, deadlock Prevention algorithms devices and their characteristics, device drivers, Device handling, disk scheduling algorithms and policies
5.	File management	File concept, types and structures, directory structure, Cases studies, access methods and matrices, file security, user Authentication
6.	UNIX and Linux operating systems as case studies	Time OS and case Studies of Mobile OS

POORNIMA COLLEGE OF ENGINEERING, JAIPUR

DEPARTMENT OF COMPUTER ENGINEERING

COURSE PLAN

Campus: Poornima College of Engineering

Class/Section: V- Sem B

Date: 20/9/2021

Program: B.Tech.


Year/ Section –III Year /B

Name of Faculty: Mr. Manish Choubisa

Name of Subject:-

Subject Code: 5CS4-03

S. No.	Lect. No	Points to cover	CO/LO	Proposed Date	Actual Date	Ref.Book/Journal with
1	L0	Zero Lecture: Introduction to the subject, Objective, scope and outcome of the course, Text, Reference Books	CO1	20/9/21	22/9/21	
2	L1	Unit 1: Introduction and History of Operating systems: Operating system Structure	CO1	22/9/21	23/9/21	T1: 3-12
3	L2	Operating System operations;	CO1	23/9/21	27/9/21	T1:20
4	L3	processes and files, Process State	CO1	27/9/21	30/9/21	T123
5	L4	Processor management: inter process communication, Shared Memory, Message Passing	CO1	29/9/21	10-04-2021	T1: 101-128
6	L5	Critical Section, Critical section problem	CO1	30/9/21	10-06-2021	
7	L6	mutual exclusion, Methods: disable interrupts, lock variable, strict alternation, Peterson solutions	CO1	10-04-2021	10-07-2021	
8	L7	semaphores, wait and signal procedures, Binary Semaphore, Counting Semaphore, Priority inversion	CO1	10-06-2021	10-11-2021	T1: 227-244
9	L8	process scheduling and algorithms,FCFS,	CO1	10-07-2021	13/10/21	T1: 183-206
10	L9	SJF Algo, Priority Scheduling, Round robin Algo	CO1	10-11-2021	14/10/21	
11	L10	critical sections, threads, multithreading	CO1	13/10/21	18/10/21	
12	L11	Unit 2: Memory management: Introduction, contiguous memory allocation,	CO2	14/10/21	20/10/21	T1: 315-342
13	L12	virtual memory, paging,	CO2	18/10/21	21/10/21	T1:328
14	L13	page table structure, demand paging,	CO2	20/10/21	25/10/21	T1:337
15	L14	page replacement policies, Basic page replacement	CO2	21/10/21	27/10/21	T1:369-382
16	L15	FIFO page replacement, optimal page replacement	CO2	25/10/21	28/10/21	
17	L16	LRU page replacement	CO2	27/10/21	29/10/21	
18	L17	thrashing, segmentation, case study	CO2	28/10/21	29/10/21	T1:386
19	L18	Class Test/Assignment		29/10/21		
20	L19	Unit-3: Deadlock: Deadlock introduction, Deadlock characterization,	CO3	11-08-2021	11-10-2021	T1: 283-285
21	L20	Shared resources,resource allocation and scheduling	CO3	11-10-2021	18/11/21	
22	L21	deadlock detection-Single instance of each resource type,deadlock avoidance-resource allocation graph algorithms,	CO3	18/11/21	22/11/21	T1: 290-304
23	L22	The Ostrich Algorithm, Deadlock avoidance- Banker's Algorithms	CO3	22/11/21	24/11/21	
24	L23	deadlock prevention algorithms	CO3	24/11/21	25/11/21	
25	L24	Device management: devices and their characteristics,	CO3	25/11/21	25/11/21	
26	L25	device drivers, device handling	CO3	29/11/21	29/11/21	
27	L26	disk scheduling algorithms and policies	CO3	12-01-2021	12-01-2021	
28	L27	Unit-4: File management: file concept, types and structures,	CO4	12-02-2021	12-02-2021	T1:421-451
29	L28	directory structures- one level, two level, hierarchical/tree, acyclic graph	CO4	12-06-2021	12-02-2021	
30	L29	cases studies, access methods and matrices,	CO4	12-08-2021	12-06-2021	
31	L30	file security, user authentication	CO4	12-09-2021	12-08-2021	


Dr. Mahesh Bunde
 B.E., M.E., Ph.D.
 Director
 Poornima College of Engineering
 ISI-0, PIIICO Institutional Area
 Sitapura, JAIPUR

32	L31	File System Implementation	CO4	13/12/21	12-09-2021	
33	L32	Unit-5:UNIX and Linux operating systems as case studies: Linux History, Design principle	CO5	15/12/21	13/12/21	T1: 801-845
34	L33	Kernel modules, process managent, memory management	CO5	16/12/21	15/12/21	
35	L34	File System, network strycture	CO5	20/12/21	16/12/21	
36	L35	Time OS	CO5	22/12/21	20/12/21	
37	L36	case studies of Mobile OS	CO5	23/12/21	22/12/21	
38	L37	Class Test/Assignment		24/12/21		
39	L38	Revision		26/12/21	22/12/21	

Study material

Textbook:

- T1: A. Silberschatz and Peter B Galvin: Operating System Principals, Wiley India Pvt. Ltd.
T2: Tanenbaum: Modern Operating System, Prentice Hall.

Reference books:

- R1: Achyut S Godbole: Operating Systems, Tata McGraw Hill
R2: DM Dhamdhare: Operating Systems – A Concepts Based Approach, Tata McGraw Hill
R3: Charles Crowley: Operating System A Design – Oriented Approach, Tata McGraw Hill.

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

Campus: Poornima College of Engineering

Course: B.Tech.

Name of Faculty: Mr. Manish Choubisa

Year/Section: 3rd

Semester/ Section – 5B

Name of Subject: Operating System

Date: 18 Sept 2021

Code: 5CS4-03

COURSE OUTCOME :


- **5CS4-03.1 (CO1)**
Students will able to solve the problem based on basic concepts of operating system, deadlocks, memory management and file structures.
- **5CS4-03.2 (CO2)**
Students will be able to apply process management, deadlocks and memory management problems.
- **5CS4-03.3 (CO3)**
Students will be able to analyze various security and user authentication techniques in file management system.
- **5CS4-03.4 (CO4)**
Students will be able to create case study (the relative matrix some functionality) of the Windows, IOS and Linux and mobile OS.

CO-PO-PSO Mapping: Mapping Levels: 1- Low, 2- Moderate, 3-Strong

CO	PO												PSO		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	-	-	-	-	-	-	-	-	-	-	-	1	-	2
CO2	-	-	3	-	-	-	-	-	-	-	-	-	1	-	2
CO3	-	-	-	3	-	-	-	-	-	-	-	-	1	-	2
CO4	-	-	-	-	3	-	-	-	-	-	-	-	1	-	2

CO-PO MAPPING JUSTIFICATION

3CS4-06	CO1	PO1	3	Since programming knowledge and fundamentals are required in operating system, memory management, deadlock, mutual exclusion etc. hence CO1 is strongly mapped with PO1.
	CO2	PO3	3	Since concepts of deadlock, file management, are identified, formulated and analysis engineering hence CO2 mapped with PO3.
	CO3	PO4	3	Real world problem that are complex in nature will be analyzed and designed using various deadlock detection and prevention hence CO3 is mapped PO4.
	CO4	PO5	3	Evaluation of Unix and Linux operating system will be done hence CO4 is mapped PO5.


Dr. Mahesh Bunde
 B.E., M.E., Ph.D.
 Director
 Poornima College of Engineering
 ISI-0, FIICO Institutional Area
 Sitapura, JAIPUR

				strongly mapped with PO5.
--	--	--	--	---------------------------

CO-PSO MAPPING JUSTIFICATION

3CS4-06	CO1	PSO1, PSO3	1,2	Semaphores, process scheduling, critical section, paging can be provided for that hence CO1 is mapped with PSO1 and PSO3.
	CO2	PSO1, PSO3	1,2	Thrashing, contiguous memory allocation and disk scheduling algorithms are different but is possible so CO2 is mapped with PSO1 and PSO3.
	CO3	PSO1, PSO3	1,2	Design and development of directory structure, file concept are possible hence CO3 is mapped with PSO1 and PSO3.
	CO4	PSO1, PSO3	1,2	Case study of Unix and Linux operating system are possible so that CO4 is mapped with PSO1 and PSO3.

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

MID SEMESTER EXAMS: CO ATTAINMENT LEVELS

Course Category	Level 3	Level 2	Level 1
A	60 % of students getting > 60% marks	50-60 % of students getting > 60% marks	40-50 % of students getting > 60% marks

END TERM RTU COMPONENT: CO ATTAINMENT LEVELS

Course Category	Level 3	Level 2	Level 1
A	50 % of students getting > 60% marks	40-50 % of students getting > 60% marks	30-40 % of students getting > 60% marks

CO ATTAINMENT LEVELS FOR THEORY OF COMPUTATION

S. No.	Course Type	Attainment Level=1	Attainment Level=2	Attainment Level=3
1	Theory Courses Mid Semester Exams	40-50 % of students getting > 60% marks	50-60 % of students getting > 60% marks	60 % of students getting > 60% marks
2	Theory Courses University Exam	30-40 % of students getting > 60% marks	40-50 % of students getting > 60% marks	50 % of students getting > 60% marks
3	Assignments/Unit Test	40-50 % of students getting > 60% marks	50-60 % of students getting > 60% marks	60 % of students getting > 60% marks

CO WISE ASSESSMENT ACTIVITIES (AS MENTIONED IN SESSION PLAN)

	Activities			
CO	Class Test	Assignment	Mid 1	Mid 2
CO1	Yes	Yes	Yes	Yes
CO2	Yes	Yes	Yes	Yes
CO3	Yes	Yes	Yes	Yes
CO4	Yes	Yes	Yes	Yes

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

PO/PSO MAPPING AND TARGETS

CO	PO												Avg.	PSO		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	CO Targets	PSO1	PSO2	PSO3
CO1	3	-	-	-	-	-	-	-	-	-	-	-	3	1	-	2
CO2	-	-	3	-	-	-	-	-	-	-	-	-	3	1	-	2
CO3	-	-	-	3	-	-	-	-	-	-	-	-	3	1	-	2
CO4	-	-	-	-	3	-	-	-	-	-	-	-	3	1	-	2

ACTIVITY WISE ASSESSMENT TOOLS

S.No.	Activity	Assessment Method	Tools	Weightage Marks	Recommendation
1.	Mid Term 1	Direct	Marks	50	For CO1,2,3,4
2.	Class Test	Direct	Marks	40	For CO1,2,3,4
3.	Assignment	Direct	Marks	20	For CO1,2,3,4,
4.	Mid Term 2	Direct	Marks	50	For CO1,2,3,4
Note that for every rubrics you need to decide assessment criteria, range of marks or weightage – above values are indicative					

Teacher Mr.Manish Choubisa

Minha Escola

	1 8:30 - 9:30	2 9:30 - 10:30	3 10:30 - 11:30	LUNCH 11:30 - 12:10	4 12:10 - 13:10	5 13:10 - 14:10	6 14:10 - 15:10	7 15:10 - 16:00
Mo	5CS4-03-OS CF-04 III YEAR V-B	7CS4-22-CS Lab AF-8A IV YEAR VII-C VII-C2				7CS4-22-CS Lab AF-7A IV YEAR VII-C VII-C2		
Tu		NSP CF-03 III YEAR V-A				7CS4-22-CS Lab AF-8A IV YEAR VII-C VII-C1		3CS7-30-INDUSTRIAL TRAINING II YEAR III-B
We		7CS4-22-CS Lab AF-8A IV YEAR VII-C VII-C1				5CS4-03-OS CF-04 III YEAR V-B	7CS7- PROJECT AF-1C IV YEAR VII-A VII-A2	
Th								3CS7-30-INDUSTRIAL TRAINING II YEAR III-A
Fr	5CS4-03-OS CF-04 III YEAR V-B							
Sa								

Timetable generated:01-Nov-21


Dr. Mahesh Bunde
 B.E., M.E., Ph.D.
 Director
 Poornima College of Engineering
 ISI-0, FIICO Institutional Area
 Sikapura, JAIPUR

2. Timetables

5E1353

Roll No. _____

Total No. of Pages: **3**

5E1353

B. Tech. V - Sem. (Main / Back) Exam., January - 2022
Computer Science & Engineering
5CS4 – 03 Operating System
CS, IT

Time: 3 Hours

Maximum Marks: 120
Min. Passing Marks: 42

Instructions to Candidates:

Attempt all ten questions from Part A, five questions out of seven questions from Part B and four questions out of five from Part C.

Schematic diagrams must be shown wherever necessary. Any data you feel missing may suitably be assumed and stated clearly. Units of quantities used /calculated must be stated clearly.

Use of following supporting material is permitted during examination. (Mentioned in form No. 205)

1. NIL

2. NIL

PART – A

(Answer should be given up to 25 words only)

[10×2=20]

All questions are compulsory

Q.1 What is kernel?

Q.2 What is thread?

Q.3 What is deadlock?

Q.4 Define logical and physical address.

Q.5 What are context switches?

[5E1353]

Page 1 of 3

[4220]

<https://www.rtuonline.com>


Dr. Mahesh Bunde
B.E., M.E., Ph.D.
Director
Poornima College of Engineering
ISI-0, RICO Institutional Area
Ghatapada, JAIPUR

- Q.6 Differentiate between pager and swapper.
Q.7 Explain the features of Operating System.
Q.8 What are frames?
Q.9 What is thrashing?
Q.10 Explain 'valid' and 'invalid' bit in page table.

PART – B

(Analytical/Problem solving questions)

[5×8=40]

Attempt any five questions

- Q.1 What are preemptive and non-preemptive scheduling process? Explain the process state diagram in detail.
- Q.2 What are the necessary conditions for deadlock? Explain resource graph model and safe-unsafe states with a suitable example.
- Q.3 Explain the followings -
- (a) Inter-process Communication.
 - (b) Mutual Exclusion and Race Condition.
 - (c) Critical Section.
- Q.4 What do you mean by demand paging? Explain virtual memory and page fault concept in detail.
- Q.5 What is file management? Explain its types and structures.
- Q.6 Differentiate between Windows and Linux based operating system.
- Q.7 What is Memory Management Unit (MMU)? Explain Best Fit, Worst Fit and Quick Fit Algorithms in detail.

PART – C

(Descriptive/Analytical/Problem Solving/Design Questions)

[4×15=60]

Attempt any four questions

- Q.1 Consider the following page reference string 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
Compare the number of page faults with frame size 3, 4 with FIFO & LRU page replacement algorithm. Also explain Belady's anomaly in detail.
- Q.2 (a) Explain the difference between long term, short term and medium term schedulers.
(b) Explain the layered approach of the Operating System.
- Q.3 For the following set of process, find the average waiting time and turn around time using Gantt chart for –
(a) SJF
(b) Priority scheduling process

Process	Burst time (ms)	Priority
P1	5	5
P2	3	4
P3	8	3
P4	2	1
P5	1	2

- Q.4 Suppose a disk drive has 200 cylinders. The drive is initially at cylinder position 98.
The queue with request from I/O to blocks on cylinders –
86, 147, 91, 177, 94, 150, 102, 175, 130
What is the total head movement needed to satisfy the request for SCAN and C-SCAN scheduling algorithm?
- Q.5 Explain the followings –
(a) Data Structure of Bankers Algorithm
(b) Segmentation
(c) File Security

POORNIMA COLLEGE OF ENGINEERING, JAIPUR
DEPARTMENT OF COMPUTER ENGINEERING

Zero Lecture Session: 2021-22 (Odd Sem)

Name of Faculty: Mr. Manish Choubisa

Branch: Computer Engineering

1) Name of Subject with Code: Operating System (5CS4-03)

2) Self-Introduction:

- a) **Name:** Mr. Manish Choubisa
- b) **Qualification:** BE (CSE), M Tech (IC)
- c) **Designation:** Assistant Professor
- d) **Research Area:** Image Processing, Computer Network
- e) **E-mail Id:** manish.choubisa@poornima.org
- f) **Other details:**
 - More than 10 years of teaching experience.
 - Around 10 Papers in National, International journals and Conferences

3) Introduction of Students: III year V Semester (Computer Engineering)

- a) Identifying and keeping records of students based on meritorious/weak in academics.

4) Instructional Language: - 100% English

5) Introduction to subject: An operating system acts as an intermediary between the user of a computer and computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs conveniently and efficiently.

An operating system is a software that manages computer hardware. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.

Objective:

The objective of this course is

- To understand the services provided by and the design of an operating system.
- To understand the structure and organization of the file system.
- To understand what a process is and how processes are synchronized and scheduled.
- To understand different approaches to memory management.
- Students should be able to use system calls for managing processes, memory and the file system.
- Students should understand the data structures and algorithms used to implement an OS.

6) Syllabus of Rajasthan Technical University, Kota

RAJASTHAN TECHNICAL UNIVERSITY, KOTA	
Syllabus	
III Year-V Semester: B.Tech. Computer Science and Engineering	
5CS4-03: Operating System	
Credit: 3	Max. Marks: 150(IA:30, ETE:120)
3L+0T+0P	End Term Exam: 3 Hours
Unit-1	Introduction: Objective, scope and outcome of the course. Introduction and History of Operating systems: Structure and operations; processes and files Processor management: inter process communication, mutual exclusion, semaphores, wait and signal procedures, process scheduling and algorithms, critical sections, threads, multithreading
Unit-2	Memory management: contiguous memory allocation, virtual memory, paging, page table structure, demand paging, page replacement policies, thrashing, segmentation, case study
Unit-3	Deadlock: Shared resources, resource allocation and scheduling, resource graph models, deadlock detection, deadlock avoidance, deadlock prevention algorithms Device management: devices and their characteristics, device drivers, device handling, disk scheduling algorithms and policies
Unit-4	File management: file concept, types and structures, directory structure, cases studies, access methods and matrices, file security, user authentication
Unit-5	UNIX and Linux operating systems as case studies; Time OS and case studies of Mobile OS

7) Books/ Website/Journals & Handbooks/ Association & Institution

S. N.	Title of Book	Authors	Publisher
Text Books			
T1	Operating System Principals,	A. Silberschatz and Peter B Galvin	Wiley India Pvt. Ltd.
T2	:Modern Operating System,	Tanenbaum	Prentice Hall.
Reference Books			
R1	Operating Systems,	Achyut S Godbole	Tata Mc Graw Hill.


Dr. Mahesh Bunde
 B.E., M.E., Ph.D.
 Director
 Poonima College of Engineering
 ISI-0, FIICO Institutional Area
 Sitapura, JAIPUR

R2	Operating Systems – A Concepts Based Approach,	DM Dhamdhere:	Tata McGraw Hill.
R3	Operating System A Design – Oriented Approach,	Charles Crowley	Tata McGraw Hill.
Websites related to subject			
1	https://archive.nptel.ac.in/courses/106/105/106105214/		
2	https://mrcet.com/downloads/digital_notes/CSE/III%20Year/OPERATING%20SYSTEMS%20DIGITAL%20%20NOTES-18.pdf		

8) Syllabus Deployment: -

a). Total no of Lectures: - 39

Unit 1: 11 lecture

Unit 2: 8 lecture

Unit 3: 8 lecture

Unit 4: 5 lecture

Unit 5: 6 lecture

b) Special Activities (To be approved by HOD, Dean & Campus Director & must be mentioned in deployment):

- Open Book Test- Once in a semester
- Quiz (100% Technical)- One in a semester
- Special Lectures (SPL)- 10% of total no. of lectures including following
 - i. Few PPT Lecture
- Revision classes:- 1 to 3 turn at the end of semester (Before II Mid Term Exam)
- Solving Important Question Bank- 1 Turn before I & II Mid Term Exam (each) - Total Two turn.

c) **Lecture schedule per week:**

i). University scheme (L+T+P) = 3L+0T+0P

ii). PGC scheme (L+T+P) = 4L+0T+0P

d) **Introduction & Conclusion:** Each subject, unit and topic shall start with introduction & close with conclusion.

e) **Time Distribution in lecture class:** - Number of chapters is beginning with objective and end of course/chapter/lecture with summary and quiz (Time allotted: 60 min.)

- First 5 min. should be utilized for paying attention towards students who were absent for last lecture or continuously absent for many days + taking attendance by calling the names of the students and also sharing any new/relevant information.
- Actual lecture delivery should be of 45 minutes
- Last 5 min. should be utilized by recapping of the topic. Providing brief introduction of the coming up lecture and suggesting portion to read.
- After completion of any Unit/Chapter a short quiz should be organized
- During lecture student should be encouraged to ask the question.

7) **University Examination systems: -**

Sr. No.	Name of the exam	Max. Marks	passing marks	Nature of paper	Syllabus Coverage	Conducted by
1.	I Mid Term Exam	30	10	Theory + Numerical	60%	PCE
2.	II Mid Term Exam	30	10	Theory + Numerical	40%	PCE
3.	University (End)Term exam	120	48	Theory + Numerical	100%	RTU

Place: Jaipur

Mr. Manish Choubisa
Assistant Professor


Dr. Mahesh Bunde
B.E., M.E., Ph.D.
Director
Poonima College of Engineering
ISI-0, RICO Institutional Area
Sitapura, JAIPUR

FIRST MID TERM EXAMINATION 2021-22

Code: 5CS4-03 Category: PCC Subject Name-Operating System
(BRANCH – COMPUTER ENGINEERING)Course Credit: 03
Max. Marks: 60

Max. Time: 2 hrs.

NOTE:- Read the guidelines given with each part carefully.**Course Outcomes (CO):**

At the end of the course the student should be able to:

CO1: Understand the basic concepts of Operating System and solve problem based on process scheduling and inter process communication.

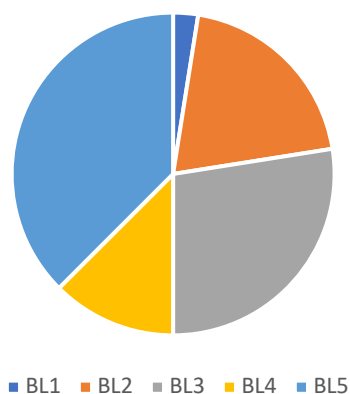
CO2: Evaluate and identify various memory management techniques in term of paging and segmentation.

CO3: Apply the methods for deadlock handling in operating system and device management policies for disk scheduling.

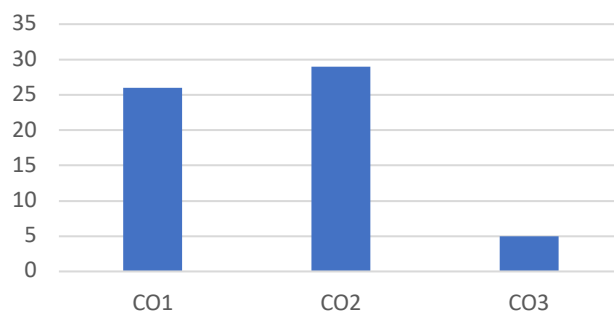
PART - A: (All questions are compulsory) Max. Marks (10)					
		Marks	CO	BL	PO
Q.1	What are the main functions of Memory Management module in Operating System?	2	CO1	BL2	PO1
Q.2	What do you mean by process? Explain some of the differences between process and program?	2	CO2	BL2	PO12
Q.3	State various differences between Multiprogramming and Multitasking? Explain with the help of any example with respect to the operating system?	2	CO1	BL1	PO1
Q.4	Explain some of the shortcomings of Round Robin Scheduling algorithm? Explain with the help of any example?	2	CO1	BL3	PO1
Q.5	Explain the term Convoy effect, Starvation and Aging with respect to CPU Scheduling Algorithm?	2	CO2	BL2	PO12
PART - B: (Attempt 4 questions out of 6) Max. Marks (20)					
Q.6	What do you mean by the state of a Process? Explain the Process State Diagram in detail?	5	CO2	BL3	PO12
Q.7	What do you mean by term “Semaphores”? How it can be useful to achieve mutual exclusion? Explain the counting Semaphore with its pseudo code with the help of any example?	5	CO2	BL4	PO12
Q.8	What is Mutual exclusion? Discuss some of the approaches to achieve the mutual exclusion? Explain some of the approaches with the help of any example?	5	CO1	BL4	PO1
Q.9	What do you mean by Contiguous Memory Allocation? What are the different techniques used in Contiguous Memory Allocation? Explain in detail.	5	CO2	BL3	PO12
Q.10	What is Deadlock in Operating System? List four necessary condition for occurrence of the Deadlock in the system?	5	CO3	BL2	PO3
Q.11	Explain the Process Control Block (PCB) in Process Management? How it can be useful to manage all process related activities? Explain in detail?	5	CO1	BL2	PO1
PART - C: (Attempt 3 questions out of 4) Max. Marks (30)					

Q.12	Consider the following set of processes-		10	CO1	BL5	PO1	
	Process	CPU Burst Time					Arrival Time
	P1	25					0
	P2	10					5
	P3	5					10
	P4	15					10
(A) Create a Gantt chart for the Non-Preemptive and Preemptive SJF Scheduling Algorithm?							
(B) Find the Waiting Time and Turnaround Time of each Process by considering the Non-Preemptive and Preemptive SJF Scheduling Algorithm?							
Q.13	Consider the following page reference string: 1, 2, 3, 4, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 3, 6. How many page faults would occur for the Optimal Page Replacement algorithm? Consider the number of frames in physical memory is: 4. Find the Fault rate for the algorithm?		10	CO2	BL5	PO12	
Q.14	What is Paging? How a logical address is converted in to physical address in Paging? Explain Address Translation Architecture diagram used in Paging with the help of any example?		10	CO2	BL3	PO12	
Q.15	Consider the following page reference string: 4, 5, 1, 2, 3, 5, 4, 3, 7, 1, 2, 3, 4, 1, 2. How many page faults would occur for the LRU Page Replacement algorithm? Consider the number of frames in physical memory is: 3. Find the Fault rate for the algorithm?		10	CO2	BL5	PO12	

Bloom's Taxonomy Levels



COURSE OUTCOME WISE MARKS DISTRIBUTION



BL – Bloom's Taxonomy Levels (1- Remembering, 2- Understanding, 3 – Applying, 4 – Analyzing, 5 – Evaluating, 6 - Creating)

CO – Course Outcomes; PO – Program Outcomes

Assignment-I

Code: 5CS4-03 Category: PCC Subject Name-Operating System
(BRANCH – COMPUTER ENGINEERING)

Max. Marks: 20

Note: Attempt any eight questions.

Don't copy the assignment. It leads to reduction of marks.

Last date to submit the assignment is:31-11-21.

Q.1	CO1	PO1	What do you mean by term “Semaphores”? How it can be useful to achieve mutual exclusion? Explain the counting Semaphore with its pseudo code with the help of any example?	5																		
Q.2	CO1	PO1	Explain the term Convoy effect, Starvation and Aging with respect to CPU Scheduling Algorithm? Explain some of the shortcomings of Round Robin Scheduling algorithm? Explain with the help of any example?	5																		
Q.3	CO2	PO12	What do you mean by the state of a Process? Explain the Process State Diagram in detail? Explain the Process Control Block (PCB) in Process Management? How it can be useful to manage all process related activities? Explain in detail?	5																		
Q.4	CO1	PO1	What do you mean by the term OS? What are the different services provided by the operating system to its users? Explain different type of OS.	5																		
Q.5	CO1	PO1	What do you mean by Context switching? Also explain the difference between Multiprogramming and Multitasking Operating System?	5																		
Q.6	CO2	PO12	<div>Consider the following set of processes-</div> <table><tr><td>Process</td><td>CPU Burst Time</td><td>Arrival Time</td></tr><tr><td>P1</td><td>6</td><td>2</td></tr><tr><td>P2</td><td>2</td><td>5</td></tr><tr><td>P3</td><td>8</td><td>1</td></tr><tr><td>P4</td><td>3</td><td>0</td></tr><tr><td>P5</td><td>4</td><td>4</td></tr></table> <div>(A) Create a Gantt chart for the Non-Preemptive and Preemptive SJF Scheduling Algorithm?</div> <div>Find the Waiting Time and Turnaround Time of each Process by considering the Non-Preemptive and Preemptive SJF Scheduling Algorithm?</div>	Process	CPU Burst Time	Arrival Time	P1	6	2	P2	2	5	P3	8	1	P4	3	0	P5	4	4	5
Process	CPU Burst Time	Arrival Time																				
P1	6	2																				
P2	2	5																				
P3	8	1																				
P4	3	0																				
P5	4	4																				
Q.7	CO2	PO12	<div>Consider the following set of processes-</div> <table><tr><td>Process</td><td>CPU Burst Time</td><td>Arrival Time</td></tr><tr><td>P1</td><td>15</td><td>0</td></tr><tr><td>P2</td><td>14</td><td>3</td></tr><tr><td>P3</td><td>22</td><td>7</td></tr><tr><td>P4</td><td>09</td><td>9</td></tr><tr><td>P5</td><td>18</td><td>16</td></tr></table> <div>Dr. Mahesh Bundela</div>	Process	CPU Burst Time	Arrival Time	P1	15	0	P2	14	3	P3	22	7	P4	09	9	P5	18	16	5
Process	CPU Burst Time	Arrival Time																				
P1	15	0																				
P2	14	3																				
P3	22	7																				
P4	09	9																				
P5	18	16																				

			(B) Create a Gantt chart for the Round Robin Scheduling Algorithm? Find the Waiting Time and Turnaround Time of each Process by using the Round Robin Scheduling Algorithm? Consider TQ=10.	
Q.8	CO1	PO1	What is Mutual exclusion? Discuss some of the approaches to achieve the mutual exclusion?	5
Q.9	CO1	PO1	What is IPC with respect to OS? What do you mean by race condition? How the race condition will occur in Spooler Directory of a printer? Explain in detail.	5
Q.10	CO3	PO4	What is Belady's anomaly in page replacement algorithm? What are disadvantages of FIFO and Optimal Page replacement algorithm?	5
Q.11	CO3	PO4	Consider the following page reference string: 4, 5, 1, 2, 3, 5, 4, 3, 7, 1, 2, 3, 4, 1, 2. How many page faults would occur for the LRU Page Replacement algorithm? Consider the number of frames in physical memory is: 3. Find the Fault rate for the algorithm?	5

SECOND MID TERM EXAMINATION 2021-22

Code: 5CS4-03 Category: PCC Subject Name-Operating System
(BRANCH – COMPUTER ENGINEERING)Course Credit: 03
Max. Marks: 60

Max. Time: 2 hrs.

NOTE:- Read the guidelines given with each part carefully.**Course Outcomes (CO):**

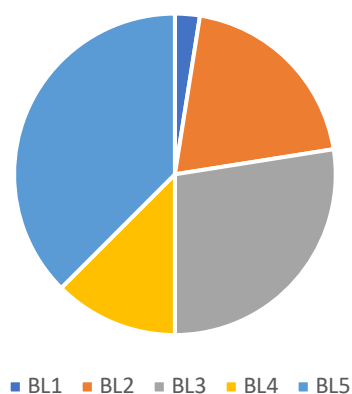
At the end of the course the student should be able to:

CO 1. Understand the basic concepts of Operating System and solve problem based on process scheduling and inter process communication.**CO 2.** Evaluate and identify various memory management techniques in term of paging and segmentation.**CO 3.** Apply the methods for deadlock handling in operating system and device management policies for disk scheduling.**CO 4.** Analyze various security and user authentication techniques in file management system**CO 5.** Create case study (the relative matrix some functionality) of the RTOS, Linux and Mobile OS.

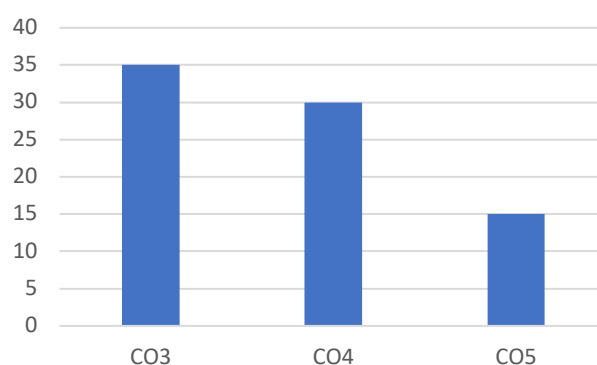
PART - A: (All questions are compulsory) Max. Marks (10)					
		Marks	CO	BL	PO
Q.1	Define the term Disk scheduling? Give some of the names of disk scheduling algorithms?	2	CO3	BL2	PO3
Q.2	Explain the term “Safe State or Unsafe State” with the help of any example?	2	CO3	BL3	PO3
Q.3	Explain Device management? Why we need to install the device driver in the operating system? Justify the specific reason for that.	2	CO3	BL3	PO3
Q.4	What do you mean by term File and File Systems? What is the need of file system in an operating system?	2	CO4	BL2	PO4
Q.5	What are the distinct operations performed by the operating systems on the file ? List them all.	2	CO4	BL2	PO4
PART - B: (Attempt 4 questions out of 6) Max. Marks (20)					
Q.6	Cite the different file allocation method? Explain some of merits and demerits of each file allocation method in detail?	5	CO4	BL3	PO4
Q.7	How the deadlock is detected in the system? Explain some of the methods that are used to recover the deadlock?	5	CO3	BL4	PO3
Q.8	Explain Free space management in File System? How the free space managed in the file system? Explain different methods to handle free space?	5	CO4	BL2	PO4
Q.9	Discuss the Serial and Index sequential access methods for accessing a file in the system? How they are different from the direct access.	5	CO4	BL3	PO4
Q.10	State some of the differences between UNIX and Linux operating system?	5	CO5	BL3	PO2
Q.11	Suppose a disk drive has 200 tracks, numbered from 0 to 199. The read / write head at track number 53. The queue with requests from I/O to blocks on tracks are as- 98 , 183, 37 , 122, 14, 124, 65, 67 The read write direction is upward from the current position. Find the total seek time using the Scan Disk Scheduling algorithm.	5	CO3	BL5	PO3
PART - C: (Attempt 3 questions out of 4) Max. Marks (30)					
Q.12	How the operating system Handle the deadlock in the system? Explain deadlock prevention method of handling deadlock in the system?	10	CO3	BL4	PO3
Q.13	Write a short note on a case study of the following topics: (Any Two). (a) Unix	5+5	CO5	BL3	PO2

	(b) Linux (c) RTOS (d) Mobile OS.																																																															
Q.14	Consider the following snapshot of the system at time T0: <table><tr><th rowspan="2">Process</th><th colspan="3">Allocation</th><th colspan="3">Maximum</th><th colspan="3">Available</th></tr><tr><th>R1</th><th>R2</th><th>R3</th><th>R1</th><th>R2</th><th>R3</th><th>R1</th><th>R2</th><th>R3</th></tr><tr><td>P0</td><td>1</td><td>2</td><td>1</td><td>2</td><td>5</td><td>1</td><td>2</td><td>2</td><td>1</td></tr><tr><td>P1</td><td>2</td><td>1</td><td>0</td><td>3</td><td>4</td><td>0</td><td></td><td></td><td></td></tr><tr><td>P2</td><td>1</td><td>1</td><td>2</td><td>5</td><td>3</td><td>4</td><td></td><td></td><td></td></tr><tr><td>P3</td><td>2</td><td>1</td><td>0</td><td>4</td><td>2</td><td>1</td><td></td><td></td><td></td></tr></table> <p>According to Banker's algorithm answer the following:</p> <p>(a). Show is the content of need Matrix?</p> <p>(b). Suppose process p3 request (2,1,1) resources. Can this request be granted? If yes ,then find out the safe state of the system.</p>	Process	Allocation			Maximum			Available			R1	R2	R3	R1	R2	R3	R1	R2	R3	P0	1	2	1	2	5	1	2	2	1	P1	2	1	0	3	4	0				P2	1	1	2	5	3	4				P3	2	1	0	4	2	1				10	CO3	BL5	PO3
Process	Allocation			Maximum			Available																																																									
	R1	R2	R3	R1	R2	R3	R1	R2	R3																																																							
P0	1	2	1	2	5	1	2	2	1																																																							
P1	2	1	0	3	4	0																																																										
P2	1	1	2	5	3	4																																																										
P3	2	1	0	4	2	1																																																										
Q.15	Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The head is initially at cylinder number 53. The cylinders are numbered from 0 to 199. Find the total seek time and average disk service by using the SSTF and FCFS algorithm.	10	CO3	BL5	PO3																																																											

Bloom's Taxonomy Levels



COURSE OUTCOME WISE MARKS DISTRIBUTION



BL – Bloom's Taxonomy Levels (1- Remembering, 2- Understanding, 3 – Applying, 4 – Analyzing, 5 – Evaluating, 6 - Creating)

CO – Course Outcomes; PO – Program Outcomes

Poornima College of Engineering, Jaipur

Department of Computer Engineering

Assignment-II

Sub: OPERATING SYSTEM (5CS 4-03)

Semester: V SEM-B

Date: 17/12/2021

Max. Marks: 20

*Last date of Submission: 23/12/2021

Q No.	CO	PO	Marks	Questions																				
Q1.	CO 3	PO 3	4	<p>By using the Banker's Algorithm, consider a system with five processes P0 through P4 and three resource types A, B and C. resource type A has 10 instances, resource type B has 5 instances and resource type C has 7 instances.</p> <table><tr><th rowspan="2">PROCESS</th><th>ALLOCATION</th><th>MAX</th></tr><tr><th>ABC</th><th>ABC</th></tr><tr><td>P₀</td><td>010</td><td>753</td></tr><tr><td>P₁</td><td>200</td><td>322</td></tr><tr><td>P₂</td><td>302</td><td>902</td></tr><tr><td>P₃</td><td>211</td><td>222</td></tr><tr><td>P₄</td><td>002</td><td>433</td></tr></table> <p>a) What is the content of the Need matrix?</p> <p>b) Is the system in a safe state?</p>	PROCESS	ALLOCATION	MAX	ABC	ABC	P ₀	010	753	P ₁	200	322	P ₂	302	902	P ₃	211	222	P ₄	002	433
PROCESS	ALLOCATION	MAX																						
	ABC	ABC																						
P ₀	010	753																						
P ₁	200	322																						
P ₂	302	902																						
P ₃	211	222																						
P ₄	002	433																						
Q2.	CO 3	PO 3	4	<p>consider a disk with 200 tracks and the queue has random requests from different processes in the order:</p> <p>25, 58, 39, 18, 80, 165, 150, 38, 182</p> <p>Initially arm is at 68. Find the Average Seek length using SSTF and SCAN Algorithms.</p>																				
Q3.	CO 3	PO 3	4	<p>Explain the following;</p> <p>a. Resource allocation graph</p> <p>b. Recovery from Deadlock</p>																				
Q4.	CO 4	PO 4	4	What are attributes of File? Explain																				
Q5.	CO 5	PO 3	4	Describes the file authentication process in Linux operating system																				



POORNIMA

COLLEGE OF ENGINEERING

Summery Sheet

Name of faculty	Manish Choubisa
Class-	B. Tech – V SEM CS
Branch	Computer Engineering
Course Code	5CS4-03
Course Name	Operating System
Session	2021-22

COURSE OUTCOMES

After completion of this course, students should be able to:	
CO1	Understand the basic concepts of Operating System and solve problem based on process scheduling and inter process communication.
CO2	Evaluate and identify various memory management techniques in term of paging and segmentation.
CO3	Apply the methods for deadlock handling in operating system and device management policies for disk scheduling.
CO4	Analyse various security and user authentication techniques in file management system
CO5	Create case study (the relative matrix some functionality) of the RTOS, Linux and Mobile OS.

CO-PO/PSO MAPPING AND TARGETS

CO	PO												Avg.	PSO		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	CO Targets	PSO1	PSO2	PSO3
CO1	3	-	-	-	-	-	-	-	-	-	-	-	3	3	-	2
CO2	-	-	-	-	-	-	-	-	-	-	-	2	2	2	-	-
CO3	-	-	3	-	-	-	-	-	-	-	-	-	3	3	-	2
CO4	-	-	-	3	-	-	-	-	-	-	-	-	3	2	2	-
CO5	-	3	-	-	-	-	-	-	-	-	-	-	3	1	-	2

Level of course

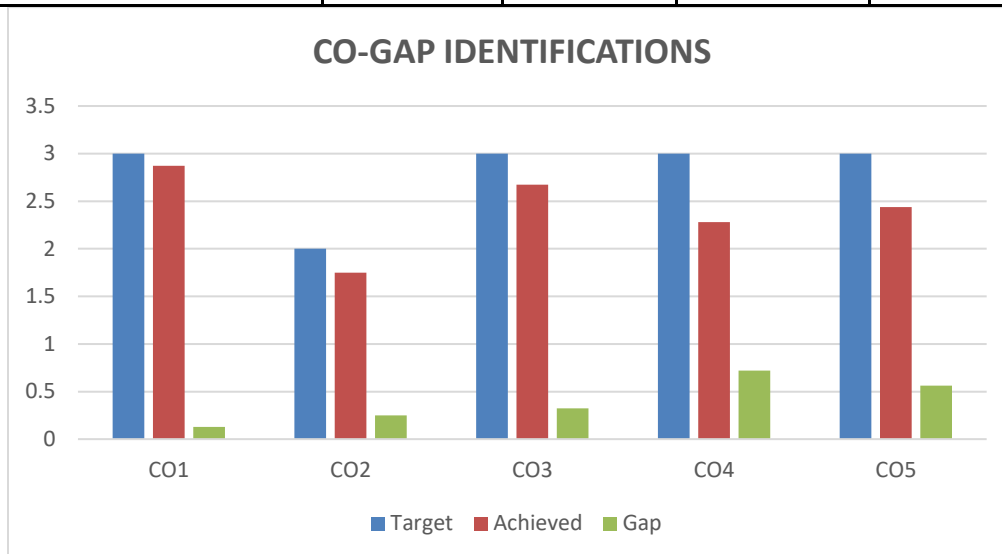
Course Category	Level 3	Level 2	Level 1
A	60% of students getting >60% marks	50-60% of students getting >60% marks	40-50% of students getting >60% marks

ACTIVITY WISE ASSESSMENT TOOLS

Sr. No.	Activity	Assessment Method	Tools	Weightage Marks	Recommendation
1.	Assignment I	Direct	Marks	30	For CO1-CO3
2.	Assignment II	Direct	Marks	30	For CO4-CO5
3.	MidTerm1	Direct	Marks	60	For CO1-CO3
4.	MidTerm2	Direct	Marks	60	For CO4-CO5

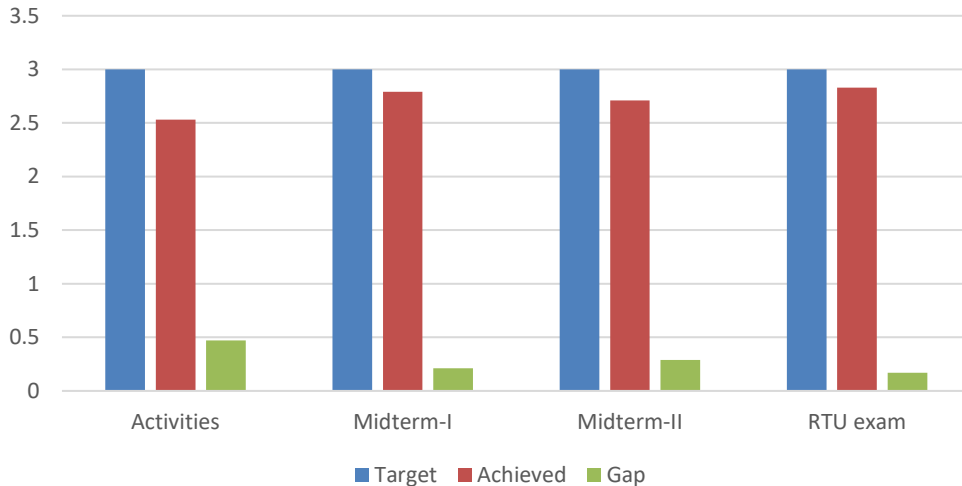
CO-GAP IDENTIFICATIONS

COs	CO1	CO2	CO3	CO4	CO5
Target	3	2	3	3	3
Achieved	2.871	1.749	2.675	2.2807	2.4386
Gap	0.129	0.251	0.325	0.7193	0.5614



Overall COs wise Attainment				
	Activities	Midterm-I	Midterm-II	RTU exam
Target	3	3	3.00	3
Achieved	2.53	2.79	2.71	2.83
Gap	0.47	0.21	0.29	0.17

Overall COs wise Attainment



Gaps Identified:

Describe what the reasons for gaps are

1. Numerical part of scheduling are need more practice
2. deadlock topic is hard to understand

Activities decided to bridge the gap

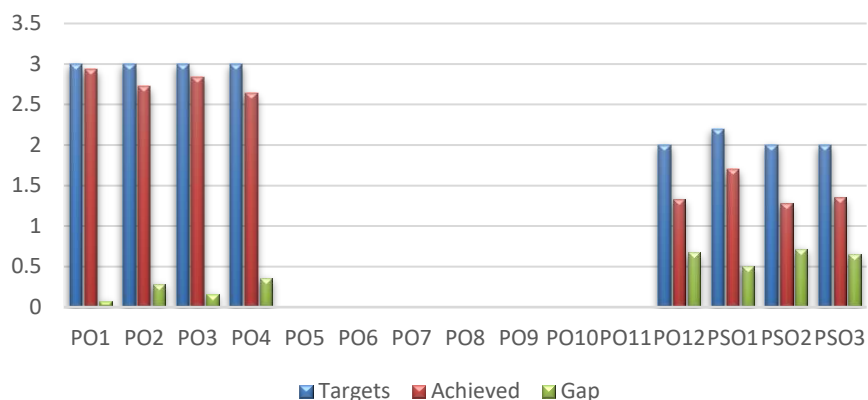
1. Extra class of scheduling topic example of GATE questions for more clarification
2. Give case study of deadlock of various OS

POs and PSOs GAP IDENTIFICATION

Attainment of PO through CO(Class Test, OBT and Quiz) Component															
5CS4-03	PO												PSO		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
Targets	3	3	3	3								2	2.2	2	2
Achieve	2.93	2.72	2.84	2.64								1.33	1.70	1.28	1.35
Gap	0.07	0.28	0.16	0.36								0.67	0.50	0.72	0.65
Attainment of PO through CO(MIDTERM-I) Component															
5CS4-03	PO												PSO		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
Targets	3	3	3	3								2	2.2	2	2
Achieved	2.87		2.49									1.27	1.93		1.32
Gap	0.13		0.51									0.73	0.27		0.68
Attainment of PO through CO(MIDTERM-II) Component															
5CS4-03	PO												PSO		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
Targets	3	3	3	3								2	2.2	2	2
Achieved	2.87	2.44	2.68	2.28								1.27	1.65	0.97	1.30
Gap	0.13	0.56	0.32	0.72								0.73	0.55	1.03	0.70
Attainment of PO through CO(RTU) Component															
5CS4-03	PO												PSO		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
Targets	3	3	3	3								2	2.2	2	2
Achieve	2.81	2.81	2.81	2.81								1.69	1.86	1.69	1.69
Gap	0.19	0.19	0.19	0.19								0.31	0.34	0.31	0.31
Attainment & Gap of Overall PO Session 2021-22															

5CS4-03	PO												PSO		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
Targets	3	3	3	3								2	2.2	2	2
Achieved	2.81	2.76	2.79	2.72								1.36	1.37	1.18	1.19
Gap	0.19	0.24	0.21	0.28								0.64	0.83	0.82	0.81

Attainment of PO through CO(Class Test, OBT and Quiz) Component



Gaps Identified:

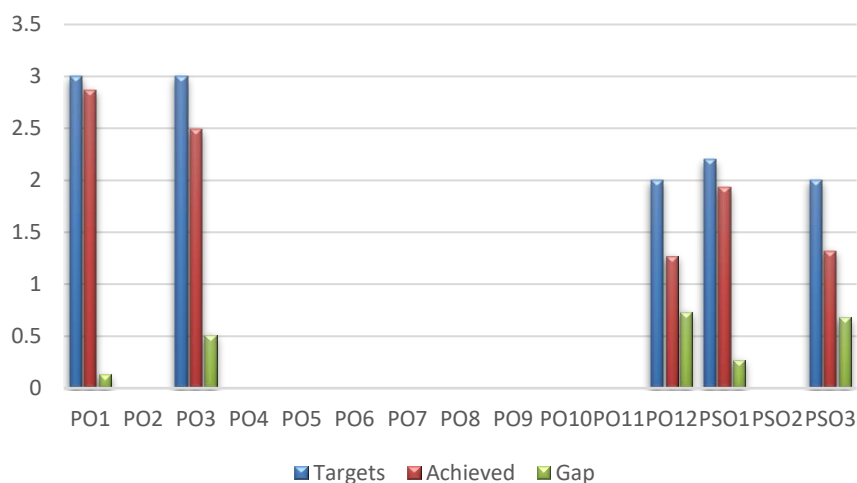
Describe what the reasons for gaps are

1. Minor gap identify
- 2.

Activities decided to bridge the gap

1. Discuss assignment in class
- 2.

Attainment of PO through CO(MIDTERM-I) Component



Gaps Identified:

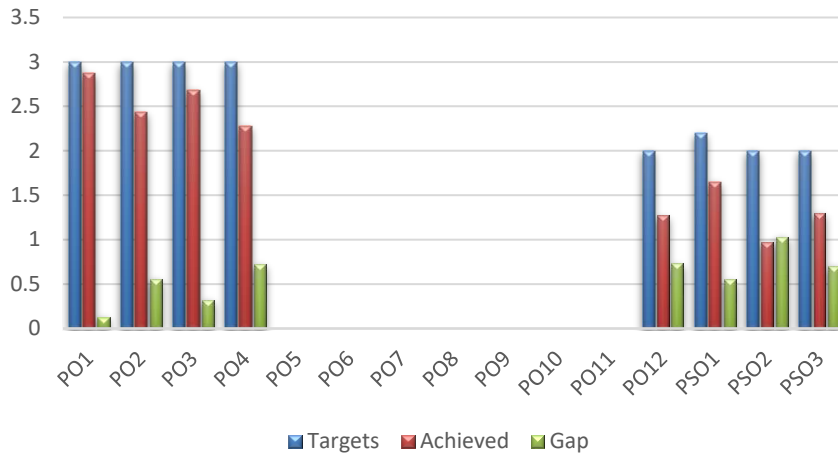
Describe what the reasons for gaps are

1. Numerical part CO achieved less
- 2.

Activities decided to bridge the gap

1. Extra class of scheduling topic example of GATE questions for more clarification
- 2.

Attainment of PO through CO(MIDTERM-II) Component



Gaps Identified:

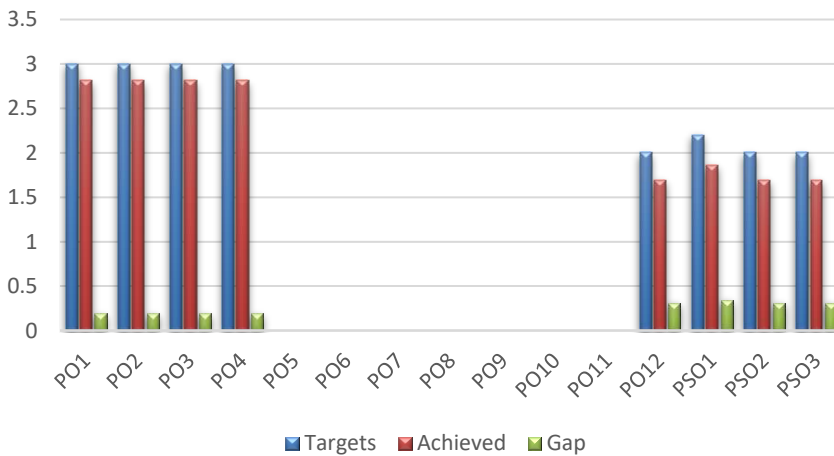
Describe what the reasons for gaps are

1. for PO2, some complex part of dead lock algorithms are less to under stand by students
- 2.

Activities decided to bridge the gap

1. Discuss deadlock with case study for different operating systems.
- 2.

Attainment of PO through CO(RTU) Component



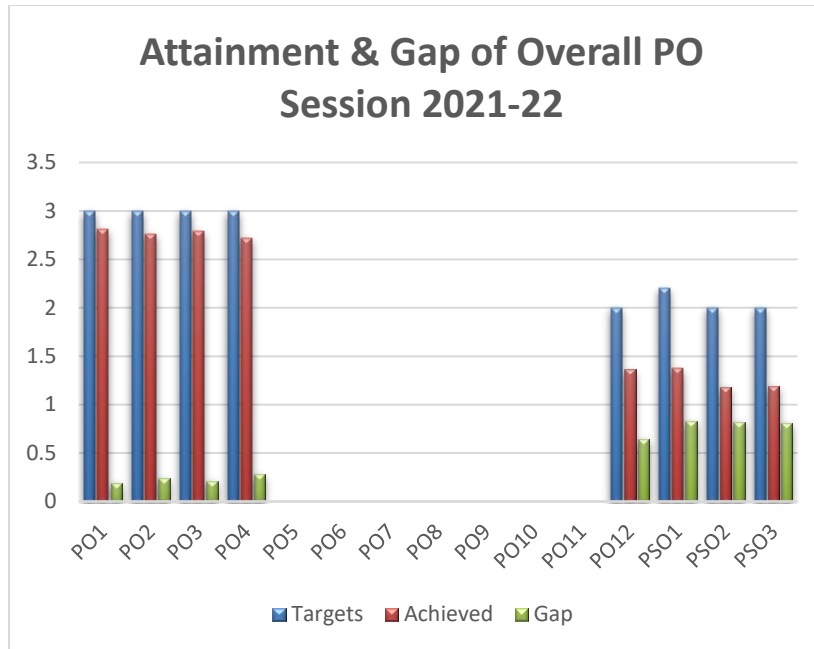
Gaps Identified:

Describe what the reasons for gaps are

1. Minor gap identify
2. some gap in PSO regarding software practices

Activities decided to bridge the gap

1. will Plan expert talk for next session regarding advance development in OS software



Gaps Identified:

Describe what the reasons for gaps are

1. Numerical part of scheduling are need more practice
2. deadlock topic is hard to understand

Activities decided to bridge the gap

1. Extra class of scheduling topic example of GATE questions for more clarification
2. Give case study of deadlock of various OS

Overall Comments:

- Minor gap identifies for Operating System course in PO1, PO2, PO3 and PO5.



POORNIMA

COLLEGE OF ENGINEERING

LECTURE NOTES

Chapter: PCE Course: CSE- Class/Section: 5th Date: _____
Name of Faculty: Manish Choudhary Name of Subject: Operating Systems Code: SCSU-03
Date (Prep.): _____ Date (Del.): _____ Unit No./Topic: 1 Lect. No: _____

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., which will be taught in this lecture)

- Operating System
- Types of O.S.
- CPU Scheduling
- Priority Scheduling

IMPORTANT & RELEVANT QUESTIONS:

- Definition & function of O.S.
- Types of O.S. in Detail - multitasking, multiprocessor
- Scheduling Based New merical Question
- Kernel level & User level thread

FEED BACK QUESTIONS (AFTER 20 MINUTES):

- Q. What is O.S & their types.
- Q. Explain CPU scheduling & Priority scheduling

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

Student should be learn to know about O.S., types of O.S., CPU scheduling

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:

Dr. Mahesh Bunde
B.E., M.E., Ph.D.

Director
Poornima College of Engineering
131-0, FIICO Institutional Area
Ghatapada, JAIPUR



POORNIMA

COLLEGE OF ENGINEERING

LECTURE NOTES

Campus: PCE Course: CSE Class/Section: 5th Date: _____
Name of Faculty: Manish Choudhary Name of Subject: O.S. Code: SSC403
Date (Prep.): _____ Date (Del.): _____ Unit No./Topic: 2 Lect. No: _____

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc. which will be taught in this lecture)

- Memory Management
- paging technique.
- Demand Paging
- Thrashing

IMPORTANT & RELEVANT QUESTIONS:

Contiguous & Non Contiguous Mem. allocation

FEED BACK QUESTIONS (AFTER 20 MINUTES)

- What is paging in OS.
- Explain Demand Paging
- What is thrashing & why it is used.

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

Student should be learn about the Memory Management & Paging technique.
Student should be learn about Demand Paging.

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites.



POORNIMA

COLLEGE OF ENGINEERING

LECTURE NOTES

Campus: PCE Course: SCSE Class/Section: 5th Date:
Name of Faculty: Manish Chauhan Name of Subject: D.S. Code: SCSE-03
Date (Prep.): Date (Del.): Unit No./Topic: 3 Lect. No:

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc.. which will be taught in this lecture)

- Deadlock Prevention, Avoidance
- Resource Allocation & scheduling
- Methods for Handling Deadlock
- Device Management

IMPORTANT & RELEVANT QUESTIONS:

Deadlock advantage & disadvantage
Method for Handling Deadlock
(Deadlock prevention, avoidance, recovery)
Devices & their characteristics)

FEED BACK QUESTIONS (AFTER 20 MINUTES):

- Q. What is deadlock & their characteristics
- Q. Explain device driver & characteristic of devices.
- Q. Disk scheduling algorithm.

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

Student should be learn about the Deadlock
Device management & disk scheduling.

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:



POORNIMA

COLLEGE OF ENGINEERING

LECTURE NOTES

Campus: PCF Course: CSE Class/Section: 5th Date:
Name of Faculty: Manish Choudhary Name of Subject: O.S. Code: 5CS4-03
Date (Prep.): Date (Del.): Unit No./Topic: 4 Lect. No:

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc.. which will be taught in this lecture)

- File Management
- File Concept, types & structure
- File Security
- File Access methods.

IMPORTANT & RELEVANT QUESTIONS:

- Detail about the file management system, file types, structure of file
- File Security
- File Access method

FEED BACK QUESTIONS (AFTER 20 MINUTES):

- Q.1 What is File system?
- Q.2 Explain types of file?
- Q.3 Explain different types of File Access method.

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

- Student should be learn about the file concepts, types of file.
- Student should be learn about the file Access method.

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:



POORNIMA

COLLEGE OF ENGINEERING

LECTURE NOTES

Campus: PCB Course: CSE Class/Section: 5th Date:
Name of Faculty: Manish Choudhary Name of Subject: O.S. Code: SCSY03
Date (Prep.): Date (Del.): Unit No./Topic: 5 Lect. No:

OBJECTIVE: To be written before taking the lecture (Pl. write in bullet points the main topics/concepts etc., which will be taught in this lecture)

Case study on
Unix, Linux
Mobile OS
RTOS

IMPORTANT & RELEVANT QUESTIONS:

- unix & linux Advantage & Disadvantage
- mobile O.S. Features.
- RTOS details

FEED BACK QUESTIONS (AFTER 20 MINUTES):

- What is unix, difference between unix & linux.
- Explain different features of O.S.
- what is RTOS.

OUTCOME OF THE DELIVERED LECTURE: To be written after taking the lecture (Pl. write in bullet points about students' feedback on this lecture, level of understanding of this lecture by students etc.)

- Student should be learn about the unix, linux programming.
- Student should be learn about the mobile O.S. & RTOS.

REFERENCES: Text/Ref. Book with Page No. and relevant Internet Websites:



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

①

UN27-1

Operating System

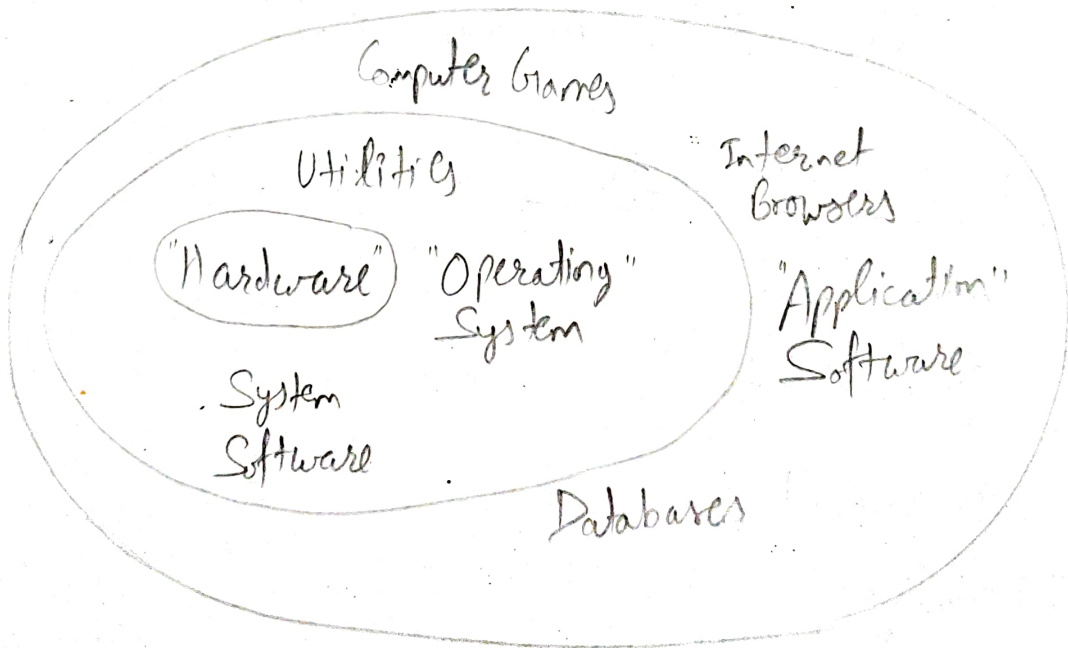
5CS4-03

Process Management

* Operating System Definition & Function:

In the comp. system hardware can only understand machine code (in the form of 0 and 1) which doesn't make any sense of naive user.

We need a system which can act as an intermediary and manage all the processes and resources present in the system.



An operating system can be defined as an interface b/w user and h/w. It is responsible for the execution of all the processes,

Dr. Mahesh Bundele
B.E., M.E., Ph.D.

Director
Poornima College of Engineering
ISO 9001:2015 Institutional Area
Gulapura, JAIPUR

CPU management, File management and many other tasks.

→ The purpose of operating system is to provide an environment in which a user can execute programs in convenient and efficient manner.

Sy Structure of a Computer System:

A comp sys. consist of:

- users (people who are using the computer)
- Application program (compilers, Databases, Games, Video player, browsers etc.)
- System programs (Shells, Editors, Compilers etc.)
- Operating System (A special prog. which act as an interface b/w user and h/w.)
- Hardware (CPU, disks, memory etc.)

What does an OS do?

- 1) Process management
- 2) Process Synchronization
- 3) Memory management
- 4) CPU Scheduling
- 5) File Management
- 6) Security



DETAILED LECTURE NOTES

②

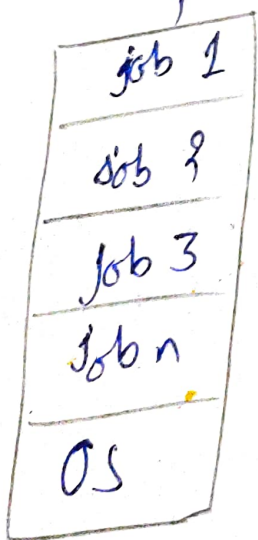
* Types of Operating System:

1) Batch OS:

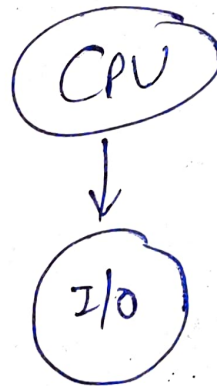
→ In this access is given to more than one person, they submit their respective jobs to the system for the execution.

→ The system put all the jobs in a queue on the basis of FCFS and then executes the job one by one. The users collect their respective output when all the jobs get executed.

Selection of job for the execution



job queue



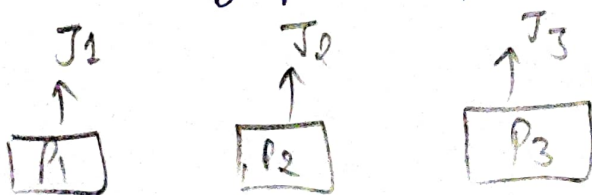
Disadvantages:

1) Starvation: If there are five jobs J_1, J_2, J_3, J_4 and J_5 present in the batch. If the execution time of J_1 is very high than other four jobs will never be going to get executed or they will have to wait for a very high time. Hence, the other processes get starved.

2) Not interactive: Batch processing is not suitable for the jobs which are dependent on the user's input.

3) Multiprogramming Operating System: Multiprogramming is an extension to the batch processing where the CPU is kept always busy. Each process needs two types of system time: CPU time and IO time. If a process does its IO, the CPU can start the execution of other processes. Therefore multiprogramming improves the efficiency of the system.

4) Multiprocessing OS: In this parallel computing is achieved. There are more than one processors present in the system which can execute more than one process at the same time. This will increase the throughput of the system.



(multiprocessing)



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES ③

4) Real Time OS: Each job carries a certain deadline within which the job is supposed to be completed, otherwise huge loss will be there or even if the result is produced then it will be completely useless.

* Process Management Introduction: There may exist more than one process in the system which may require the same resource at the same time. therefore, the operating system has to manage all the processes and the resources in a convenient & efficient way.

Some resources may need to be executed by one process at one time to maintain the consistency otherwise the system can become inconsistent and deadlock may occur.

The OS is responsible for the following activities in connections with process management.

- 1) Scheduling processes and threads on the CPUs.
- 2) Creating and deleting both user and system processes.
- 3) Suspending & resuming processes.

- 4) Providing mechanism for process synchronization.
- 5) Providing mechanisms for process communication.

* Attributes of a Process: The attributes of the process are used by the OS to create the process control block (PCB) for each of them. This is also called context of the process. Attributes are stored in PCB are described below:

- 1) Process ID: When a process is created, a unique id is assigned to the process which is used for unique identification of the process in the system.
- 2) Program Counter: A prog. counter stores the add. of the last instruction of the process on which the process was suspended. The CPU uses this add. when the execution of this process is resumed.
- 3) Process State: The process, from its creation to the completion, goes through various states which are new, ready, running and waiting.
- 4) Priority: Every process has its own priority. The process with the highest priority among the processes gets the CPU first. This is also stored on the process control block.
- 5) General Purpose Register: Every process has its own set of registers which are used to



POORNIMA

COLLEGE OF ENGINEERING

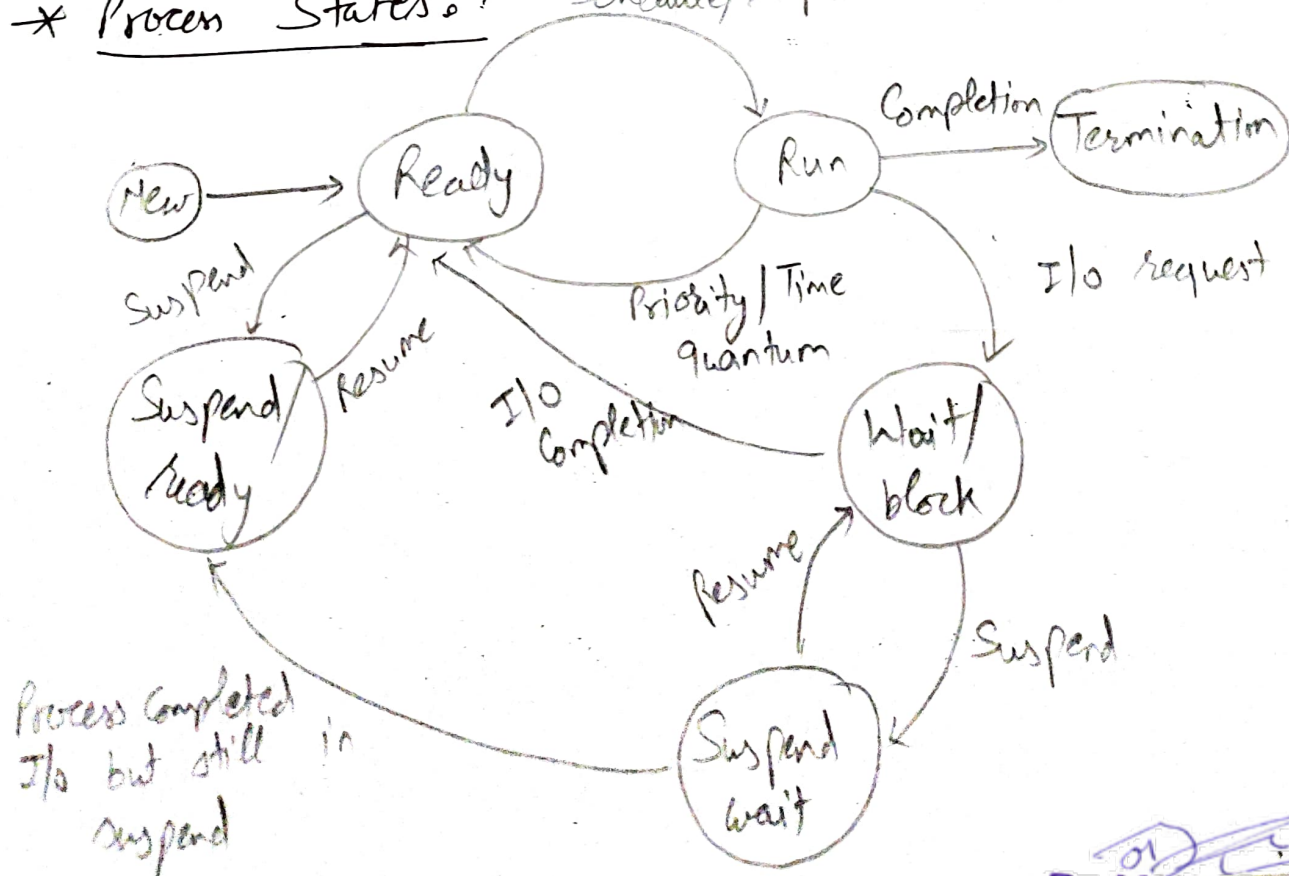
DETAILED LECTURE NOTES (4)

which is generated during the execution of the process.

6) List of open files: During the execution, every process uses some files which need to be present in the main memory. OS also maintains a list of open files in the PCB.

7) List of open devices: OS also maintain the list of all open devices which are used during the execution of the process.

* Process States: Schedule/Dispatch



1) New: A prog. which is going to be picked up by the OS into the main memory is called a new process.

2) Ready: Whenever a process is created, it directly enters in the ready state, in which, it waits for the CPU to be assigned.

The processes which are ready for the execution and reside in the main memory are called ready state processes. There can be many processes present in the ready state.

3) Running: One of the processes from the ready state will be chosen by the OS depending upon the scheduling algo. Hence, if we have only one CPU in our sys., the no. of running processes for a particular time will always be one. If we have n processors in the sys. then we can have n processes running simultaneously.

4) Block or Wait: From the running state, a process can make the transition to the block or wait state depending upon the scheduling algo or the intrinsic behaviour of the process.

When a process waits for a certain resource to be assigned or for the input



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES ⑤

then the OS move this process to the block or wait state and assign the CPU to the other processes.

5) Completion or termination : When a process finishes its execution, it comes in the termination state. All the context of the process (PCB) will also be deleted the process will be terminated by the OS.

6) Suspend Ready : A process in the ready state, which is moved to secondary memory from the main memory due to lack of the resources is called in the suspend ready state.

7) Suspend wait : Instead of removing the process from the ready queue, it's better to remove the blocked process which is waiting for some resources in the main memory. Hence processes complete their execution once the main memory get available and their wait is finished.

Dr. Mahesh Bunde
B.E., M.E., Ph.D.

Director

Poornima College of Engineering
131-0, RICO Institutional Area
Bhopal, Madhya Pradesh

Operations on the process:

- 1) Creation: Once the process is created, it will be ready and come into the ~~queue~~ ready queue and will be ready for execution.
- 2) Scheduling: Out of the many processes present in the ready queue, the OS chooses one process and start executing it. Selecting the process which is to be executed & next, is known as scheduling.
- 3) Execution: Once the process is scheduled for the execution, the processor starts executing it. Process may come to the blocked or wait state during the execution then in that case the processor starts executing the other processes.
- 4) Deletion / Killing: Once the ~~pro~~ purpose of the process get over then the OS will kill the process. The context of the process (PCB) will be deleted and the process gets terminated by the OS.

* Process Schedulers:

- 1) Long term scheduler: It is also known as job scheduler. It chooses the processes from the pool (secondary memory) and keeps ~~the~~ ^{the} the ready queue maintained in the



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES ⑥

2) Short Term Scheduler: It is also known as CPU scheduler. It selects one of the jobs from the ready queue and dispatch to the CPU for the execution.

3) Medium Term Scheduler: It takes care of the swapped out processes. If the running state processes needs some I/O time for the completion then there is a need to change its state from running to waiting.

* Process Queue: The OS manages various types of queues for each of the process states. If the process is moved from one state to another state then its PCB is also unlinked from the corresponding queue and added to the other state queue in which the transition is made.

1) Job Queue: In starting, all the processes get stored in the job queue. It is maintained in secondary memory. The long term scheduler picks

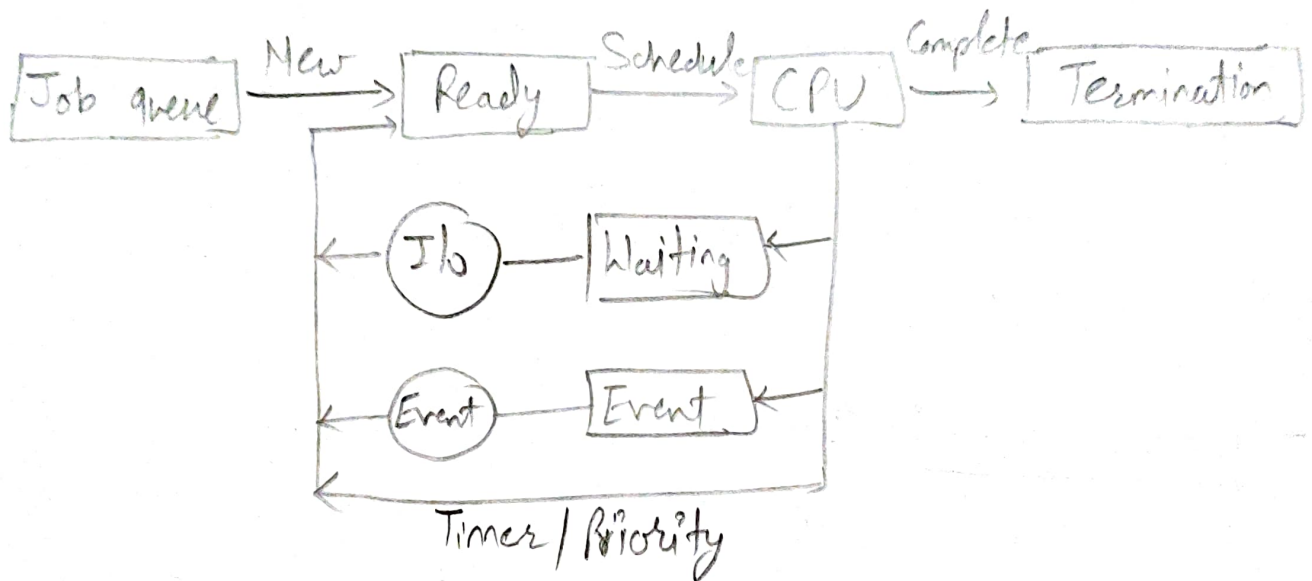
Dr. Mahesh Bunde
B.E., M.E., Ph.D.
Director

Poornima College of Engineering
1316, Rajco Institutional Area
Sitapura, JAIPUR

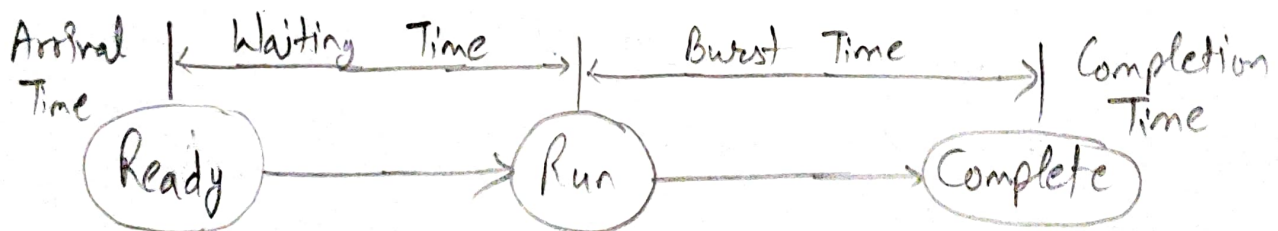
Some of the jobs and put them in the primary memory.

2) Ready Queue: Ready queue is maintained in primary queue. The short term scheduler picks the job from the ready queue and dispatch to the CPU for the execution.

3) Waiting Queue: When the process needs some IO operation in order to complete its execution, OS changes the state of the process from running to waiting.



* Times related to the process:



CT → Completion Time

AT → Arrival Time

TAT → Turn Around Time

WT → Waiting Time

BT → Burst Time



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

⑦

$$CT - AT = WT + BT$$

$$TAT = CT - AT$$

$$WT = TAT - BT$$

- 1) Arrival Time : The time at which the process enters into the ready queue is called the arrival time.
- 2) Burst Time : The total amount of time required by the CPU to execute the whole process is called the Burst time. It does not include the waiting time.
- 3) Completion Time : The time at which the process enters into the completion state or the time at which the process completes its execution, is called completion Time.
- 4) Turn around Time : The total amount of time spent by the process from its arrival to its completion is called Turn around Time.

Dr. Mahesh Bunde
B.E., M.E., Ph.D.
Director

Poornima College of Engineering
1310, Rajco Institutional Area
Sitapura, Jaipur

5) Waiting Time: The total amount of time for which the process waits for the CPU to be assigned is called waiting time.

6) Response Time: The difference b/w the arrival time and the time at which the process first gets the CPU is called response time.

* CPU Scheduling:

→ In the uniprogramming sys. like MS DOS, When a process waits for any I/O operation to be done, the CPU remains idle. This is an overhead since it wastes the time and causes the prob. of starvation.

→ In multiprogramming sys, the OS schedules the processes. CPU doesn't remain idle during the waiting time of the process and it starts executing other processes.

→ In multiprogramming sys, the OS schedules the processes on the CPU to have the max. utilization of it and this procedure is called CPU scheduling. The OS uses various scheduling algo. to schedule the processes.

→ This is a task of the short term scheduler to schedule the CPU for the no. of processes present in the job pool. Whenever the running process requests some I/O operations then the ~~short term~~ scheduler saves the current context of



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES ⑧

and changes its state from running to waiting. During the time, process is in waiting state; the short term scheduler picks another process from the ready queue and assign the CPU to this process. This procedure is called context switching.

Why do we need Scheduling?

→ In multiprogramming, if the long term scheduler picks more I/O bound processes then most of the time, the CPU remains idle. The task of OS is to optimize the utilization of resources.

→ If most of the running processes changes their state from running to waiting then there may always be a possibility of deadlock in the system. Hence to reduce this overhead, the OS needs to schedule the jobs to get the optimal utilization of CPU and to avoid the possibility of the deadlock.

Dr. Mahesh Bunde
B.E., M.E., Ph.D.

Director

Poornima College of Engineering
131-6, RIICO Institutional Area
Jaipur, JAIPUR

* Scheduling Algorithms:

There are various algo which are used by the OS to schedule the processes on the processor in an efficient way.

Purpose of a Scheduling Algorithm?

- Max. CPU scheduling
- Fair allocation of CPU
- Max. throughput
- Min turnaround time
- Min waiting time
- Min response time

There are following algo. which can be used to schedule the jobs.

- 1) First Come First Serve: It is the simplest algo. to implement. The process with the minimal arrival time will get the CPU first. The lesser the arrival time, the sooner will the process get the CPU. It is the non-preemptive type of scheduling.

Advantages

- Simple
- Easy
- FCFS



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

③

Disadvantages:

- 1) • the scheduling method is ^(without stop) non preemptive, the process will run to the completion.
- Due to non-preemptive nature of the algo, the prob of starvation may occur.
- It is easy to implement but it is poor in performance since the avg. waiting time is higher as compare to other scheduling algo.

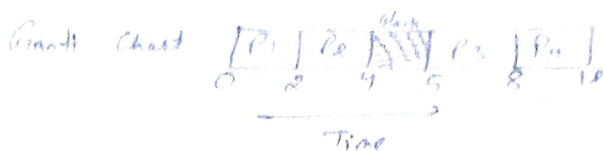
Example: there are 4 processes with the process ID P₁, P₂, P₃, P₄. arrival time & Burst time are given

Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P ₁	0	2	2	2	0
P ₂	1	2	4	3	1
P ₃	5	3	8	3	0
P ₄	6	4	12	6	2

Dr. Mahesh Bunde
B.E., M.E., Ph.D.

Director

Poornima College of Engineering
1316, Rajeev Institutional Area
Sitapura, Jaipur



Completion time
of a process
is the point
where it reaches
the right
side of the
process.

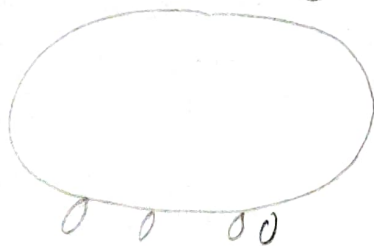
$$TAT = CT - AT$$

$$WT = TAT - BT$$

Convoy Effect in FCFS: FCFS may suffer from the Convoy effect (to travel with and protect) if the burst time of the first job is the highest among all. As in real life, if a convoy is passing through the road then the other persons may get blocked until it passes completely. This can be simulated in OS also.

→ If the CPU gets the processes of higher burst time at the front end of the ready queue then the processes of lower burst time may get blocked which means they may never get the CPU if the job in the execution has a very high burst time. This is called Convoy effect or starvation.

Convoy Effects or Starvation



Shorter jobs



Example: We have 3 processes named as P1, P2 and P3. The Burst time of process P1 is highest.

Dr. Mahesh Bunde
B.E., M.E., Ph.D.

Director

Poornima College of Engineering
131-0, RICO Institutional Area
Sitapura, JAIPUR



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES 10

i) In this process P1 arrives at the first in the queue although, the burst time of the process is the highest among all. Since we are following FCFS hence the CPU will execute the process P1 first.

→ In this schedule, the avg. waiting time of the system will be very high. This is because of the convey effect. The other processes P2, P3 have to wait for their turn for 40 units of time although their ~~burst~~ burst time is very low. This schedule suffers from starvation.

Process Id	Arrival time	Burst time	Completion Time	Turn Around Time	Waiting Time
1	0	40	40	40	0
2	1	3	43	42	39
3	1	1	44	43	42

Gantt Chart

P1	P2	P3
----	----	----

0 40 43 44

$$\text{Avg. waiting time} = 81/3$$

ii) In this, if process P1 would have arrived at the last of the queue and other processes P2 and P3 at earlier then the prob. of starvation would not be there

2) Shortest Job First: The job with the shortest burst time will get the CPU first. The longer the burst time, the sooner will the process get the CPU. It is the non-preemptive type of scheduling. However, it is very difficult to predict the burst time needed for a process hence this algorithm is very difficult to implement in the system.

Advantages?

- Max. throughput
- Min avg. waiting and turn around time

Disadvantages?

- may suffer with the prob. of starvation.
- It is not implementable bcoz the exact burst time for a process can't be known in advance.

<u>Q.</u>	Process ID	Arrival time	Burst time	Completion time	TAT	WT	RT
	P ₁	1	3	6	5	2	2
	P ₂	2	4	10	8	4	4
	P ₃	1	2	3	2	0	0
	P ₄	4	4	14	10	6	6

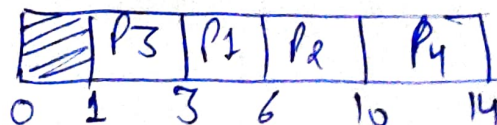
Criteria: "Burst time" | Response time: the time at which a process get the CPU first time. (diff. b/w arrival time & response time)

Mode: Non-preemptive (if one process get the CPU then it will execute complete without interruption. Avg TAT = $\frac{25}{4}$)

TAT: CT - AT

WT: TAT - BT

Gantt Chart



Avg WT = $\frac{12}{4}$

Time →
Arrive at P3 P2 P4



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

11

3) Shortest Remaining time first: It is the preemptive form of SJF. In this algo, the OS schedules the job according to the remaining time of the execution.
→ The execution of the process can be stopped after certain amount of time.

→ At the arrival of every process, the short term scheduler schedules the process with the least remaining burst time among the list of available processors and running process.

→ Once all the processes are available in the ready queue, No preemption will be done and the algo. will work as SJF scheduling. The context of the process is saved in the process control block. When the process is removed from the execution and the next process is scheduled. This PCB is accessed on the next execution of this process.

Dr. Mahesh Bunde
B.E., M.E., Ph.D.

Director

Poornima College of Engineering
131-0, RICO Institutional Area
Sikapur, Jaipur

Process ID	Arrival time	Burst time	Completion time	TAT	WT	AT
P1	0	8	3	3	4	0
P2	1	3	4	3	0	0
P3	2	4	13	11	7	7
P4	4	2	5	1	0	0

Criteria: Burst time

Mode: Preemptive

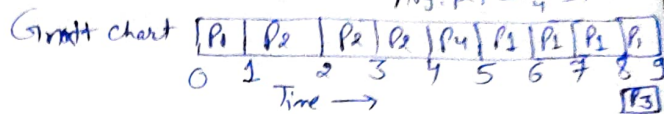
$$TAT = CT - AT$$

$$WT = TAT - BT$$

$$AT = \{CPU \text{ first time} - AT\}$$

$$Avg. TAT = \frac{24}{4} = 6, Avg. WT = \frac{11}{4} = 2.75$$

$$Avg. AT = \frac{7}{4} = 1.75$$



Ready queue: P1 P2 | P1 P2 P4
P1 P2 P3 | P1 P3
P1 P2 P3

→ if P1 is alone at 0 on arrival time then no need to check burst time.

→ P2 has less burst time 3 with P2 i.e. 4.

→ Now P2 has less burst time 1

→ When burst time same of two process then check arrival time.

→ Run P1 process together but we run single single for easyness.

→ Completion time check of right hand side of last execute process in gantt chart.

in preemptive case response time is most imp., check left hand side of first execute process in gantt chart.



4) Round
→ the OS processes

→ Each amount back

→ It

→ If

And

of

N

Dr. Mahesh Bunde
B.E., M.E., Ph.D.

Director

Poornima College of Engineering
131-0, RICO Institutional Area
Sitapura, JAIPUR



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

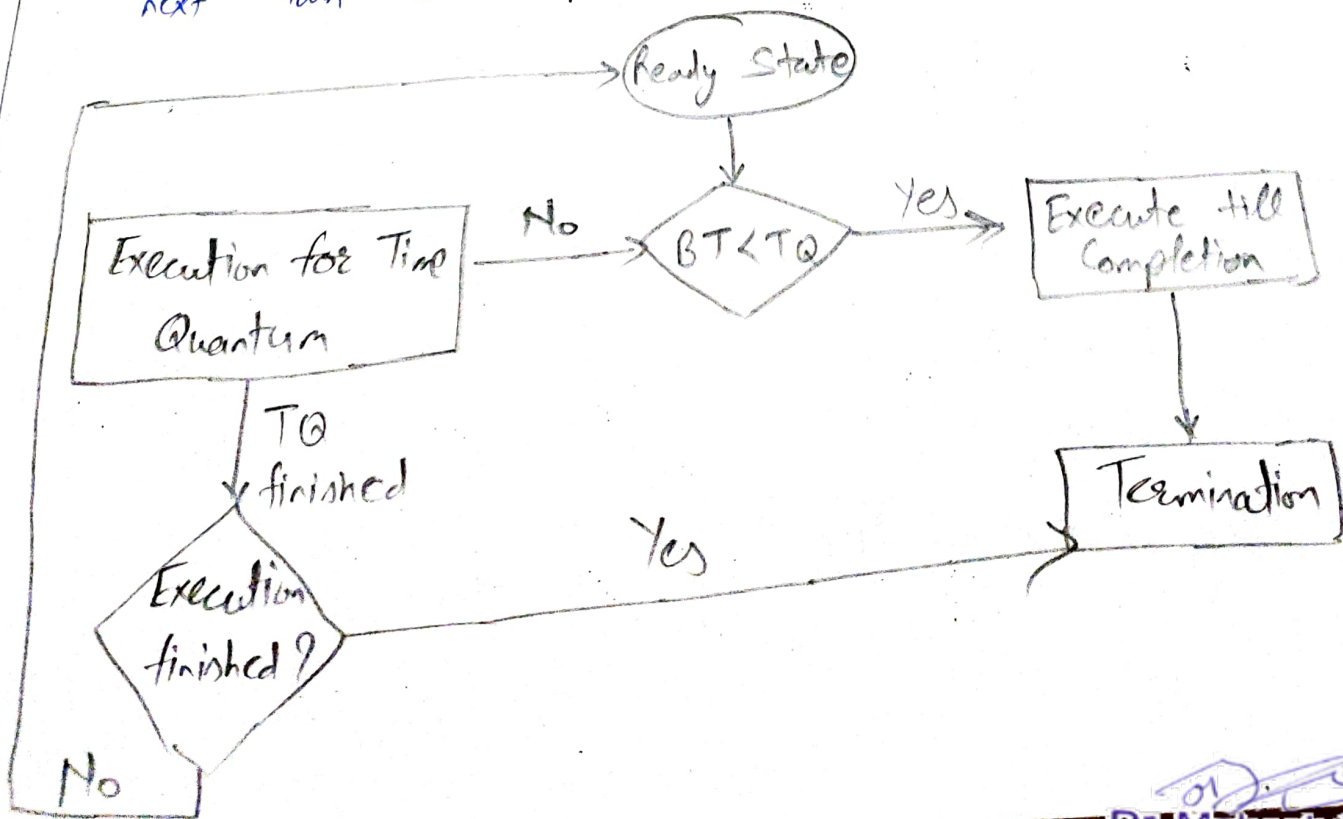
4) Round Robin Scheduling Algorithm?

→ the OS defines a time quantum (slice). All the processes will get executed in the cyclic way.

→ Each of the process will get the CPU for a small amount of time (called quantum time) and then get back to the ready queue to wait for its next turn.

→ It is a preemptive type of scheduling.

→ If the execution of the process is completed during that time then the process will terminate else the process will go back to the ready queue and waits for the next turn to complete the execution.



Dr. Mahesh Bunde
B.E., M.E., Ph.D.
Director

Poornima College of Engineering
1310, Rajeev Institutional Area
Sitapura, Jaipur

Advantages:

- It can be actually implementable in the sys. because it is not depending on the burst time.
- It does not suffer from starvation or convoy effect.
- all jobs get a fair allocation of CPU.

Disadvantages:

- Higher the time quantum, the higher the response time in the system.
- the lower the time quantum, the higher the context switching overhead in the sys.
- Deciding a perfect time quantum is really a very difficult task in the system.

<u>Eg.</u>	Process ID	Arrival time	Burst time	Completion time	TAT	WT	RT
P1		0	5	12	12	7	0
P2		1	4	11	10	6	1
P3		2	3	6	4	2	2
P4		4	1	9	5	4	4

Criteria: Time Quantum

Mode : Preemptive

TAT = CT - AT

WT = TAT - BT

RT: {CPU first time - AT}

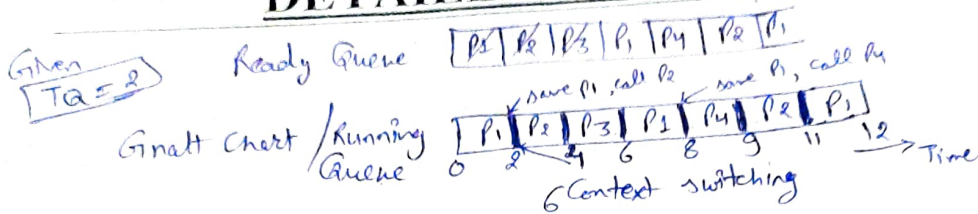


POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

(13)



- Process are in ready queue & select one process and give to the CPU (Running state).
- In Round Robin all the process run till time quantum, doesn't care about burst time.
- Imp. is "sequence of process in ready queue".
- check on 0 time in ready queue based on arrival time.
- if process is left acc. to burst time then put it into ready queue.

Dr. Mahesh Bunde
B.E., M.E., Ph.D.
Director

Poornima College of Engineering
1310, Rajco Institutional Area
Sikapur, Jaipur

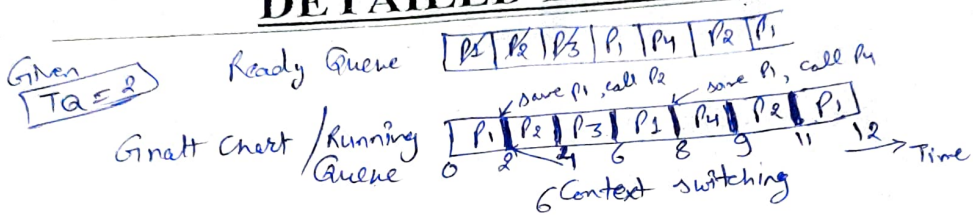


POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

(13)



- Process are in ready queue & select one process and give to the CPU (Running state).
- In Round Robin all the process run till time quantum, doesn't care about burst time.
- Imp. is "sequence of process in ready queue"
- check on 0 time in ready queue based on arrival time.
- if process is left acc. to burst time then put it into ready queue.

Dr. Mahesh Bunde
B.E., M.E., Ph.D.

Director

Poornima College of Engineering
131-0, RICO Institutional Area
Sikapura, JAIPUR

5) Highest Response Ratio Next (HRRN) Scheduling?

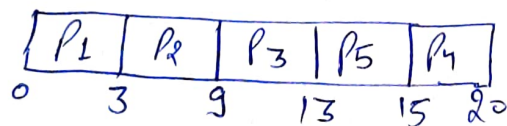
It is the most optimal scheduling algo. This is a non-preemptive algo in which scheduling is done on the basis of an extra parameter called Response Ratio.

A response ratio is calculated for each of the available jobs and the job with the highest response ratio is given priority over the others.

$$\text{Response Ratio} = \frac{W + S}{S} \quad \text{where } W : \text{Waiting time} \\ S : \text{Service time or Burst time}$$

Process No.	AT	BT	CT	TAT	WT
P1	0	3	3	3	0
P2	2	6	9	7	1
P3	4	4	13	9	5
P4	6	5	20	14	9
P5	8	2	15	7	5

Gantt Chart



→ At time 3 P2 process arrive.

→ At time 9 all the processes have arrived. So which process should be executed first.

$$RR_3 = \frac{(9-4) + 4}{4} = 2.25$$

$$RR_5 = \frac{(9-8) + 2}{2} = 1.5$$

$$RR_4 = \frac{(9-6) + 5}{5} = 1.6$$

$$WT = \text{Current time} - AT$$

Dr. Mahesh Bunde
B.E., M.E., Ph.D.

Director

Poornima College of Engineering
131-0, RICO Institutional Area
Sikapura, JAIPUR



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES (19)

→ P₃ or R₄ having highest response ratio. So it is executed first.

$$R_{P_4} = \frac{(13-6)+5}{5} = 2.4$$

$$R_{R_5} = \frac{(13-8)+2}{2} = 3.5$$

→ R₅ or P₅ having highest Response Ratio so it is executed first.

$$\text{Avg. WT.} = \frac{0+1+5+9+5}{5} = \frac{20}{5} = 4$$

6) Priority Scheduling: There is a priority no. assigned to each process. In some system, the lower the no., the higher the priority. While, in the others, the higher the number, the higher will be priority.

→ The process with the higher priority among the available processes is given the CPU.

→ There are two types of priority scheduling also exists.

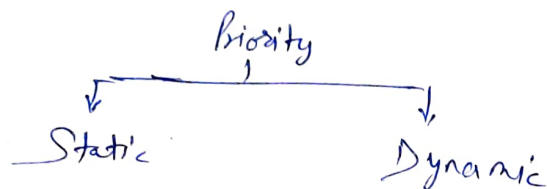
- i) Preemptive
- ii) Non-preemptive.

Priority Scheduling

↓
Preemptive

↓
Non-Preemptive

- the priority no. assigned to each of the process may or may not vary.
- if the priority no. doesn't change itself throughout the process, it is called Static priority. While if it keeps changing itself at the regular intervals, it is called dynamic priority.



Preemptive Priority Scheduling:

- At the time of arrival of a process in the ready queue, its priority is compared with the priority of the other processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time.
- The one with the highest priority among all the available processes will be given the CPU next.
- The difference b/w preemptive priority scheduling and non-preemptive priority scheduling is that, in the preemptive priority scheduling, the job which

Dr. Mahesh Bunde
B.E., M.E., Ph.D.
Director

Poornima College of Engineering
ISO-9001:2015 Institutional Area
Jaipur, RAIPUR



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES 15

executed can be stopped at the arrival of a higher priority job.

→ Once all the jobs get available in the ready queue, the algo. will behave as non-preemptive priority scheduling, which means the job scheduled will run till the completion and no preemption will be done.

Example:

Priority	Process No.	Arrival time	Burst time	Completion time	TAT	WT
10	P1	0	8	12	12	7
20	P2	1	4	8	7	3
30	P3	2	2	4	2	0
40	P4	4	1	5	1	0

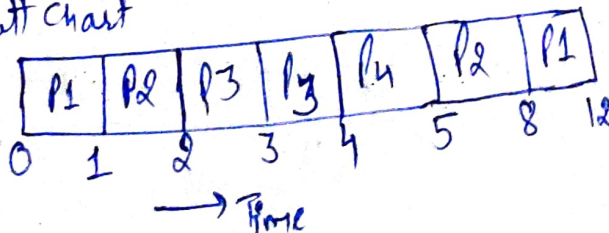
Criteria: Priority
Mode: Preemptive
 $TAT = CT - AT$
 $WT = TAT - BT$

$$\text{Avg. TAT} = \frac{28}{4} = 7$$

$$\text{Avg. WT} = \frac{10}{4} = 2.5$$

Gantt chart

Higher the no.
higher the priority



Dr. Mahesh Bunde
B.E., M.E., Ph.D.
Director

Poornima College of Engineering
1316, Rajco Institutional Area
Sitapura, Jaipur

- In preemptive scheduling, when a process is ready to execute, the scheduler checks if there is any process with higher priority than the process currently running. If yes, it preempt the process and execute the process with higher priority.

Non-Preemptive Priority Scheduling:

- The processes are scheduled acc. to the priority no. assigned to them.
- Once the process get scheduled, it will run till the completion.
- Generally, the lower the priority no, the higher is the priority of the process.

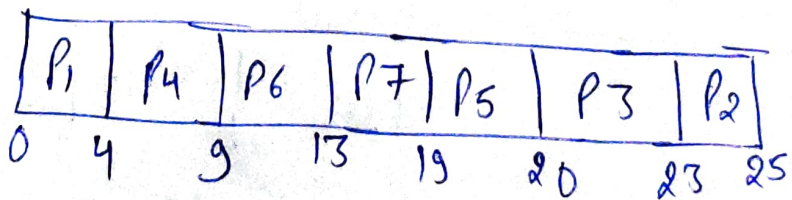
Example:

Priority	Process No.	AT	BT	CT	TAT	WT	RT
2 (L)	1	0	4	4	4	0	0
4	2	1	2	25	24	22	22
6	3	2	3	23	21	18	18
10	4	3	5	9	6	1	1
8	5	4	1	20	16	15	15
12 (H)	6	5	4	13	8	4	4
9	7	6	6	19	13	7	7

Criteria: Priority

Mode: Non-preemptive

Gantt Chart



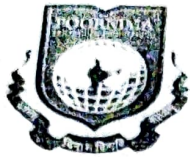
$$TAT = CT - AT$$

$$BWT = TAT - BT$$

$$RT = (CPU \text{ at first time} - AT)$$

Dr. Mahesh Bunde
B.E., M.E., Ph.D.
Director

Poonima College of Engineering
1316, Rajco Institutional Area
Sikar, RAJASTHAN



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

7) Shortest Remaining Time first (SRTF) Scheduling:- OR SJF with preemption

Criteria - Burst time
Mode - Preemptive

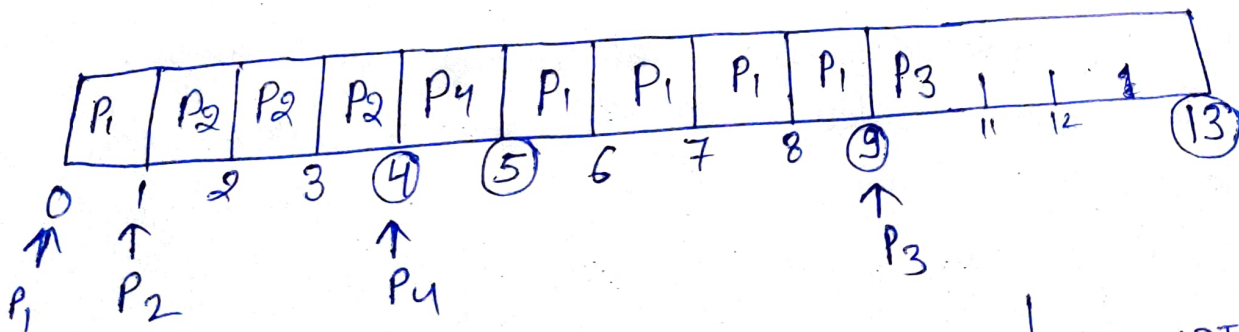
Process No.	Arrival time	Burst time	Completion Time	TAT	WT	RT
P ₁	0	5	9	9	4	0
P ₂	1	3	4	3	0	0
P ₃	2	4	13	11	7	7
P ₄	4	1	5	1	0	0

Gantt Chart :-

$$TAT = CT - AT$$

$$WT = TAT - BT$$

$$RT = (CPU \text{ first time} - AT)$$



$$Avg RT = \frac{7}{4} = 1.75 \quad \Bigg| \quad Avg TAT = \frac{24}{4} = 6$$

$$Avg WT = \frac{11}{4} = 2.75$$

* Inter Process Communication:

- It is provided by the operating system.
- Aim : provide the comm. b/w several processes. The intercommunication allows a process letting another process know that some event has occurred.

Role of Synchronization in Inter Process Communication:

It is one of the essential part of inter process comm. This is provided by interprocess comm. control mechanism, but sometimes it can also be controlled by comm. processes.

There are following methods that used to provide the synchronization.

1) Mutual Exclusion : It is generally required that only one process thread can enter the critical section at a time. This also helps in synchronization and creates a stable state to avoid the race condition.

2) Semaphore : It is a type of variable that usually controls the access to the shared resources by several processes.

types of semaphore:

- Binary Semaphore

- Counting Semaphore

3) Barrier : A barrier typically not allows an individual process to proceed unless all the processes does not reach it. It is used by many parallel ~~per~~ languages and collective routines imposes barriers.



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

(17)

4) Spinlocks: It is a type of lock as its name implies. The processor is trying to acquire the spinlock waits or stays in a loop while checking that the lock is available or not. It is known as busy waiting because even though the process is active, the process does not perform any functional operation (or task).

* Approaches to Interprocess Communication:

1) Pipe: The type is a type of data channel that is unidirectional in nature. It means that the data in this type of data channel can be moved in only a single direction at a time. Still, one can use two-channel of this type, so that can be send and receive data in two processes. It uses the standard methods for input and output. These pipes are used in all types of POSIX systems and in different versions of windows operating system as well.

2) Shared Memory:

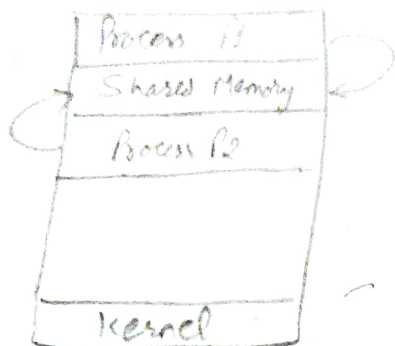
→ accessed by multiple processes simultaneously.

→ It is primarily used so that processes can comm. with each other.

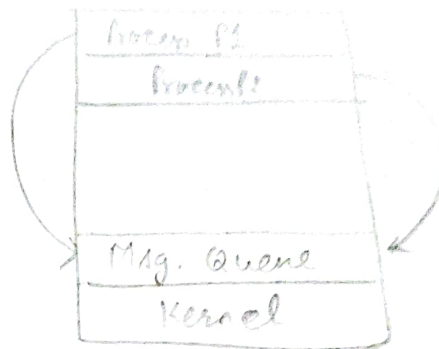
→ It used by almost all POSIX and similar operating system as well.

3) Message Queue:

- Several msg. are allowed to read and write the data to the msg. queue.
- the msg. are stored or stay in the queue until their recipients retrieve them.



Shared Memory



Msg. Queue

4) Message Passing:

- It allows the processes to synchronize and communicate with each other.
- the processes can comm. with each other without restoring the shared variables.
- Inter-process comm. mechanism provides two operations that are as follow:
 - Send msg.
 - Received msg.

Note: the size of the msg. can be fixed or variable.



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

2

5) Direct Communication

- A link is created or established b/w two communicating processes.
- In every pair of communicating processes, only one link can exist.

6) Indirect Communication

- It can only exist or be established when processes share a common mailbox, and each pair of these processes shares multiple comm. links.
- These shared links can be unidirectional or bi-directional.

7) FIFO

- Comm. b/w 2 unrelated processes.
- Considered as a full duplex, which means that one process can comm. with another process and vice-versa.

Other approaches:

- 1) Socket: It act as a type of endpoint for receiving or sending data in a n/w. on the same comp. or diff. comp.

2) File: A file is a type of data record or a document stored on the disk and can be acquired on demand by the file server.

3) Signal: they are the msgs of sys. that are sent by one process to another. Therefore, they are not used for sending data but for remote commands b/w multiple processes.

Usually, they are not used to send the data but to remote commands in b/w several processes.

Why we need Interprocess Communication?

Reasons to use interprocess comm. for sharing the data.

- helps to speedup modularity.
- Computational
- Privilege separation
- Convenience
- helps OS to comm. each other and synchronize their actions as well.



POORNIMA

COLLEGE OF ENGINEERING

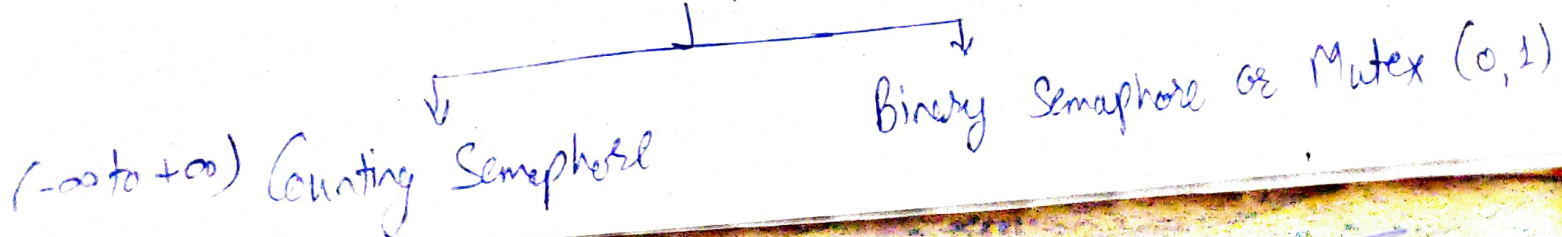
DETAILED LECTURE NOTES

(12)

* Semaphore :

- To get rid of the prob of wasting the wake-up signals, Dijkstra proposed an approach which involves storing all the wake-up calls.
- Dijkstra states that, instead of giving the wake-up calls directly to the consumer, producer can store the wake-up call in a variable. Any of consumers can read it whenever it needs to do so.
- Semaphore is the integer variables which store the entire wake up calls that are being transferred from producer to consumer.
- It is a variable on which read, modify and update happens automatically in kernel mode.
- Semaphore cannot be implemented in the user mode bcoz race condition may always arise when two or more processes try to access the variable simultaneously.

Semaphore



Dr. Mahesh Bunde
B.E., M.E., Ph.D.

Director

Poornima College of Engineering
131-0, RICO Institutional Area
Sitapura, JAIPUR

11 Counting Semaphore / Part 2 Critical Section

→ More than one processes need to execute in critical section simultaneously. However, counting semaphore can be used when we need to have more than one process in the critical section at the same time.

```
Down (Semaphore S)
{
    S value = S value - 1,
    if (S value < 0)
    {
        put process (PCB) in
        suspended list Sleep();
    }
    else
        return;
}
```

```
Up (Semaphore S)
{
    S value = S value + 1,
    if (S value ≤ 0)
    {
        Select a process
        from suspended list &
        wake up();
    }
}
```

→ In this mechanism, the entry and exist in the critical section are performed on the basis of the value of counting semaphore. The value of counting semaphore at any point of time indicates the max. no. of processes that can enter in the critical section at the same time.

→ A process which wants to enter in the critical section first decrease the semaphore value by 1

critical
can
process



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

(20)

+1,
And then check whether it gets negative or not. If it gets negative then the process is pushed in the list of blocked processes (i.e. 2) otherwise it gets enter in the critical section.

→ When a process exists from the critical section, it increases the counting semaphore by 1 and then checks whether it is negative or zero. If it is negative then that means that at least one process is waiting in the blocked state - hence, to ensure bounded waiting, the first process among the list of blocked processes will wake up and gets enter in the critical section.

→ The processes in the blocked list will get waked in the order in which they slept. If the value of counting semaphore is negative then it states the no. of processes in the blocked state while if it is positive it states the no. of slots available in the critical section.

Dr. Mahesh Bunde
B.E., M.E., Ph.D.
Director

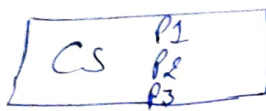
Poornima College of Engineering
1376, Rajco Institutional Area
Sitapura, Jaipur

Q. If we have processes P_1, P_2, P_3, P_4, P_5 they want to use critical section but first apply the entry code & after completion apply exit code.

Operation: $\left[\begin{array}{c} P() \text{ , Down , wait} \\ \downarrow \\ V() \text{ , up , Signal / Post / Release} \end{array} \right] \rightarrow \text{entry code}$
 these are synonyms.

Semaphore value $S = \text{let initialize.}$
 -4

P_1, P_2, P_3, P_4, P_5
 entry action



exit code

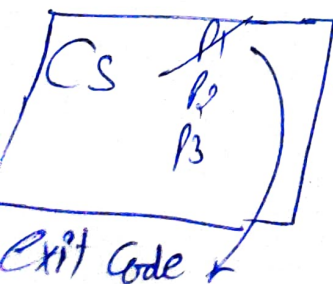
block list

P_4
 P_5

all process checked by down code that are used for entry section.

→ if Semaphore value $S = -4$ then we say that 4 processes are in the block state or in suspended list.

→ if Semaphore value $S = 0$ then no other process can enter in the critical section. this is the last value.



if P_1 want to say that I want to come out my work is complete then execute the exit code first. (up code)

Dr. Mahesh Bunde
 B.E., M.E., Ph.D.

Director
 Poornima College of Engineering
 131-0, RICO Institutional Area
 Sikapur, Jaipur



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES (21)

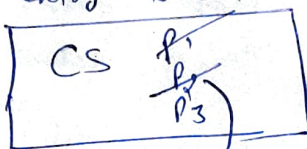
Current semaphore value $S = -2$ $\neq 0$

wake up mean those processes are in block state first put in the ready queue acc. to FIFO.

Block list

P5

P4, P1
entry section

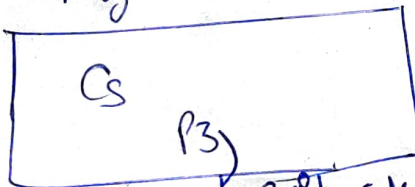


Exit code

Block list

P5, P4, P1

entry section



exit code

P4 want to come in CS then again suspend coz semaphore value S is -1 . So we work on P3 that is in CS. so semaphore value $S = -2$ $\neq 0$

Block list

P4

Q. If 6P, 2V, 4V semaphore value $[S=10]$
 what will be final value of semaphore?

$$[S=10] - 6 = 4 + 4 = (8) \checkmark$$

Q. $[S=17]$ 8 5P, 3V, 1P

$$S=17-5 = 12+3 = 15-2 = (13) \checkmark$$

*) Binary Semaphore/Mutex:

→ In Counting Semaphore, Mutual exclusion was not provided
 bcoz we have the set of processes which required to
 execute in the critical section simultaneously.

→ However, Binary Semaphore strictly provides mutual exclusion.
 Here, instead of having more than 1 slots available
 in the critical section, we can only have at most
 1 process in the critical section. The semaphore can
 have only two values 0 and 1.

operation: $\boxed{\text{Down, } P, \text{ wait}}$

$\boxed{\text{up, } V, \text{ signal}}$

→ let $[S=1]$ and check in down process

so if $S \neq 0$ we get then it is called successful
operation.



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES (93)

Down(Semaphore S)

```
{
  if (S value == 1)
  {
    S value = 0;
  }
  else
  {
    Block this process and
    place in Suspend list;
    sleep();
  }
}
```

Up(Semaphore S)

```
{
  if (suspend list is empty)
  {
    S value = 1;
  }
  else
  {
    Select a process from
    suspend list and
    wake up();
  }
}
```

But if $S=0$ then ~~process~~ check in down.
process will go into suspend list or unsuccessful
operation.

if $S=0$ then make the value of $S=1$.
if $S=1$ then make the value of $S=1$.

Dr. Mahesh Bundele
B.E., M.E., Ph.D.

Director

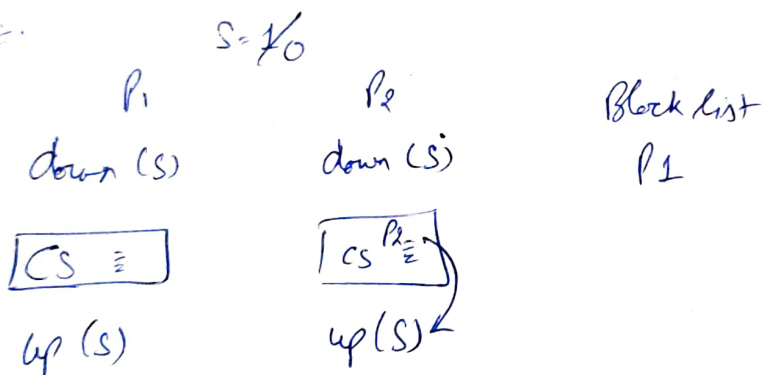
Poornima College of Engineering
131-0, RICO Institutional Area
Sitapura, JAIPUR

if $S \neq 0$ then P_1 is not critical section after that process comes in block list.

up now $S = 0$ make $S = 1$. if suspended list is empty. But if suspended list is not empty then select any one process from block list & wake up that process & that process come into ready queue.

eg But $S = 1$ check suspend list is empty or not. if process is in suspend list then wake up that process.

Eg.



Let say P_2 want to execute then come in CS & P_1 also want but go into block list

up P_2 want to execute up. then check suspend list i.e. not empty so wake up the P_1 process & put into ready queue.



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

(23)

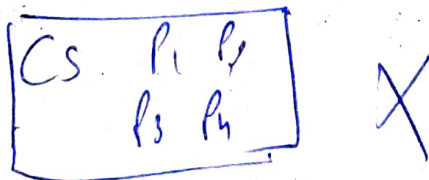
* Critical Section Problem

- Critical section is the part of prog. which tries to access shared resources. That resource may be any resource in a comp. like a memory location, Data Structure, CPU or any I/O device.
- The critical section cannot be executed by more than one process at the same time; OS faces the difficulties in allowing and disallowing the processes from entering the critical section.
- The critical section prob. is used to design a set of protocols which can ensure that the Race condition among the processes will never arise.

Requirements of Synchronization mechanisms:

- 1) Mutual Exclusion: If one process is executing inside critical section then other process must not enter in critical section.

Primary



2) Priority: If one process has more priority to execute into critical section then it should not stop other processes to get into the critical section. [Primary]

3) Bounded Waiting: We should be able to predict the waiting time for every process to get into the critical section. The process must not be endlessly waiting for getting into the critical section. [Secondary]

4) Architectural Neutrality: Our mechanism must be architectural natural. It means that if our solⁿ is working fine on one architecture then it should also run on the other ones as well. [Secondary]

Lock Variable: / Critical Section solution using Lock:

- This is simplest synchronization mechanism.
- In this, a lock variable lock is used.
- Two values of lock can be possible, either 0 or 1.
- Lock value 0 means, critical section is vacant.
- Lock value 1 means, critical section occupied or full.
- A process which wants to get into the critical section first checks the value of the lock variable. If it is 0 then it sets the value of lock as 1 and enters into the critical section, otherwise it waits.



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES (24)

do
{ acquire lock
critical section
release lock
}

- Execute in user mode
no need of OS or kernel.
- Multiprocess Exclusion
- No mutual Exclusion guarantee

pseudo code:

1. While (Lock == 1); or while (Lock != 0);
2. Lock = 1

Entry
Code

3. Critical section

4. Lock = 0

Exit Code

eg. Case 1

P₁

P₂

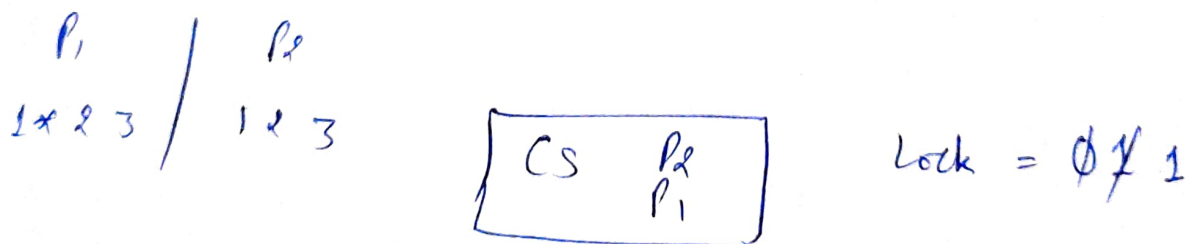
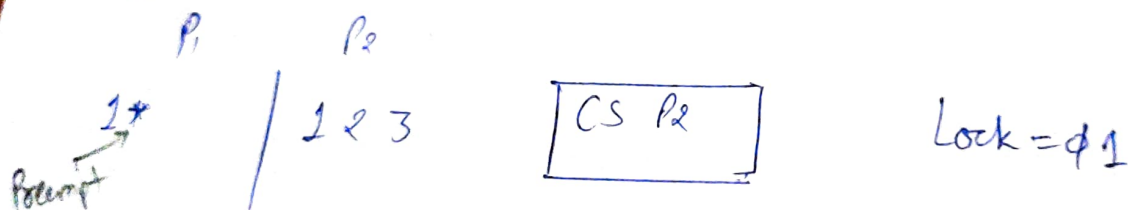
1 2 3 4 | 5

CS
P₁
P₂

Lock = 0 ≠ 1

- First P_1 wants to enter in critical section so check the condition of entry code, set the value of lock 0 to 1 & enter into critical section.
- Now P_2 wants to enter in CS so check the condition of entry code. Condition is false so P_2 is into infinite loop.
- Now P_1 wants to come out from CS, again set the value of lock 1 to 0 & execute the exit code now CS is vacant.
- At this point P_2 again wants to come in CS. this time condition is true, set the value of lock 0 to 1 and come into CS.

Case 2



- P_1 wants to enter in CS so check condition of entry code. after executing 1st line of entry code P_1 preempt due to some Emergency.



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

(25)

- In between P_2 process come P_1 wants to enter in CS so execute the entry code & set the value of lock 0 to 1. then come into CS.
- Now P_1 again come back & execute 2nd line of entry code, set the value of lock 1 to 1. (overright) & come into CS.

So in this case No mutual exclusion.

in preemption case Lock variable fail,

Critical Section Solution using 'Test_and_Set'
Instruction?

why

without waiting. Solution using "test-and-set" instruction:

while (test_and_set(&lock)); {
 [CS]
 }
 lock = false;

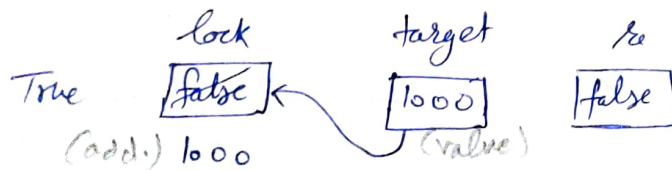
1. while (lock == 1)
2. lock = 1
3. CS
4. lock = 0

```
boolean test_and_set (boolean *target)
{
    boolean r = *target;
    *target = TRUE;
    return r;
}
```

address / pointer variable

definition of fun.

P₁ P₂
 1 2 3 1 2 3
 ↑
 preempt
 Lock = 0 ≠ 1
 initially Lock = 0



True = 1 , False = 0

Initially Lock = false

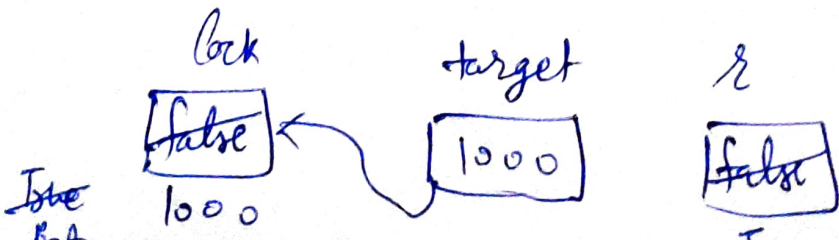
P₁ P₂

*target mean point out the lock so change the value of false to True. put the value of lock True in r, & changes its value false to true.

CS P₁

Now on P₂

CS P₂ not enter



so mutual exclusion

True
~~False~~

P₂ stuck in while enter in CS.



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES (96)

Turn Variable (Strict Alteration)

- 2 Process Solution
- Run in user mode

Mutual exclusion ✓
progress ✗
bounded waiting ✓

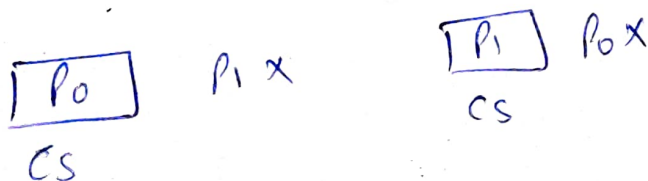
	Process "P ₀ "	Process "P ₁ "
Entry Code →	$\text{while (turn} \neq 0);$ <div>Critical section</div>	$\text{while (turn} \neq 1);$ <div>Critical Section</div>
Exit Code →	turn = 1;	turn = 0;

int turn = 0;

in process P₀ condition false so

CS P₀

process P₁ also want to come in CS but it stuck
in while condition so it can not come into CS.
Vice-versa.



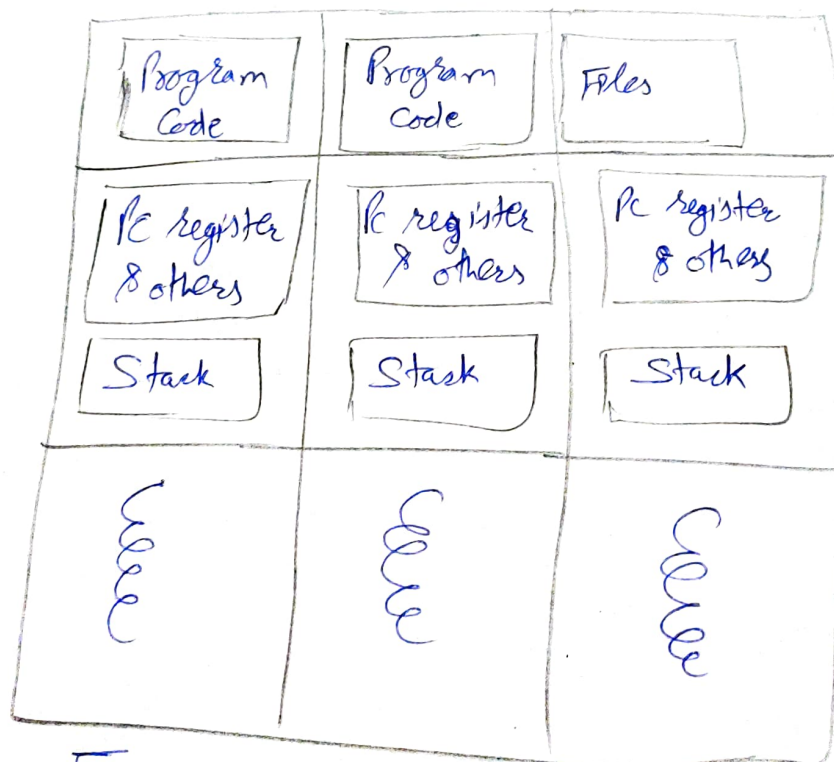
If turn = 0 then P₁ can't come in CS coz it stuck in while condition,

turn 0 → process P₀
turn 1 → process P₁ } execute acc to turn value

BW Alternate turn of every process like P₀ then P₁ or P₁ then P₀

* Thread S

- A thread is a single sequential flow of execution of tasks of a process i.e. it is also known as thread of execution or thread of control.
- A thread can execute inside the process of any O.S.
- There can be more than one thread inside a process.
- Each thread of the same process makes use of a separate prog. counter and a stack of activation records and control blocks.
- thread is often referred to as a lightweight process.



[Three Threads of Same Process]

→ The process can be split down into so many threads

Dr. Mahesh Bunde
B.E., M.E., Ph.D.

Director

Poornima College of Engineering
131-6, RICO Institutional Area
Sikar, RAJASTHAN



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES (24)

Need of Thread?

- Context switching is faster when working with threads.
- It takes less time to terminate a thread than a process.
- It takes less time to create a new thread in an existing process than to create a new process.
- Threads can share the common data, they do not need to use inter-process communication.

Types of Thread?

1) User Level Thread?

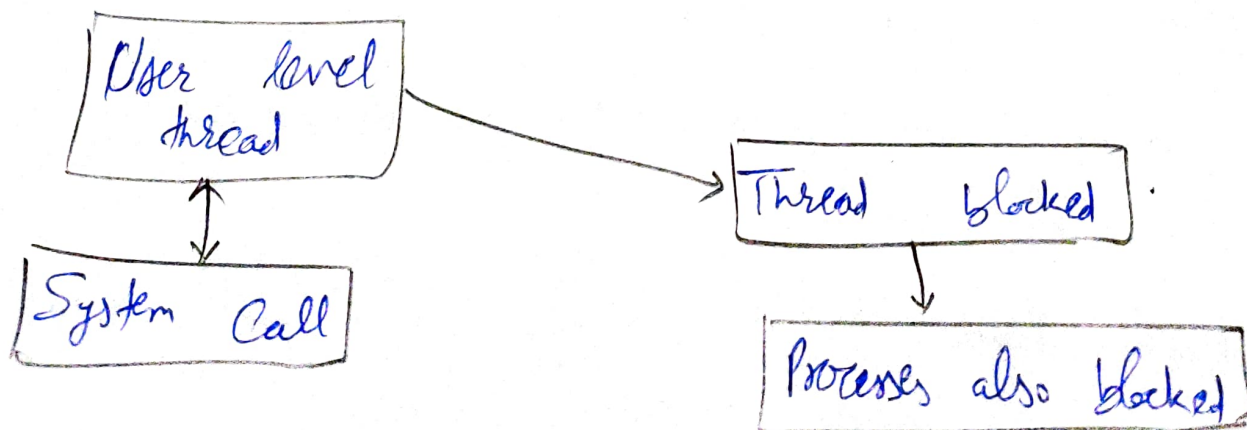
- The OS does not recognize the user-level thread.
 - User thread can be easily implemented by the user.
 - If a user performing a user-level thread blocking operation, the whole process is blocked.
- The kernel level thread does not know anything about user level thread.
- The kernel level thread manages user level threads as if they are single threaded process.

Advantages:

- User threads can be easily implemented than the kernel thread.
- User level threads can be applied to most types of OS that do not support threads at kernel level.
- It is faster & efficient.
- Context switch time is shorter than the kernel level thread.
- It does not require modification of OS.
- Representation of user level threads is very simple.
- It is simple to create, switch & synchronize threads without the intervention of the process.

Disadvantages:

- User level thread lack coordination b/w the thread & kernel.
- If a thread causes a page fault, the entire process is blocked.





POORNIMA

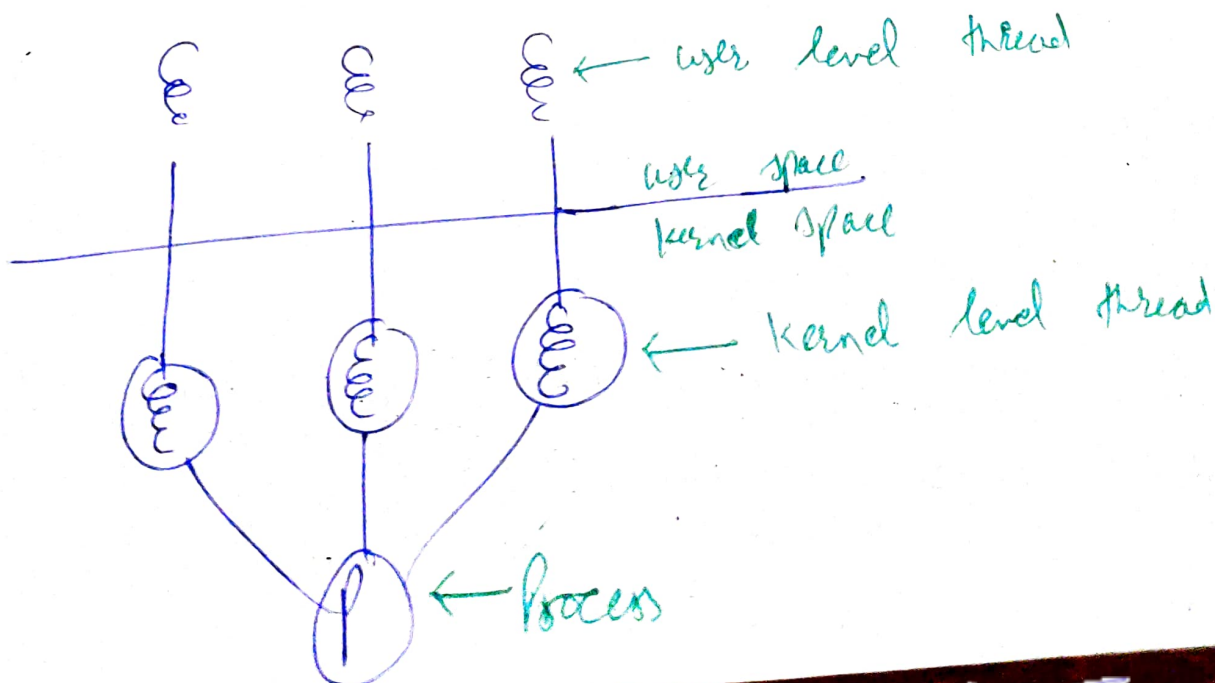
COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

(21)

Q. Kernel Level Thread:

- Recognizes the OS.
- There is a thread control block and process control block in the sys. for each thread and process in the kernel level thread.
- Implemented by OS.
- Kernel knows about all the threads and manages them.
- Kernel level thread offers a system call to create and manage the threads from user-space.
- The implementation of kernel thread is more difficult than the user thread.
- Context switch time is longer in the kernel thread.



Advantages:

- It is fully aware of all threads.
- The scheduler may decide to spend more CPU time in the process of threads being large numerical.
- The kernel level thread is good for those app. that block the frequency.

Disadvantages:

- manages & schedules all threads
- Implementation of kernel threads is difficult than the user thread.
- kernel-level thread is slower than user-level threads

Components of Threads:

- 1) Program Counter
- 2) Register set
- 3) Stack Space

Benefits of Threads:

- Enhanced throughput of the system: When the process is split into many threads, and each thread is treated as a job, the no. of jobs done in the unit time increase. That's why throughput of the system also increase.



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

(25)

• Effective utilization of Multiprocessor System: When you have more than one thread in one process, you can schedule more than one thread in more than one processor.

• Faster Context Switch: The context switching period b/w threads is less than the process context switching. The process context switch means more overhead for the CPU.

• Responsiveness: When the process is split into several threads and when a thread completes its execution, that process can be responded to as soon as possible.

• Communication: Multiple thread commⁿ is simple because the threads share the same add. space, while in ~~process~~, ~~process~~,

• Resource Sharing: Resources can be shared b/w all threads within a process such as code, data and files.

NOTES The stack & register cannot be shared b/w threads. There is a stack and register for each thread.

* Difference b/w Process and Thread:

Process

1) Process is heavy weight & resource intensive.

2) Process switching needs interaction with O.S.

3) In multiple processing environments each process executes the same code but has its own memory and file resources.

4) If one process is blocked, then no other process can execute until the first process is unblocked.

5) Multiple processes without using threads use more resources.

6) In multiple processes each process operates independently of the others.

Thread

Thread is light weight, taking fewer resources than a process.

Thread switching does not need to interact with O.S.

All threads can share same set of open files, child processes.

While one thread is blocked and waiting, a second thread in the same task can run.

Multiple threaded processes use fewer resources.

One thread can read, write or change another thread's data.

Dr. Mahesh Bunde
B.E., M.E., Ph.D.

Director

Poornima College of Engineering
13170, RICO Institutional Area
Bhopal, Madhya Pradesh



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

* Difference b/w User Level & Kernel Level Thread? 30

User Level Thread

Kernel Level Thread

- 1) User level threads are faster to create and manage. Kernel level threads are slower to create and manage.
- 2) Implementation is by a thread library at the user level. OS supports creation of kernel threads.
- 3) User level thread is generic and can run on any operating system. Kernel level thread is specific to the OS.
- 4) Multi-threaded applications can't take advantage of multiprocessing. Kernel routines themselves can be multithreaded.

Multithreading

→ Multithreading allows multiple processes (threads) to run concurrently or parallelly.

OR

→ When multiple threads run concurrently it is known as multithreading.

→ To understand the concept of multithreading, you must understand what is thread and a process.

→ A process is a program in execution. A process can further be divided into sub processes known as threads.

→ Eg: Playing a video and downloading it at the same time.

→ We have two types of thread i.e. user level thread and kernel level thread. So for these threads to run together there must exist a relationship b/w them. The relationship is established by using multithreading models.

→ There are three common ways of establishing this relationship.

- 1) Many - to - One Model
- 2) One - to - One Model
- 3) Many - to - many Model.


Dr. Mahesh Bunde
B.E., M.E., Ph.D.
Director

Poonima College of Engineering
1316, Rajco Institutional Area
Sitapura, Jaipur



POORNIMA

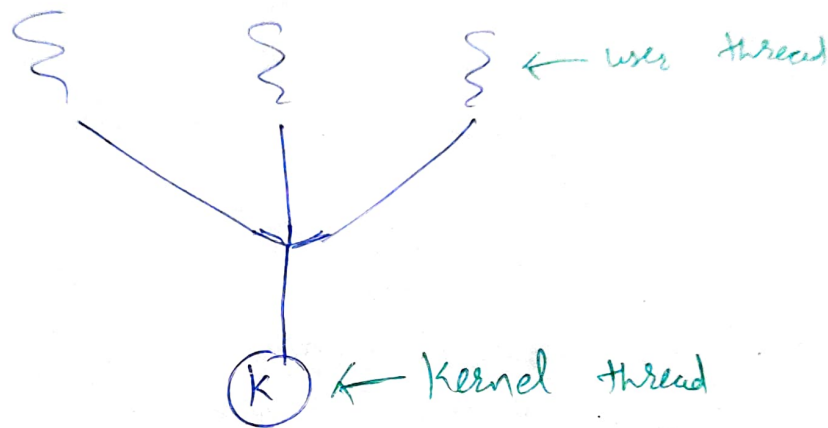
COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

PAGE NO. (31)

Many - to - One Model:

- multiple user threads are associated or mapped with one kernel thread.
- The thread management is done on the user level so it is more efficient.

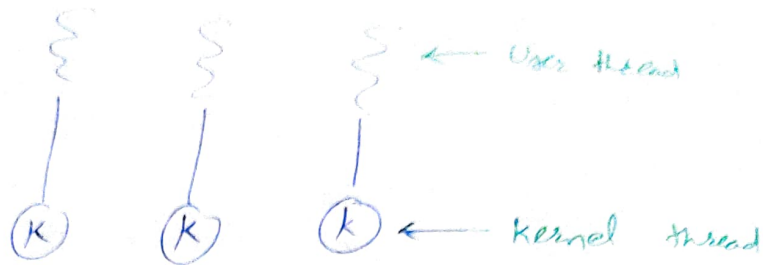


Drawbacks:

- One thread can access the kernel thread at a time so multiple threads are unable to run in parallel in the multiprocessor system. Even though we have multiple processors one kernel thread will run on only one processor. Hence, the user thread will also run in that processor only in which the mapped kernel thread is running.

One to One Model:

→ One user thread is mapped to one kernel thread.



Advantage:

- In this model, the first drawback of many-to-one model is solved. As each user thread is mapped to diff. kernel threads so even if any user thread makes a blocking system call, the other user threads won't be blocked.
- the second drawback is also overcome. It allows the threads to run parallel on a multiprocessor. Since it has each kernel threads mapped to one user thread. So each kernel thread can run on diff processors. Hence, each user thread can run on one of the processors.

Drawback of Many-to-One:

- As multiple user threads are mapped to one kernel thread. So, if one user thread makes a blocking sys call, it will block the kernel thread which will in turn block all the other threads.

Dr. Mahesh Bundele
B.E., M.E., Ph.D.

Director
Poonima College of Engineering
1316, Rajco Institutional Area
Sitapura, Jaipur



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES

PAGE NO. 29

Disadvantages:

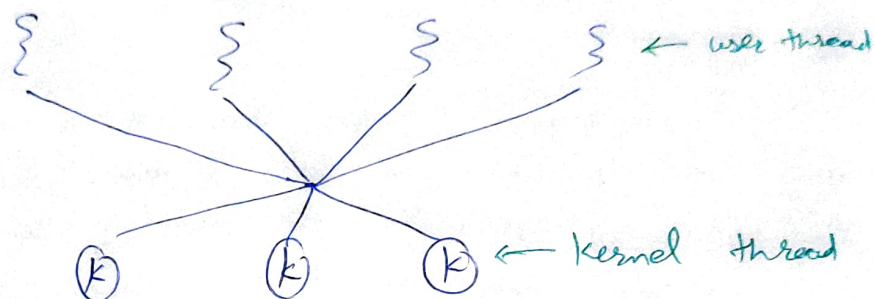
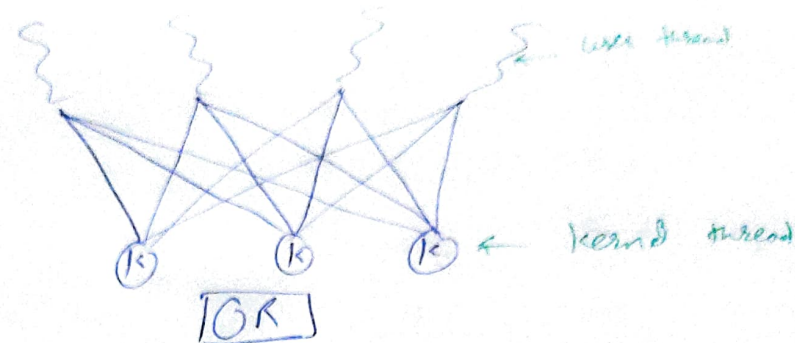
- Each time we create a user thread we have to create a kernel thread. So, the overhead of creating a kernel thread can affect the performance of the app.
- In a multiprocessor system, there is a limit of how many threads can run at a time. Suppose if there are four - processors in the system then only max. four threads can run at the time. So, if you are having 5 threads and trying to run them at a time then it may not work. Therefore, the app. should restrict the no. of kernel threads that it supports.

Many-to-Many Model:

- Many user threads to mapped to a smaller or equal no. of kernel threads.
- The no. of kernel threads is specific to a particular app. or m/c.

Dr. Mahesh Bunde
B.E., M.E., Ph.D.
Director

Poornima College of Engineering
1310, P.O. Institutional Area
Jaipur, RAJASTHAN



Advantages:

- When a user thread makes a blocking system call other threads are not blocked.
- There can be as many user threads as necessary. Also, the threads can run parallel on multiple processors.
- Here, we don't have need to create as many kernel threads as the user thread. So, there is no prob. with any extra overhead which was caused due to creating kernel thread.



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES 33

- The no. of kernel threads supported here is specific to the app. or m/c.

So, this is the best model that we can have in a multithreading sys. to establish the relationship b/w user-thread and kernel thread.

Benefits of Multithreading:-

- 1) Resource Sharing: As the threads can share the memory and resources of any process it allows any app. to perform multiple activities inside the same add. space.
- 2) Utilization of Multiprocessor Architecture: The diff. threads can run parallel on the multiple processors. Hence, this enables the utilization of the processor to a large extent and efficiency.
- 3) Reduced Context Switching Time: The threads minimized the context switching time as in the context switching, the virtual memory space remains

2) Efficient: The allocation of memory and resources during process creation comes with a cost. As the threads can distribute resources of the process it is more economical to create context switch threads.

Thread Libraries:

- Thread libraries provide programmers with API for the creation and management of threads.
- Thread libraries may be implemented either in user space or in kernel space, with ~~no kernel support~~.
- The user space involves API funⁿ implemented solely within the user space, with no kernel support.
- The kernel space involves system calls and requires a kernel with thread library support.

Three types of Thread:

- 1) POSIX Pthreads: may be provides as either a user or kernel library, as an extension to the POSIX standard.
- 2) Win32 threads: are provided as a kernel-level library on Windows systems.
- 3) Java threads: Since java generally runs on a Java virtual Mlc, the implementation of threads is based upon whatever OS and hardware the JVM is



POORNIMA

COLLEGE OF ENGINEERING

DETAILED LECTURE NOTES (34)

running on i.e. either Pthreads or Win32 threads depending on the system.

Multithreading Issues:

1) Thread Cancellation: Thread Cancellation means terminating a thread before it has finished working. There can be two approaches for this; one is Asynchronous Cancellation, which terminates the target thread immediately. The other is, Deferred cancellation allows the target thread to periodically check if it should be cancelled.

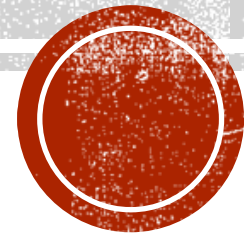
2) Signal Handling: Signals are used in UNIX Systems to notify a process that a particular event has occurred. Now when a multithreaded process receives a signal, to which thread it must be delivered? It can be delivered to all or a single thread.

3) fork() System call: It executed in kernel space through which a process create a copy of itself. Now the prob. in the multithreaded process is, if one thread forks, will the entire process be copied or not?

4) Security Issues: Yes, there can be security issues because of the extensive sharing of resources b/w multiple threads.

UNIT- IV

FILE MANAGEMENT



CONTENTS:

- File concept, File Attributes
- File types and structures,
- Directory structure,
- File Access methods and matrices,
- File security,
- User authentication
- Cases studies

FILE CONCEPT

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.
- File is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

FILE ATTRIBUTES

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring
- Information about files are kept in the directory structure, which is maintained on the disk
- Many variations, including extended file attributes such as file checksum
- Information kept in the directory structure

FILE OPERATIONS

- File is an **abstract data type**, To define a file properly, we need to consider the operations that can be performed on files.

Six basic operations:

1. **Create**
 2. **Write** – at **write pointer** location
 3. **Read** – at **read pointer** location
 4. **Reposition within file - seek**
 5. **Delete**: delete a file
 6. **Truncate**: erase the contents of a file but keep its attributes
- **Open(F_i)** – search the directory structure on disk for entry F_i , and move the content of entry to memory
 - **Close (F_i)** – move the content of entry F_i in memory to directory structure on disk

OPEN FILES

- Several pieces of data are needed to manage open files:
 - **Open-file table**: tracks open files
 - File pointer: pointer to last read/write location, per process that has the file open
 - **File-open count**: counter of number of times a file is open – to allow removal of data from open-file table when last processes closes it
 - Disk location of the file: cache of data access information
 - Access rights: per-process access mode information

OPEN FILE LOCKING

- **File locks allow one process to lock a file and prevent other processes from gaining access to it**
- File locks are useful for files that are shared by several processes—for example, a system log file that can be accessed and modified by a number of processes in the system.
- Provided by some operating systems and file systems
 - Similar to reader-writer locks
 - **Shared lock** similar to reader lock – several processes can acquire concurrently
 - **Exclusive lock** similar to writer lock
- Mediates access to a file
- Mandatory or advisory:
 - **Mandatory** – If a lock is mandatory, then once a process acquires an exclusive lock, the operating system will prevent any other process from accessing the locked file
 - **Advisory** – processes can find status of locks and decide what to do

FILE TYPES

- A common technique for implementing file types is to include the type as part of the file name.
- The name is split into two parts—
 - name and
 - Extension
- This two parts usually separated by a period
- The system uses the extension to indicate the type of the file and the type of operations that can be done on that file.

FILE TYPES — NAME, EXTENSION

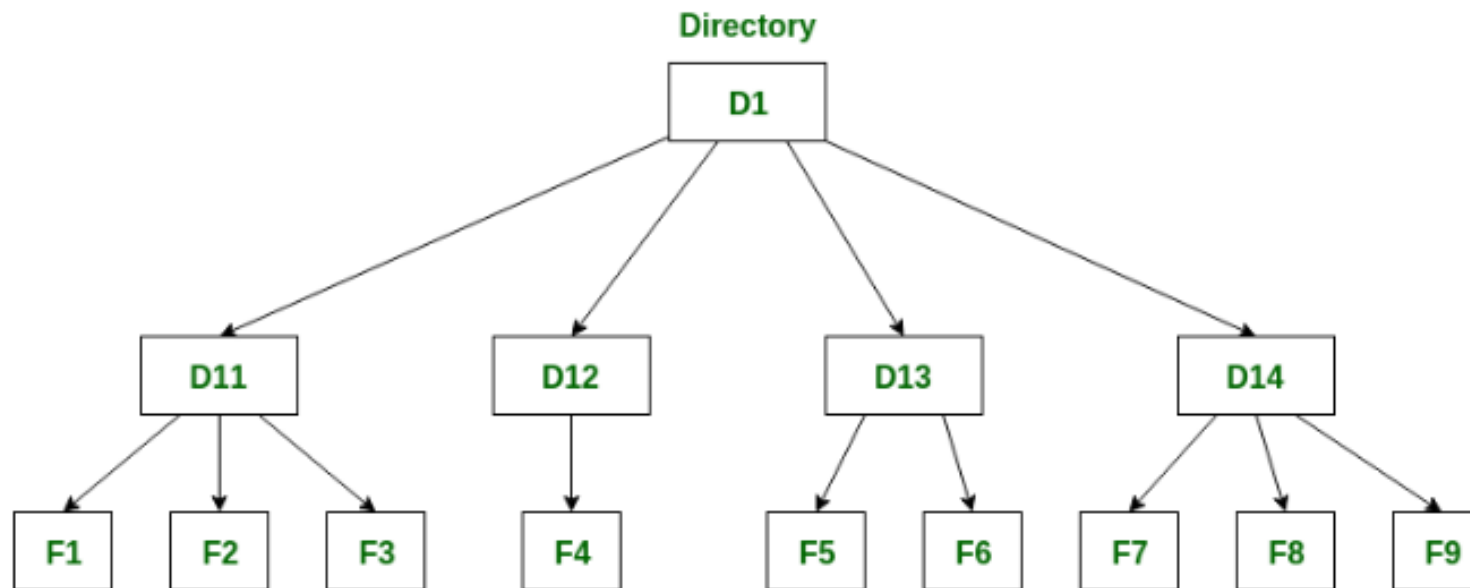
file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

FILE STRUCTURE

- File types also can be used to indicate the internal structure of the file
- A File Structure should be according to a required format that the operating system can understand.
- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

DIRECTORY STRUCTURE

- A **directory** is a container that is used to contain folders and file. It organizes files and folders into a hierarchical manner.
- The directory can be viewed as a symbol table that translates file names into their directory entries



OPERATIONS PERFORMED ON DIRECTORY

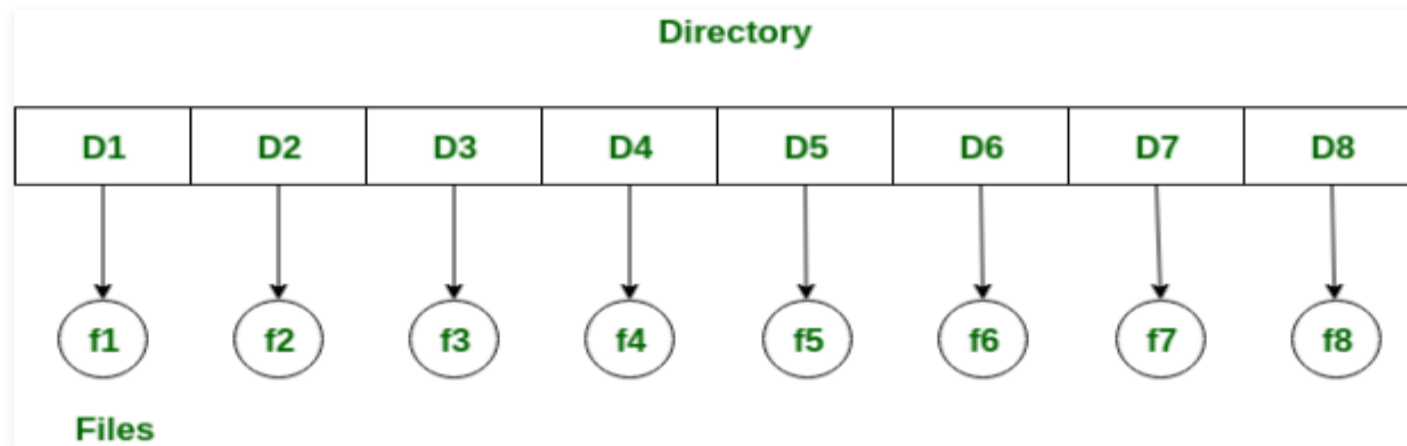
- **Search for a file.**
- **Create a file**
- **Delete a file**
- **List a directory**
- **Rename a file**
- **Traverse the file system**

DIRECTORY STRUCTURE TYPES

- There are several logical structures of a directory, these are given below.
- 1. **Single-level directory**
- 2. **Two-level directory**
- 3. **Tree-structured directory**
- 4. **Acyclic graph directory**
- 5. **General graph directory structure**

1. SINGLE-LEVEL DIRECTORY

- Single level directory is simplest directory structure.
- In it all files are contained in same directory which make it easy to support and understand.
- A single level directory has a significant limitation, however, when the number of files increases or when the system has more than one user.
- Since all the files are in the same directory, they must have the unique name .
- if two users call their dataset test, then the unique name rule violated.



SINGLE-LEVEL DIRECTORY

- **Advantages:**

- Since it is a single directory, so its implementation is very easy.
- If the files are smaller in size, searching will become faster.
- The operations like file creation, searching, deletion, updating are very easy in such a directory structure.

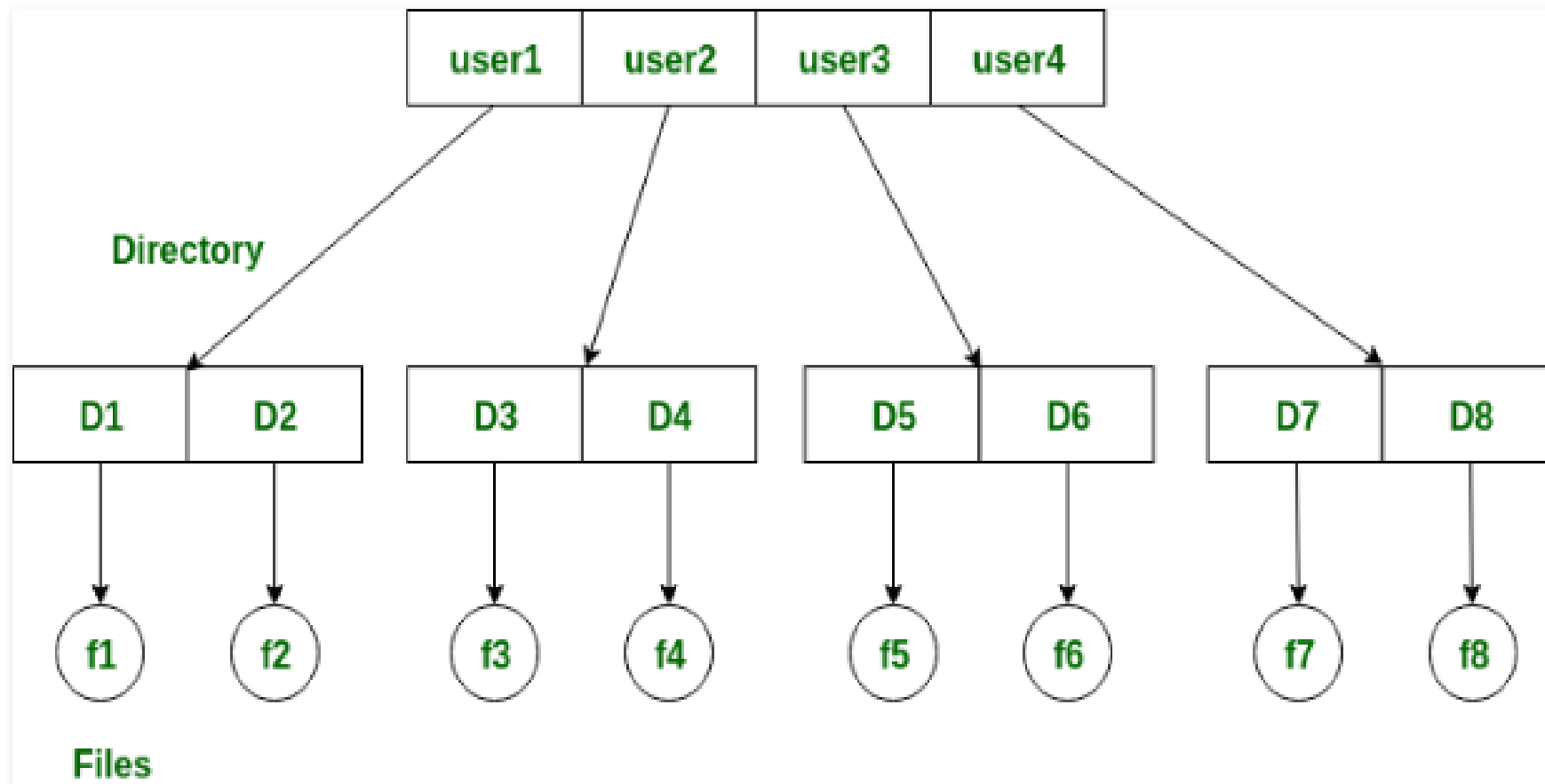
- **Disadvantages:**

- There may chance of name collision because two files can not have the same name.
- Searching will become time taking if the directory is large.
- In this can not group the same type of files together.

2. TWO-LEVEL DIRECTORY

- As we have seen, a single level directory often leads to confusion of files names among different users.
- the solution to this problem is to create a separate directory for each user.
- In the two-level directory structure, each user has there own *user files directory* (UFD).
- The UFDs has similar structures, but each lists only the files of a single user.
- System's *master file directory* (MFD) is searches whenever a new user ids logged in.
- The MFD is indexed by username or account number, and each entry points to the UFD for that user.

TWO-LEVEL DIRECTORY



TWO-LEVEL DIRECTORY

- **Advantages:**

- We can give full path like /User-name/directory-name/.
- Different users can have same directory as well as file name.
- Searching of files become more easy due to path name and user-grouping.

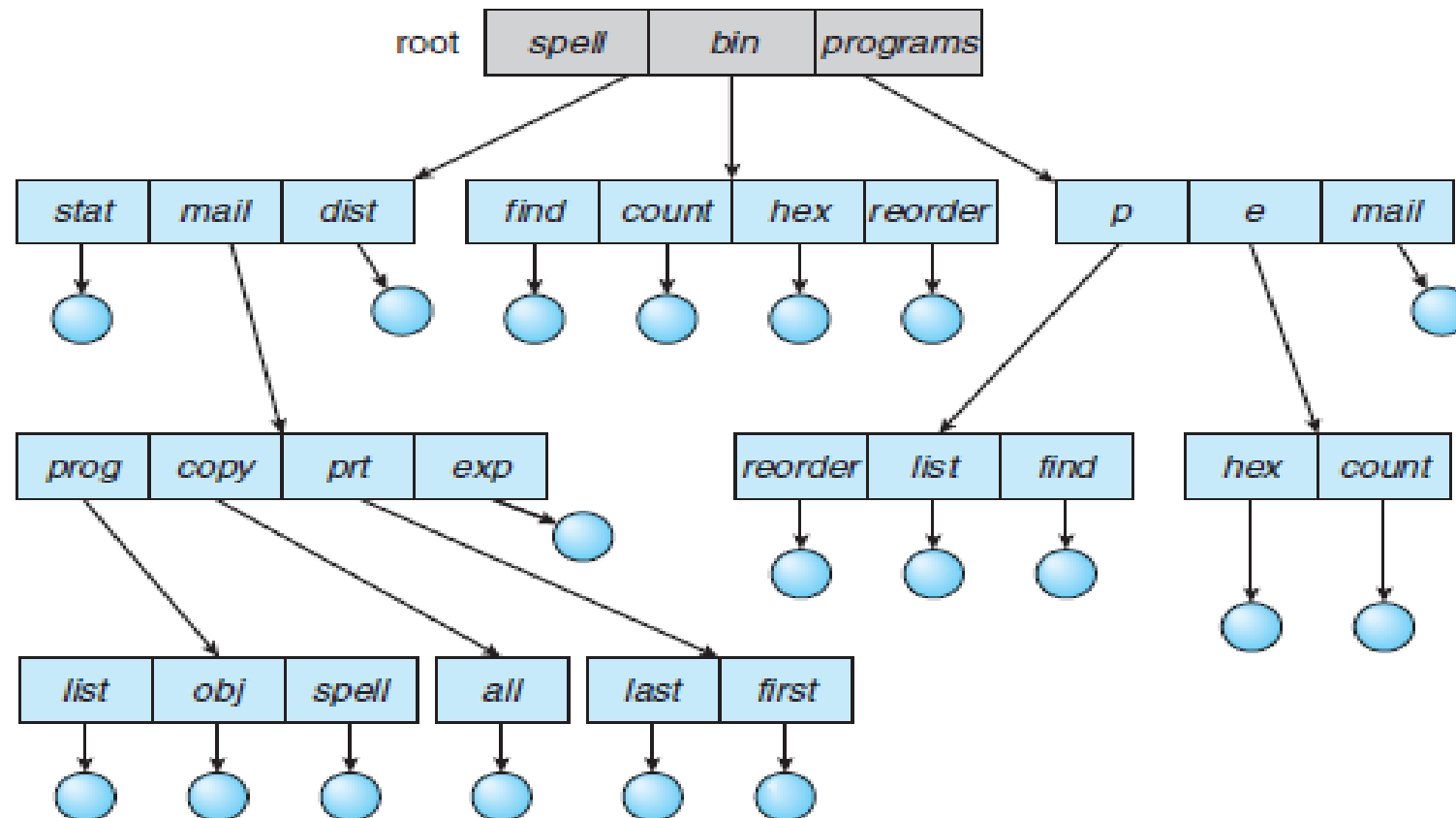
- **Disadvantages:**

- A user is not allowed to share files with other users.
- Still it not very scalable, two files of the same type cannot be grouped together in the same user.

3. TREE-STRUCTURED DIRECTORY

- Once we have seen a two-level directory as a tree of height 2, the natural generalization is to extend the directory structure to a tree of arbitrary height.
- This generalization allows the user to create their own subdirectories and to organize their files accordingly.
- A tree structure is the most common directory structure. The tree has a root directory, and every file in the system has a unique path.
- A directory (or subdirectory) contains a set of files or subdirectories.
- A directory is simply another file, but it is treated in a special way. All directories have the same internal format.
- One bit in each directory entry defines the entry as a file (0) or as a subdirectory (1).
- Special system calls are used to create and delete directories.

TREE-STRUCTURED DIRECTORY



TREE-STRUCTURED DIRECTORY

- **Advantages:**

- Very generalize, since full path name can be given.
- Very scalable, the probability of name collision is less.
- Searching becomes very easy, we can use both absolute path as well as relative.

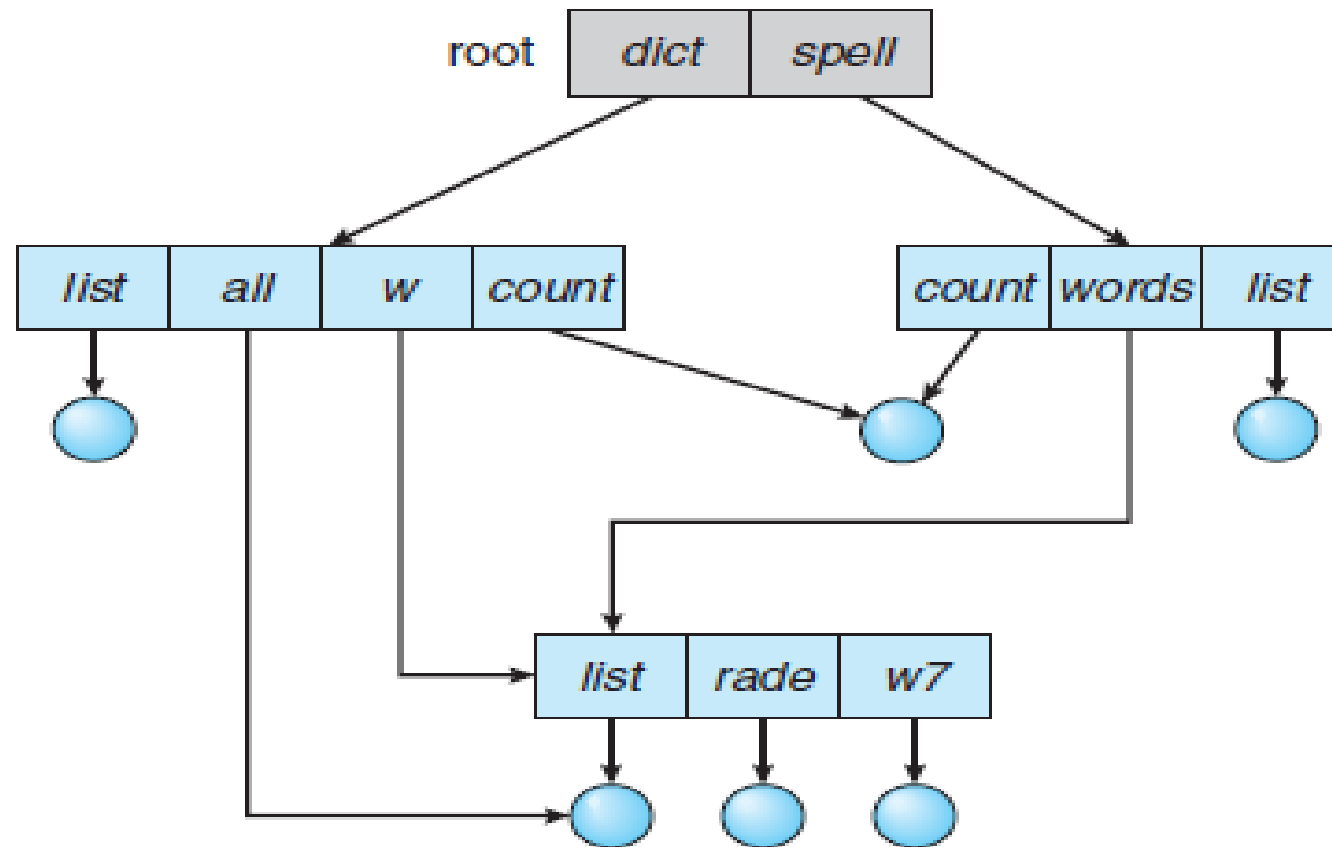
- **Disadvantages:**

- Every file does not fit into the hierarchical model, files may be saved into multiple directories.
- We can not share files.
- It is inefficient, because accessing a file may go under multiple directories.

4. ACYCLIC GRAPH DIRECTORY

- An acyclic graph is a graph with no cycle and allows to share subdirectories and files.
- The same file or subdirectories may be in two different directories.
- It is a natural generalization of the tree-structured directory. It is used in the situation like when two programmers are working on a joint project and they need to access files.
- The associated files are stored in a subdirectory, separating them from other projects and files of other programmers, since they are working on a joint project so they want the subdirectories to be into their own directories.
- The common subdirectories should be shared. So here we use Acyclic directories.
- It is the point to note that shared file is not the same as copy file . If any programmer makes some changes in the subdirectory it will reflect in both subdirectories.

ACYCLIC GRAPH DIRECTORY



ACYCLIC GRAPH DIRECTORY

- **Advantages:**

- We can share files.
- Searching is easy due to different-different paths.

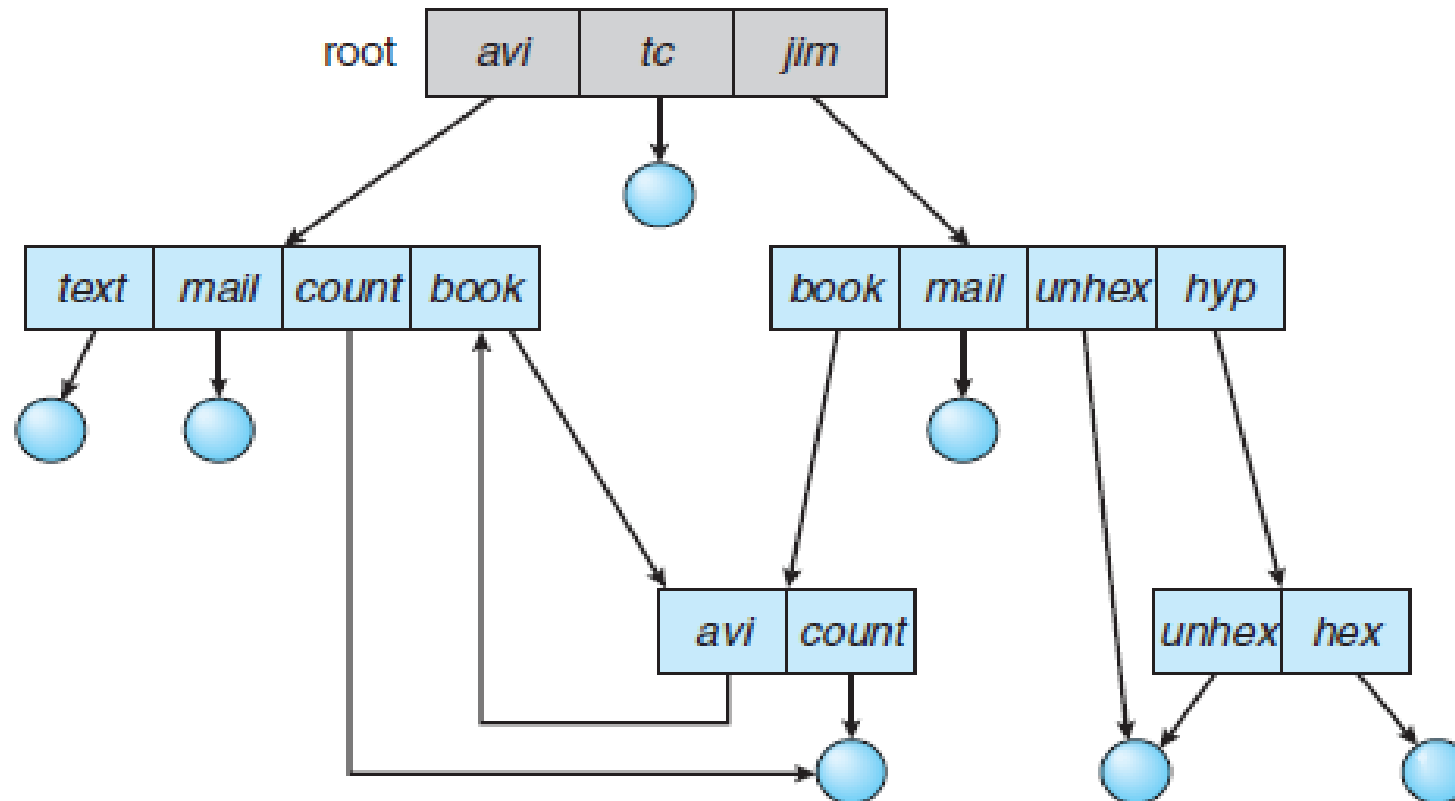
- **Disadvantages:**

- We share the files via linking, in case of deleting it may create the problem,
- If the link is softlink then after deleting the file we left with a dangling pointer.
- In case of hardlink, to delete a file we have to delete all the reference associated with it.

5. GENERAL GRAPH DIRECTORY STRUCTURE

- In general graph directory structure, cycles are allowed within a directory structure where multiple directories can be derived from more than one parent directory.
- The main problem with this kind of directory structure is to calculate total size or space that has been taken by the files and directories.
- If cycles are allowed to exist in the directory, we likewise want to avoid searching any component twice, for reasons of correctness as well as performance.
- A poorly designed algorithm might result in an infinite loop continually searching through the cycle and never terminating

GENERAL GRAPH DIRECTORY STRUCTURE



GENERAL GRAPH DIRECTORY STRUCTURE

- **Advantages:**

- It allows cycles.
- It is more flexible than other directories structure.

- **Disadvantages:**

- It is more costly than others.
- It needs garbage collection.

FILE ACCESS METHODS AND MATRICES

- The way that files are accessed and read into memory is determined by Access methods.
- Usually a single access method is supported by systems while there are OS's that support multiple access methods.
- There are **three ways** to access a file into a computer system:
 1. **Sequential-Access,**
 2. **Direct Access,**
 3. **Index sequential Method.**

1. SEQUENTIAL-ACCESS

- Data is accessed one record right after another in an order.
- Read command causes a pointer to be moved ahead by one.
- Write command allocates space for the record and moves the pointer to the new End Of File.
- Such a method is reasonable for tape.

2. DIRECT ACCESS

- This method is useful for disks.
- The file is viewed as a numbered sequence of blocks or records.
- There are no restrictions on which blocks are read/written, it can be done in any order.
- User now says "read n" rather than "read next".
- "n" is a number relative to the beginning of file, not relative to an absolute physical disk location.

3. INDEXED SEQUENTIAL ACCESS

- It is the other method of accessing a file which is built on the top of the sequential access method.
- These methods construct an index for the file.
- The index, like an index in the back of a book, contains the pointer to the various blocks.
- To find a record in the file, we first search the index and then by the help of pointer we access the file directly.
- It is built on top of Sequential access.
- It uses an Index to control the pointer while accessing files.

FILE SECURITY AND USER AUTHENTICATION

- To keep safe the data of the user from the improper access to the system.
- Protection can be provided in number of ways. For a single laptop system, we might provide protection by locking the computer in a desk drawer or file cabinet.
- For multi-user systems, different mechanisms are used for the protection.
- The files which have direct access of the any user have the need of protection.
- The files which are not accessible to other users doesn't require any kind of protection.
- The mechanism of the protection provide the facility of the controlled access by just limiting the types of access to the file

FILE SECURITY AND USER AUTHENTICATION

- **Types of Access :**
- **Read –**
Reading from a file.
- **Write –**
Writing or rewriting the file.
- **Execute –**
Loading the file and after loading the execution process starts.
- **Append –**
Writing the new information to the already existing file, editing must be end at the end of the existing file.
- **Delete –**
Deleting the file which is of no use and using its space for the another data.
- **List –**
List the name and attributes of the file.
- Operations like renaming, editing the existing file, copying; these can also be controlled.

FILE SECURITY AND USER AUTHENTICATION

- **Access Control :**

- There are different methods used by different users to access any file.
- The general way of protection is to associate *identity-dependent access* with all the files and directories an list called **access-control list (ACL)** which specify the names of the users and the types of access associate with each of the user.
- The main problem with the access list is their length. If we want to allow everyone to read a file, we must list all the users with the read access.

FILE SECURITY AND USER AUTHENTICATION

- The access to any system is also controlled by the password. If the use of password are is random and it is changed often, this may be result in limit the effective access to a file.

The use of passwords has a few disadvantages:

- The number of passwords are very large so it is difficult to remember the large passwords.
- If one password is used for all the files, then once it is discovered, all files are accessible; protection is on all-or-none basis.

CASES STUDIES

LINUX - FILE MANAGEMENT

I Hear and I Forget.
I See and I Remember.
I Do and I Understand.



Good Luck

GOOD LUCK

5CS4-03: OPERATING SYSTEM

Er. Manish Choubisa

Assistant Professor

Department of Computer Engineering

UNIT- II

Memory Management

CONTENTS

- **Memory management:**
- contiguous memory allocation,
- virtual memory,
- Paging
- page table structure
- demand paging
- page replacement policies
- Thrashing
- Segmentation
- case study

Contents: Unit-2

- **Memory management:**
- Contiguous memory allocation,
- Virtual memory,
- Paging
- Page table structure
- demand paging
- Page replacement policies
 - FIFO page replacement
 - Optimal page replacement
 - LRU page replacement
- Thrashing
- Segmentation
- Case study



MEMORY MANAGEMENT

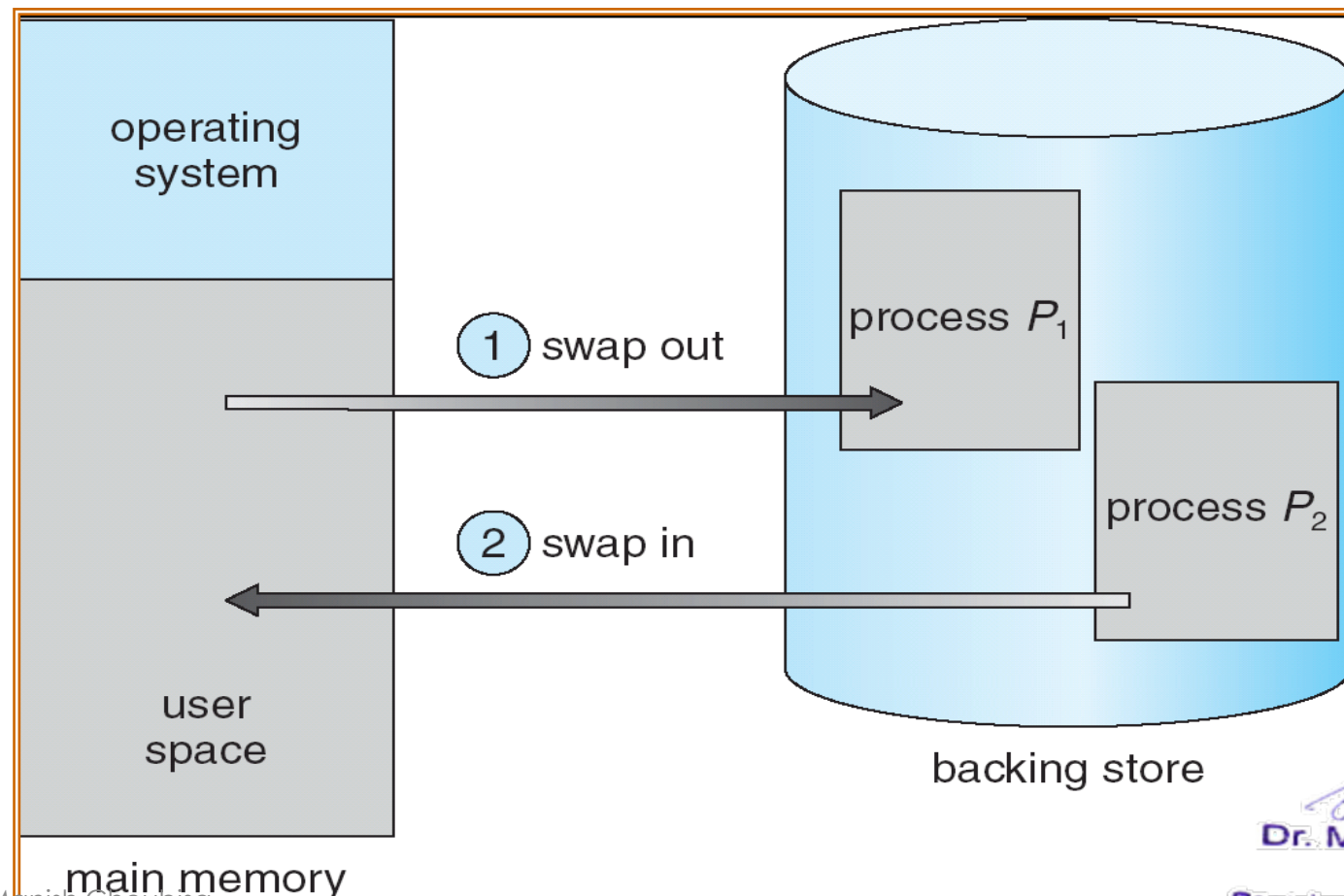
MEMORY MANAGEMENT

- Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution.
- Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free.
- It checks how much memory is to be allocated to processes.
- It decides which process will get memory at what time.
- It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

SWAPPING

- Swapping is a mechanism in which a process can be swapped temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes.
- At some later time, the system swaps back the process from the secondary storage to main memory.
- Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason **Swapping is also known as a technique for memory compaction.**

SCHEMATIC VIEW OF SWAPPING



MEMORY ALLOCATION

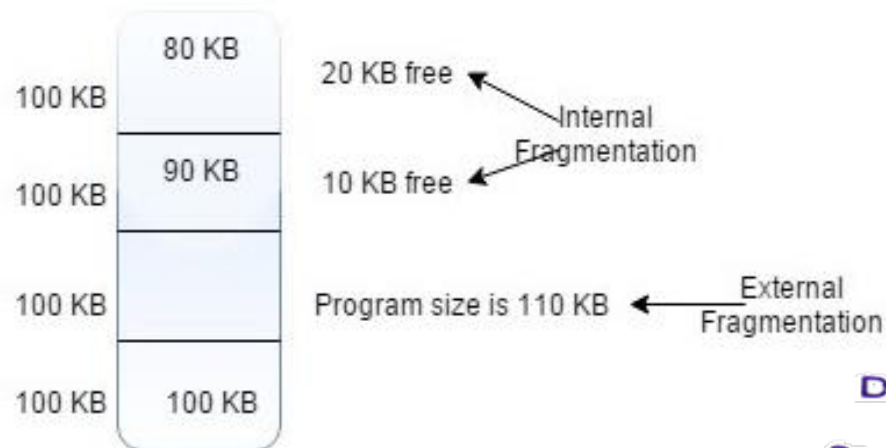
- Memory allocation is a process by which computer programs are assigned memory or space.
- Main memory usually has two partitions –
 1. **Low Memory** - Operating system resides in this type of memory.
 2. **High Memory**- User processes are held in high memory.

FRAGMENTATION

- In computer storage, **fragmentation** is a phenomenon in **which storage space is used inefficiently**, reducing capacity or performance and often both.
- ❖ As processes are loaded and removed from memory, the free memory space is broken into little pieces.
- ❖ It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused.
- ❖ This problem is known as Fragmentation.

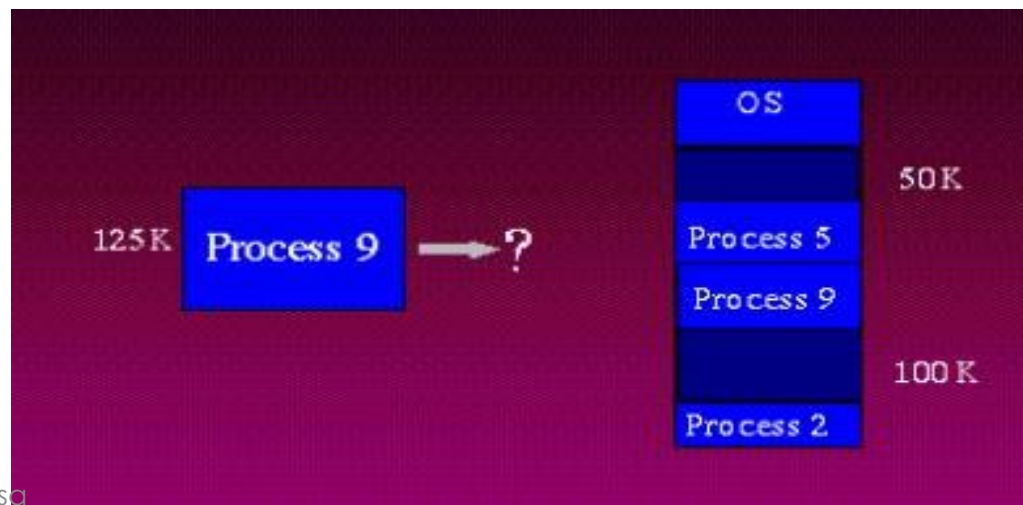
INTERNAL FRAGMENTATION

- When a program is allocated to a memory block, if that program is lesser than this memory block and remaining space goes wasted, this situation is called internal fragmentation.
- Generally, internal fragmentation memory partition is static or fixed.



EXTERNAL FRAGMENTATION

- Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used.
- **External fragmentation** arises when free memory is separated into small blocks and is interspersed by allocated memory.



PARTITION ALLOCATION

- Memory is divided into different blocks or partitions.
- Each process is allocated according to the requirement.
- Partition allocation is an ideal method to avoid internal fragmentation.

PARTITION ALGORITHM

- **First Fit:** In this type fit, the partition is allocated, which is the first sufficient block from the beginning of the main memory.
- **Best Fit:** It allocates the process to the partition that is the first **smallest partition** among the free partitions.
- **Worst Fit:** It allocates the process to the partition, which is the **largest sufficient** freely available partition in the main memory.

MEMORY ALLOCATION

1. Contiguous

- Fixed Partition
- Variable Partition

2. Non-Contiguous

- Paging
- Multilevel paging
- Segmentation
- Segmented with paging

Contents: Unit-2

- Memory management:
- **Contiguous memory allocation**
- Virtual memory,
- Paging
- Page table structure
- demand paging
- Page replacement policies
 - FIFO page replacement
 - Optimal page replacement
 - LRU page replacement
- Thrashing
- Segmentation
- Case study



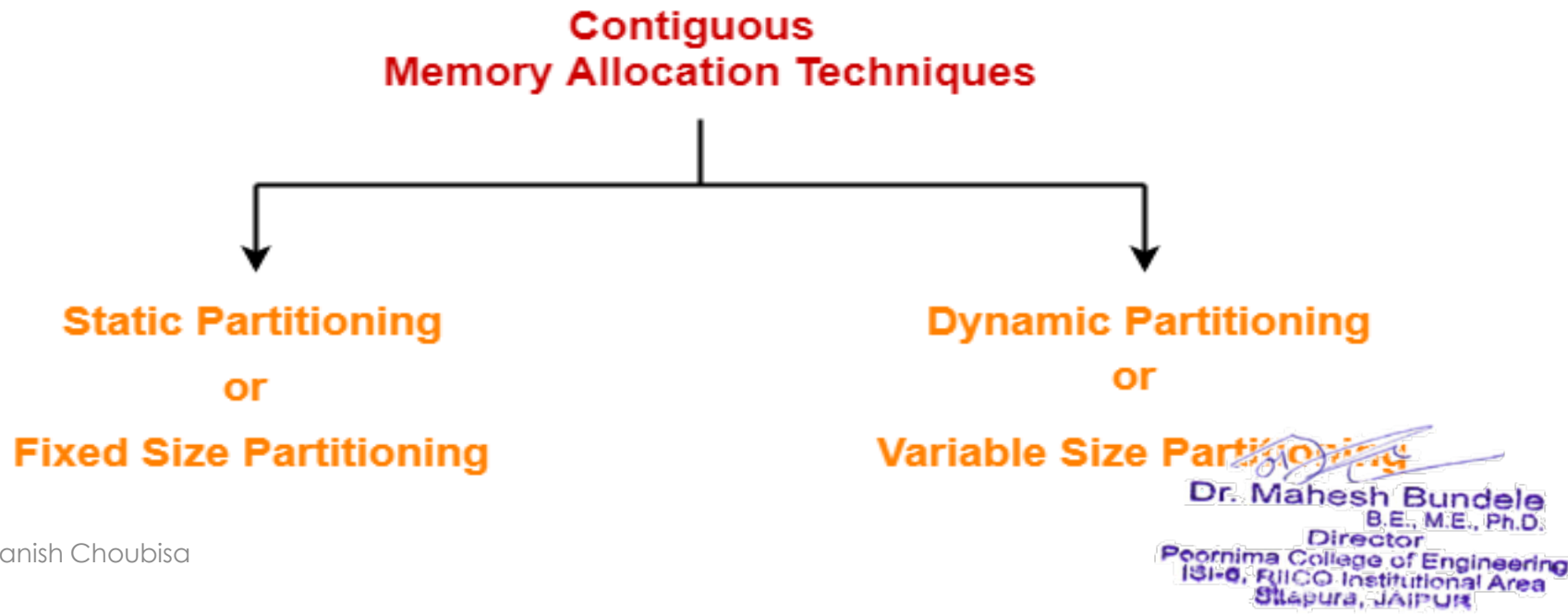
CONTIGUOUS MEMORY ALLOCATION

CONTIGUOUS MEMORY ALLOCATION

- Each process occupies a contiguous block of physical memory
- It Allocates a **single contiguous section of memory** to a process or a file
- It allows to store the process only in a contiguous fashion.
- Thus, entire process has to be stored as a single entity at one place inside the memory.

CONTIGUOUS MEMORY ALLOCATION (CONT...)

- To allocate the **contiguous** space to user processes, the memory can be divide either in the **fixed-sized partition** or in the **variable-sized partition**.



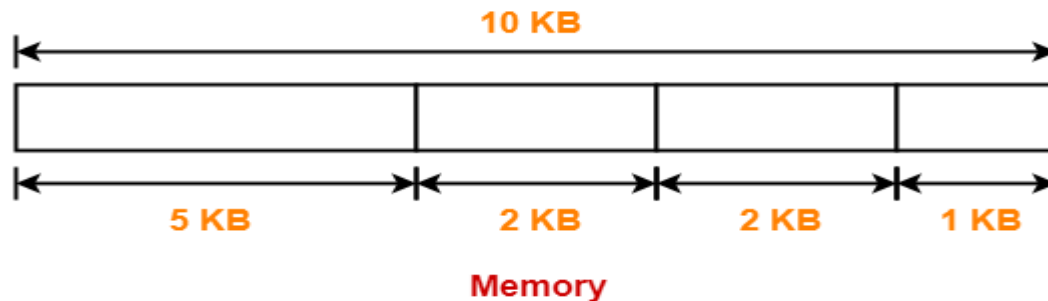
STATIC PARTITIONING

- Static partitioning is a fixed size partitioning scheme.
- In this technique, main memory is pre-divided into fixed size partitions.
- The size of each partition is fixed and can not be changed.
- Each partition is allowed to store only one process.
- **Internal Fragmentation occurs** only in static partitioning.

STATIC PARTITIONING

- **Example:**

Under fixed size partitioning scheme, a memory of size 10 KB may be divided into fixed size partitions as-



- These partitions are allocated to the processes as they arrive.
- The partition allocated to the arrived process depends on the algorithm followed.

ADVANTAGES OF STATIC PARTITIONING

- It is simple and easy to implement.
- It supports multiprogramming since multiple processes can be stored inside the main memory.
- Only one memory access is required which reduces the access time.

DISADVANTAGES OF STATIC PARTITIONING

- It suffers from both **internal fragmentation and external fragmentation**.
- It utilizes memory inefficiently.
- The degree of multiprogramming is limited equal to number of partitions.
- There is a limitation on the size of process since processes with size greater than the size of largest partition can't be stored and executed.

DYNAMIC PARTITIONING

- Dynamic partitioning is a **variable size partitioning** scheme.
- It performs the allocation dynamically.
- When a process arrives, a partition of size equal to the size of process is created.
- Then, that partition is allocated to the process.

ADVANTAGES OF DYNAMIC PARTITIONING

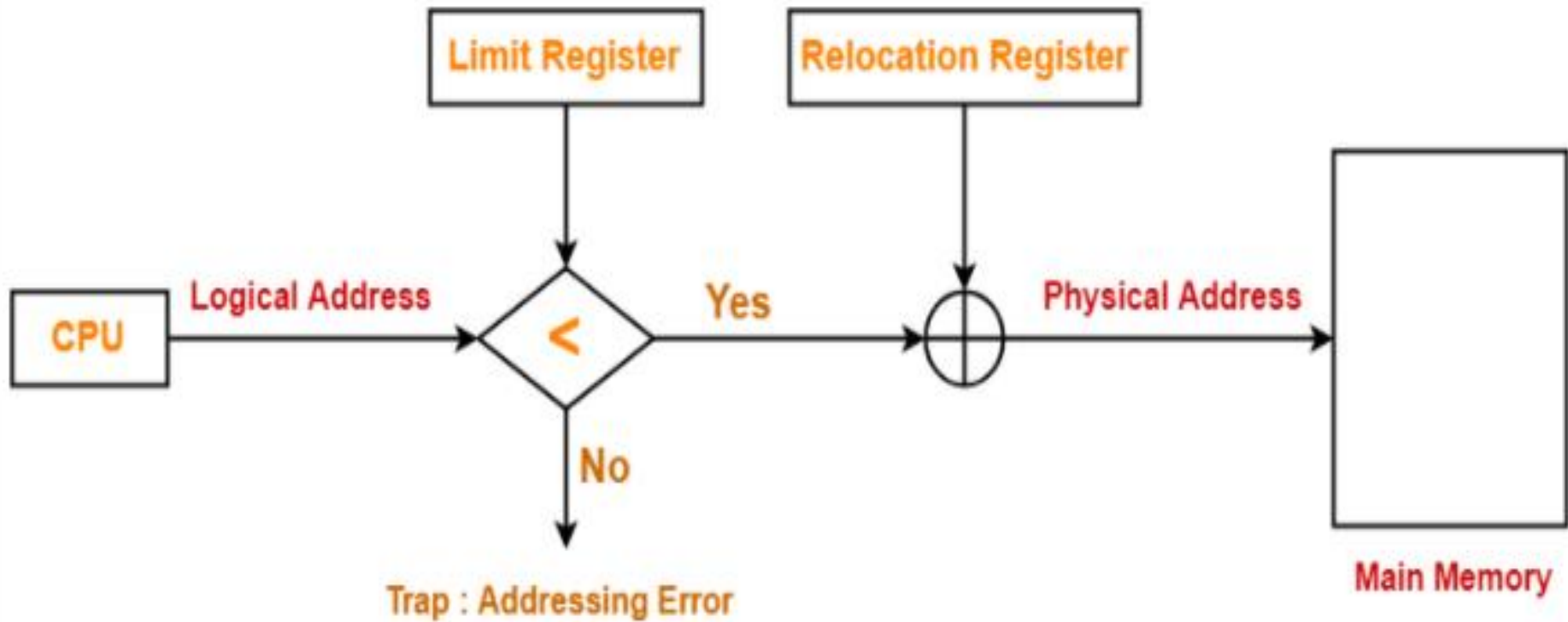
- It does not suffer from internal fragmentation.
- Degree of multiprogramming is dynamic.
- There is no limitation on the size of processes.

DISADVANTAGES OF DYNAMIC PARTITIONING

- It suffers from external fragmentation.
- Allocation and deallocation of memory is complex.

TRANSLATING LOGICAL ADDRESS INTO PHYSICAL ADDRESS

- **Memory Protection:**
- CPU always generates a logical address.
- A physical address is needed to access the main memory.
- The translation scheme uses two registers:
 - Relocation Register
 - Limit Register
- Relocation Register stores the base address or starting address of the process in the main memory.
- Limit Register stores the size or length of the process.



Translating Logical Address into Physical Address

Practice question1

- (Contiguous memory allocation)

Consider six memory partitions of size 200 KB, 400 KB, 600 KB, 500 KB, 300 KB and 250 KB. These partitions need to be allocated to four processes of sizes 357 KB, 210 KB, 468 KB and 491 KB in that order.

Perform the allocation of processes using-

- First Fit Algorithm
- Best Fit Algorithm
- Worst Fit Algorithm

Practice question 2

- (Contiguous memory allocation)

Consider the following heap (figure) in which blank regions are not in use and hatched regions are in use-



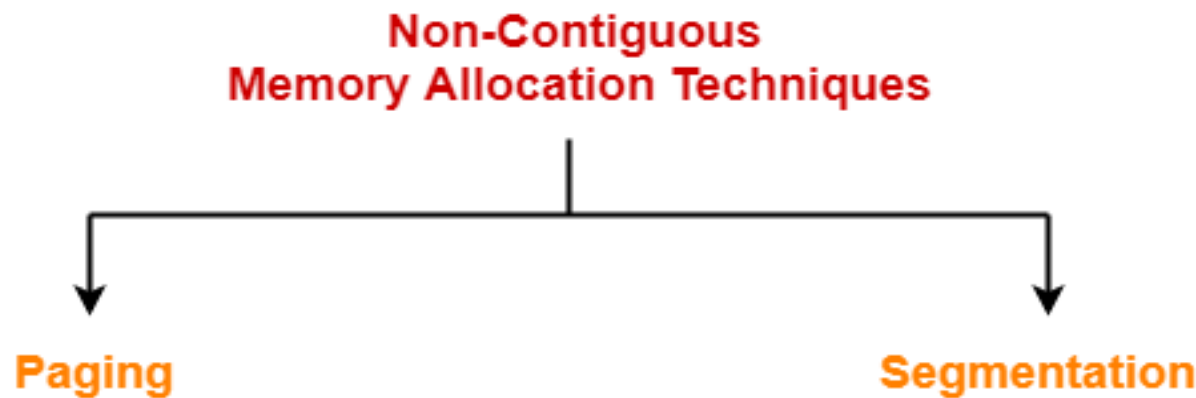
The sequence of requests for blocks of size 300, 25, 125, 50 can be satisfied if we use-

- Either first fit or best fit policy (any one)
- First fit but not best fit policy
- Best fit but not first fit policy
- None of the above


Non-Contiguous Memory Allocation

NON-CONTIGUOUS MEMORY ALLOCATION-

- It allows to store parts of a single process in a non-contiguous fashion.
- Thus, different parts of the same process can be stored at different places in the main memory.



Contents: Unit-2

- **Memory management:**
- Contiguous memory allocation
- **Virtual memory** 
- Paging
- Page table structure
- demand paging
- Page replacement policies
 - FIFO page replacement
 - Optimal page replacement
 - LRU page replacement
- Thrashing
- Segmentation
- Case study

VIRTUAL MEMORY

VIRTUAL MEMORY

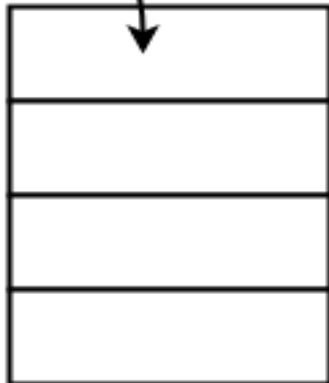
- A computer can address more memory than the amount physically installed on the system.
- This extra memory is actually called **virtual memory**
- it is a section of a hard disk that's set up to emulate the computer's RAM.
- Virtual memory is commonly implemented by demand paging.
- Demand segmentation can also be used to provide virtual memory.

Virtual memory	A storage allocation scheme in which secondary memory can be addressed as though it were part of main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites, and program-generated addresses are translated automatically to the corresponding machine addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of secondary memory available and not by the actual number of main storage locations.
Virtual address	The address assigned to a location in virtual memory to allow that location to be accessed as though it were part of main memory.
Virtual address space	The virtual storage assigned to a process.
Address space	The range of memory addresses available to a process.
Real address	The address of a storage location in main memory.

PAGING

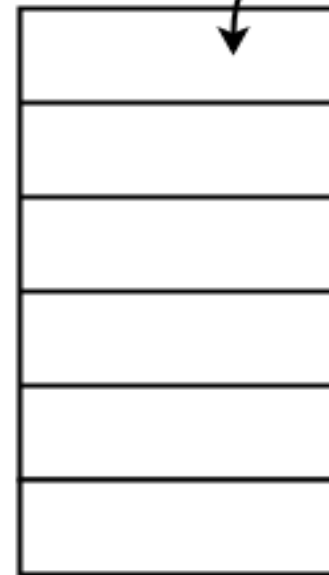
- Paging is a fixed size partitioning scheme.
- In paging, secondary memory and main memory are divided into **equal fixed size partitions**.
- The partitions of secondary memory are called as **pages**.
- The partitions of main memory are called as **frames**.
- Each process is divided into parts where **size of each part is same as page size**.
- The size of the last part may be less than the page size.
- The pages of process are stored in the frames of main memory depending upon their availability.

Frames



Main Memory

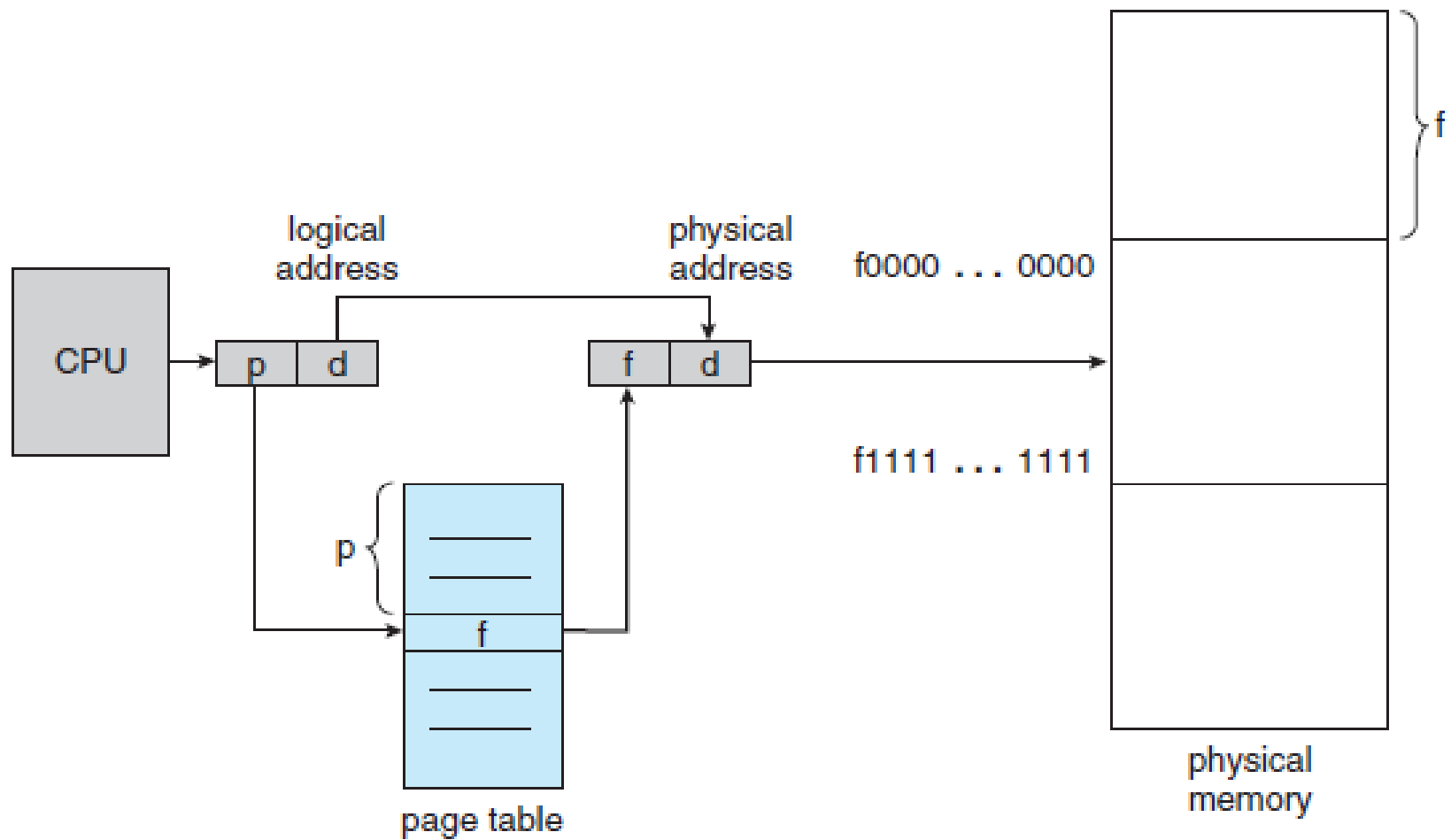
Pages



Secondary Memory

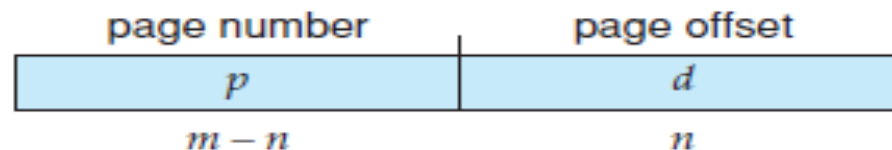
TRANSLATING LOGICAL ADDRESS INTO PHYSICAL ADDRESS

- CPU always generates a logical address.
- A physical address is needed to access the main memory.
- Every address generated by the CPU is divided into two parts: a **page number (p)** and a **page offset (d)**.



Paging hardware.

- The page number is used as an index into a **page table**.
- The **page table** contains the base address of each page in physical memory.
- This base address is combined with the page offset to define the physical memory address that is sent to the memory unit.
- If the size of the logical address space is 2^m , and a page size is 2^n bytes, then the high-order $m - n$ bits of a logical address designate the page number, and the n low-order bits designate the page offset



page 0
page 1
page 2
page 3

logical
memory

0	1
1	4
2	3
3	7

page table

frame
number

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

physical
memory

Paging model of logical and physical


ADVANTAGES OF PAGING

- It allows to store parts of a single process in a non-contiguous fashion.
- It solves the problem of external fragmentation.

DISADVANTAGES OF PAGING

- It suffers from internal fragmentation.
- There is an overhead of maintaining a page table for each process.
- The time taken to fetch the instruction increases since now two memory accesses are required.

Contents: Unit-2

- **Memory management:**
- Contiguous memory allocation
- Virtual memory
- Paging
- **Page table structure** 
- demand paging
- Page replacement policies
 - FIFO page replacement
 - Optimal page replacement
 - LRU page replacement
- Thrashing
- Segmentation
- Case study

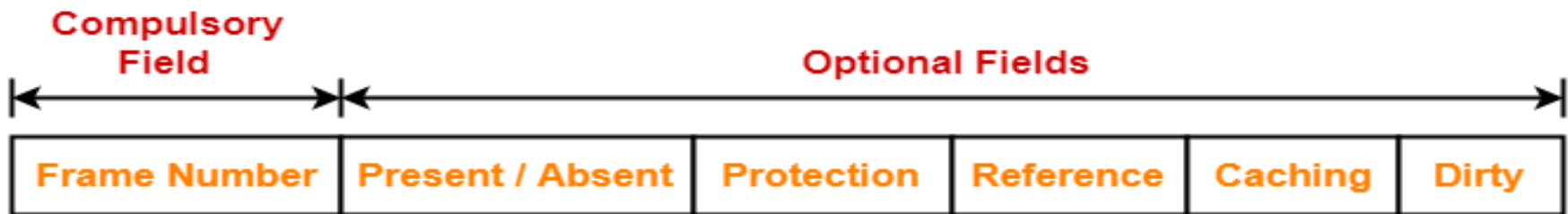
PAGE TABLE STRUCTURE

PAGE TABLE STRUCTURE

- Page table is a data structure.
- It maps the page number referenced by the CPU to the frame number where that page is stored.
- Page table is stored in the main memory.
- Number of entries in a page table = Number of pages in which the process is divided.
- Page Table Base Register (PTBR) contains the base address of page table.
- Each process has its own independent page table.

PAGE TABLE ENTRY

- A page table entry contains several information about the page.
- The information contained in the page table entry varies from operating system to operating system.
- The most important information in a page table entry is frame number.



Page Table Entry Format

1. Frame Number-

- Frame number specifies the frame where the page is stored in the main memory.
- The number of bits in frame number depends on the number of frames in the main memory.

2. Present / Absent Bit-

- This bit is also sometimes called as **valid / invalid bit**.
- This bit specifies whether that page is present in the main memory or not.
- If the page is not present in the main memory, then this bit is set to 0 otherwise set to 1.

3. Protection Bit-

- This bit is also sometimes called as “**Read / Write bit**”.
- This bit is concerned with the page protection.
- It specifies the permission to perform read and write operation on the page.
- If only read operation is allowed to be performed and no writing is allowed, then this bit is set to 0.
- If both read and write operation are allowed to be performed, then this bit is set to 1.

4. Reference Bit-

- Reference bit specifies whether that page has been referenced in the last clock cycle or not.
- If the page has been referenced recently, then this bit is set to 1 otherwise set to 0.

5. Caching Enabled / Disabled-

- This bit enables or disables the caching of page.
- Whenever freshness in the data is required, then caching is disabled using this bit.
- If caching of the page is disabled, then this bit is set to 1 otherwise set to 0.

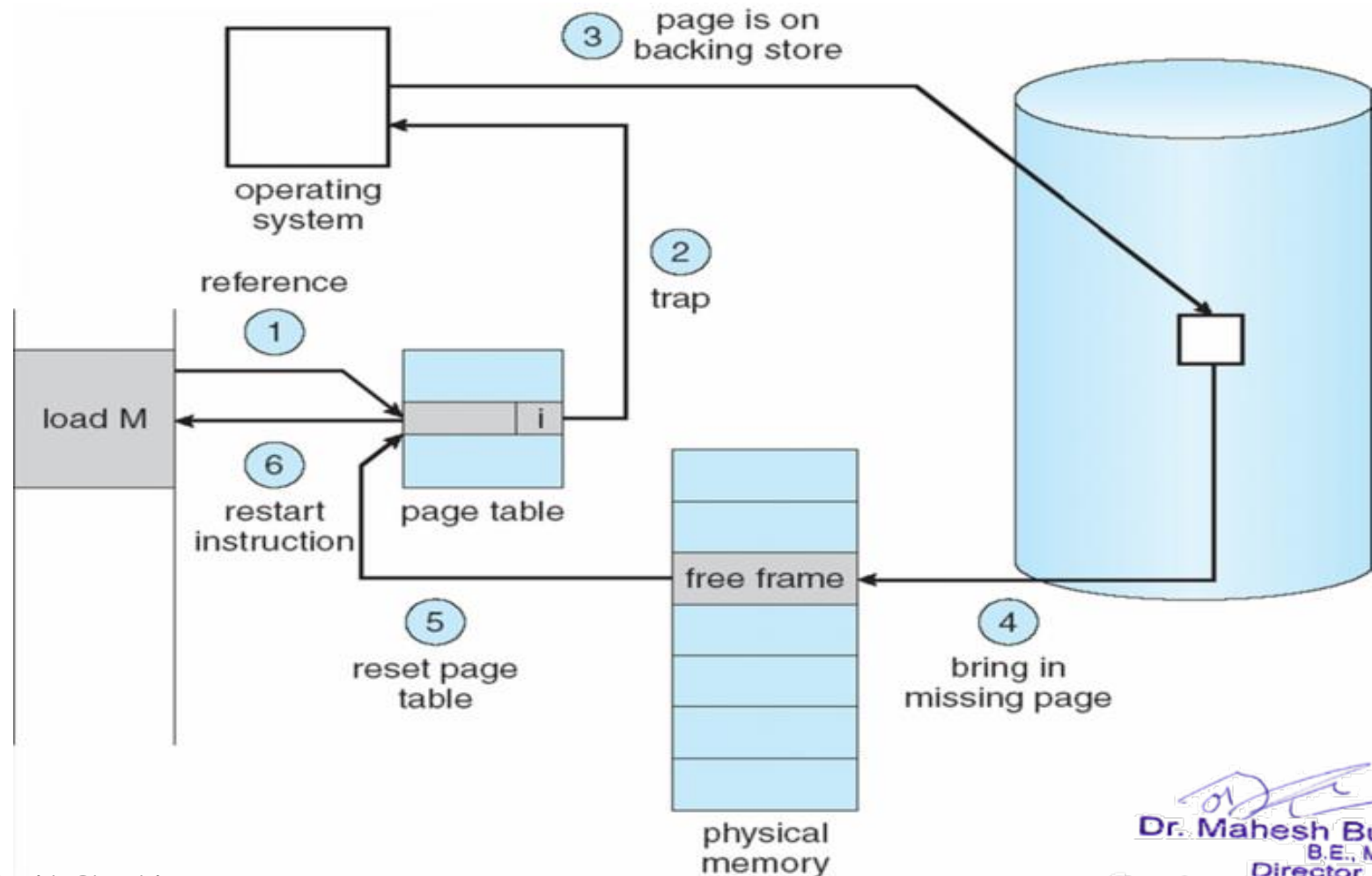
6. Dirty Bit-

- This bit is also sometimes called as “**Modified bit**”.
- This bit specifies whether that page has been modified or not.
- If the page has been modified, then this bit is set to 1 otherwise set to 0.


PAGE FAULT

- When a page referenced by the CPU is not found in the main memory, it is called as a **page fault**.
- When a page fault occurs, the required page has to be fetched from the secondary memory into the main memory.

PAGE FAULT (CONT..)



Contents: Unit-2

- **Memory management:**
- Contiguous memory allocation
- Virtual memory
- Paging
- Page table structure
- **demand paging** 
- Page replacement policies
 - FIFO page replacement
 - Optimal page replacement
 - LRU page replacement
- Thrashing
- Segmentation
- Case study

DEMAND PAGING

DEMAND PAGING

- Demand paging is a technique used in virtual memory systems where the pages are brought in the main memory only when required or demanded by the CPU.
- Hence, it is also named as **lazy swapper** because the swapping of pages is done only when required by the CPU

- **Advantages**

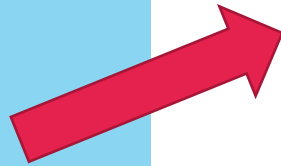
- It increases the degree of multiprogramming as many processes can be present in the main memory at the same time.
- There is a more efficient use of memory as processes having size more than the size of the main memory can also be executed using this mechanism because we are not loading the whole page at a time.

- **Disadvantages**

- The amount of processor overhead and the number of tables used for handling the page faults is greater than in simple page management techniques.

Contents: Unit-2

- **Memory management:**
- Contiguous memory allocation
- Virtual memory
- Paging
- Page table structure
- demand paging
- **Page replacement policies**
 - FIFO page replacement
 - Optimal page replacement
 - LRU page replacement
- Thrashing
- Segmentation
- Case study



PAGE REPLACEMENT POLICIES

PAGE REPLACEMENT ALGORITHMS

- Page replacement is a process of swapping out an existing page from the frame of a main memory and replacing it with the required page.
- Page replacement is required when-
 - ✓ All the frames of main memory are already occupied.
 - ✓ Thus, a page has to be replaced to create a room for the required page.
- Page replacement algorithms help to decide which page must be swapped out from the main memory to create a room for the incoming page

PAGE REPLACEMENT ALGORITHMS

1. FIFO Page Replacement Algorithm
2. Optimal Page Replacement Algorithm
3. LRU Page Replacement Algorithm

A good page replacement algorithm is one that minimizes the number of page faults

1. FIFO PAGE REPLACEMENT ALGORITHM

- As the name suggests, this algorithm works on the principle of “**First in First out**”.
- It replaces the oldest page that has been present in the main memory for the longest time.
- It is implemented by keeping track of all the pages in a queue

BELADY'S ANOMALY

- Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm.
- For example, if we consider reference string 3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4 and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10 page faults.

2. OPTIMAL PAGE REPLACEMENT ALGORITHM-

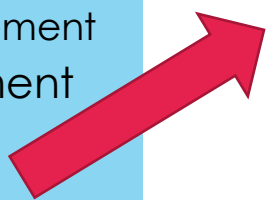
- This algorithm replaces the page that will not be referred by the CPU in future for the longest time.
- It is practically impossible to implement this algorithm.
- This is because the pages that will not be used in future for the longest time can not be predicted.
- However, it is the best known algorithm and gives the least number of page faults.
- Hence, it is used as a performance measure criterion for other algorithms.

3. LRU PAGE REPLACEMENT ALGORITHM

- As the name suggests, this algorithm works on the principle of “**Least Recently Used**”.
- It replaces the page that has not been referred by the CPU for the longest time.

Contents: Unit-2

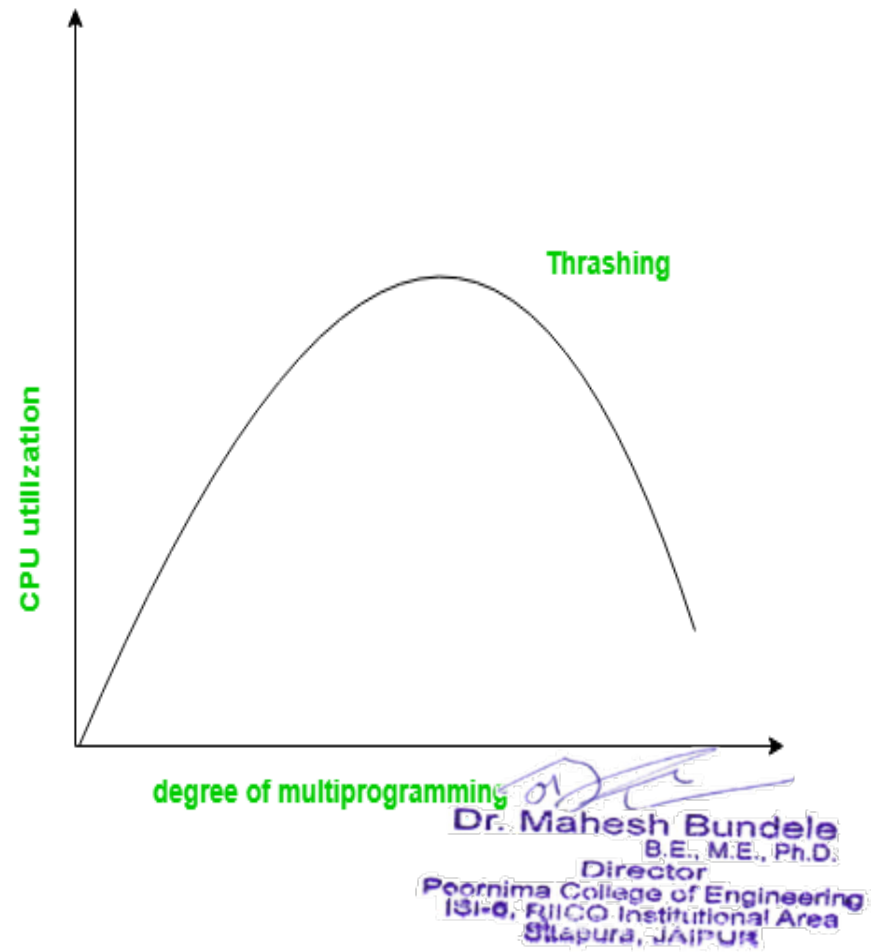
- **Memory management:**
 - Contiguous memory allocation
 - Virtual memory
 - Paging
 - Page table structure
 - demand paging
 - Page replacement policies
 - FIFO page replacement
 - Optimal page replacement
 - LRU page replacement
- **Thrashing**
- Segmentation
- Case study



THRASHING

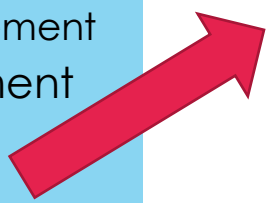
THRASHING

- If this page fault and then swapping happening very frequently at higher rate, then operating system has to spend more time to swap these pages.
- This state is called thrashing.
- Because of this, CPU utilization is going to be reduced.



Contents: Unit-2

- **Memory management:**
- Contiguous memory allocation
- Virtual memory
- Paging
- Page table structure
- demand paging
- Page replacement policies
 - FIFO page replacement
 - Optimal page replacement
 - LRU page replacement
- Thrashing
- **Segmentation**
- Case study



SEGMENTATION

SEGMENTATION

- Like Paging, Segmentation is another non-contiguous memory allocation technique.
- In segmentation, process is not divided blindly into fixed size pages.
- Rather, the process is divided into modules for better visualization.
- Segmentation is a variable size partitioning scheme.
- In segmentation, secondary memory and main memory are divided into partitions of unequal size.
- The size of partitions depend on the length of modules.
- The partitions of secondary memory are called as **segments**.

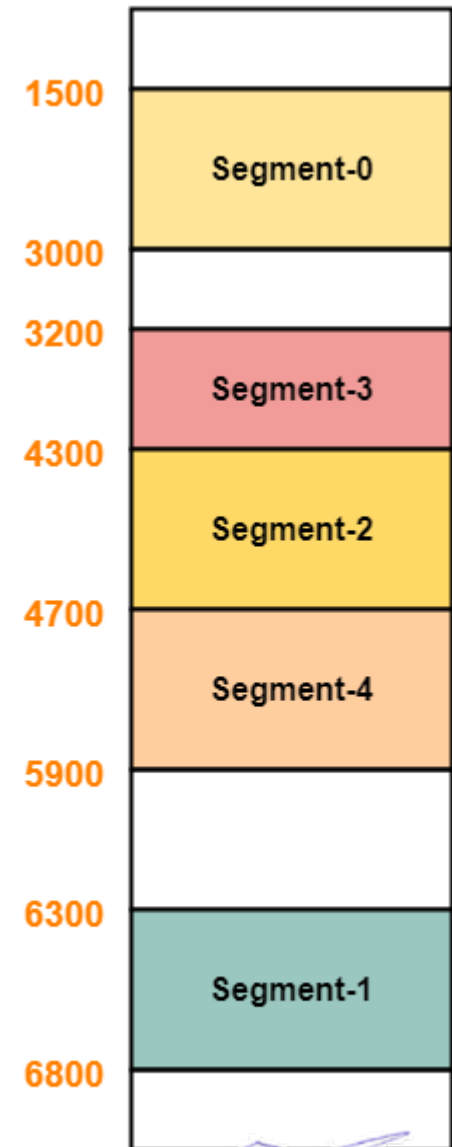
SEGMENT TABLE

- Segment table is a table that stores the information about each segment of the process.
- It has two columns.
 - First column stores the size or length of the segment.
 - Second column stores the base address or starting address of the segment in the main memory.
- Segment table is stored as a separate segment in the main memory.
- Segment table base register (STBR) stores the base address of the segment table.

EXAMPLE

	Limit	Base
Seg-0	1500	1500
Seg-1	500	6300
Seg-2	400	4300
Seg-3	1100	3200
Seg-4	1200	4700

Segment Table



- Limit indicates the length or size of the segment.
- Base indicates the base address or starting address of the segment in the main memory.

ADVANTAGES OF SEGMENTATION

- It allows to divide the program into modules which provides better visualization.
- Segment table consumes less space as compared to Page Table in paging.
- It solves the problem of internal fragmentation.

DISADVANTAGES OF SEGMENTATION

- The time taken to fetch the instruction increases since now two memory accesses are required.
- Segments of unequal size are not suited for swapping.
- It suffers from **external fragmentation** as the free space gets broken down into smaller pieces with the processes being loaded and removed from the main memory.

CASE STUDY

Intel 32 and 64-bit Architectures

IMPORTANT POINT

- Physical Address Space = Size of main memory
- Size of main memory = Total number of frames x Page size
- Frame size = Page size
- If number of frames in main memory = 2^X , then number of bits in frame number = X bits
- If Page size = 2^X Bytes, then number of bits in page offset = X bits
- If size of main memory = 2^X Bytes, then number of bits in physical address = X bits

IMPORTANT POINT

- Virtual Address Space = Size of process
- Number of pages the process is divided = $\text{Process size} / \text{Page size}$
- If process size = 2^X bytes, then number of bits in virtual address space = X bits
- Size of page table = Number of entries in page table \times Page table entry size
- Number of entries in pages table = Number of pages the process is divided
- Page table entry size = Number of bits in frame number + Number of bits used for optional fields if any

I Hear and I Forget.
I See and I Remember.
I Do and I Understand.



Good Luck

GOOD LUCK



Thank You

Happy Learning

5CS4-03: Operating System

Er. Manish Choubisa

Assistant Professor

Department of Computer Engineering

Poornima College of Engineering, Jaipur

UNIT-3

Deadlocks

Content

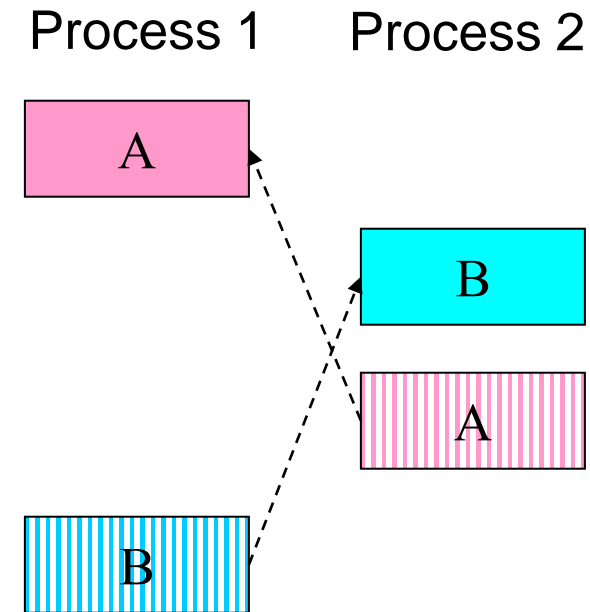
- ▶ **Deadlock:**
- ▶ Shared resources,
- ▶ resource allocation and scheduling,
- ▶ resource graph models,
- ▶ Methods for Handling Deadlocks
 1. Deadlock prevention
 2. Deadlock avoidance
 3. Deadlock detection and recovery
 4. Deadlock Ignorance
- ▶ **Device management:**
- ▶ devices and their characteristics,
- ▶ device drivers,
- ▶ device handling,
- ▶ disk scheduling algorithms and policies

System Model

- ▶ System consists of resources
- ▶ Resource types R_1, R_2, \dots, R_m
CPU cycles, memory space, I/O devices
- ▶ Each resource type R_i has W_i instances.
- ▶ Each process utilizes a resource as follows:
 - ▶ request
 - ▶ use
 - ▶ release

When do deadlocks happen?

- ▶ Suppose
 - ▶ Process 1 holds resource A and requests resource B
 - ▶ Process 2 holds B and requests A
 - ▶ Both can be blocked, with neither able to proceed
- ▶ Deadlocks occur when ...
 - ▶ Processes are granted exclusive access to devices or software constructs (resources)
 - ▶ Each deadlocked process needs a resource held by another deadlocked process



DEADLOCK!

What is a deadlock?

- ▶ Formal definition:
“A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.”
- ▶ Usually, the event is release of a currently held resource
- ▶ In deadlock, none of the processes can
 - ▶ Run
 - ▶ Release resources
 - ▶ Be awakened

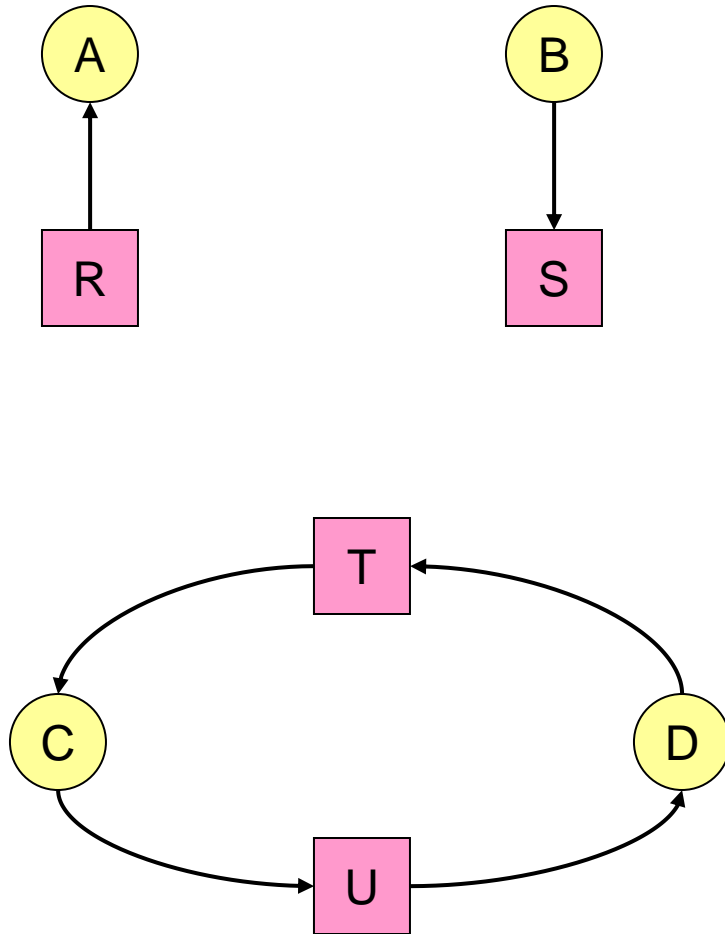
Deadlock Characterization

- ▶ Four conditions for deadlock:
- ▶ **Mutual exclusion**
 - ▶ Each resource is assigned to at most one process
- ▶ **Hold and wait**
 - ▶ A process holding resources can request more resources
- ▶ **No preemption**
 - ▶ Previously granted resources cannot be forcibly taken away
- ▶ **Circular wait**
 - ▶ There must be a circular chain of 2 or more processes where each is waiting for a resource held by the next member of the chain

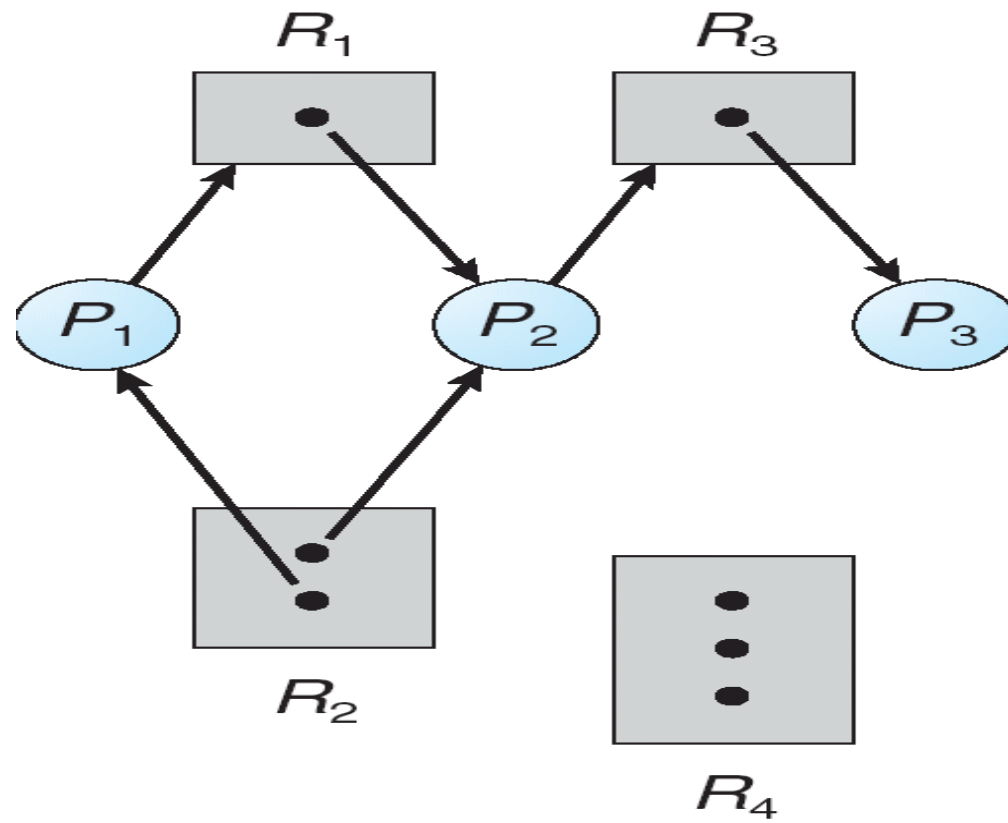
Resource-Allocation Graph

- ▶ A set of vertices V and a set of edges E .
- ▶ V is partitioned into two types:
 - ▶ $P = \{P_1, P_2, \dots, P_n\}$, the set consisting of all the processes in the system
 - ▶ $R = \{R_1, R_2, \dots, R_m\}$, the set consisting of all resource types in the system
- ▶ **request edge** - directed edge $P_i \rightarrow R_j$
- ▶ **assignment edge** - directed edge $R_j \rightarrow P_i$

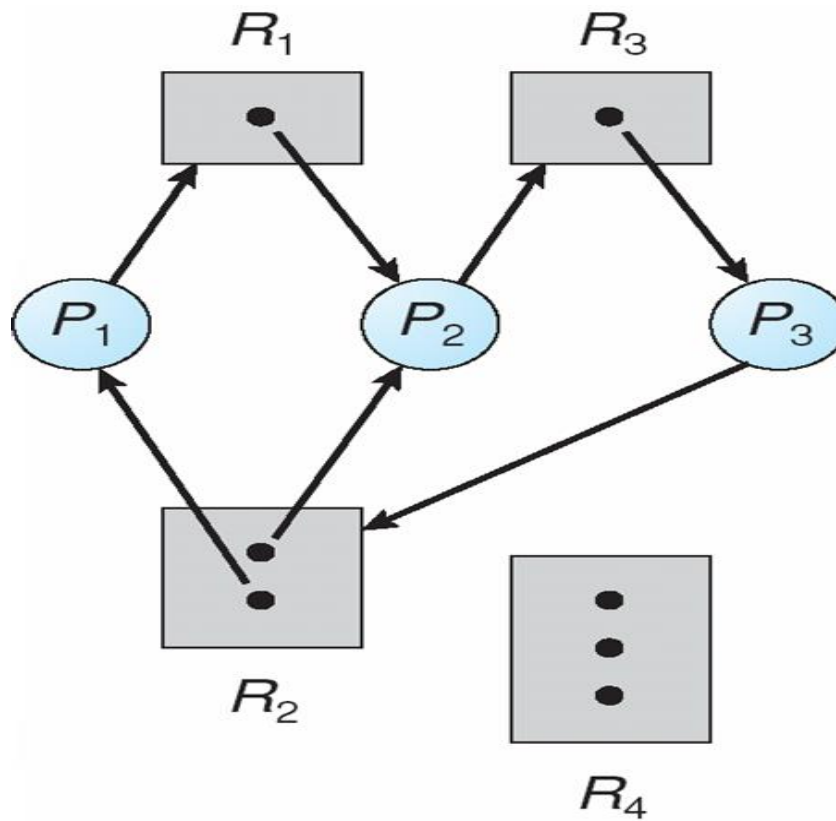
Resource allocation graphs



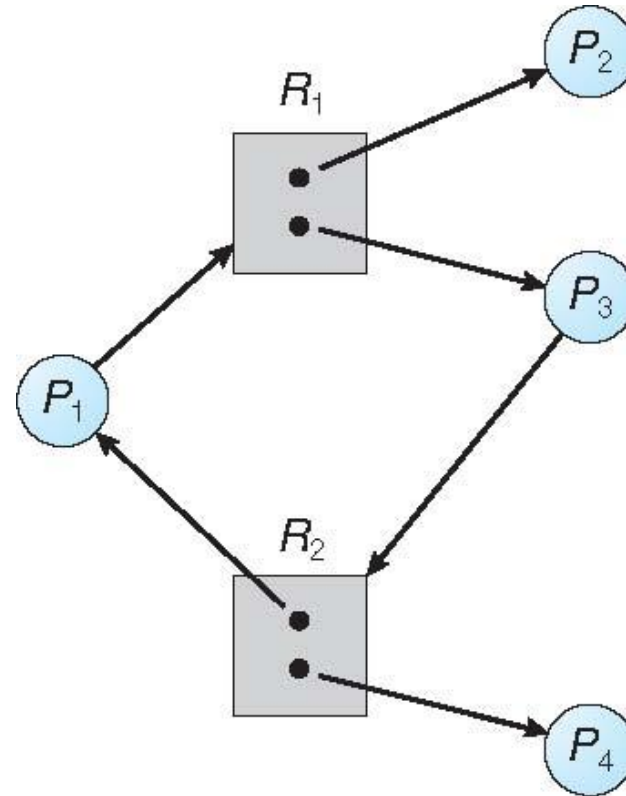
- ▶ Resource allocation modeled by directed graphs
- ▶ Example 1:
 - ▶ Resource R assigned to process A
- ▶ Example 2:
 - ▶ Process B is requesting / waiting for resource S
- ▶ Example 3:
 - ▶ Process C holds T, waiting for U
 - ▶ Process D holds U, waiting for T
 - ▶ C and D are in deadlock!



Graph With A deadlock



Graph With A Cycle But No Deadlock



Basic Facts

- ▶ If graph contains no cycles \Rightarrow no deadlock
- ▶ If graph contains a cycle \Rightarrow
 - ▶ if only one instance per resource type, then deadlock
 - ▶ if several instances per resource type, possibility of deadlock

Methods for Handling Deadlocks

1. Deadlock prevention
2. Deadlock avoidance
3. Deadlock detection and recovery
4. Deadlock Ignorance

1. Deadlock prevention

- ▶ This strategy involves designing a system that violates one of the four necessary conditions required for the occurrence of deadlock.
- ▶ This ensures that the system remains free from the deadlock.
- ▶ **The various conditions of deadlock occurrence may be violated as-**

1. Mutual Exclusion-

To violate this condition, all the system resources must be such that they can be used in a shareable mode.

- In a system, there are always some resources which are mutually exclusive by nature.
- So, this condition can not be violated.

Deadlock prevention

2. Hold and Wait-

This condition can be violated in the following ways-

Approach-01:

In this approach,

- A process has to first request for all the resources it requires for execution.
- Once it has acquired all the resources, only then it can start its execution.
- This approach ensures that the process does not hold some resources and wait for other resources.

Drawbacks-

The drawbacks of this approach are-

- It is less efficient.
- It is not implementable since it is not possible to predict in advance which resources will be required during execution.

Deadlock prevention

Approach-02:

In this approach,

- A process is allowed to acquire the resources it desires at the current moment.
- After acquiring the resources, it starts its execution.
- Now before making any new request, it has to compulsorily release all the resources that it holds currently.
- This approach is efficient and implementable.

Approach-03:

In this approach,

- A timer is set after the process acquires any resource.
- After the timer expires, a process has to compulsorily release the resource.

Deadlock prevention

3. No Preemption-

- This condition can be violated by forceful preemption.
- Consider a process is holding some resources and request other resources that can not be immediately allocated to it.
- Then, by forcefully preempting the currently held resources, the condition can be violated.

Deadlock prevention

4. Circular Wait-

- This condition can be violated by not allowing the processes to wait for resources in a cyclic manner.
- To violate this condition, the following approach is followed-

Approach-

- A natural number is assigned to every resource.
- Each process is allowed to request for the resources either in only increasing or only decreasing order of the resource number.
- In case increasing order is followed, if a process requires a lesser number resource, then it must release all the resources having larger number and vice versa.
- This approach is the most practical approach and implementable.
- However, this approach may cause starvation but will never lead to deadlock.

Deadlock Prevention

- ▶ **Mutual Exclusion** - not required for sharable resources (e.g., read-only files); must hold for non-sharable resources
- ▶ **Hold and Wait** - must guarantee that whenever a process requests a resource, it does not hold any other resources
 - ▶ Require process to request and be allocated all its resources before it begins execution, or allow process to request resources only when the process has none allocated to it.
 - ▶ Low resource utilization; starvation possible

Deadlock Prevention (Cont.)

- ▶ **No Preemption** -
 - ▶ If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are released
 - ▶ Preempted resources are added to the list of resources for which the process is waiting
 - ▶ Process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting
- ▶ **Circular Wait** - impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration

Deadlock Avoidance

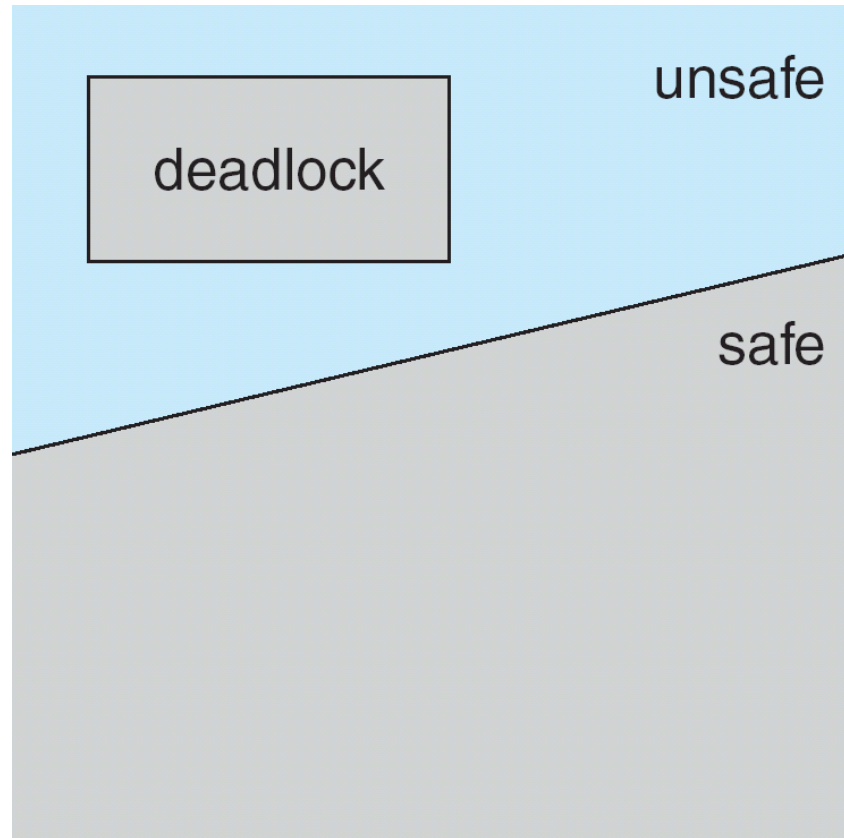
Requires that the system has some additional ***a priori*** information available

- ▶ Simplest and most useful model requires that each process declare the ***maximum number*** of resources of each type that it may need
- ▶ The deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that there can never be a circular-wait condition
- ▶ Resource-allocation *state* is defined by the number of available and allocated resources, and the maximum demands of the processes

Safe State

- ▶ When a process requests an available resource, system must decide if immediate allocation leaves the system in a safe state
- ▶ System is in **safe state** if there exists a sequence $\langle P_1, P_2, \dots, P_n \rangle$ of ALL the processes in the systems such that for each P_i , the resources that P_i can still request can be satisfied by currently available resources + resources held by all the P_j , with $j < i$
- ▶ That is:
 - ▶ If P_i resource needs are not immediately available, then P_i can wait until all P_j have finished
 - ▶ When P_j is finished, P_i can obtain needed resources, execute, return allocated resources, and terminate
 - ▶ When P_i terminates, P_{i+1} can obtain its needed resources, and so on

Safe, Unsafe, Deadlock State



Basic Facts

- ▶ If a system is in safe state \Rightarrow no deadlocks
- ▶ If a system is in unsafe state \Rightarrow possibility of deadlock
- ▶ Avoidance \Rightarrow ensure that a system will never enter an unsafe state.

Deadlock Avoidance Algorithms

1. Resource-allocation graph Algorithm:

for Single instance of a resource type

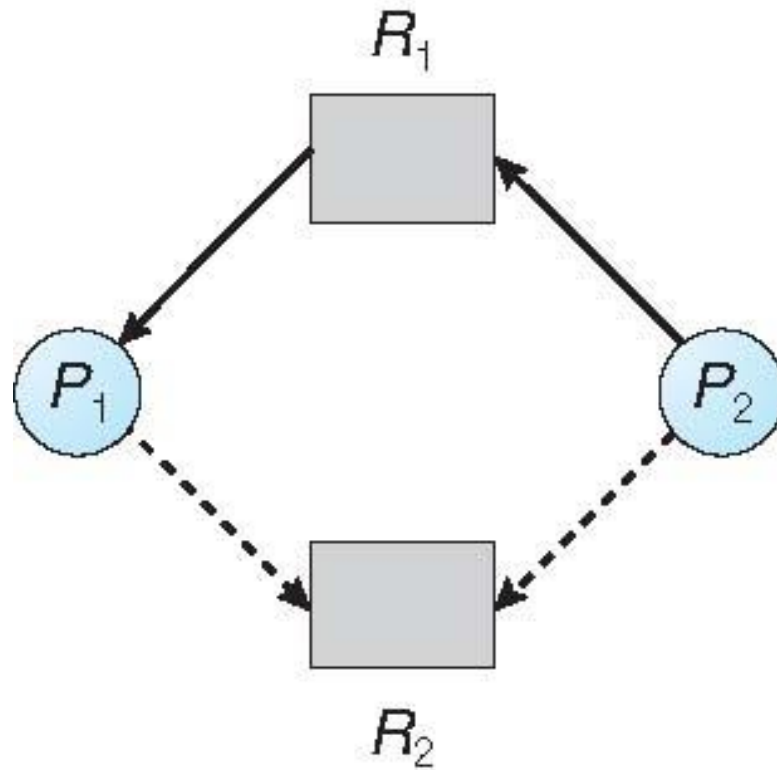
2. Banker's algorithm :

for Multiple instances of a resource type

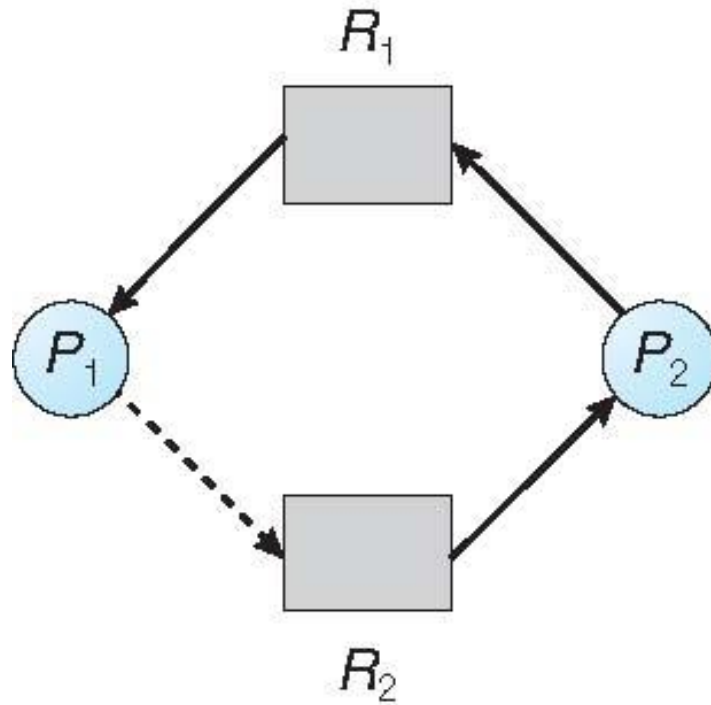
1. Resource-Allocation Graph Algorithm

- ▶ **Claim edge** $P_i \rightarrow R_j$ indicated that process P_j may request resource R_j in future; represented by a dashed line
- ▶ Claim edge converts to request edge when a process requests a resource
- ▶ **Request edge** converted to an **assignment edge** when the resource is allocated to the process
- ▶ When a resource is released by a process, assignment edge reconverts to a claim edge
- ▶ Resources must be claimed *a priori* in the system

Resource-Allocation Graph



Unsafe State In Resource-Allocation Graph



Resource-Allocation Graph Algorithm

- ▶ Suppose that process P_i requests a resource R_j
- ▶ The request can be granted only if converting the request edge to an assignment edge does not result in the formation of a cycle in the resource allocation graph

2. Banker's Algorithm

- ▶ Multiple instances
- ▶ Each process must a priori claim maximum use
- ▶ When a process requests a resource it may have to wait
- ▶ When a process gets all its resources it must return them in a finite amount of time

Data Structures for the Banker's Algorithm

Let n = number of processes, and m = number of resources types.

- ▶ **Available:** Vector of length m . If available $[j] = k$, there are k instances of resource type R_j available
- ▶ **Max:** $n \times m$ matrix. If $Max[i, j] = k$, then process P_i may request at most k instances of resource type R_j
- ▶ **Allocation:** $n \times m$ matrix. If $Allocation[i, j] = k$ then P_i is currently allocated k instances of R_j
- ▶ **Need:** $n \times m$ matrix. If $Need[i, j] = k$, then P_i may need k more instances of R_j to complete its task

$$Need[i, j] = Max[i, j] - Allocation[i, j]$$

Example of Banker's Algorithm

- 5 processes P_0 through P_4 ;

3 resource types:

A (10 instances), B (5 instances), and C (7 instances)

- Snapshot at time T_0 :

	<u>Allocation</u>	<u>Max</u>	<u>Available</u>
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

Example (Cont.)

- The content of the matrix *Need* is defined to be *Max - Allocation*

	<u>Need</u>
	A B C
P_0	7 4 3
P_1	1 2 2
P_2	6 0 0
P_3	0 1 1
P_4	4 3 1

- The system is in a safe state since the sequence $\langle P_1, P_3, P_4, P_2, P_0 \rangle$ satisfies safety criteria

Example: P_1 Request (1,0,2)

- Check that Request \leq Available (that is, $(1,0,2) \leq (3,3,2) \Rightarrow$ true

	<u>Allocation</u>	<u>Need</u>	<u>Available</u>
	A B C	A B C	A B C
P_0	0 1 0	7 4 3	2 3 0
P_1	3 0 2	0 2 0	
P_2	3 0 2	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

- Executing safety algorithm shows that sequence $\langle P_1, P_3, P_4, P_0, P_2 \rangle$ satisfies safety requirement
- Can request for (3,3,0) by P_4 be granted?
- Can request for (0,2,0) by P_0 be granted?

Deadlock detection and recovery

- ▶ Allow system to enter deadlock state
- ▶ The OS periodically checks if there is any existing deadlock in the system and take measures to remove the deadlocks.
- ▶ Detection algorithm
- ▶ Recovery scheme

Deadlock Detection

- ▶ There are 2 different cases in case of Deadlock detection -
 1. If resource has single Instance
 - ▶ Wait- For graph
 2. If resources have multiple instances
 - ▶ Deadlock Detection Algorithm

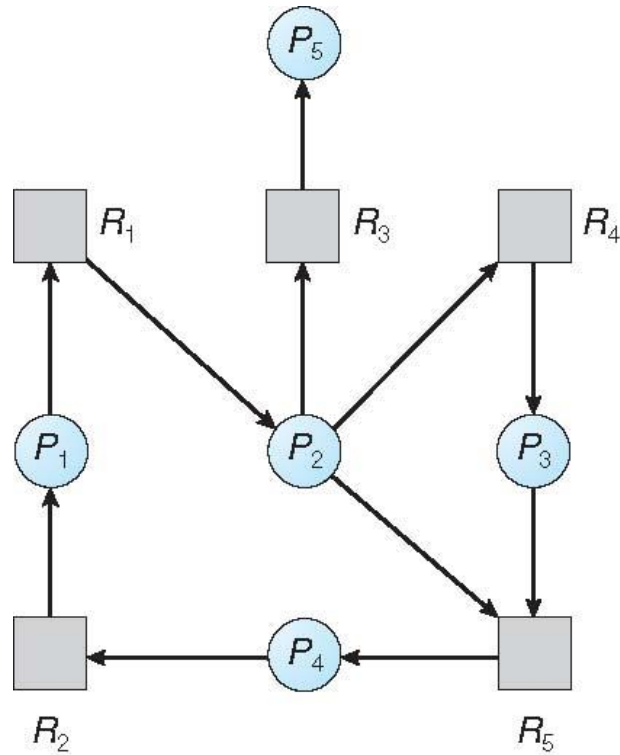
1. Single Instance of Each Resource Type

- ▶ Maintain **wait-for** graph
 - ▶ Nodes are processes
 - ▶ $P_i \rightarrow P_j$ if P_i is waiting for P_j
- ▶ Periodically invoke an algorithm that searches for a cycle in the graph. If there is a cycle, there exists a deadlock
- ▶ An algorithm to detect a cycle in a graph requires an order of n^2 operations, where n is the number of vertices in the graph

Wait for Graph:

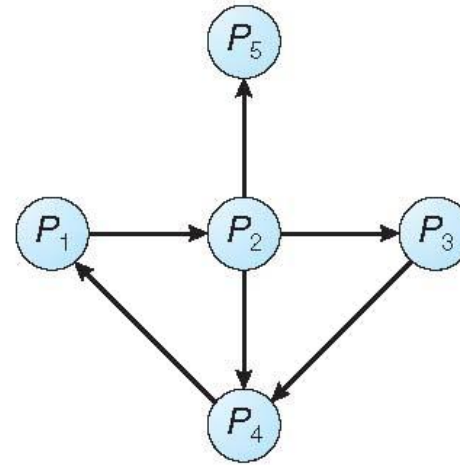
- ▶ If Resources has Single Instance a wait-for graph is made.
- ▶ A Wait-for graph vertex denotes process. The edge implies one process waiting for other Process to release resource.
- ▶ A deadlock is detected if one wait-for graph contains a cycle.
- ▶ Wait-for graph is made by looking at resource allocation graph.
- ▶ An edge between P1 to P2 exists if P1 needs some resource which P2 has.
- ▶ To detect cycle, system maintains the wait state of graph and periodically invoke an algorithm to detect cycle in graph.

Resource-Allocation Graph and Wait-for Graph



(a)

Resource-Allocation Graph



(b)

Corresponding wait-for graph

2. Several Instances of a Resource Type

- ▶ By multiple instance we mean one resource may allow 2 or more accesses concurrently, in such cases Wait for graph is not applicable.
- ▶ In this, a new algorithm is used. It's a bit similar to Banker's Algorithm, but Bankers Algorithm is different.
- ▶ It too uses 3 data structures -
 - ▶ Available
 - ▶ Vector of length m
 - ▶ Indicates number of available resources of each type.
 - ▶ Allocation
 - ▶ Matrix of size $n \times m$
 - ▶ $A[i,j]$ indicates the number of j th resource type allocated to i th process.
 - ▶ Request
 - ▶ Matrix of size $n \times m$
 - ▶ Indicates request of each process.
 - ▶ $Request[i,j]$ tells number of instance P_i process is request of j th resource type.

Detection Algorithm

1. Let **Work** and **Finish** be vectors of length m and n , respectively
Initialize:

(a) **Work** = **Available**

(b) For $i = 1, 2, \dots, n$, if $\text{Allocation}_i \neq 0$, then
Finish[i] = **false**; otherwise, **Finish**[i] = **true**

2. Find an index i such that both:

(a) **Finish**[i] == **false**

(b) $\text{Request}_i \leq \text{Work}$

If no such i exists, go to step 4

Detection Algorithm (Cont.)

3. $Work = Work + Allocation_i$
 $Finish[i] = true$
go to step 2
4. If $Finish[i] == false$, for some i , $1 \leq i \leq n$, then the system is in deadlock state. Moreover, if $Finish[i] == false$, then P_i is deadlocked

Algorithm requires an order of $O(m \times n^2)$ operations to detect whether the system is in deadlocked state

Example of Detection Algorithm

- Five processes P_0 through P_4 ; three resource types A (7 instances), B (2 instances), and C (6 instances)

- Snapshot at time T_0 :

	<u>Allocation</u>	<u>Request</u>	<u>Available</u>
	A B C	A B C	A B C
P_0	0 1 0	0 0 0	0 0 0
P_1	2 0 0	2 0 2	
P_2	3 0 3	0 0 0	
P_3	2 1 1	1 0 0	
P_4	0 0 2	0 0 2	

- Sequence $\langle P_0, P_2, P_3, P_1, P_4 \rangle$ will result in $Finish[i] = true$ for all i

Example (Cont.)

- ▶ P_2 requests an additional instance of type C

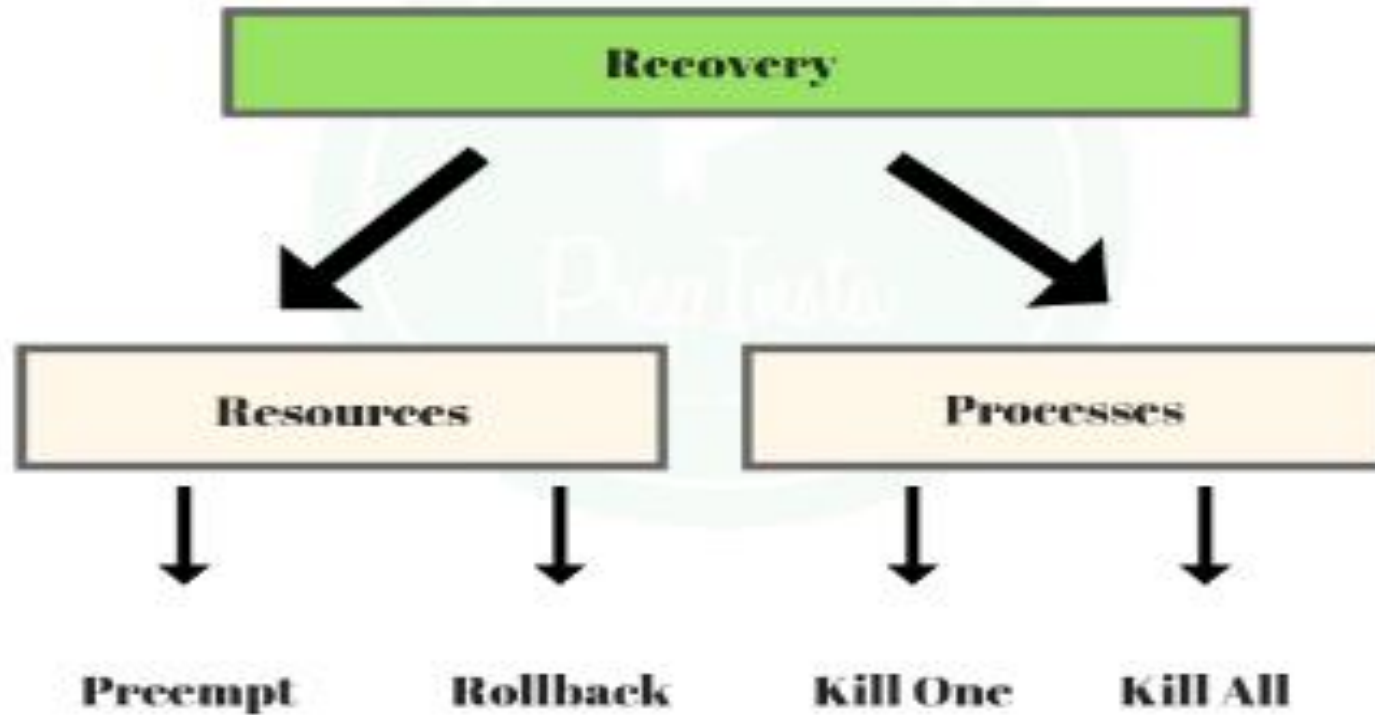
	<u>Request</u>		
	A	B	C
P_0	0	0	0
P_1	2	0	2
P_2	0	0	1
P_3	1	0	0
P_4	0	0	2

- ▶ State of system?
 - ▶ Can reclaim resources held by process P_0 , but insufficient resources to fulfill other processes; requests
 - ▶ Deadlock exists, consisting of processes P_1 , P_2 , P_3 , and P_4

Detection-Algorithm Usage

- ▶ When, and how often, to invoke depends on:
 - ▶ How often a deadlock is likely to occur?
 - ▶ How many processes will need to be rolled back?
 - ▶ one for each disjoint cycle
- ▶ If detection algorithm is invoked arbitrarily, there may be many cycles in the resource graph and so we would not be able to tell which of the many deadlocked processes “caused” the deadlock.

Deadlock Recovery in OS



Deadlock Recovery

- ▶ Deadlock can be recovered by:
- ▶ **Kill the Process** - One way is to kill all the process in deadlock or the second way kill the process one by one, and check after each if still deadlock exists and do the same till the deadlock is removed.
- ▶ **Preemption** - The resources that are allocated to the processes involved in deadlock are taken away(preempted) and are transferred to other processes. In this way, system may recover from deadlock as we may change system state.
- ▶ **Rollback** - The OS maintains a database of all different states of system, a state when the system is not in deadlock is called safe state. A rollback to previous 'n' number of safe states in iterations can help in the recover.

Recovery from Deadlock: Process Termination

- ▶ Abort all deadlocked processes
- ▶ Abort one process at a time until the deadlock cycle is eliminated
- ▶ In which order should we choose to abort?
 1. Priority of the process
 2. How long process has computed, and how much longer to completion
 3. Resources the process has used
 4. Resources process needs to complete
 5. How many processes will need to be terminated
 6. Is process interactive or batch?

Recovery from Deadlock: Resource Preemption

- ▶ **Selecting a victim** - minimize cost
- ▶ **Rollback** - return to some safe state, restart process for that state
- ▶ **Starvation** - same process may always be picked as victim, include number of rollback in cost factor

Deadlock Ignorance

- ▶ This strategy involves ignoring the concept of deadlock and assuming as if it does not exist.
- ▶ This strategy helps to avoid the extra overhead of handling deadlock.
- ▶ Windows and Linux use this strategy and it is the most widely used method.
- ▶ It is also called as **Ostrich approach**.

Device Management

Content

- ▶ **Device management:**
- ▶ devices and their characteristics,
- ▶ device drivers,
- ▶ device handling,
- ▶ disk scheduling algorithms and policies

Device Management Functions

- ▶ Track status of each device (such as tape drives, disk drives, printers, plotters, and terminals).
- ▶ Use preset policies to determine which process will get a device and for how long.
- ▶ Allocate the devices.
- ▶ Deallocate the devices at 2 levels:
 - ▶ At process level when I/O command has been executed & device is temporarily released
 - ▶ At job level when job is finished & device is permanently released.

Types of Devices

► Three categories:

1. Dedicated,
2. Shared, and
3. Virtual

Dedicated Devices

- ▶ Assigned to only one job at a time and serve that job for entire time it's active.
 - ▶ E.g., tape drives, printers, and plotters, demand this kind of allocation scheme, because it would be awkward to share.
- ▶ Disadvantage -- must be allocated to a single user for duration of a job's execution.
 - ▶ Can be quite inefficient, especially when device isn't used 100 % of time.

Shared Devices

- ▶ Assigned to several processes.
 - ▶ E.g., disk pack (or other direct access storage device) can be shared by several processes at same time by interleaving their requests.
- ▶ Interleaving must be carefully controlled by Device Manager.
- ▶ All conflicts must be resolved based on predetermined policies to decide which request will be handled first.

Virtual Devices

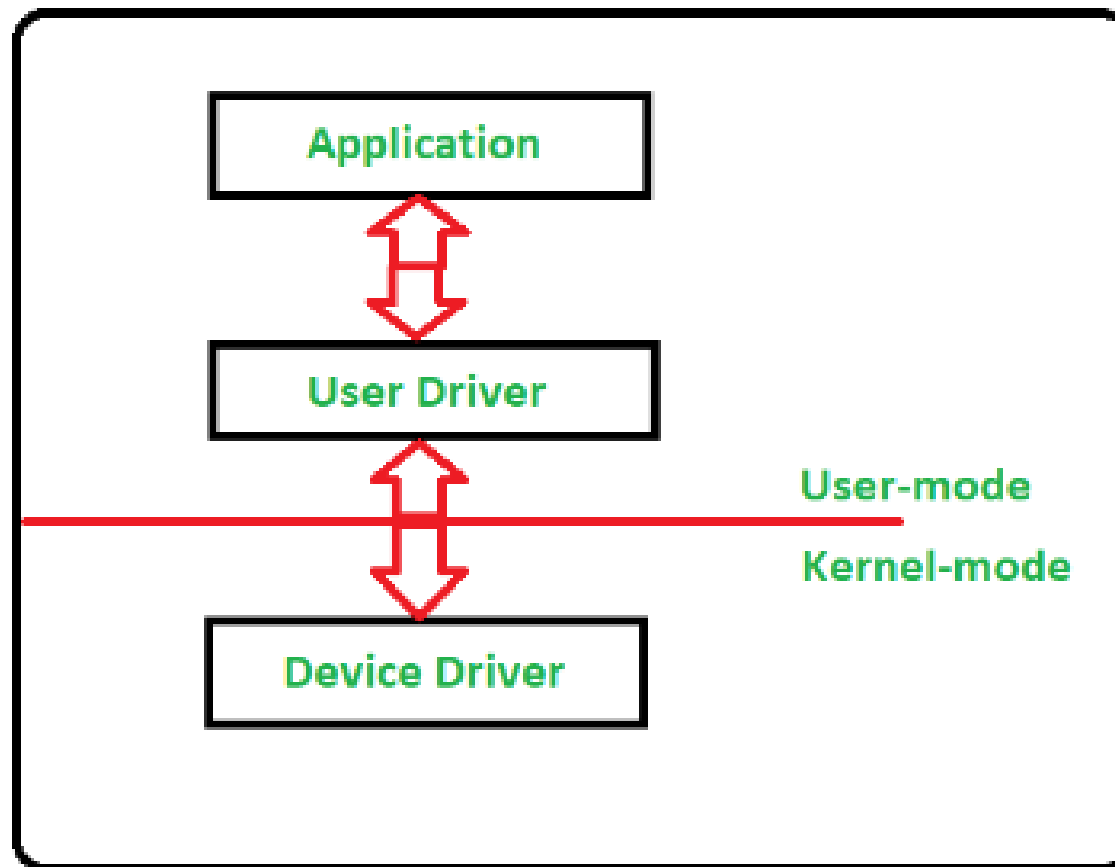
- ▶ Combination of dedicated devices that have been transformed into shared devices.
 - ▶ E.g, printers are converted into sharable devices through a spooling program that reroutes all print requests to a disk.
 - ▶ Output sent to printer for printing only when all of a job's output is complete and printer is ready to print out entire document.
 - ▶ Because disks are sharable devices, this technique can convert one printer into several “virtual” printers, thus improving both its performance and use.

Device Driver

- ▶ More commonly known as a driver, a device driver or hardware driver
- ▶ It is a group of files that enable one or more hardware devices to communicate with the computer's operating system.
- ▶ Without drivers, the computer would not be able to send and receive data correctly to hardware devices, such as a printer.

Device Driver

- ▶ **Device Drivers** are very essential for a computer system to work properly because without device driver the particular hardware fails to work accordingly means it fails in doing a particular function/action for which it has been created.



What devices need drivers?

- ▶ Hardware devices that are unknown by the operating system or that have features that are unknown by the operating system all require drivers.
- ▶ Below is a list of hardware devices and peripherals that require drivers.
- ▶ Card reader
- ▶ Controller
- ▶ Modem
- ▶ Motherboard chipset
- ▶ Network card
- ▶ Printer
- ▶ Scanner
- ▶ Sound card
- ▶ Tape drive
- ▶ USB devices
- ▶ Video card

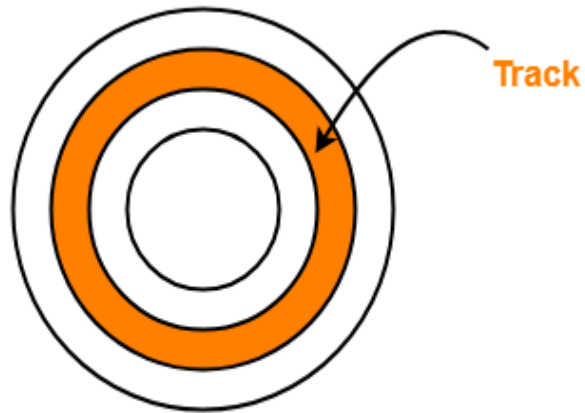
What devices may not need drivers?

- ▶ Today's operating systems have a lot of generic drivers that allow hardware to work at a basic level without needing drivers or software.
- ▶ However, if that device has features unknown to the operating system, it will not work without drivers.
- ▶ For example, you could plug any keyboard into a computer and expect it to work. However, if that keyboard has any special keys or features, they will not work until the drivers are installed.
- ▶ CPU, Disc drive, Fan, Hard drive, Heat sink, Joystick, Keyboard, Mouse, Monitor, Power supply, RAM, Speakers, UPS

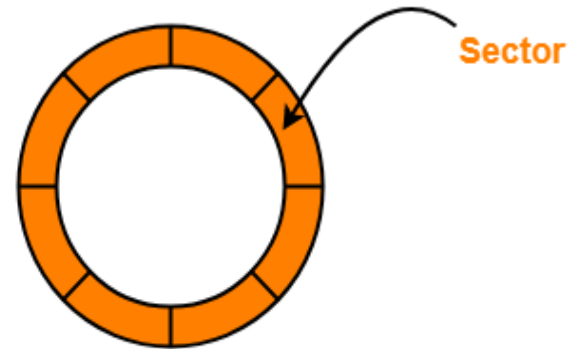
Disk Scheduling Algorithms and Policies

- The algorithms used for disk scheduling are called as **disk scheduling algorithms**.
 - The purpose of disk scheduling algorithms is to reduce the total seek time.
1. FCFS Algorithm
 2. SSTF Algorithm
 3. SCAN Algorithm
 4. C-SCAN Algorithm
 5. LOOK Algorithm
 6. C-LOOK Algorithm

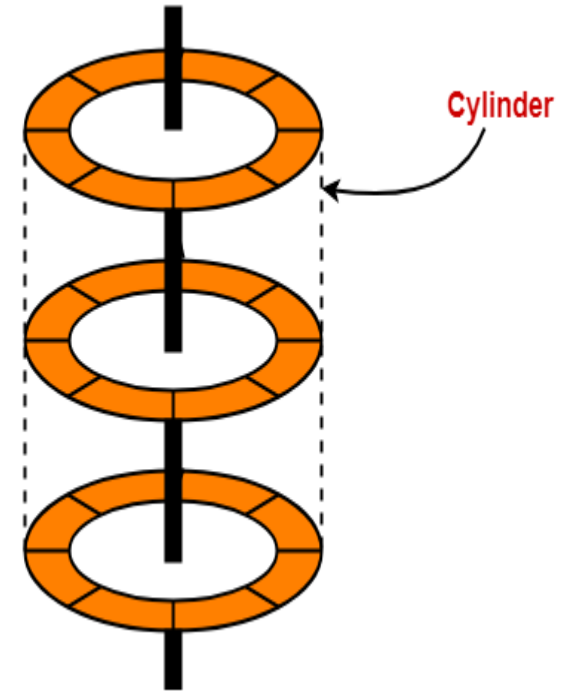
Disk Architecture



Disk divided into tracks



Track divided into sectors



Disk Performance Parameters

1. Seek Time-

- The time taken by the read / write head to reach the desired track is called as **seek time**.
- It is the component which contributes the largest percentage of the disk service time.
- The lower the seek time, the faster the I/O operation.

2. Rotational Latency-

- The time taken by the desired sector to come under the read / write head is called as **rotational latency**.
- It depends on the rotation speed of the spindle.

3. Data Transfer Rate-

- The amount of data that passes under the read / write head in a given amount of time is called as **data transfer rate**.
- The time taken to transfer the data is called as **transfer time**

► **Disk Access Time**

Disk access time = Seek time + Rotational Latency + Transfer Rate

1. FCFS Disk Scheduling Algorithm

- As the name suggests, this algorithm entertains requests in the order they arrive in the disk queue.
- It is the simplest disk scheduling algorithm.

► Advantages-

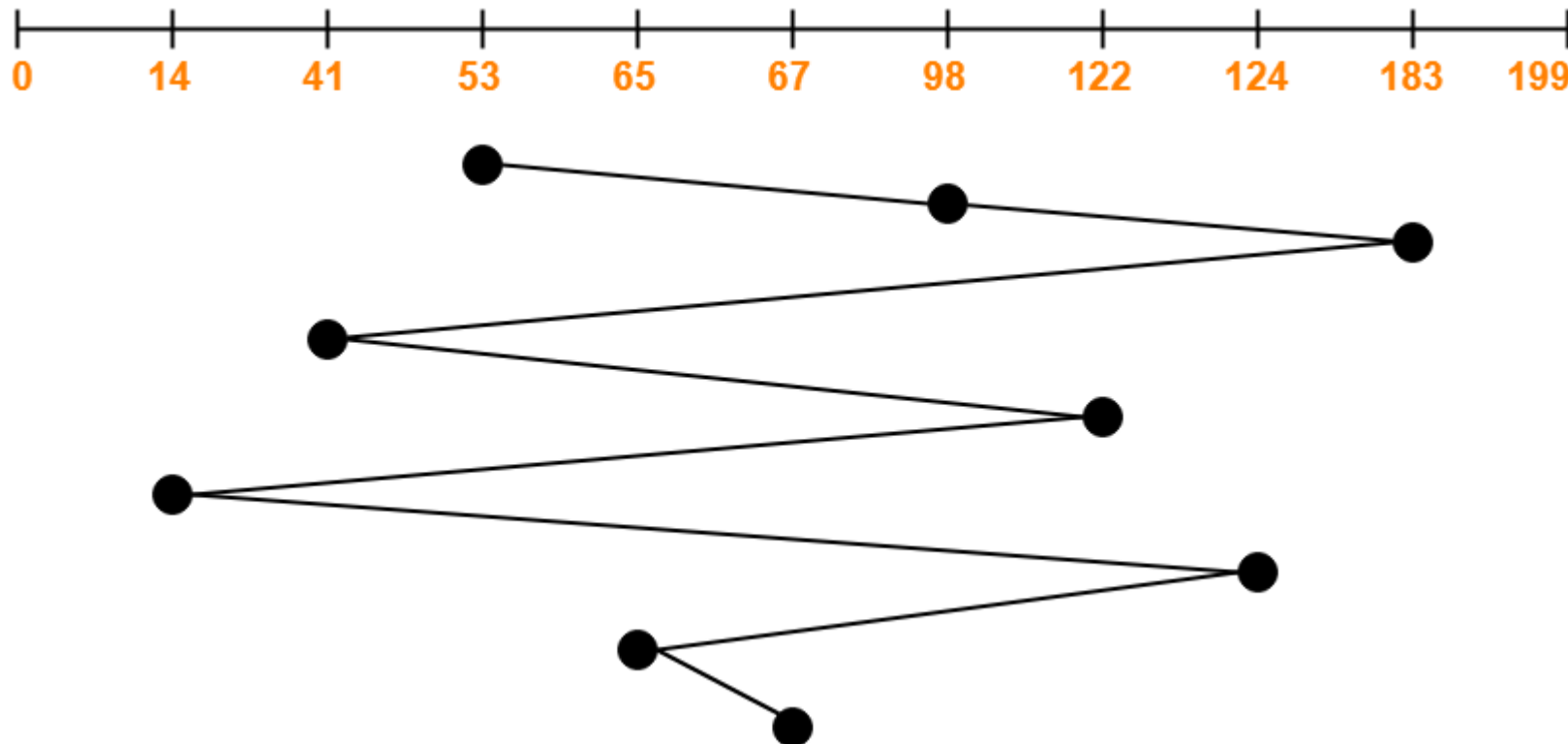
- It is simple, easy to understand and implement.
- It does not cause starvation to any request.

► Disadvantages-

- It results in increased total seek time.
- It is inefficient.

Example

- **Problem-** Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The FCFS scheduling algorithm is used. The head is initially at cylinder number 53. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is



Total head movements incurred while servicing these requests

$$= (98 - 53) + (183 - 98) + (183 - 41) + (122 - 41) + (122 - 14) + (124 - 14) + (124 - 65) + (67 - 65)$$

$$= 45 + 85 + 142 + 81 + 108 + 110 + 59 + 2 = 632$$

2. SSTF Disk Scheduling Algorithm

- SSTF stands for **Shortest Seek Time First**.
- This algorithm services that request next which requires least number of head movements from its current position regardless of the direction.
- It breaks the tie in the direction of head movement.

► Advantages-

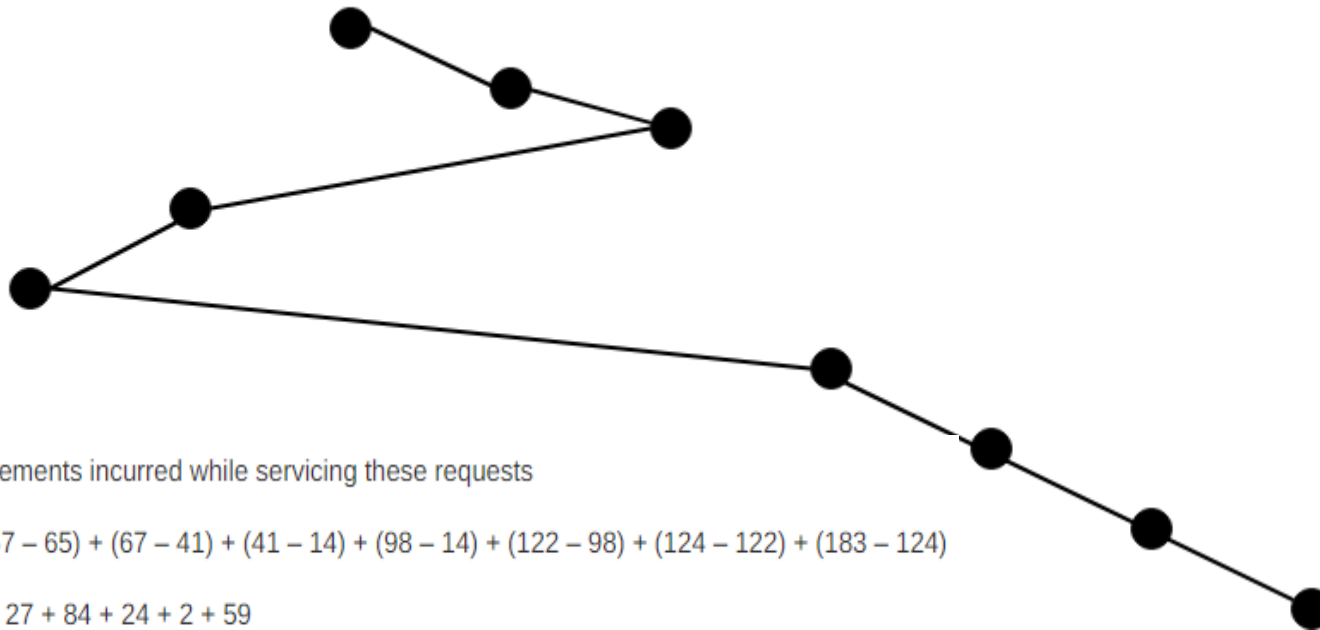
- It reduces the total seek time as compared to **FCFS**.
- It provides increased throughput.
- It provides less average response time and waiting time.

► Disadvantages-

- There is an overhead of finding out the closest request.
- The requests which are far from the head might starve for the CPU.
- It provides high variance in response time and waiting time.
- Switching the direction of head frequently slows down the algorithm.

Example

- **Problem:** Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The SSTF scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is _____.



Total head movements incurred while servicing these requests

$$= (65 - 53) + (67 - 65) + (67 - 41) + (41 - 14) + (98 - 14) + (122 - 98) + (124 - 122) + (183 - 124)$$

$$= 12 + 2 + 26 + 27 + 84 + 24 + 2 + 59$$

$$= 236$$

3. SCAN Disk Scheduling Algorithm

- As the name suggests, this algorithm scans all the cylinders of the disk back and forth.
- Head starts from one end of the disk and move towards the other end servicing all the requests in between.
- After reaching the other end, head reverses its direction and move towards the starting end servicing all the requests in between.
- The same process repeats.
- SCAN Algorithm is also called as **Elevator Algorithm**.
- This is because its working resembles the working of an elevator.

► **Advantages-**

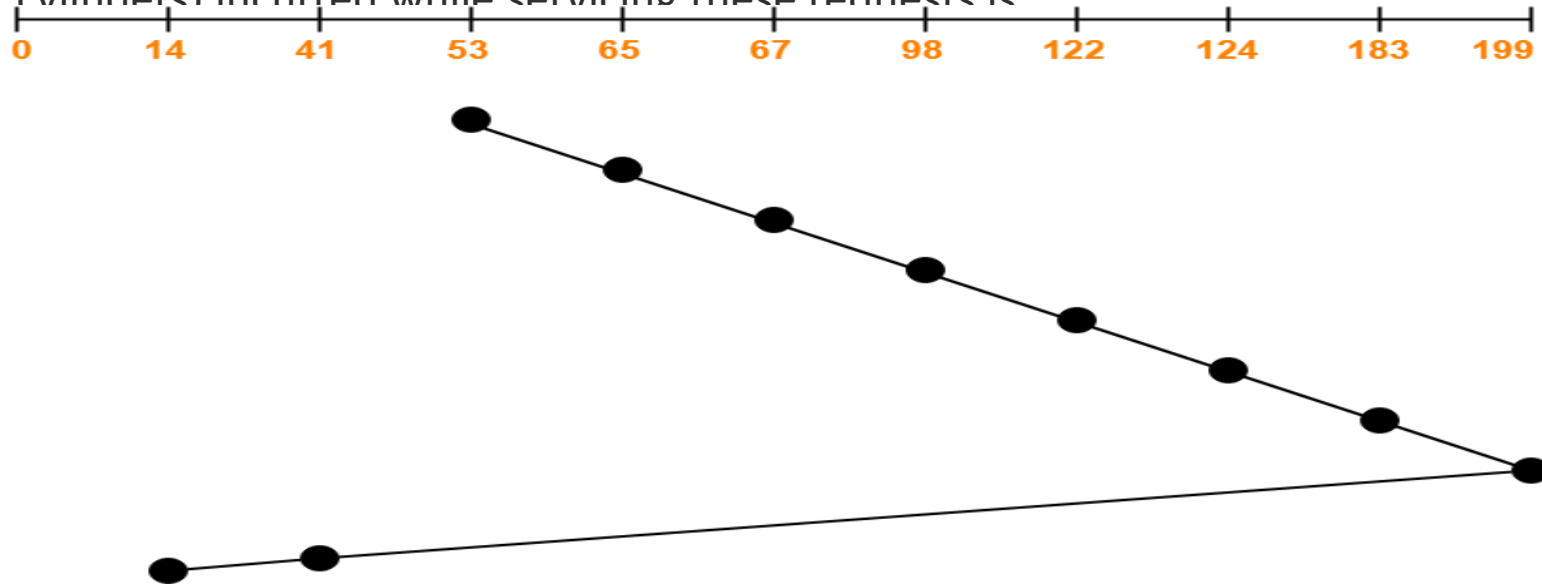
- It is simple, easy to understand and implement.
- It does not lead to starvation.
- It provides low variance in response time and waiting time.

► **Disadvantages-**

- It causes long waiting time for the cylinders just visited by the head.
- It causes the head to move till the end of the disk even if there are no requests to be serviced.

Example

- **Problem-** Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The SCAN scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is



Total head movements incurred while servicing these requests

$$= (199 - 53) + (199 - 14)$$

$$= 146 + 185 = 331$$

4. C-SCAN Disk Scheduling Algorithm

- Circular-SCAN Algorithm is an improved version of the SCAN Algorithm.
- Head starts from one end of the disk and move towards the other end servicing all the requests in between.
- After reaching the other end, head reverses its direction.
- It then returns to the starting end without servicing any request in between.
- The same process repeats.

► **Advantages-**

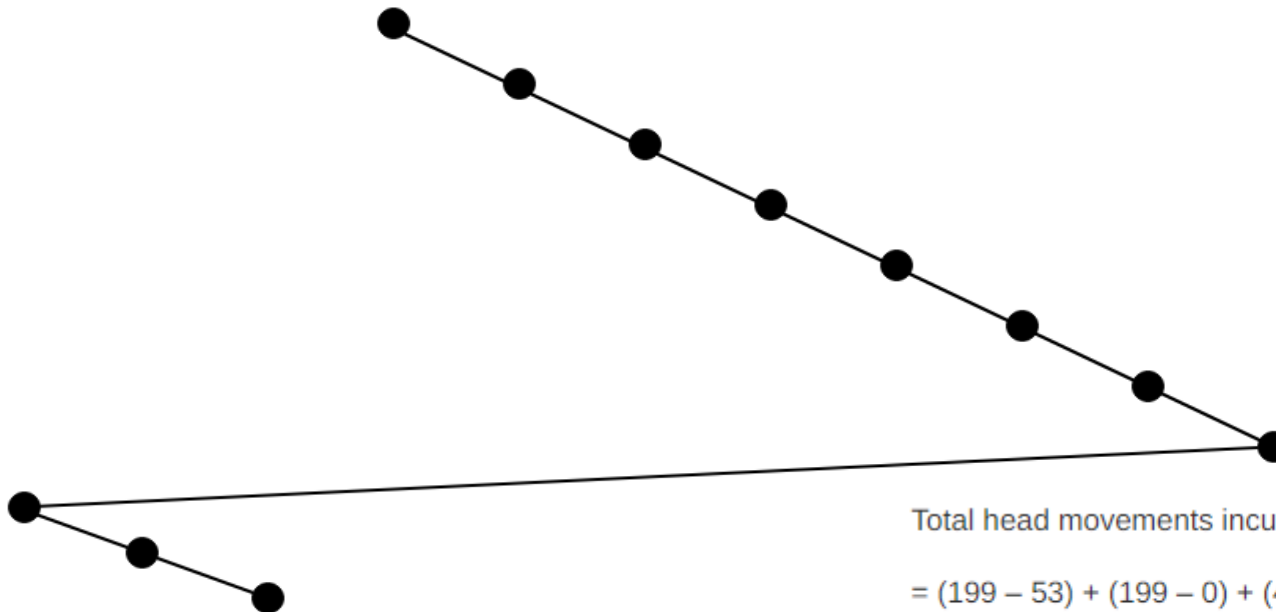
- The waiting time for the cylinders just visited by the head is reduced as compared to the SCAN Algorithm.
- It provides uniform waiting time.
- It provides better response time.

► **Disadvantages-**

- It causes more seek movements as compared to SCAN Algorithm.
- It causes the head to move till the end of the disk even if there are no requests to be serviced.

Example

- **Problem-** Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The C-SCAN scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is



Total head movements incurred while servicing these requests

$$= (199 - 53) + (199 - 0) + (41 - 0)$$

$$= 146 + 199 + 41 = 386$$

5. LOOK Disk Scheduling Algorithm

- LOOK Algorithm is an improved version of the SCAN Algorithm.
- Head starts from the first request at one end of the disk and moves towards the last request at the other end servicing all the requests in between.
- After reaching the last request at the other end, head reverses its direction.
- It then returns to the first request at the starting end servicing all the requests in between.
- The same process repeats.

- ▶ The main difference between SCAN Algorithm and LOOK Algorithm is-
 - SCAN Algorithm scans all the cylinders of the disk starting from one end to the other end even if there are no requests at the ends.
 - LOOK Algorithm scans all the cylinders of the disk starting from the first request at one end to the last request at the other end.

► **Advantages-**

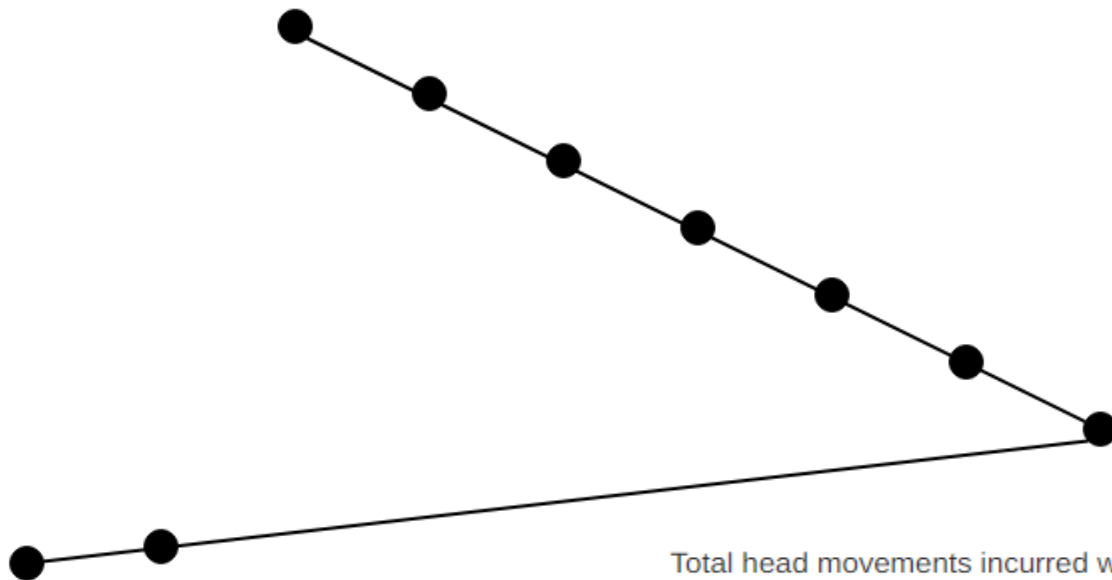
- It does not causes the head to move till the ends of the disk when there are no requests to be serviced.
- It provides better performance as compared to SCAN Algorithm.
- It does not lead to starvation.
- It provides low variance in response time and waiting time.

► **Disadvantages-**

- There is an overhead of finding the end requests.
- It causes long waiting time for the cylinders just visited by the head.

Example

- **Problem-** Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The LOOK scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is _____.



Total head movements incurred while servicing these requests

$$= (183 - 53) + (183 - 14)$$

$$= 130 + 169 = 299$$

6. C-LOOK Disk Scheduling Algorithm

- Circular-LOOK Algorithm is an improved version of the LOOK Algorithm.
- Head starts from the first request at one end of the disk and moves towards the last request at the other end servicing all the requests in between.
- After reaching the last request at the other end, head reverses its direction.
- It then returns to the first request at the starting end without servicing any request in between.
- The same process repeats.

► **Advantages-**

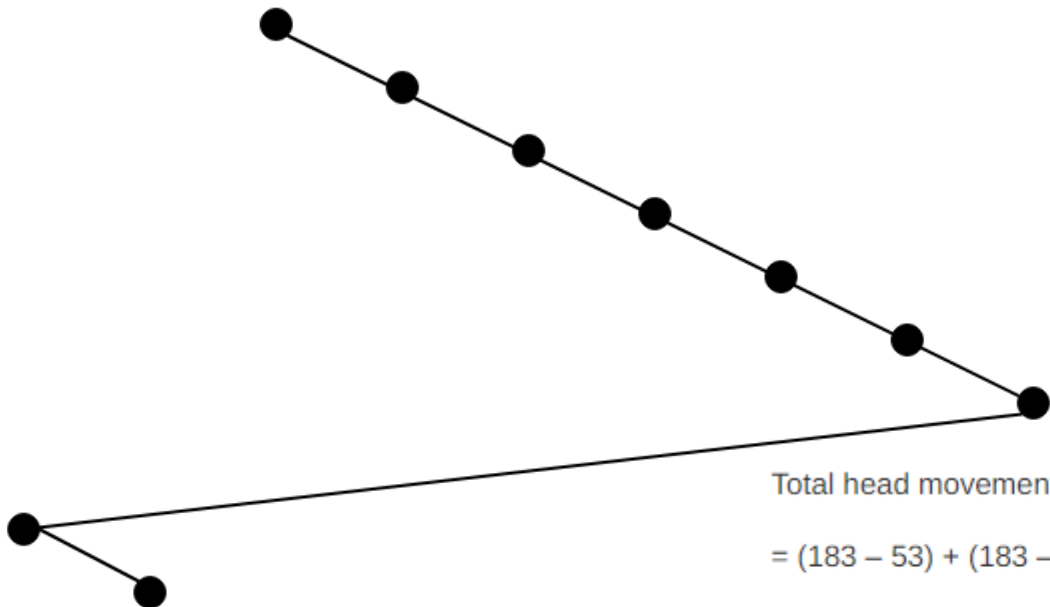
- It does not causes the head to move till the ends of the disk when there are no requests to be serviced.
- It reduces the waiting time for the cylinders just visited by the head.
- It provides better performance as compared to LOOK Algorithm.
- It does not lead to starvation.
- It provides low variance in response time and waiting time.

► **Disadvantages-**

- There is an overhead of finding the end requests.

Example

- **Problem:** Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 41, 122, 14, 124, 65, 67. The C-LOOK scheduling algorithm is used. The head is initially at cylinder number 53 moving towards larger cylinder numbers on its servicing pass. The cylinders are numbered from 0 to 199. The total head movement (in number of cylinders) incurred while servicing these requests is _____.



Total head movements incurred while servicing these requests

$$= (183 - 53) + (183 - 14) + (41 - 14)$$

$$= 130 + 169 + 27$$

$$= 326$$

I Hear and I Forget.
I See and I Remember.
I Do and I Understand.



Good Luck



5CS4-03 OPERATING SYSTEM

Unit-V

Case Study On:

- Unix, Linux
- Mobile OS
- RTOS

UNIX and Linux operating systems as case studies

What is LINUX?

It is an operating system at the top. It acts as an interface between the user and the machine to perform specific task. The user performs some task in terms of input and this OS redirects the instruction to the machine in machine language. After performing the specific task it sends the computed task back to the user as an output. Well, all the operating system performs this task. Now, we shall study more about the LINUX operating system. LINUX means “GNU/LINUX” because LINUX is almost always used with the GNU tools.

LINUX is very similar to other operating systems like windows and OS X. There are certain measures that categories this operating system among others. It is widely used operating system apart from Windows. This operating system is run over various platforms from Pc's to cellular phones. Even supercomputers runs on LINUX.

The important thing under consideration of the LINUX is that it is open source operating system. Open source simply means that its source code is available to all the users and it can be modified. No one individually owns LINUX. The competition to impart much to this software adds much to this operating system. About 1,000 developers across 100 different companies contribute to different kernels of this software. Initially it was made for server as an operating system. But this feature of open source has made LINUX to be used in any of the device in modern time. It do not include executable (.exe) files hence it is all free from viruses and hacking free software. This is the only reason that it is used in all of the major companies where security of database is highly important.

What is UNIX?

The UNIX dates back to 1969. Since then, it has changed a lot and various versions are out till today. It has built its versions for many platforms and different environments. It is the previous version of LINUX. The full form of UNIX is UNiplexed Information Computing System (UNICS), which was later termed as UNIX. This name has been given because the UNIX was

first developed in 1970s; it required the data and addresses buses to be uniplexed. They were not multiplexed, hence it was named so.

What are the differences between LINUX and UNIX?

1. The developers of UNIX have a specific target audience and platform for their operating system. They have a clear idea what applications they have to optimize to give users the best results. Commercial UNIX vendors can do anything since they can maintain consistency between different versions of UNIX.

Whereas the development of GNU/LINUX, is more diverse as compared to UNIX. Here the developers come from different backgrounds and thus have different opinions and thinking. There are no specific tools and restrictions with the LINUX. The tools can be used on new editions without much testing.

2. Since the kernel is the most important part of any operating system. The source code is not freely available for any of the commercial version of UNIX. Whereas in the case of LINUX; it is freely available. Both the terms have a different compiling and patching kernels and drivers. With Linux and other open source operating systems, a patch can be released in source code form and end users can install it, or even verify and modify it if desired. These patches tend to be far less tested than patches from UNIX vendors. Since there is not a complete list of applications and environments that need to be tested on Linux, the Linux developers have to depend on the many eyes of end users and other developers to catch errors.

Henceforth we can say that both these operating systems works fine in their fields and due to the lack of (.exe) executable files it becomes virus less platform to work with. The security issues are less in terms of hacking the database

Linux History

Linux looks and feels much like any other UNIX system; indeed, UNIX compatibility has been a major design goal of the Linux project. However, Linux is much younger than most UNIX systems. Its development began in 1991, when a Finnish university student, Linus Torvalds,

began developing a small but self-contained kernel for the 80386 processor, the first true 32-bit processor in Intel's range of PC-compatible CPUs.

History of UNIX and Linux:

- UNICS
- PDP-11 UNIX
- Portable UNIX
- Berkeley UNIX
- Standard UNIX
- MINIX
- Linux

Before Linux

- In 80's, Microsoft's DOS was the dominated OS for PC
- single-user, single-process system
- Apple MAC is better, but expensive
- UNIX is much better, but much much expensive. Only for minicomputer for commercial applications
- People was looking for a UNIX based system, which is cheaper and can run on PC
- Both DOS, MAC and UNIX are proprietary, i.e., the source code of their kernel is protected
- No modification is possible without paying high license fees

GNU project

- Established in 1984 by Richard Stallman, who believes that software should be free from restrictions against copying or modification in order to make better and efficient computer programs

- GNU is a recursive acronym for “GNU's Not Unix”
- Aim at developing a complete Unix-like operating system which is free for copying and modification
- Companies make their money by maintaining and distributing the software, e.g. optimally packaging the software with different tools (Redhat, Slackware, Mandrake, SuSE, etc)
- Stallman built the first free GNU C Compiler in 1991. But still, an OS was yet to be developed

Beginning of Linux

- A famous professor **Andrew Tanenbaum** developed **Minix**, a simplified version of UNIX that runs on PC
- Minix is for class teaching only. No intention for commercial use
- In Sept 1991, **Linus Torvalds**, a second year student of Computer Science at the University of Helsinki, developed the preliminary kernel of Linux, known as Linux version 0.0.1
- It was put to the Internet and received enormous response from worldwide software developers
- By December came version 0.10. Still Linux was little more than in skeletal form.
- It was licensed under GNU General Public License, thus ensuring that the source codes will be free for all to copy, study and to change

Linux Pros and Cons

Advantages over Windows

- It's almost free to relatively inexpensive
- Source code is included
- Bugs are fixed quickly and help is readily available through the vast support in Internet
- Linux is more stable than Windows
- Linux is truly multi-user and multi-tasking

- multiuser: OS that can simultaneously serve a number of users
- multitasking: OS that can simultaneously execute a number of programs
- Linux runs on equipment that other operating systems consider too underpowered, e.g. 386 systems, PDA, etc

Disadvantages compared with Windows

- Isn't as popular as Windows
- No one commercial company is responsible for Linux
- Linux is relatively hard to install, learn and use

Hence currently, Linux is mainly used in commercial applications, server implementation

More than 75% current network servers are developed based on Linux or UNIX systems

Linux System Architecture: Components of a Linux System

The Linux system is composed of three main bodies of code, in line with most traditional UNIX implementations:

1. **Kernel.** The kernel is responsible for maintaining all the important abstractions of the operating system, including such things as virtual memory and processes.
2. **System libraries.** The system libraries define a standard set of functions through which applications can interact with the kernel. These functions implement much of the operating-system functionality that does not need the full privileges of kernel code. The most important system library is the C library, known as libc. In addition to providing the standard C library, libc implements the user mode side of the Linux system call interface, as well as other critical system-level interfaces.
3. **System utilities.** The system utilities are programs that perform individual, specialized management tasks. Some system utilities are invoked just once to initialize and configure some aspect of the system. Others —known as daemons in UNIX terminology—run permanently, handling such tasks as responding to incoming network connections, accepting logon requests from terminals, and updating log files.

Following Figure illustrates the various components that make up a full Linux system. The most important distinction here is between the kernel and everything else. All the kernel code executes in the processor's privileged mode with full access to all the physical resources of the computer. Linux refers to this privileged mode as kernel mode. Under Linux, no user code is built into the kernel. Any operating-system-support code that does not need to run in kernel mode is placed into the system libraries and runs in user mode. Unlike kernel mode, user mode has access only to a controlled subset of the system's resources.

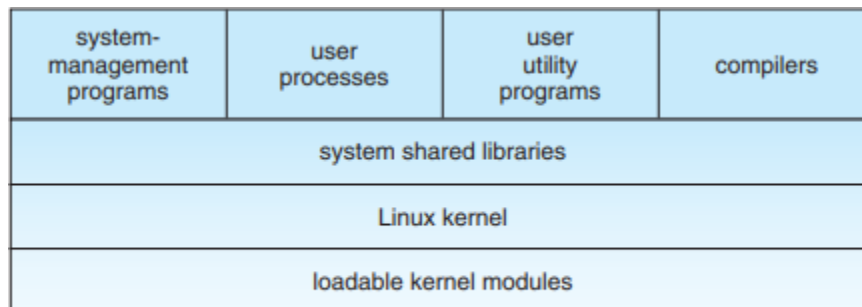


Figure Components of the Linux system.

Process Management

- For a multitask system, multiple programs can be executed simultaneously in the system
- When a program starts to execute, it becomes a process
- The same program executing at two different times will become two different processes
- Kernel manages processes in terms of creating, suspending, and terminating them
- A process is protected from other processes and can communicate with the others

The fork() and exec() Process Model:

The basic principle of UNIX process management is to separate into two steps two operations that are usually combined into one:

- the creation of a new process
- and the running of a new program.

A new process is created by the `fork()` system call, and a new program is run after a call to `exec()`.

These are two distinctly separate functions. We can create a new process with `fork()` without running a new program.

Any process may call `exec()` at any time. A new binary object is loaded into the process's address space and the new executable starts executing in the context of the existing process.

This model has the advantage of great simplicity. It is not necessary to specify every detail of the environment of a new program in the system call that runs that program. The new program simply runs in its existing environment.

If a parent process wishes to modify the environment in which a new program is to be run, it can fork and then, still running the original executable in a child process, make any system calls it requires to modify that child process before finally executing the new program.

Broadly, process properties fall into three groups:

- the process identity,
- environment,
- and context.

- **Process Identity**

A process identity consists mainly of the following items:

- Process ID (PID). Each process has a unique identifier. The PID is used to specify the process to the operating system when an application makes a system

call to signal, modify, or wait for the process. Additional identifiers associate the process with a process group (typically, a tree of processes forked by a single user command) and login session.

- **Credentials.** Each process must have an associated userID and one or more group IDs that determine the rights of a process to access system resources and files.
- **Personality.** Process personalities are not traditionally found on UNIX systems, but under Linux each process has an associated personality identifier that can slightly modify the semantics of certain system calls.
- **Namespace.** Each process is associated with a specific view of the filesystem hierarchy, called its namespace. Most processes share a common namespace and thus operate on a shared file-system hierarchy. Processes and their children can, however, have different namespaces, each with a unique file-system hierarchy—their own root directory and set of mounted file systems.

● **Process Environment**

A process's environment is inherited from its parent and is composed of two null-terminated vectors: the argument vector and the environment vector. The argument vector simply lists the command-line arguments used to invoke the running program; it conventionally starts with the name of the program itself.

The environment vector is a list of "NAME=VALUE" pairs that associates named environment variables with arbitrary textual values. The environment is not held in kernel memory but is stored in the process's own user-mode address space as the first datum at the top of the process's stack.

The argument and environment vectors are not altered when a new process is created. The new child process will inherit the environment of its parent.

However, a completely new environment is set up when a new program is invoked. On calling `exec()`, a process must supply the environment for the new program. The kernel passes these environment variables to the next program, replacing the process's current environment. The kernel otherwise leaves the environment and command-line vectors alone—their interpretation is left entirely to the user-mode libraries and applications.

- **Process Context**

The process identity and environment properties are usually set up when a process is created and not changed until that process exits. A process may choose to change some aspects of its identity if it needs to do so, or it may alter its environment. In contrast, process context is the state of the running program at any one time; it changes constantly.

Process context includes the following parts:

- **Scheduling context.** The most important part of the process context is its scheduling context—the information that the scheduler needs to suspend and restart the process. This information includes saved copies of all the process's registers. Floating-point registers are stored separately and are restored only when needed. Thus, processes that do not use floating-point arithmetic do not incur the overhead of saving that state. The scheduling context also includes information about scheduling priority and about any outstanding signals waiting to be delivered to the process

- Accounting. The kernel maintains accounting information about the resources currently being consumed by each process and the total resources consumed by the process in its entire lifetime so far.
- File table. The file table is an array of pointers to kernel file structures representing open files. When making file-I/O system calls, processes refer to files by an integer, known as a file descriptor (fd), that the kernel uses to index into this table.
- File-system context. Whereas the file table lists the existing open files, the file-system context applies to requests to open new files. The file-system context includes the process's root directory, current working directory, and namespace.
- Signal-handler table. UNIX systems can deliver asynchronous signals to a process in response to various external events. The signal-handler table defines the action to take in response to a specific signal. Valid actions include ignoring the signal, terminating the process, and invoking a routine in the process's address space.
- Virtual memory context. The virtual memory context describes the full contents of a process's private address space.

Process Scheduling

Linux has two separate process-scheduling algorithms.

One is a time-sharing algorithm for fair, preemptive scheduling among multiple processes.

The other is designed for real-time tasks, where absolute priorities are more important than fairness.

The Linux scheduler is a preemptive, priority-based algorithm with two separate priority ranges: a real-time range from 0 to 99 and a nice value ranging from -20 to 19. Smaller nice values indicate higher priorities. Thus, by increasing the nice value, you are decreasing your priority and being “nice” to the rest of the system.

Let us now see some more important differences between Linux and Unix in the below tabular format:

Features	Linux	Unix
Developer	Inspired by MINIX (a Unix-like OS), Linux was originally developed by Finnish-American software engineer Linus Torvalds. Since it is an open source, we have community developers for Linux.	Originally derived from AT&T Unix, it was developed at Bell Labs by Kenneth Lane Thompson, Dennis Ritchie, and 3 others.

Features	Linux	Unix
Written in	C and other programming languages.	C and assembly language.
OS family	Unix-like	Unix
Source Model	Open source	Mixed. Traditionally closed source, however, few Unix projects are open source which include illumosOS and BSD (Berkley Software Distribution) OS.
Available in	Multilingual	English
Initial release	Linux is newer when compared to Unix. It was derived from Unix and was released in September 1991.	Unix is older. Was released in October 1973 for outside parties. Before that, it was used internally in Bell Labs since its inception in 1970.
Kernel Type	Monolithic kernel	Kernel Type varies. It can be monolithic, microkernel and hybrid.
License	GNUv2(GPL General Public License) and others.	Licensing varies. Few versions are proprietary while others are free/OSS.
Official Website	https://www.kernel.org/	http://opengroup.org/unix
Default user interface	Unix shell	CLI (Command Line Interface) and Graphical (X Windows system)
Text Mode Interface	By default, the shell is BASH (Bourne Again Shell). Moreover, is compatible with	Originally the Bourne shell. It is also compatible with many

Features	Linux	Unix
	many command interpreters.	command interpreters.
Cost	Can be obtained and used freely. There are priced versions of Linux as well. But, generally, Linux is cheaper than Windows.	Proprietary operating systems have different cost structures set accordingly by the vendors selling it.
Examples	Debian, Ubuntu, Fedora, Red Hat, Android, etc.	IBM AIX, Solaris, HP-UX, Darwin, macOS X, etc.
Architecture	Was originally created for Intel's x86 hardware, ports available for a lot of CPU types.	Compatible with PA and Itanium machines. Solaris is also available on x86/x64. OSX is PowerPC.
Threat detection and solution	As Linux is mainly driven by open source community, many developers across different parts of the world are working on the code. Hence threat detection and solution is quite fast in case of Linux.	Due to the proprietary nature of Unix, users need to wait for proper bug fixing patches.
Security	Both Linux and Unix based OS is generally regarded as very well protected against malware. This is attributable to lack of root access, quick updates and comparatively low market share (as compared to windows). As of 2018, there has been none widespread Linux virus.	Unix is also considered to be very safe. It is even harder to infect as the source is also not available. There is no actively spreading virus for Unix nowadays.
Price	Linux is free. However, corporate support is available at a price.	Unix is not free. However, some Unix versions are free for development use (Solaris). In a collaborative environment, Unix

Features	Linux	Unix
		costs \$1,407 per user and Linux costs \$256 per user. Hence, UNIX is extremely expensive.

Mobile Operating System

1. Introduction :

A **mobile operating system** (or **mobile OS**) is an operating system for phones, tablets, smartwatches, or other mobile devices.

While computers such as typical laptops are 'mobile', the operating systems usually used on them are not considered mobile ones, as they were originally designed for desktop computers that historically did not have or need specific mobile features. This distinction is becoming blurred in some newer operating systems that are hybrids made for both uses.

Mobile operating systems combine features of a personal computer operating system with other features useful for mobile or handheld use; usually including, and most of the following considered essential in modern mobile systems; a wireless inbuilt modem and SIM tray for telephony and data connection.

Mobile devices with mobile communications abilities (e.g., smartphones) contain two mobile operating systems – the main user-facing software platform is supplemented by a second low-level proprietary real-time operating system which operates the radio and other hardware.

Mobile operating systems have majority use since 2017 (measured by web use); with even only the smartphones running them (excluding tablets) more used than any other kind of device. Thus traditional desktop OS is now a minority used kind of OS.

Popular Mobile Operating Systems

1. Android OS (Google Inc.)

The Android mobile operating system is Google's open and free software stack that includes an operating system, middleware and also key applications for use on mobile devices, including smartphones. Updates for the open source Android mobile operating system have been developed under "dessert-inspired" version names (Cupcake, Donut, Eclair, Gingerbread,

Honeycomb, Ice Cream Sandwich) with each new version arriving in alphabetical order with new enhancements and improvements.

2. Bada (Samsung Electronics)

Bada is a proprietary Samsung mobile OS that was first launched in 2010. The Samsung Wave was the first smartphone to use this mobile OS. Bada provides mobile features such as multipoint-touch, 3D graphics and of course, application downloads and installation.

3. BlackBerry OS (Research In Motion)

The BlackBerry OS is a proprietary mobile operating system developed by Research In Motion for use on the company's popular BlackBerry handheld devices. The BlackBerry platform is popular with corporate users as it offers synchronization with Microsoft Exchange, Lotus Domino, Novell GroupWise email and other business software, when used with the BlackBerry Enterprise Server.

4. iPhone OS / iOS (Apple)

Apple's iPhone OS was originally developed for use on its iPhone devices. Now, the mobile operating system is referred to as iOS and is supported on a number of Apple devices including the iPhone, iPad, iPad 2 and iPod Touch. The iOS mobile operating system is available only on Apple's own manufactured devices as the company does not license the OS for third-party hardware. Apple iOS is derived from Apple's Mac OS X operating system.

5. MeeGo OS (Nokia and Intel)

A joint open source mobile operating system which is the result of merging two products based on open source technologies: Maemo (Nokia) and Moblin (Intel). MeeGo is a mobile OS designed to work on a number of devices including smartphones, netbooks, tablets, in-vehicle information systems and various devices using Intel Atom and ARMv7 architectures.

6. Palm OS (Garnet OS)

The Palm OS is a proprietary mobile operating system (PDA operating system) that was originally released in 1996 on the Pilot 1000 handheld. Newer versions of the Palm OS have added support for expansion ports, new processors, external memory cards, improved security and support for ARM processors and smartphones. Palm OS 5 was extended to provide support for a broad range of screen resolutions, wireless connections and enhanced multimedia capabilities and is called Garnet OS.

7. Symbian OS (Nokia)

Symbian is a mobile operating system (OS) targeted at mobile phones that offers a high-level of integration with communication and personal information management (PIM) functionality. Symbian OS combines middleware with wireless communications through an integrated mailbox and the integration of Java and PIM functionality (agenda and contacts). Nokia has made the Symbian platform available under an alternative, open and direct model, to work with some OEMs and the small community of platform development collaborators. Nokia does not maintain Symbian as an open source development project.

8. webOS (Palm/HP)

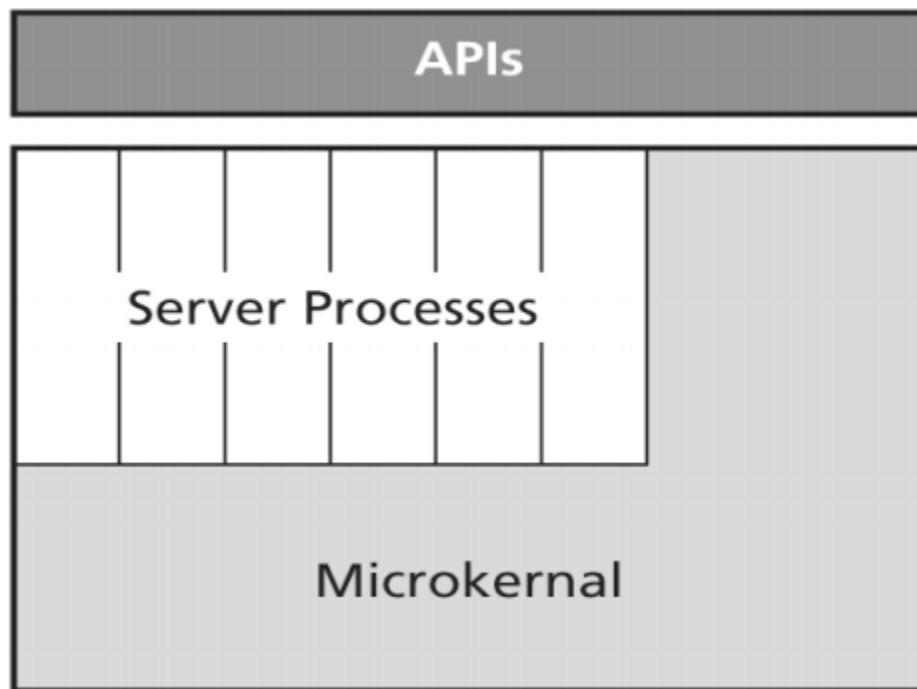
WebOS is a mobile operating system that runs on the Linux kernel. WebOS was initially developed by Palm as the successor to its Palm OS mobile operating system. It is a proprietary Mobile OS which was eventually acquired by HP and now referred to as webOS (lower-case w) in HP literature. HP uses webOS in a number of devices including several smartphones and HP TouchPads. HP has pushed its webOS into the enterprise mobile market by focusing on improving security features and management with the release of webOS 3.x. HP has also announced plans for a version of webOS to run within the Microsoft Windows operating system and to be installed on all HP desktop and notebook computers in 2012.

9. Windows Mobile (Windows Phone)

Windows Mobile is Microsoft's mobile operating system used in smartphones and mobile devices – with or without touchscreens. The Mobile OS is based on the Windows CE 5.2 kernel. In 2010 Microsoft announced a new smartphone platform called Windows Phone 7.

Microkernel Design in Mobile OS:

Microkernel assigns only a small number of essential processes to the kernel such as address space management, inter process communication and scheduling and provides all other services in separate processes called servers.



Essential components and minimalist: If the hardware provides multiple rings or CPU modes, the microkernel may be the only software executing at the most privileged level, which is generally referred to as supervisor or kernel mode. Traditional operating system functions, such as device drivers, protocol stacks and file systems, are typically removed from the microkernel itself and

are Everything else can be done in a usermode program, although device drivers implemented as user programs may on some processor architectures require special privileges to access I/O hardware.

A key component of a microkernel is a good IPC system and virtual-memory-manager design that allows implementing page-fault handling and swapping in usermode servers in a safe way. Since all services are performed by usermode programs, efficient means of communication between programs are essential, far more so than in monolithic kernels. The design of the IPC system makes or breaks a microkernel. To be effective, the IPC system must not only have low overhead, but also interact well with CPU scheduling.

File system in Mobile OS:

Mobile file management (MFM) is a type of information technology (IT) software that allows businesses to manage transfers and storage of corporate files and other related items on a mobile device, and allows the business to oversee user access.

Mobile file management software is typically installed on a corporate file server like Windows 2008, and on a mobile device such as tablet computers and smartphones, e.g., Android, iPad, iPhone, etc. Other features include the ability to remotely wipe a lost or stolen device, access, cache and store files on a mobile device and integrate with file permission solutions like those from Microsoft's Active Directory.

A main advantage of modern mobile file management solutions is that they do not need a VPN connection for the mobile devices to connect to the corporate file servers. The connection between the mobile device and the corporate file server is established via a cloud service. This way the corporate file server doesn't need to open incoming ports which would cause security issues.

The files are transferred highly encrypted - e.g. according to AES 256 Bit industry standard. Only the company server and the mobile device keep the encryption key to be able to encrypt and decrypt the files. So nobody, not even the mobile file management solution provider, can

access the files. Third-party cloud-based companies provide solutions which can be used to manage mobile files but are not controlled by corporate IT organizations.

File management is how the computer operating system keeps data organized through the use of files and folders, how they are arranged, and how they are listed in a hierarchical order. Mobile file management allows file management to be used on tablet computers. By installing it both on the tablet and the corporate server, users of mobile devices can freely access corporate servers from remote locations.

Mobile OS Architecture:

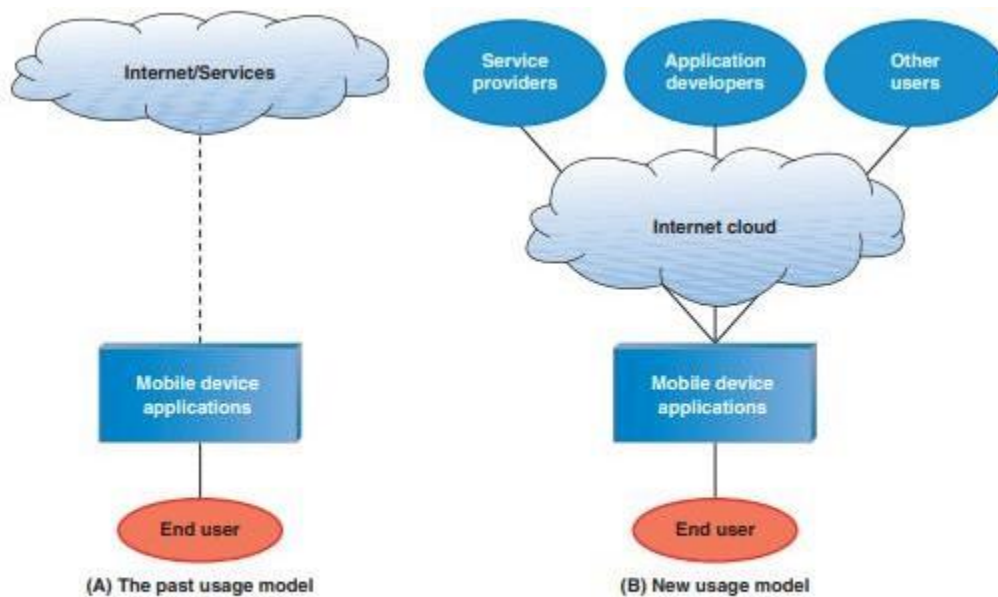
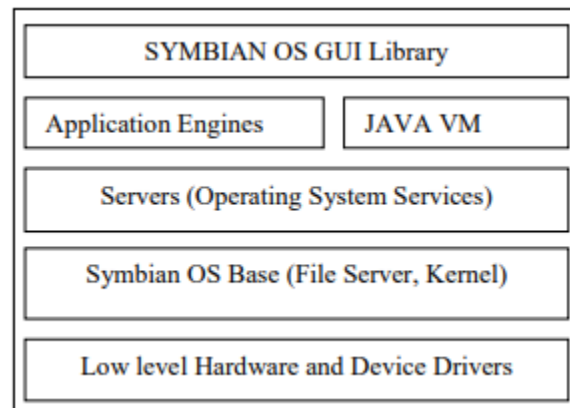


Figure 1: High-level usage models of mobile devices

- **Symbian OS Architecture:**

This Operating system was developed by NOKIA.

Architecture:



The System Kernel, File Server, Memory Management and Device drivers are located in the Base Operating System Layer. The Kernel manages system resources and responsible for time-slicing the applications and system tasks. The topmost layer of symbian provides the frameworks and libraries for constructing user interface controls and utilities.

The Application Engine layer provides services that support generic types of applications and OS Services layer provides servers, Frameworks and libraries that implement core Operating system support for graphics, communications, connectivity and Multimedia. Java VM Provides a set of APIs for mobile devices on the topmost of the OS.

Advantages:

- A. Designed from scratch for mobile platforms
- B. High Quality Games
- C. Easier and Faster Connectivity.

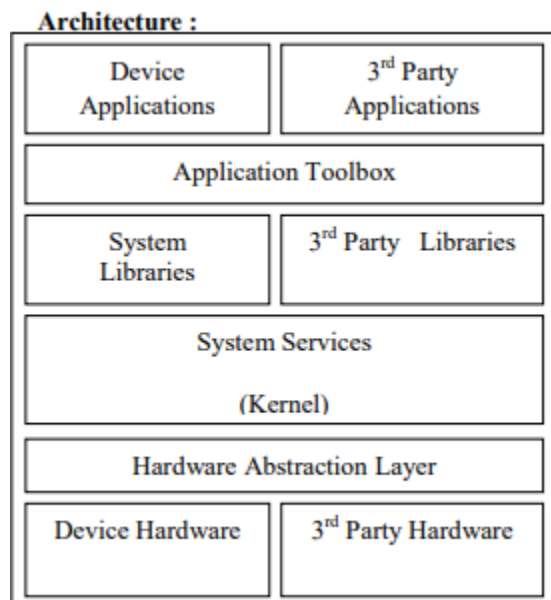
Disadvantages:

- A. Continuous Shifts in GUIs
- B. Frequent hangs and late responses.

Features :Multi Tasking, Good Performance,

• Palm OS Architecture:

This operating system is especially designed for PDAs and handheld devices.



System libraries let the developers easily extend the functionality of OS. Hardware layer is finely tuned and optimized to support a very specific range of H/W, CPU, Controller Chips and Smaller screens of Palm OS Based devices. 3rd party libraries provide support for 3rd party applications such as games, graphics drawings.

Disadvantages

- A. No Keyboard
- B. No full text Recognition

Advantages

- A. Hand writing input recognition
- B. Expansion Support
- C. Memory Management

• Android OS Architecture:

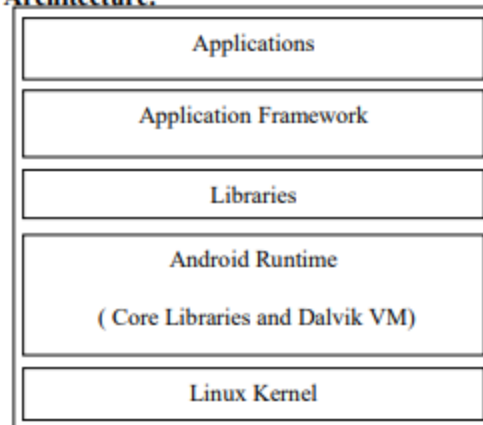
The most popular mobile operating system today in mobile market. The Android OS is an open source software, which means that any user can bring improvements to the operating system, therefore one may benefit not only from Google developers' know-how, but also from that of third-party developers. Google opened the entire source code (including the network and telephony support) so producers are free to add extensions without making them available to the open source community. Android has been criticized for the fact that some parts of libraries and APIs are not fully open source

Android version history:

Code name	Version numbers	Initial release date	API level
No codename	1.0	September 23, 2008	1
Petit Four (only internally used)	1.1	February 9, 2009	2
Cupcake	1.5	April 27, 2009	3
Donut	1.6	September 15, 2009	4

Eclair	2.0 – 2.1	October 26, 2009	5 – 7
Froyo	2.2 – 2.2.3	May 20, 2010	8
Gingerbread	2.3 – 2.3.7	December 6, 2010	9 – 10
Honeycomb	3.0 – 3.2.6	February 22, 2011	11 – 13
Ice Cream Sandwich	4.0 – 4.0.4	October 18, 2011	14 – 15
Jelly Bean	4.1 – 4.3.1	July 9, 2012	16 – 18
KitKat	4.4 – 4.4.4	October 31, 2013	19 – 20
Lollipop	5.0 – 5.1.1	November 12, 2014	21 – 22
Marshmallow	6.0 – 6.0.1	October 5, 2015	23
Nougat	7.0 – 7.1.2	August 22, 2016	24 – 25
Oreo	8.0 – 8.1	August 21, 2017	26 – 27
Pie	9.0	August 6, 2018	28

Android 10	10.0	September 3, 2019	29
------------	------	-------------------	----

Architecture:

Linux kernel acts as an abstraction layer between the hardware and the rest of the software stack. Android runtime includes core libraries and Dalvik VM. Core libraries have a set of libraries to provide the functionality of JAVA PL. Every application runs on its own Dalvik VM which executes files in .dex format.

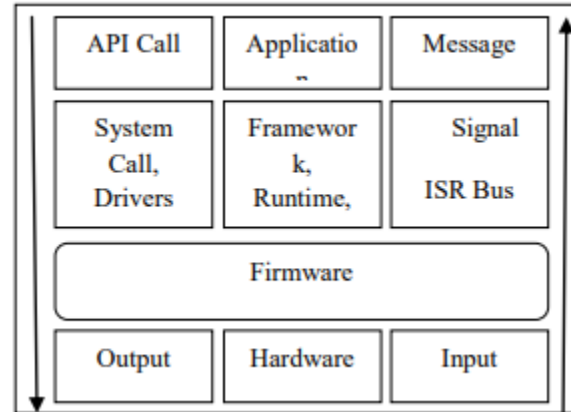
Android has a set of C/C++ libraries used by various components of the operating system. It ships with a set of core applications that offers developers the ability to build various applications with an open development.

Advantages: Multitasking, Ease of access to thousands of applications, Diverse Phone options.

Disadvantages: Needed Continuous Internet Connection, Advertising, Range of applications can still be expanded.

• IOS Architecture:

A mobile operating system developed by Apple Inc. and distributed exclusively for Apple.

Architecture:

Hardware refers to physical chips soldered to iPhone circuitry. Firmware refers to chip specific code that is either contained in with memory in/around the peripheral itself or within the drive for said peripheral. Processor refers to ARM Instruction set and interrupt descriptor table as setup by OS during boot process. iPhone OS is the kernel, drivers and services that sits between user space and Hardware. Runtime is composed of dynamic link libraries as well as underlying C libraries.

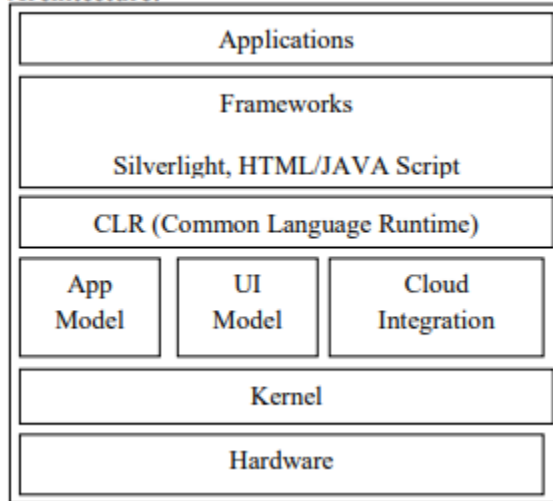
Frameworks/API has API calls which are Apple distributed headers with iPhone SDK. The Application stored in iPhone has to be purchased through App Store, This App was compiled to native code by compiler and linked with runtime by the linker. The application runs entirely within user space environment set up by the iPhone OS

Advantages: Direct Twitter Integration, Advance Voice Recognition, Facetime to make Video calls.

Disadvantages: No flash Support, Dependent on Apple hardware, App Approval process is largely a black box to developers, facetime is exclusive to iOS powered devices.

Windows Phone OS Architecture:

Developed by Microsoft.

Architecture:

The Hardware is composed of ARM 7 CPU, Direct X 9 Capable GPU, 256 MB RAM, Capacitative multi touch display with required Physical buttons. Kernel Handles low level device driver access as well as basic security, networking and storage. The three libraries App Model, UI Model and Cloud Integration Model sit above the kernel for application management and notifications.

Application facing APIs include silver light, HTML/Java Script and CLR that supports C#.Net and VB.Net applications.

Advantages: Multitasking, Feature additions, Multimedia, Camera Technology

Case Study: Android Operating System

CONTENTS

1	Introduction.....	15
2	Structure.....	16
2.1	Kernel	16
2.2	Shell.....	17
2.2.1	Libraries	17
2.2.2	Android Runtime	18
2.2.3	Application Framework	18
2.2.4	Applications	19
3	Process Management	20
3.1	Processes	20
3.2	Applications and Tasks	21
3.3	Application Internals	21
3.4	Application Life Cycle	22

Introduction

Android is a mobile operating system developed by Google based on the Linux kernel. Android is mainly designed for smartphone devices which implement a touch screen input interface. It has also been developed for other devices such as tablet computers, smart watches (Android Wear) and cars (Android Auto.)

Android is known for its OS touch inputs that correspond to real world actions such as tapping, swiping, pinching and reverse pinching.

Android is the most popular mobile operating system, competing with IOS for apple devices and Windows Phone. A developer survey conducted in April–May 2013 found that 71% of mobile developers develop for Android

Android Source code is released by Google under the open source licenses though most Android devices ship with a combination of open source software and proprietary software developed and licensed by Google. The open source nature of Android has enabled many to create and distribute their own modified version of the OS through the Android Open Source Project (ASOP). CyanogenMod is the most widely used community firmware.

Unlike other mobile operating systems, Android is written in Java and is run on virtual machines. Android features the Dalvik Runtime Machine and Android Runtime in newer versions which executes its own bytecode. Dalvik is a core component and all Android user applications and the application framework are written in Java and executed in Dalvik.

The development of Android takes place quickly with a new version being released every few months. Android release numerous updates that incrementally improve the operating system adding new features and fixing bugs in older releases. Each major release is released under the name of a dessert or sugary treat. The first version of Android was released in 2008, named cup cake and the versions to follow are Donut, Éclair, Froyo, Gingerbread, HoneyComb, Icecream Sandwich, Jelly Bean, KitKat and the latest, Lollipop.

Structure

Android software stack is subdivided into five parts: Applications, Application Framework, Libraries, Android Runtime, and the Linux Kernel.

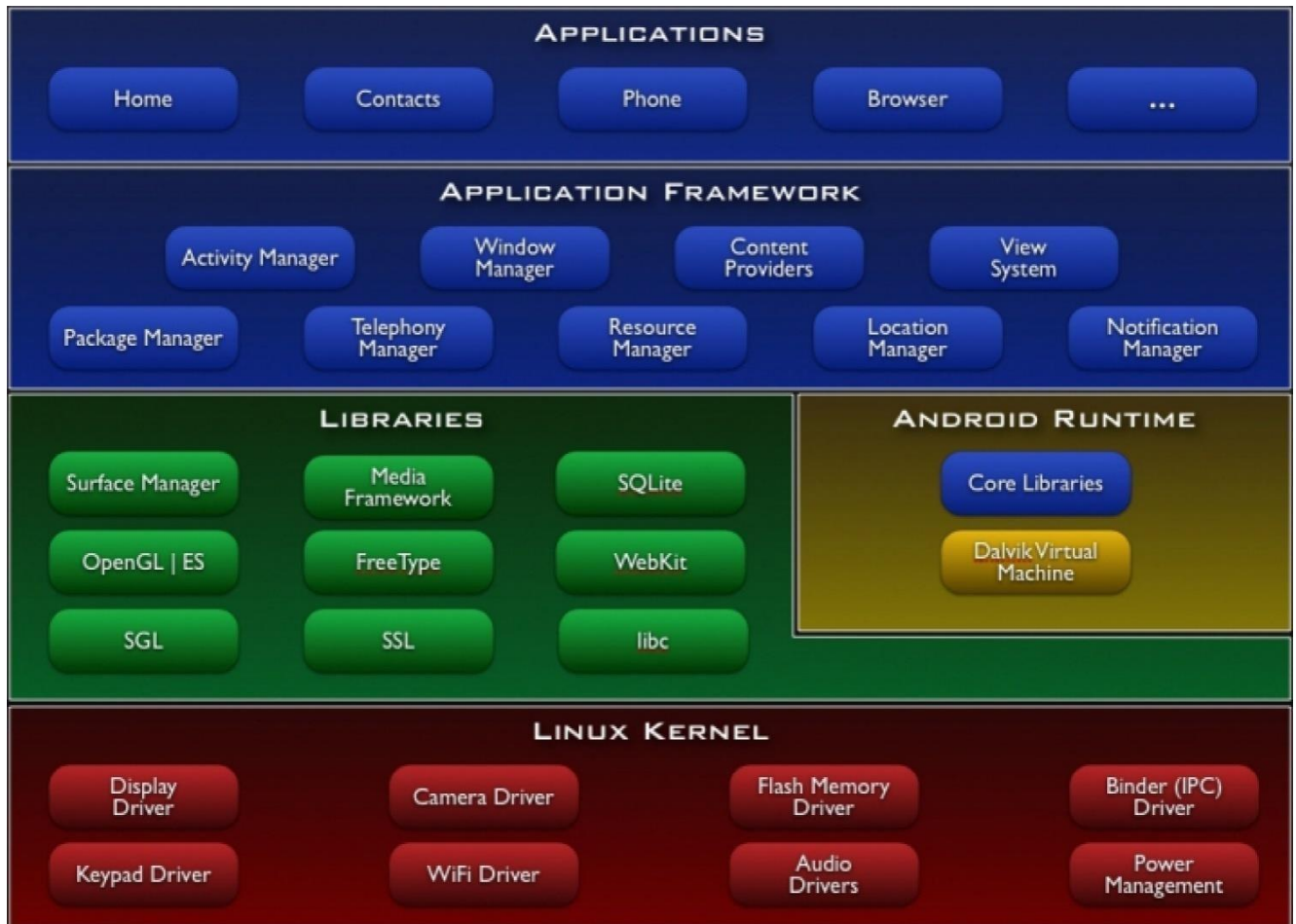


Figure 1: Structure of the Android OS

1. Kernel

The kernel is the core of the operating system. Androids kernel is built off of the Linux 2.6 kernel with some architectural modifications which are implemented by google outside the usual linux kernel development cycle. Linux kernel is a monolithic kernel, meaning that most of the operating system is found in the kernel space, such as device drivers, kernel extensions. This result in very large source code.

The Linux Kernel is what interacts with the hardware and contains all device drivers used by higher levels of the software stack to control and communicate with the hardware. Such drivers are the Display Driver, Camera Driver, Flash Memory Driver, Audio Driver e.t.c

The kernel is modified for special needs in mobile devices such as power management, memory management and the runtime management.

2. Shell

The shell is the user space of the operating system. It acts as the intermediary between the user and the operating system. The Android shell is divided into the following parts

2.1 Libraries

This is the layer that enables the device to handle different types of data. These libraries are written in C or C++ language and are specific to a particular hardware. Androids standard C library is optimized for devices with low power consumption and little memory.

The libraries used by Linux ,GNU libs (glibc), are too big and complicated for mobile devices and so Android implements its own version of libc – Bionic libc. The benefits of using Bionic are its smaller footprint and optimization for low frequency CPUs, used by mobile devices.

Some native libraries implemented in Bionic are:

- ***Surface Manager***

This handles screen access for the window manager from the framework layer. It composites screens using off screen buffering meaning that apps can't directly draw into the screen, but instead the drawings go into the off screen buffer. In the off screen buffer, drawings are combined to form the final output to screen which the user will see, including the status bar at the top and navigation buttons at the bottom of the screen. The off screen buffer is also responsible for the transparency of windows

- ***Media Framework***

This includes audio and video codecs which are heavily optimized for mobile devices. They allow for recording and playback of various formats of audio and video such as mp3, mp4, avi, wav e.t.c

- ***SQLite***

This is the database engine used by android in storage of all data

- ***Webkit***

The browser engine used in rendering HTML webpages

- ***OpenGL***

Used to render 3D images on screen

2.2 Android Runtime

The Android Runtime consists of the Dalvik Virtual Machine (DVM) or Android RunTime (ART) and core Java libraries.

The Dalvik Virtual Machine is an interpreter for bytecode that has been transformed by Java code to Dalvik bytecode. It is an optimized Java Virtual Machine optimized for low processing power and low memory requirements. The DVM runs .dex files (Dalvik executable files) instead of .class files which the JVM runs. Dalvik is compiled into native code whereas the core libraries are written in Java thus interpreted by Dalvik.

Android Runtime (ART) is a newer virtual machine introduced by Google in their newer releases of Android. ART works similarly to Dalvik with many advantages such as AOT (Ahead Of Time) compilation and improved garbage collection which boosts performance significantly.

The Core Java Libraries provide most of the functionalities defined in the Java SE libraries

2.3 Application Framework

These are frameworks written in Java that provide abstractions of the underlying native libraries and Dalvik capabilities. The frameworks are blocks which the applications directly interact with. They provide the basic functions of phones like resource management, voice call management e.t.c. They are the basic tools used to build applications. Some important blocks in the application framework layer are:

- **Activity Manager:** Manages activity life of applications

- **Content Providers:** Manages data sharing between applications
- **Telephone Manager:** Manages all voice calls
- **Location Manager:** Uses GPS or cell towers for location management
- **Resource Manager:** Manages many types of resources used in applications

2.4 Applications

This is the top most layer of the operating system containing all the apps that users interact with. Android applications run in their own Dalvik Virtual Machine and consist of many components such as activities, services, broadcast receivers and content providers. Components interact with each other in the same or different application via intents.

Various apps come pre-installed with android devices such as SMS client app, dialer, web browser, file explorer etc. There is a large community of developers who have come up with various types of applications such as games, document readers, social networks and various others

Process Management

Android provides several means on different layers to compose, execute and manage applications.

Processes

A process is an instance of a program that is being executed. In Android, there are five different stages a process goes through in its lifecycle. The various types have different importance levels which are strictly ordered. The importance hierarchy is (descending from highest importance)

1. **Foreground Process:** This is the app currently in use by the user. Other processes can be considered foreground processes if they're interacting with the process that's currently in the foreground. There are a few foreground processes at any given time
2. **Visible Process:** This is a process that isn't in the foreground but is still affecting what is seen on the screen. For example, a foreground process maybe a dialog but the visible process is the app in the background of the screen which triggered the dialog.
3. **Service Process:** This is a process that isn't tied to any app on screen but is still doing something in the background such as playing music or downloading files
4. **Background Process:** These are processes that are currently not visible to the user and therefore do not have any impact on the user experience. These are apps that are paused and are kept in memory for quick access in the future. They do not use valuable CPU time and other resources apart from memory
5. **Empty Process:** This does not contain any app data anymore. They are kept around for caching purposes to speed launch later but maybe killed by the system if necessary.

Usually only background and empty processes are killed by the system and so the user experience stays unaffected. Android only kills apps when the memory usage goes too high but usually Android does not kill apps

Processes can contain multiple threads such as in Linux based systems. Most Android applications implement thread to separate the UI from input handling and I/O operations or long running calculations.

Applications and Tasks

Android applications are made up of processes and their included threads. Tasks are a series of activities of possibly multiple applications. A task is basically a history of a user's actions. E.g. When a user reads their mail and opens a link which uses the browser application. In this case, the task is made up of two applications (mail and browser) and many activities. The importance of the task concept is that it allows users to navigate backwards like popping elements of a stack.

Application Internals

The structure of an Android applications is based on four different components which are Activity, Service, Broadcast Receiver and Content Provider. An application doesn't necessarily have all these components but it should at least have an Activity in order to present a graphical user interface.

Services and broadcast receivers allow applications to perform jobs in the background. Services usually run for a long time but broadcast receivers can be triggered by events and run for a short time.

Application Life Cycle

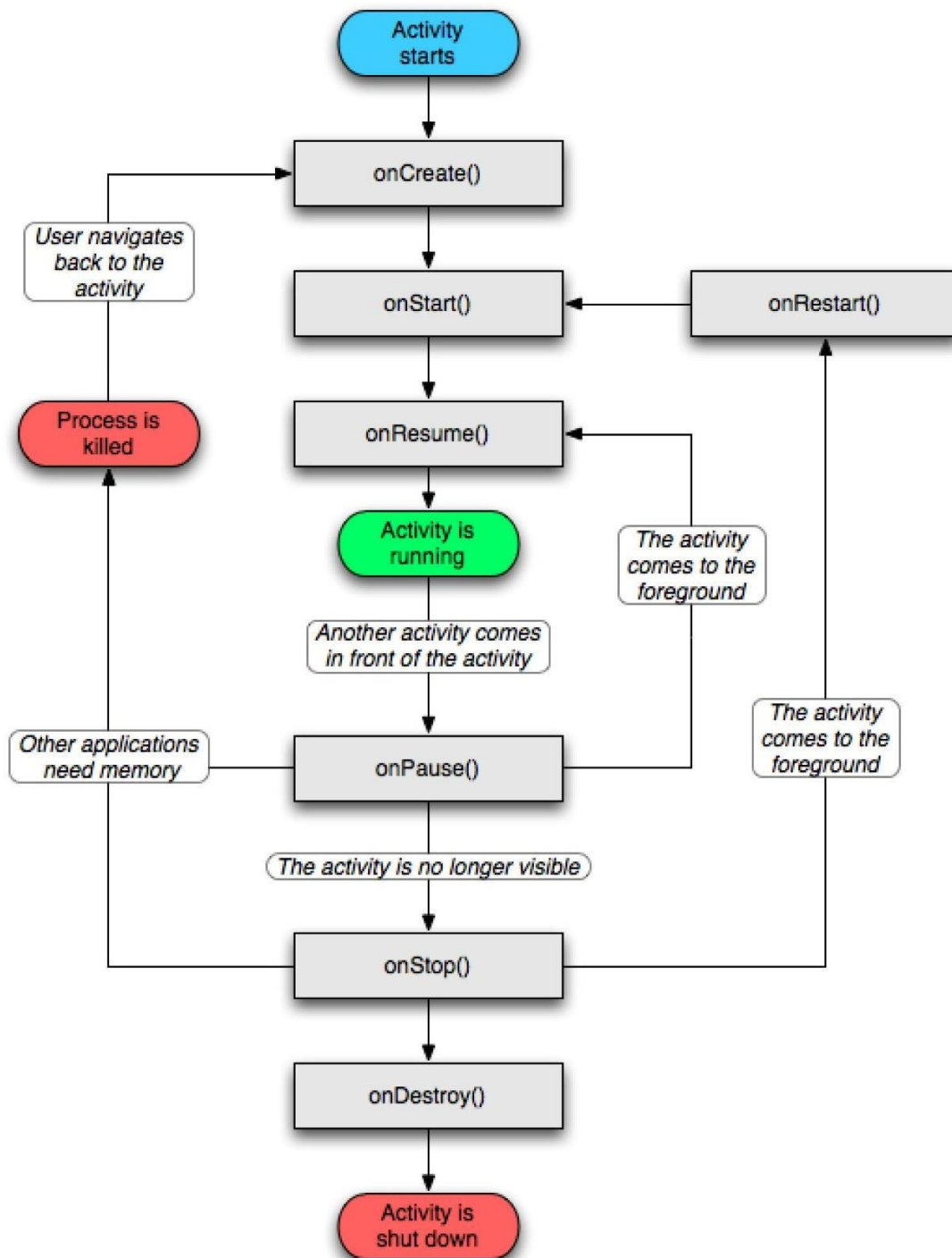


Figure 2: Activity Life Cycle

An activity is a single screen of an application. It contains visual elements that allow user interaction. An application contains many activities. The state of an android applications processes is determined by the state of the application's components, such as its activities. As the application components alter their states the underlying type of the process is changed. All activities are subclasses from `android.app.Activity` and their life cycle is controlled by the `On...()` functions.

As an application starts, the following functions are called sequentially `OnCreate()`, `OnStart()` and `OnResume()`. `OnCreate()` is called only one in the lifetime of a process but `OnStart()` and `OnResume()` are called more often. If an activity loses focus then the `OnPause()` function is called. If the activity is no longer visible then the `OnStop()` function is called. Before deleting the activity `OnDestroy()` function is called which ends the activity lifetime.

The following describes the functions in detail:

- **OnCreate():** The initial functions that initialize the activity.
- **OnStart():** The process type changes to visible and the activity is about to become visible to the user.
- **OnResume():** The process is set to foreground. The application gets focus and can get user input.
- **OnPause():** When the device goes to sleep or the application loses focus the process is set to visible. From here, the process may be resumed or stopped.
- **OnStop():** The activity is not visible and the process type is set to background and the application can be killed at any time.
- **OnDestroy():** This method is called right before the system kills the process and the application deletes the activity.

Real Time Operating System (RTOS)

What is an RTOS?

As the name suggests, there is a deadline associated with tasks and an RTOS adheres to this deadline as missing a deadline can cause affects ranging from undesired to catastrophic.

The embedded systems are becoming more and more complex today and with each passing generation their intrusion in our daily lives will become deeper. This means they will bear more and more responsibilities on their shoulders to solve real time problems to make our life easier. But, this requires more and more complex real time applications that RTOS will have to manage effectively.

Some of the most widely used RTOS are :

- LynxOS
- OSE
- QNX
- RTLinux
- VxWorks
- Windows CE

Classification of RTOS:

RTOS can be classified into three types :

1. **Hard RTOS** : These type of RTOS strictly adhere to the deadline associated with the tasks. Missing on a deadline can have catastrophic affects. The air-bag example is example of a hard RTOS as missing a deadline there could cause a life.
2. **Firm RTOS** : These type of RTOS are also required to adhere to the deadlines because missing a deadline may not cause a catastrophic affect but could cause undesired affects, like a huge reduction in quality of a product which is highly undesired.
3. **Soft RTOS** : In these type of RTOS, missing a deadline is acceptable. For example On-line Databases.

Features of RTOS

An RTOS must be designed in a way that it should strike a balance between supporting a rich feature set for development and deployment of real time applications and not compromising on the deadlines and predictability.

The following points describe the features of an RTOS :

- Context switching latency should be short. This means that the time taken while saving the context of current task and then switching over to another task should be short.
- The time taken between executing the last instruction of an interrupted task and executing the first instruction of interrupt handler should be predictable and short. This is also known as interrupt latency.

- Similarly the time taken between executing the last instruction of the interrupt handler and executing the next task should also be short and predictable. This is also known as interrupt dispatch latency.
- Reliable and time bound inter process mechanisms should be in place for processes to communicate with each other in a timely manner.
- An RTOS should have support for multitasking and task preemption. Preemption means to switch from a currently executing task to a high priority task ready and waiting to be executed.
- Real time Operating systems but support kernel preemption where-in a process in kernel can be preempted by some other process.

Some Misconceptions related to RTOS

- *RTOS should be fast.* This is not true. An RTOS should have a deterministic behavior in terms of deadlines but its not true that the processing speed of an RTOS is fast. This ability of responsiveness of an RTOS does not mean that they are fast.
- *All RTOS are same.* As already discussed we have three types of RTOS (Hard, firm and soft).
- *RTOS cause considerable amount of CPU overhead.* Well, again this is not true. Only 1%-4% of CPU time is required by an RTOS.

RTOS Architecture

RTOS Architecture

For simpler applications, RTOS is usually a kernel but as complexity increases, various modules like networking protocol stacks debugging facilities, device I/Os are included in addition to the kernel.

The general architecture of RTOS is shown in the fig.

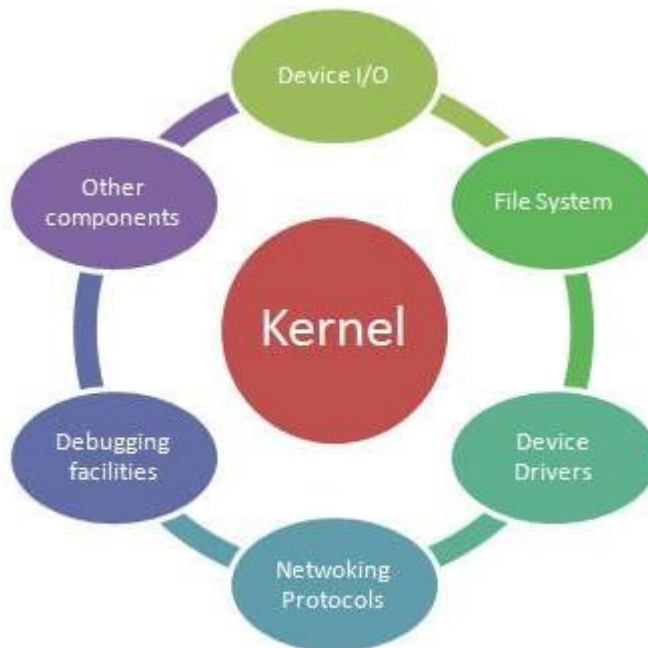


Fig. 3: A Figure Illustrating the General Architecture of RTOS

A) Kernel

RTOS kernel acts as an abstraction layer between the hardware and the applications.

There are three broad categories of kernels

Monolithic kernel

Monolithic kernels are part of Unix-like operating systems like Linux, FreeBSD etc. A monolithic kernel is one single program that contains all of the code necessary to

perform every kernel related task. It runs all basic system services (i.e. process and memory management, interrupt handling and I/O communication, file system, etc) and provides powerful abstractions of the underlying hardware. Amount of context switches and messaging involved are greatly reduced which makes it run faster than microkernel.

· **Microkernel**

It runs only basic process communication (messaging) and I/O control. It normally provides only the minimal services such as managing memory protection, Inter process communication and the process management. The other functions such as running the hardware processes are not handled directly by microkernels. Thus, micro kernels provide a smaller set of simple hardware abstractions. It is more stable than monolithic as the kernel is unaffected even if the servers failed (i.e. File System). Microkernels are part of the operating systems like AIX, BeOS, Mach, Mac OS X, MINIX, and QNX. Etc

· **Hybrid Kernel**

Hybrid kernels are extensions of microkernels with some properties of monolithic kernels. Hybrid kernels are similar to microkernels, except that they include additional code in kernel space so that such code can run more swiftly than it would were it in user space. These are part of the operating systems such as Microsoft Windows NT, 2000 and XP. DragonFly BSD, etc

· **Exokernel**

Exokernels provides efficient control over hardware. It runs only services protecting the resources (i.e. tracking the ownership, guarding the usage, revoking access to resources, etc) by providing low-level interface for library operating systems and leaving the management to the application.

Six types of common services are shown in the following figure below and explained in subsequent sections

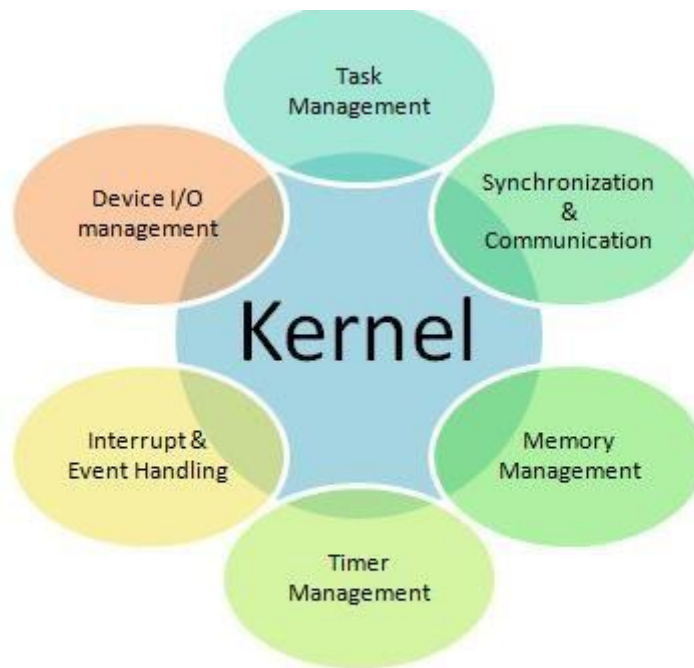


Fig. 4: A Figure Showing Common Services Offered By a RTOS System

B) Task Management

• Task Object

In RTOS, The application is decomposed into small, schedulable, and sequential program units known as “Task”, a basic unit of execution and is governed by three time-critical properties; release time, deadline and execution time. Release time refers to the point in time from which the task can be executed. Deadline is the point in time by which the task must complete. Execution time denotes the time the task takes to execute.

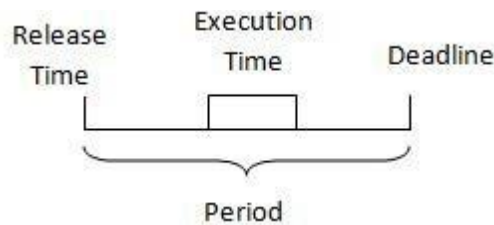


Fig. 5: A Diagram Illustrating Use of RTOS for Time Management Application

Each task may exist in following states

- Dormant : Task doesn't require computer time
- Ready: Task is ready to go active state, waiting processor time
- Active: Task is running
- Suspended: Task put on hold temporarily
- Pending: Task waiting for resource.

C) Synchronisation and communication

Task Synchronisation & intertask communication serves to pass information amongst tasks.

· Task Synchronisation

Synchronization is essential for tasks to share mutually exclusive resources (devices, buffers, etc) and/or allow multiple concurrent tasks to be executed (e.g. Task A needs a result from task B, so task A can only run till task B produces it).

Task synchronization is achieved using two types of mechanisms:

- Event Objects

Event objects are used when task synchronization is required without resource sharing.

They allow one or more tasks to keep waiting for a specified event to occur. Event

object can exist either in triggered or non-triggered state. Triggered state indicates resumption of the task.

- **Semaphores.**

A semaphore has an associated resource count and a wait queue. The resource count indicates availability of resource. The wait queue manages the tasks waiting for resources from the semaphore. A semaphore functions like a key that define whether a task has the access to the resource. A task gets an access to the resource when it acquires the semaphore.

There are three types of semaphore:

- Binary Semaphores
- Counting Semaphores
- **Mutually Exclusion(Mutex) Semaphores**

Semaphore functionality (Mutex) represented pictorially in the following figure

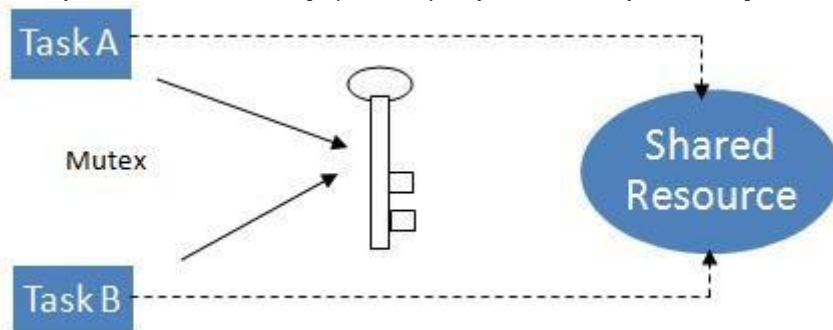


Fig. 12: A Diagram Showing Architecture of Semaphore Functionality

- **Intertask communication**

Intertask communication involves sharing of data among tasks through sharing of memory space, transmission of data, etc. Intertask communications is executed using following mechanisms

- Message queues – A message queue is an object used for intertask communication through which task send or receive messages placed in a shared memory. The queue may follow 1) First In First Out (FIFO), 2) Last in First Out (LIFO) or 3) Priority (PRI) sequence. Usually, a message queue comprises of an associated queue control block (QCB), name, unique ID, memory buffers, queue length, maximum message length and one or more task waiting lists. A message queue with a length of 1 is commonly known as a mailbox.

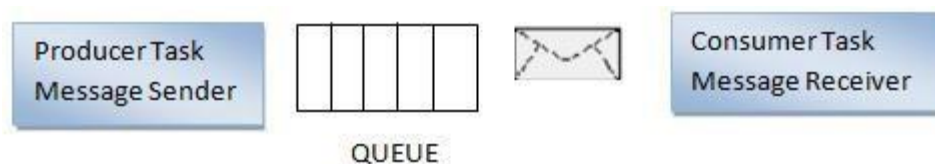


Fig. 13: A Diagram Showing Flow of a Message Queue in a Mailbox

- Pipes – A pipe is an object that provide simple communication channel used for unstructured data exchange among tasks. A pipe does not store multiple messages but stream of bytes. Also, data flow from a pipe cannot be prioritized.
- Remote procedure call (RPC) – It permits distributed computing where task can invoke the execution of another task on a remote computer.

D) Memory Management

Two types of memory managements are provided in RTOS – Stack and Heap.

Stack management is used during context switching for TCBs. Memory other than memory used for program code, program data and system stack is called heap memory and it is used for dynamic allocation of data space for tasks. Management of this memory is called heap management.

E) Timer Management

Tasks need to be performed after scheduled durations. To keep track of the delays, timers- relative and absolute- are provided in RTOS.

F) Interrupt and event handling

RTOS provides various functions for interrupt and event handling, viz., Defining interrupt handler, creation and deletion of ISR, referencing the state of an ISR, enabling and disabling of an interrupt, etc. It also restricts interrupts from occurring when modifying a data structure, minimise interrupt latencies due to disabling of interrupts when RTOS is performing critical operations, minimises interrupt response times.

G) Device I/O Management

RTOS generally provides large number of APIs to support diverse hardware device drivers

Popular RTOS

POPULAR RTOS

There are number of commercially available RTOS, each with some distinct features and targeted for a specific set of applications. Following table lists some of the widely used commercially available RTOS.

RTOS	Applications/Features
Windows CE	Used for Small footprint, mobile and connected devices Supported by ARM,MIPS, SH4 & x86 architectures
LynxOS	Complex, hard real-time applications POSIX-compatible, multiprocess, multithreaded OS. Supported by x86, ARM, PowerPC architectures
VxWorks	Most widely adopted RTOS in the embedded industry. Used in famous NASA rover robots Spirit and Opportunity Certi?ed by several agencies and international standards for real time systems, reliability and security-critical applications.
Micrium µC/OS-II	Ported to more than a hundred architectures including x86, mainly used in microcontrollers with low resources. certi?ed by rigorous standards, such as RTCADO-178B
QNX	Most traditional RTOS in the market. Microkernel architecture; completely compatible with the POSIX Certi?ed by FAADO-278 and MIL-STD-1553 standards.
RT Linux	Hard realtime kernel Good real time performance, but no certification
Jbed	Developed for embedded systems and Internet applications under the Java platform.

	Allows an entire application including the device drivers to be written using Java
Symbian	Designed for Smartphones Supported by ARM, x86 architecture
VRTX	Suitable for traditional board based embedded systems and SoC architectures Supported by ARM, MIPS, PowerPC & other RISC architectures



ENG224

INFORMATION TECHNOLOGY – Part I

2. Operating System Case Study: Linux



2. Operating System Case Study: Linux



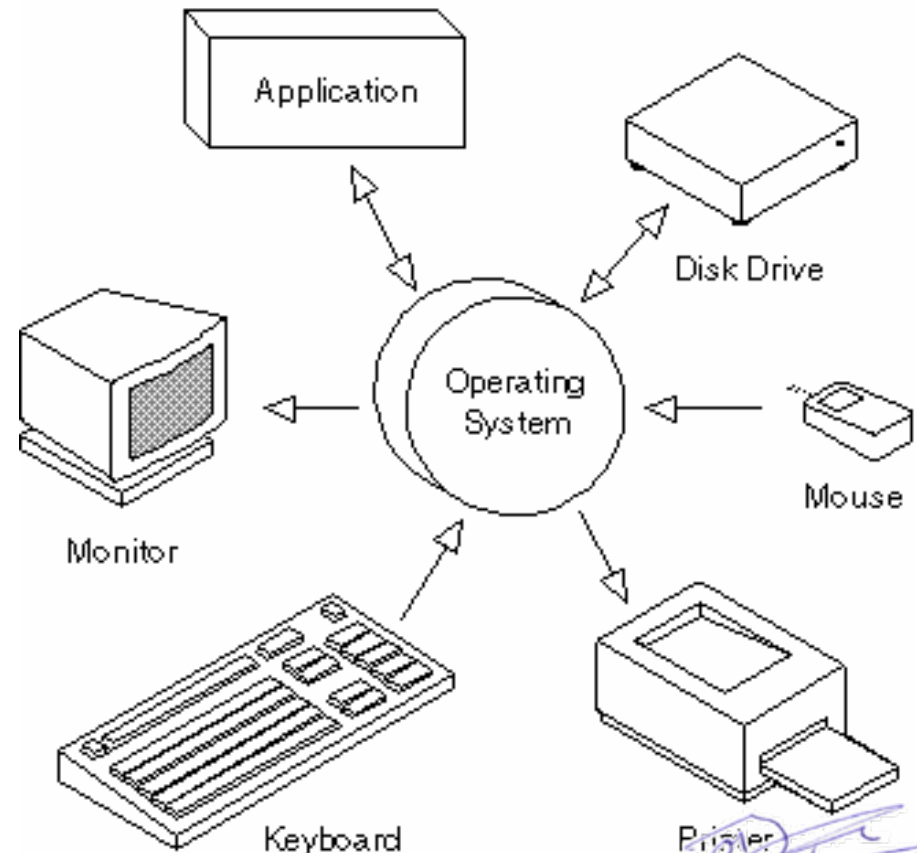
Reference

- S.M. Sarwar, R. Koretsky and S.A. Sarwar, *Linux – The Textbook*, Addison Wesley, 1st ed, 2002



Features of modern OS

- To facilitate easy, efficient, fair, orderly, and secure use of resources
 - Provide a user interface
 - Organize files on disk
 - Allocating resource to different users with security control
 - Co-ordinate programs to work with devices and other programs





Case study: Linux

A. Development of Linux



2. Operating System Case Study: Linux

- Before Linux

- In 80's, Microsoft's DOS was the dominated OS for PC
 - single-user, single-process system
- Apple MAC is better, but expensive
- UNIX is much better, but much much expensive.
Only for minicomputer for commercial applications
- People was looking for a UNIX based system, which is cheaper and can run on PC
- Both DOS, MAC and UNIX are **proprietary**, i.e., the source code of **their kernel is protected**
 - No modification is possible without paying **high** license fees



2. Operating System Case Study: Linux

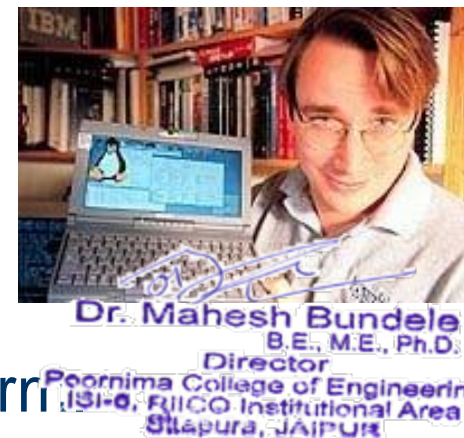
- GNU project

- Established in 1984 by **Richard Stallman**, who believes that software should be free from restrictions against copying or modification in order to make better and efficient computer programs
- **GNU** is a recursive acronym for “**G**NU's **N**ot **U**nix”
- Aim at developing a complete Unix-like operating system which is free for copying and modification
- Companies make their money by maintaining and distributing the software, e.g. optimally packaging the software with different tools (Redhat, Slackware, Mandrake, SuSE, etc)
- Stallman built the first free GNU C Compiler in 1991
But still, an OS was yet to be developed



- Beginning of Linux

- A famous professor Andrew Tanenbaum developed **Minix**, a simplified version of UNIX that runs on PC
- Minix is for class teaching only. No intention for commercial use
- In Sept 1991, **Linus Torvalds**, a second year student of Computer Science at the University of Helsinki, developed the preliminary kernel of Linux, known as Linux version 0.0.1
- It was put to the Internet and received enormous response from worldwide software developers
- By December came version 0.10. Still Linux was little more than in skeletal form





- Confrontation and Development
 - Message from Professor Andrew Tanenbaum
 - " I still maintain the point that designing a monolithic kernel in 1991 is a fundamental error. Be thankful you are not my student. You would not get a high grade for such a design :-)" (Andrew Tanenbaum to Linus Torvalds)
 - "Linux is obsolete".
(Remark made by Andrew Tanenbaum)
 - But work went on. Soon more than a hundred people joined the Linux camp. Then thousands. Then hundreds of thousands
 - It was licensed under **GNU General Public License**, thus ensuring that the source codes will be free for all to copy, study and to change.



- Linux Today

- Linux has been used for many computing platforms
 - PC, PDA, Supercomputer,...
- Current kernel version 2.6.13
- Not only character user interface but graphical user interface, thanks to the X-Window technology
- Commercial vendors moved in Linux itself to provide freely distributed code. They make their money by compiling up various software and gathering them in a distributable format
 - Red Hat, Slackware, etc
- Chinese distribution of Linux also appeared in Taiwan and China - CLE, Red Flag Linux



Linux Pros and Cons

- **Advantages** over Windows
 - It's almost free to relatively inexpensive
 - Source code is included
 - Bugs are fixed quickly and help is readily available through the vast support in Internet
 - Linux is more stable than Windows
 - Linux is truly multi-user and multi-tasking
 - **multiuser**: OS that can simultaneously serve a number of users
 - **multitasking**: OS that can simultaneously execute a number of programs
 - Linux runs on equipment that other operating systems consider too underpowered, e.g. 386 systems.



Linux Pros and Cons (Cont)

- **Disadvantages** compared with Windows
 - Isn't as popular as Windows
 - No one commercial company is responsible for Linux
 - Linux is relatively hard to install, learn and use
- Hence currently, Linux is mainly used in commercial applications, **server implementation**
- More than 75% current network servers are developed based on Linux or Unix systems
 - Due to the relatively **high reliability**



Case study: Linux

B. Linux System Architecture



2. Operating System Case Study: Linux

AUI

Applications: Compilers, word processors, X-based GUI

LINUX Shell: Bourne Again (bash), TC, Z, etc.

Language libraries

API

System call interface

Kernel

Memory
management

**File
management**

**Process
Management**

Device Drives

BIOS

Computer Hardware



- Kernel
 - The part of an OS where the real work is done
- System call interface
 - Comprise a set of functions (often known as Application Programmer's Interface API) that can be used by the applications and library routines to use the services provided by the kernel
- Application User's Interface
 - Interface between the kernel and user
 - Allow user to make commands to the system
 - Divided into text based and graphical based



2. Operating System Case Study: Linux

- File Management

- Control the creation, removal of files and provide directory maintenance
- For a **multiuser system**, every user should have its own right to access files and directories

- Process Management

- For a **multitask system**, multiple programs can be executed simultaneously in the system
- When a program starts to execute, it becomes a **process**
- The same program executing at two different times will become two different processes
- Kernel manages processes in terms of creating, suspending, and terminating them
- A process is protected from other processes and can communicate with the others



- Memory management
 - Memory in a computer is divided into **main memory** (RAM) and **secondary storage** (usually refer to hard disk)
 - Memory is small in capacity but fast in speed, and hard disk is vice versa
 - Data that are not currently used should be saved to hard disk first, while data that are urgently needed should be retrieved and stored in RAM
 - The mechanism is referred as **memory management**
- Device drivers
 - Interfaces between the kernel and the BIOS
 - Different device has different driver



Case study: Linux

B.1 User interface



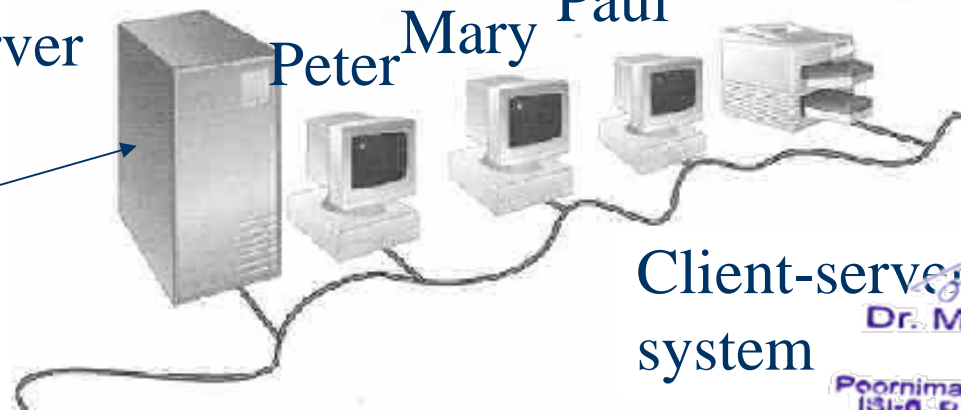
Linux User Login

- Linux is a **multiuser OS**
- Allow multiple users to use the resource of a computer at the same time
- Every user needs to **login** the system with the **password** provided to identify their right in using the resource
- Require for both client-server based system or desktop

Linux
Server

Peter Mary Paul

Peter: admin
Paul : general
Mary : intruder
:



Client-server based
system



Linux User Interface

- Traditional Linux (Unix also) uses command-driven interface (or text-based interface)
 - User needs to type lines of command to instruct the computer to work, similar to DOS
 - **Advantage:** fast in speed. Very few resource is required for its implementation
 - **Disadvantages:** user needs to type, hence can easily make error. Besides, user needs to memorize all commands
 - Suitable for expert users and for the systems that interaction with user is not frequent, such as servers



2. Operating System Case Study: Linux

- By adopting the **X-Window technology**, graphical user interface (**GUI**) is available for Linux:
 - Uses pointing devices (e.g. mouse) to control the system, similar to Microsoft's Windows
 - Provide menu-driven and/or icon-driven interfaces
 - **menu-driven**: user is provided with a menu of choices. Each choice refers to a particular task
 - **icon-driven**: tasks are represented by pictures (icon) and shown to user. Click on an icon invokes one task
 - **Advantages**: No need to memorize commands. Always select task from menus or icons
 - **Disadvantages**: Slow and require certain resource for its implementation
 - Suitable for general users and systems, such as...

2. Operating System Case Study: Linux

The collage illustrates a typical Linux GUI environment based on GNOME. It includes a desktop with standard icons, a GIMP image editor, a file manager, and a terminal window. The terminal window displays the output of a file listing command, showing details such as permissions, owner, group, size, and date for various files and directories.

- A typical Linux GUI based on GNOME
- Similar to Microsoft's Windows, however, different window systems can be chosen (e.g. GNOME, KDE, etc.)



2. Operating System Case Study: Linux

Linux text-based interface

The prompt \$ shows that bash shell is using

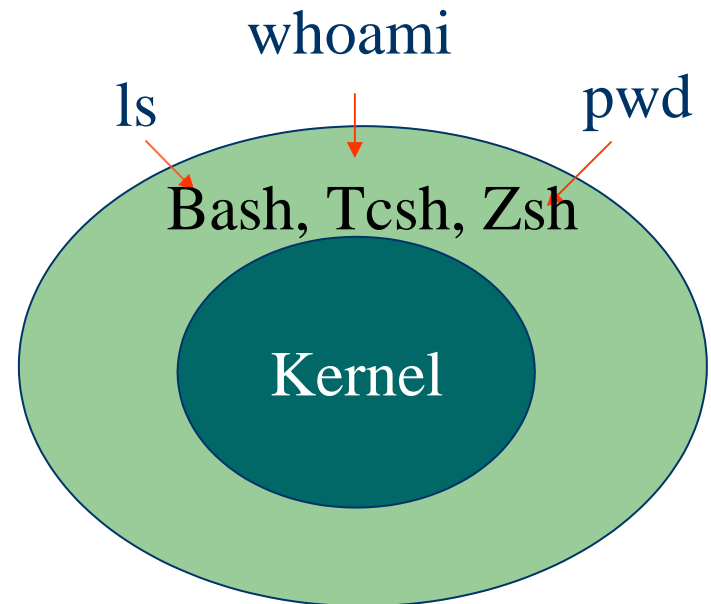
command to show the content of current directory

command to show the content of current directory with option -al



Linux Shell

- **Shell** interprets the command and request service from kernel
- Similar to DOS but DOS has only one set of interface while Linux can select different shell
 - Bourne Again shell (Bash), TC shell (Tcsh), Z shell (Zsh)
- Different shell has similar but different functionality
- **Bash** is the default for Linux
- Graphical user interface of Linux is in fact an application program work on the shell





2. Operating System Case Study: Linux

- Frequently used commands available in most shells:
 - **ls** : to show (**list**) the names of the file in the current directory
 - **cd** : **c**hange **d**irectory,
 - e.g. `cd /` change to the root directory
 - `cd ..` change to the parent of that directory
 - **cp** : **c**opy one file to another
 - e.g. `cp abc.txt xyz.txt` copy abc.txt to xyz.txt
 - **rm** : **r**emove a file
 - **man** : ask for the **m**anual (or help) of a command
 - e.g. `man cd` ask for the manual of the command cd
 - **pwd** : show the name of the **p**resent **w**orking **d**irectory
 - **cat** : to show the content of a text file
 - e.g. `cat abc.txt` show the content of abc.txt
 - **whoami** : to show the username of the current user



Case study: Linux

B.2 File management



Linux File Management

- In Linux, file is defined as simply the thing that deals with a sequence of bytes
- Hence **everything are files**
 - An ordinary file is a file; a directory is also file; a network card, a hard disk, any device are also files since they deal with a sequence of bytes
- Linux supports five types of files
 - simple/ordinary file (text file, c++ file, etc)
 - directory
 - symbolic (soft) link
 - special file (device)
 - named pipe (FIFO)



2. Operating System Case Study: Linux

```
dlun@enpklun.polyu.edu.hk: /home/dlun/Desktop
File Edit Settings Help

[dlun@enpklun Desktop]$ ls
Autostart          Red Hat Support.kdeInk  cdrom.kdeInk
Printer.kdeInk     Templates              floppy.kdeInk
Red Hat Errata.kdeInk  Trash
[dlun@enpklun Desktop]$
[dlun@enpklun Desktop]$
[dlun@enpklun Desktop]$ ls -al
total 44
drwxr-xr-x  5 dlun  dlun      4096 May 17  2001 .
drwx----- 15 dlun  dlun      4096 Jan  4 15:15 ..
drwxr-xr-x  2 dlun  dlun      4096 May 17  2001 Autostart
-rw-r--r--  1 dlun  dlun      230 May 17  2001 Printer.kdeInk
-rw-r--r--  1 dlun  dlun      159 May 17  2001 Red Hat Errata.kdeInk
-rw-r--r--  1 dlun  dlun      153 May 17  2001 Red Hat Support.kdeInk
drwxr-xr-x  2 dlun  dlun      4096 May 17  2001 Templates
drwxr-xr-x  2 dlun  dlun      4096 May 17  2001 Trash
-rw-r--r--  1 dlun  dlun      388 May 17  2001 cdrom.kdeInk
-rw-r--r--  1 dlun  dlun      395 May 17  2001 floppy.kdeInk
-rw-r--r--  1 dlun  dlun      144 May 17  2001 www.redhat.com.kdeInk
[dlun@enpklun Desktop]$
```

The concept of simple file and directory is similar to DOS

Names in blue are directories, indicated by a letter d at the beginning



2. Operating System Case Study: Linux

- Symbolic (soft) link
 - Not a real file, just a **link** to another file
 - Allow giving another name to a file without actually duplicates it – hence **save memory space**
- Special file (device)
 - Each hardware device, e.g. keyboard, hard disk, CD-ROM, etc is associated with at least one file
 - Usually store in /dev directory
 - Applications can read and write any devices by reading and writing their associate file – hence the access method is known as **device independent**
 - Divide into two types: character special files, e.g. keyboard, and block special files, e.g. disk



2. Operating System Case Study: Linux

Command that sets a symbolic link to a file called CUI to anotherCUI

```
dlun@enpk1un.polyu.edu.hk: /home/dlun/Desktop
File Edit Settings Help
[dlun@enpk1un Desktop]$ ln -s ../CUI anotherCUI
[dlun@enpk1un Desktop]$ ls -al
total 44
drwxr-xr-x  5 dlun  dlun  4096 Jan  4 18:36 .
drwx----- 16 dlun  dlun  4096 Jan  4 18:26 ..
drwxr-xr-x  2 dlun  dlun  4096 May 17  2001 Autostart
-rw-r--r--  1 dlun  dlun   230 May 17  2001 Printer.kdeInk
-rw-r--r--  1 dlun  dlun   159 May 17  2001 Red Hat Errata.kdeInk
-rw-r--r--  1 dlun  dlun   153 May 17  2001 Red Hat Support.kdeInk
drwxr-xr-x  2 dlun  dlun  4096 May 17  2001 Templates
drwxr-xr-x  2 dlun  dlun  4096 May 17  2001 Trash
lrwxrwxrwx  1 dlun  dlun    6 Jan  4 18:36 anotherCUI -> ../CUI
-rw-r--r--  1 dlun  dlun   388 May 17  2001 cdrom.kdeInk
-rw-r--r--  1 dlun  dlun   395 May 17  2001 floppy.kdeInk
-rw-r--r--  1 dlun  dlun   144 May 17  2001 www.redhat.com.kdeInk
[dlun@enpk1un Desktop]$
```

File size is only 6 bytes

A symbolic link begins with a letter *l*



2. Operating System Case Study: Linux

```
dlun@enpkun.polyu.edu.hk: /dev
File Edit Settings Help
[dlun@enpkun /dev]$ ls -al *d0
brw-rw---- 1 dlun floppy 2, 0 Aug 24 2000 fd0
brw-rw---- 1 root disk 9, 0 May 17 2001 md0
brw-rw---- 1 root disk 46, 0 Aug 24 2000 pc0
crw-rw-rw- 1 root tty 2, 224 Aug 24 2000 ptyd0
brw-rw---- 1 root disk 25, 0 Aug 24 2000 sbpcd0
brw-rw---- 1 root disk 11, 0 Aug 24 2000 scd0
crw----- 1 root sys 110, 0 Aug 24 2000 srnd0
crw-rw-rw- 1 root tty 3, 224 Aug 24 2000 ttyd0
[dlun@enpkun /dev]$
```

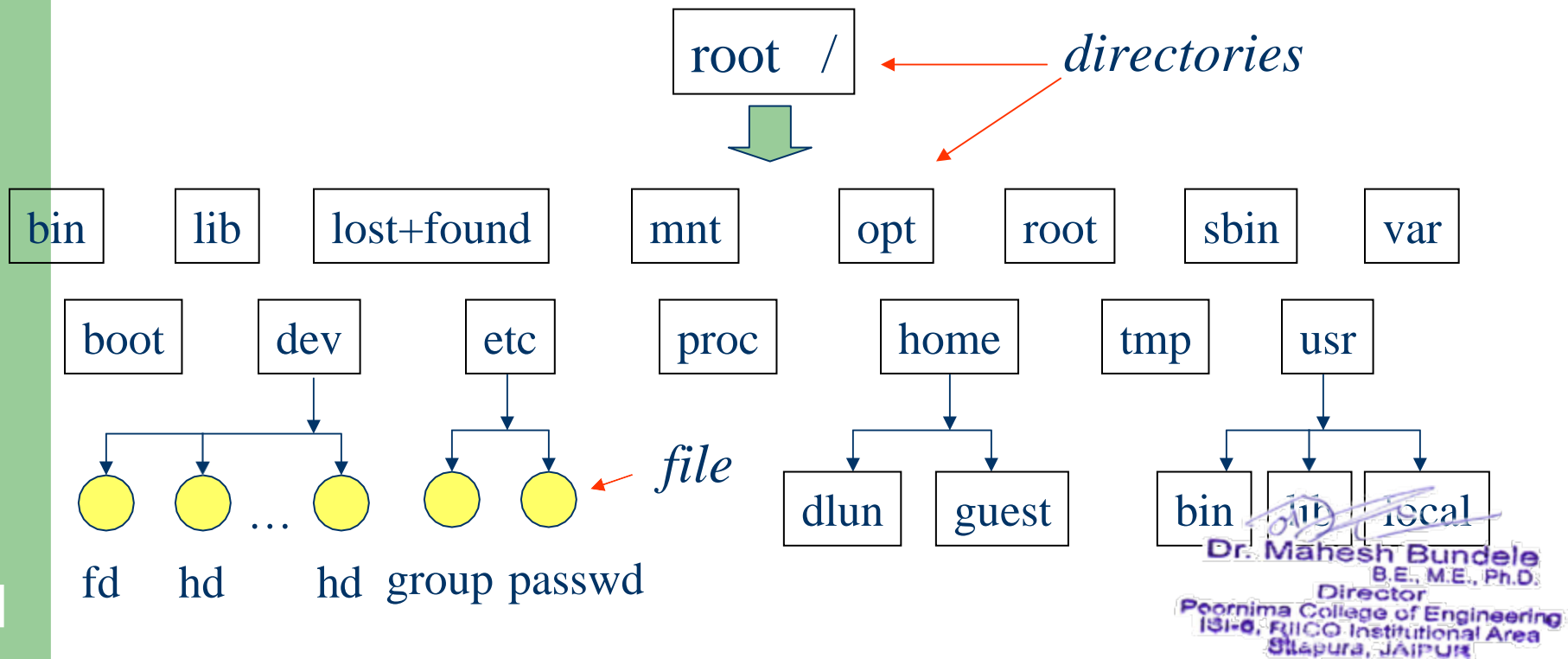
Some are character devices, hence start with a letter c

Some of the special device files in /dev
fd0 – floppy disk
md0 – CD-Rom
Both of them are block devices, hence start with a letter b



Linux File System Structure

- According to the **File System Standard (FSSTND)** proposed in 1994, every LINUX system should contain a set of standard files and directories





2. Operating System Case Study: Linux

- Root Directory (/)
 - Top of the file system. Similar to \ in DOS
- /bin
 - Contain the binary (executable code) of most essential Linux commands, e.g. bash, cat, cp, ln, ls, etc.
- /boot
 - Contain all the files needed to boot the Linux system, including the binary of the Linux kernel. E.g., on Red Hat Linux 6.1, the kernel is in /boot/vmlinux-2.2.5-15 file
- /dev
 - Contain the special files for devices, e.g. fd0, hd0 etc



2. Operating System Case Study: Linux

- /etc
 - Contain host-specific files and directories, e.g. information about system configuration
 - /etc/passwd
 - This file contains login information of users in the system
 - For every user, one line of record is stored in the following format:

login_name : dummy_or_encrypted_password : user_ID :
group_ID : user_info : home_directory : login_shell



2. Operating System Case Study: Linux

- E.g. davis:x:134:105:James A Davis:/home/davis:/bin/bash
 - **davis** : login name
 - **x** : means that it is a dummy password. The encrypted password is stored in /etc/shadow. This field can also be used to store the actual encrypted password. In any case, the original (unencrypted) password cannot be seen by anyone, including the administrator
 - **134** : a user id given to that user. Range from 0 to 65535. 0 is assigned to super-user. 1 to 99 are reserved
 - **105** : a group id given to that user to indicate which group he belongs to. Range from 0 to 65535. 0 to 99 reserved
 - **James A Davis** : user info, usually user's full name
 - **/home/davis** : home directory of the user
 - **/bin/bash** : the location of the shell the user is using



2. Operating System Case Study: Linux

- /home
 - Contain the **home directories of every user** in the system, e.g. dlun, guest, etc
- /lib
 - Store all **essential libraries** for different language compilers
- /lost+found
 - Contain all the files on the system **not connected to any directory.**
 - System administrator should determine the fate of the files in this directory



2. Operating System Case Study: Linux

- /mnt

- Use by system administrator to mount file systems temporarily by using the mount command
- Before using any devices, they have to be **mounted** to the system for registration
- For example, **after mounting a CD-ROM**, the file system in it will be **mapped to /mnt/cdrom directory**
- User can then read and write files in the CD-ROM by accessing this directory
- **Similar to mapping a drive letter to a CD-ROM in Windows**
- Different from the special file in /dev. Special file is only a place where data of the CD-ROM is transferred or stored. No file system concept



2. Operating System Case Study: Linux

- /opt
 - Use to install add-on software packages, e.g. star office, etc.
- /proc
 - Contain process and system information
- /root
 - Home directory of the user root, usually the administrator
- /sbin
 - The directories /sbin, /usr/sbin, and /usr/local/sbin contain **system administration tools, utilities and general root only commands**, such as halt, reboot and shutdown



2. Operating System Case Study: Linux

- /tmp
 - Contain **temporary files**. Usually files in this directory will be deleted from time to time to avoid the system fills with temp files
- /usr
 - One of the largest sections of the Linux file system
 - Contain **read-only data that are shared between various users**, e.g. the manual pages needed for the command man. Stored in /usr/man directory
- /var
 - Contain **data that keeps on changing** as the system is running. E.g. /var/spool/mail directory keeps the mail of user



Linux File Access Privilege

- Linux is a multiuser system, the files of all users are stored in a single file structure
- Mechanism is required to restrict one user to access the files of another user, if he is not supposed to
- User can impose **access permission** to each file to restrict its access
- The term “access permission” refers to
 - read permission
 - write permission
 - execute permission



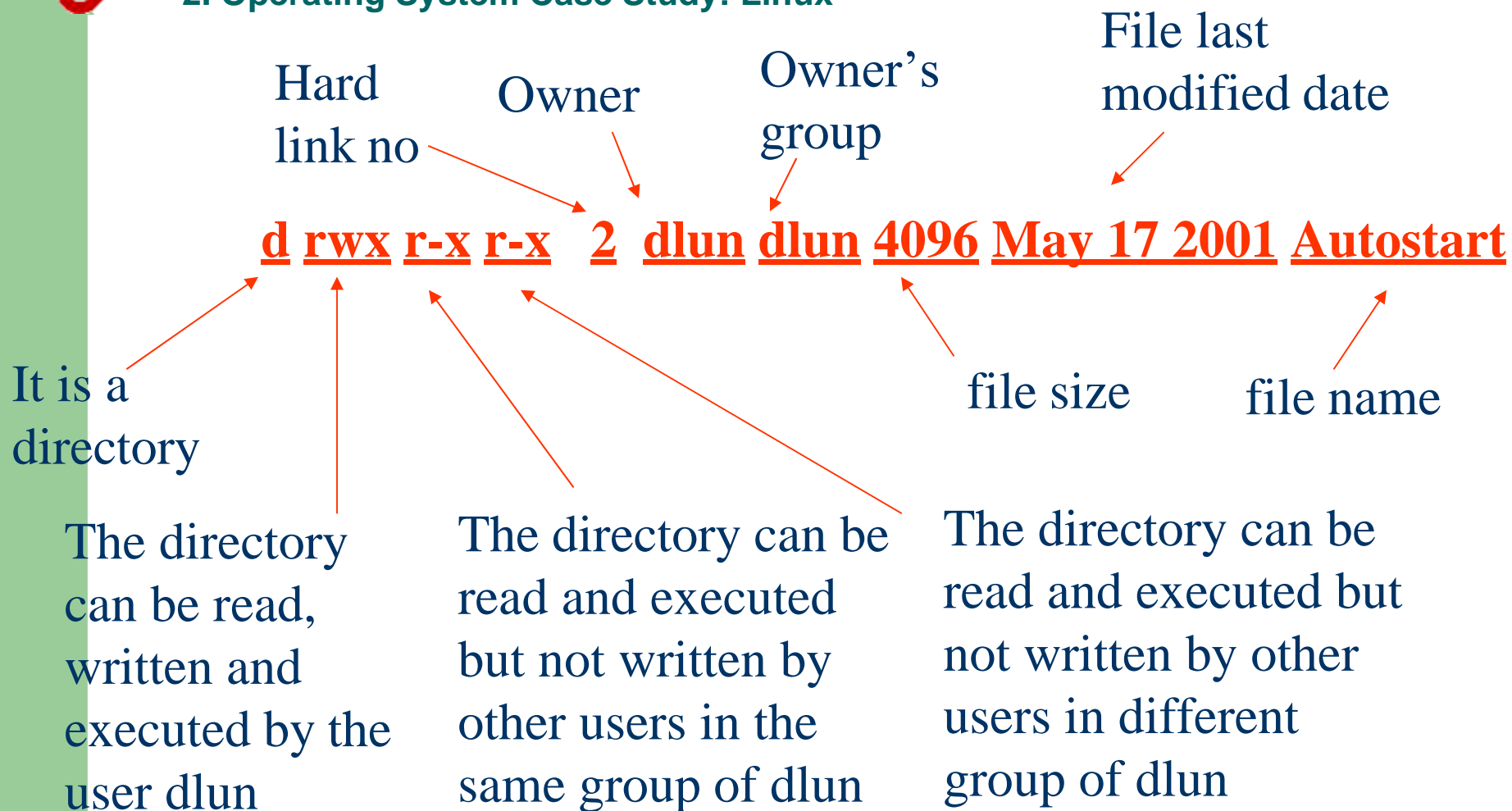
2. Operating System Case Study: Linux

```
dlun@enpkun.polyu.edu.hk: /home/dlun/Desktop
File Edit Settings Help
[dlun@enpkun Desktop]$ ln -s ../CUI anotherCUI
[dlun@enpkun Desktop]$ ls -al
total 44
drwxr-xr-x  5 dlun  dlun  4096 Jan  4 18:36 .
drwx----- 16 dlun  dlun  4096 Jan  4 18:26 ..
drwxr-xr-x  2 dlun  dlun  4096 May 17  2001 Autostart
-rw-r--r--  1 dlun  dlun   230 May 17  2001 Printer.kdeInk
-rw-r--r--  1 dlun  dlun   159 May 17  2001 Red Hat Errata.kdeInk
-rw-r--r--  1 dlun  dlun   153 May 17  2001 Red Hat Support.kdeInk
drwxr-xr-x  2 dlun  dlun  4096 May 17  2001 Templates
drwxr-xr-x  2 dlun  dlun  4096 May 17  2001 Trash
lrwxrwxrwx  1 dlun  dlun    6 Jan  4 18:36 anotherCUI -> ../CUI
-rw-r--r--  1 dlun  dlun   388 May 17  2001 cdrom.kdeInk
-rw-r--r--  1 dlun  dlun   395 May 17  2001 floppy.kdeInk
-rw-r--r--  1 dlun  dlun   144 May 17  2001 www.redhat.com.kdeInk
[dlun@enpkun Desktop]$
```

The file access permission can be seen by using the command `ls -l` or `ls -al`



2. Operating System Case Study: Linux



The group of a user is assigned by the administrator
user is added to the system



2. Operating System Case Study: Linux

- Access permission can also be assigned to a directory
- **Directory is also a file** that contains the attributes of the files inside it
- If **read permission** is not given to a directory
 - cannot show the structure of this directory
 - e.g. cannot use ls
- If **write permission** is not given to a directory
 - cannot modify anything of the directory structure
 - e.g. cannot copy a file into this directory since it will modify the directory structure by adding one more file
- If **execute permission** is not given to a directory
 - nearly nothing can be done with this directory



- The access permission of a file or directory can be changed by using the command

chmod xyz filename/directory name

- xyz refers 3 digit in octal form
- E.g.

660 : 110 110 000

⇒ rw- rw- ---

545 : 101 100 101

⇒ r-x r-- r-x



ENG224

INFORMATION TECHNOLOGY – Part I

2. Operating System Case Study: Linux

dlun@enpklun.polyu.edu.hk: /home/dlun/Desktop/test/temp

File Edit Settings Help

temp does not have execution right

```
[dlun@enpklun test]$ ls -l
total 12
-rw-r--r-- 1 dlun dlun 395 Jan 7 16:36 floppy.kdeInk
drw----- 2 dlun dlun 4096 Jan 9 11:06 temp
-rw-rw-r-- 1 dlun dlun 16 Jan 7 16:05 test1.txt
```

even cd is not workable

```
[dlun@enpklun test]$ cd temp
bash: cd: temp: Permission denied
```

execution right is added

```
[dlun@enpklun test]$
[dlun@enpklun test]$ chmod 700 temp
[dlun@enpklun test]$ ls -l
total 12
-rw-r--r-- 1 dlun dlun 395 Jan 7 16:36 floppy.kdeInk
drwx----- 2 dlun dlun 4096 Jan 9 11:06 temp
-rw-rw-r-- 1 dlun dlun 16 Jan 7 16:05 test1.txt
[dlun@enpklun test]$ cd temp
[dlun@enpklun temp]$
```

now we can change the directory to temp

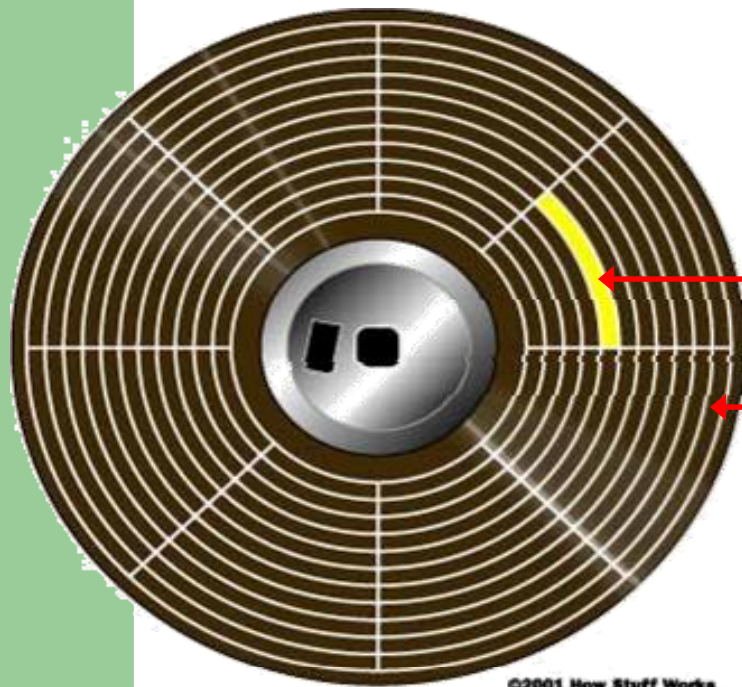


File Storage in Linux

- Data storage on hard disk
 - Data in a hard disk are stored on a magnetic flat plate
 - Disk's surface needs to be partitioned and labeled so that computer can go directly to a specific point on it
 - Achieve by low level **formatting** the disk
 - Create magnetic concentric circles called **tracks**
 - Each track is split into smaller parts called **sectors** and numbered
- Each sector: hold 512 bytes data
 - E.g. 80 tracks (from outer to inner 0 .. 79), 18 sectors disk can store 80x18x512 bytes data.



Formatted Disk

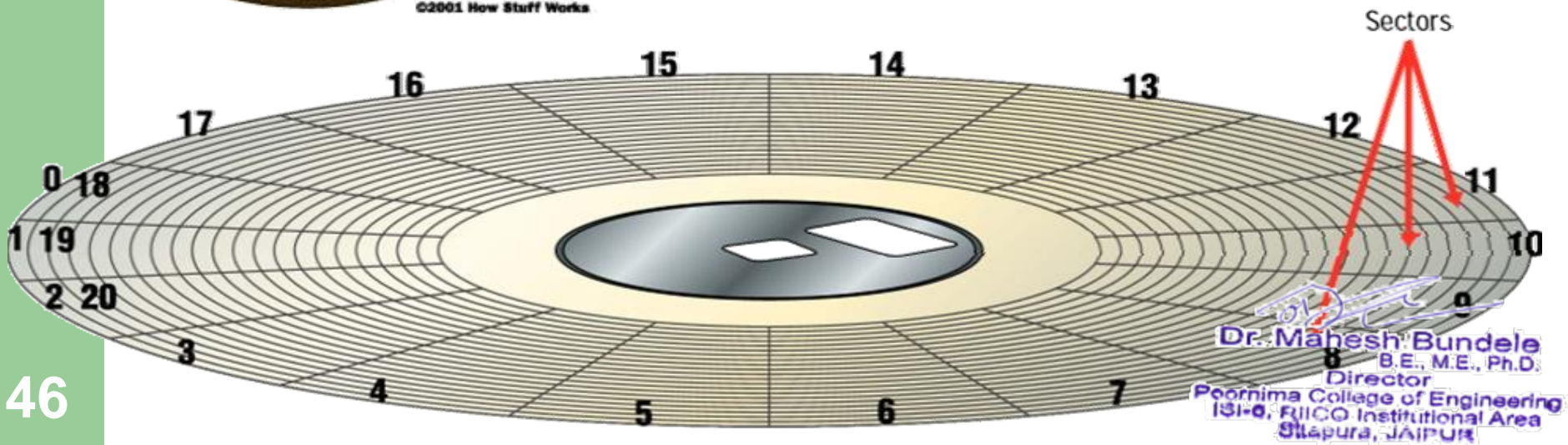


©2001 How Stuff Works

Sector

Track

Density of data is higher for inner tracks than outer tracks





2. Operating System Case Study: Linux

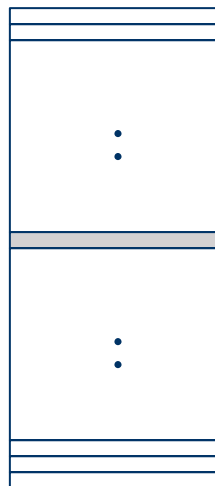
- Must read or write whole sector at a time
- OS allocates groups of sectors called cluster to files
- Files smaller than the cluster will still be allocated the whole cluster, but the rest left unused
- In Linux, every file is associated with an **inode** that records its location in the disk
- The inode of all files are put together in a data structure called **inode table**
- In the **directory**, every file is associated with a **inode number** that points to an entry of the **inode table**



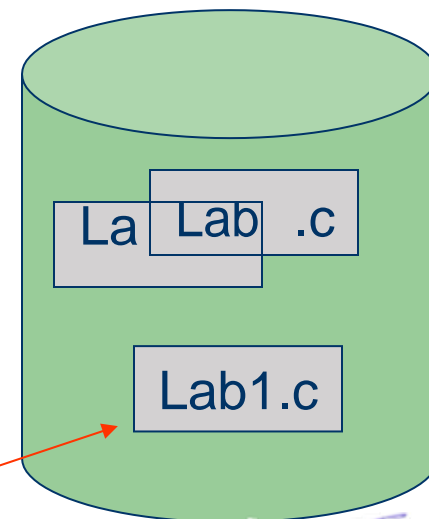
2. Operating System Case Study: Linux

Contents of the directory /home/dlun

1076	...
2083	...
13059	lab1.c
17488	lab2.c
18995	lab3.c



Number of links
File mode
User ID
Time created
Time last updated
:
Location on disk





Case study: Linux

B.3 Process management

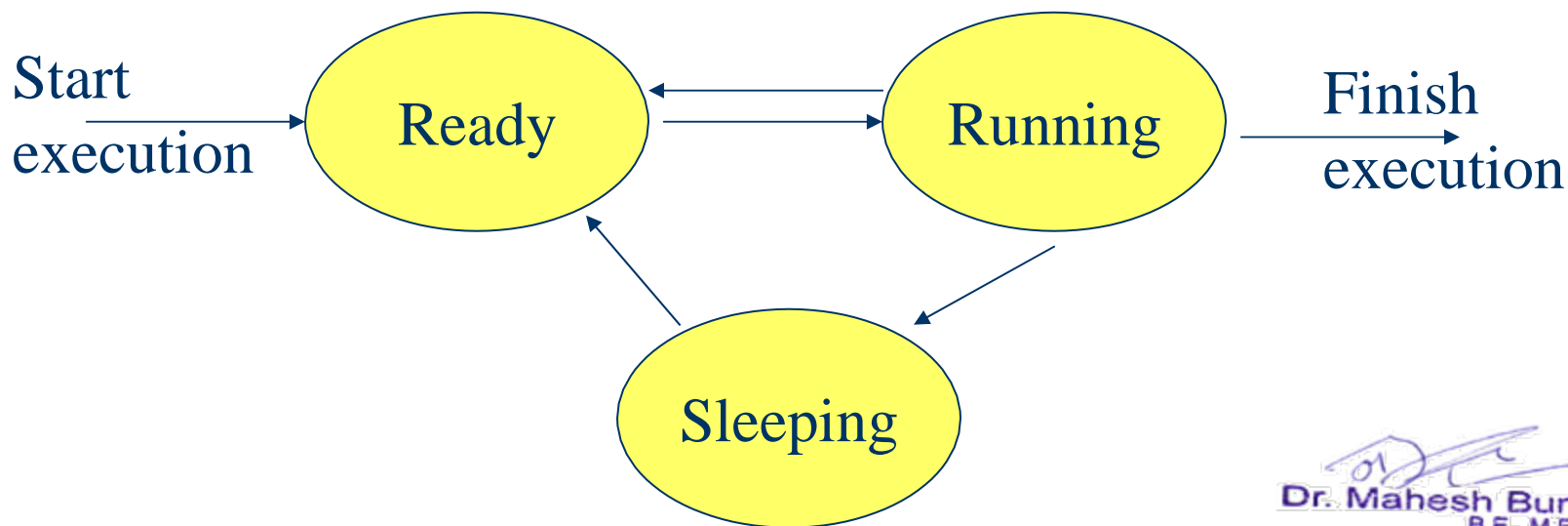


Linux Process Management

- Linux is a **multitasking** system
- Multiple programs can be executed at the same time
- Ultimately, a program needs to be executed by a CPU
- If there is only one CPU, how multiple programs can be executed at the same time?
⇒ By **time sharing**
- That is, all programs are claimed to be **executing**.
In fact, most of them are **waiting** for the CPU.



- A program that is claimed to be executing is called a **process**
- For a multitasking system, a process has at least the following three **states**:





2. Operating System Case Study: Linux

- Ready state
 - All processes that are ready to execute but **without the CPU** are at the ready state
 - If there is only 1 CPU in the system, all processes except one are at the ready state
- Running state
 - The process that **actually possesses the CPU** is at the running state
 - If there is only 1 CPU in the system, at most there is only one process is at the running state
- Sleeping state
 - The process that is **waiting for other resource** I/O, is at the sleeping state

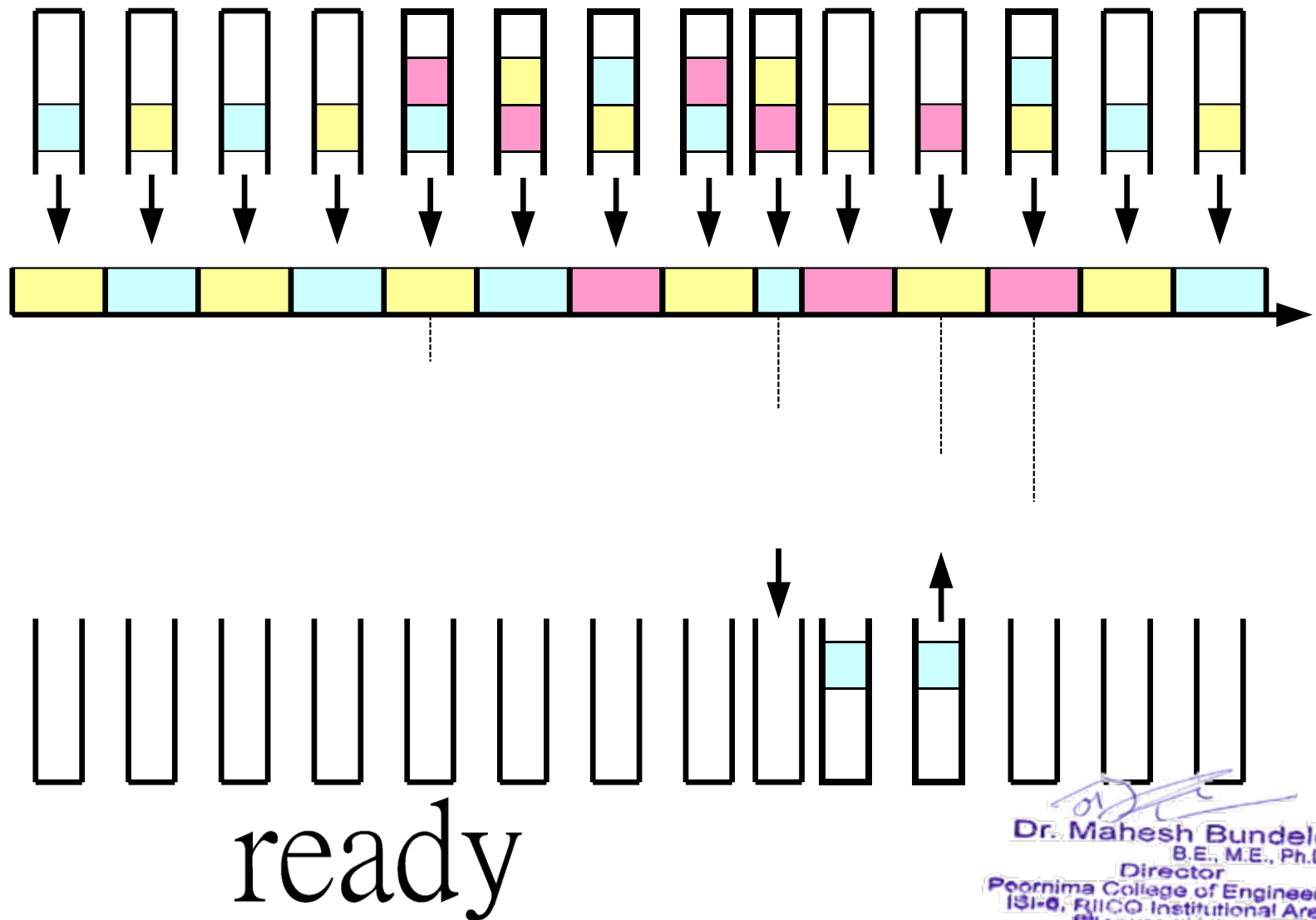


2. Operating System Case Study: Linux

- Processes will alternatively get into the CPU one after the other (called the **round robin scheme**)
- A process will be “in” a CPU for a very short time (**quantum**)
 - For Linux, each quantum is about 100msec
- At the time that a process is selected to be “in” the CPU
 - It goes from **ready state to running state**
- After that, it will be swapped out
 - It goes from **running state back to ready state**
- Or it may due to the waiting of an I/O device, e.g. mouse
 - It goes from **running state to sleeping state**
- When obtaining the required resource
 - It goes from **sleeping state to ready state**



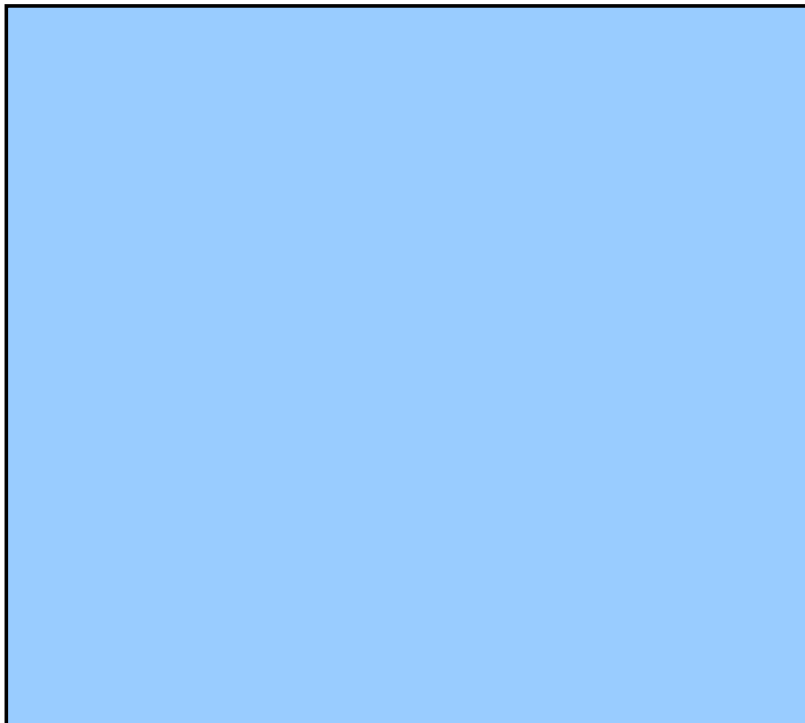
2. Operating System Case Study: Linux



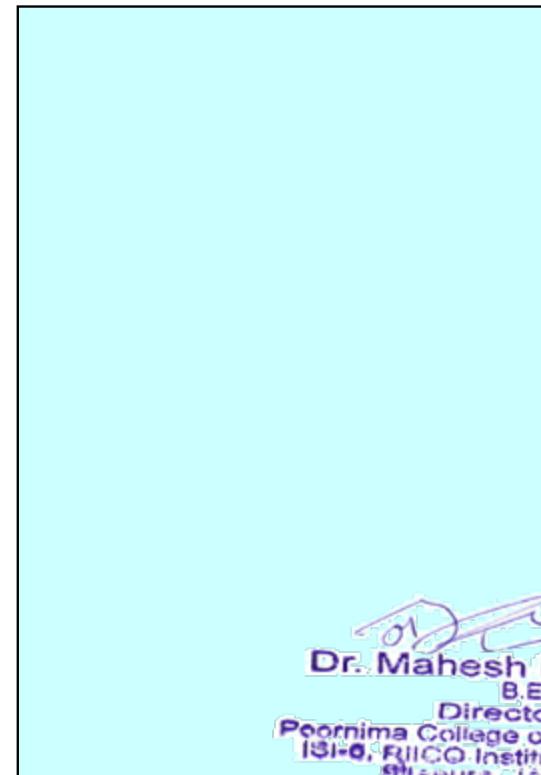


- The mechanism to determine which process should “get into” the CPU is called **Process scheduling**
- For example,

Program A



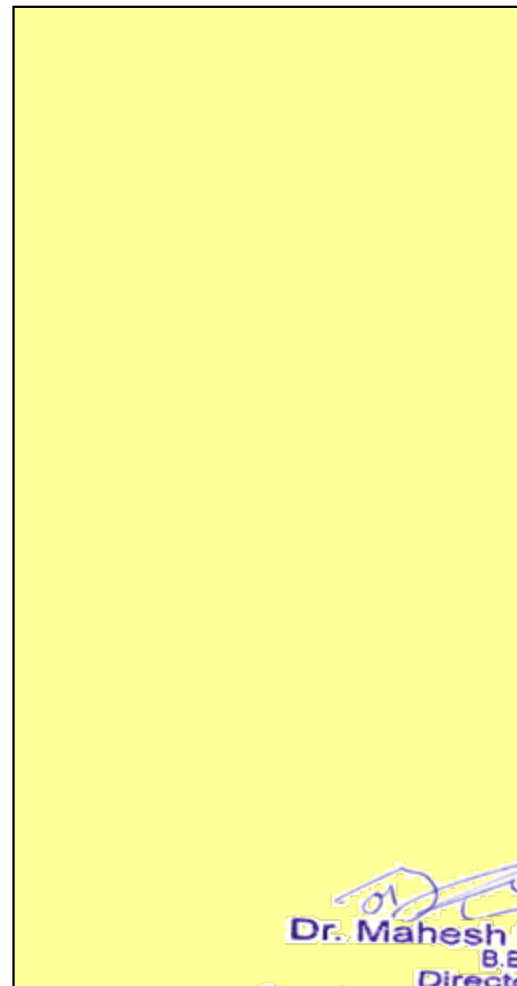
Actual sequence of operations





Actual sequence of operations

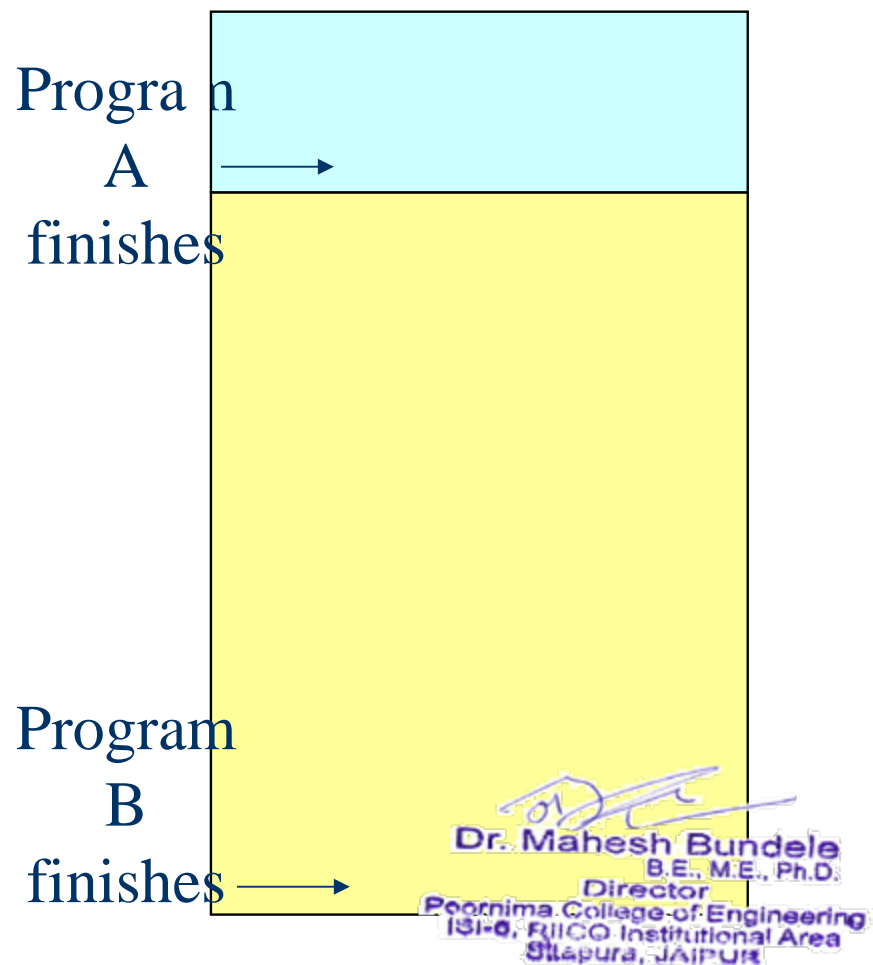
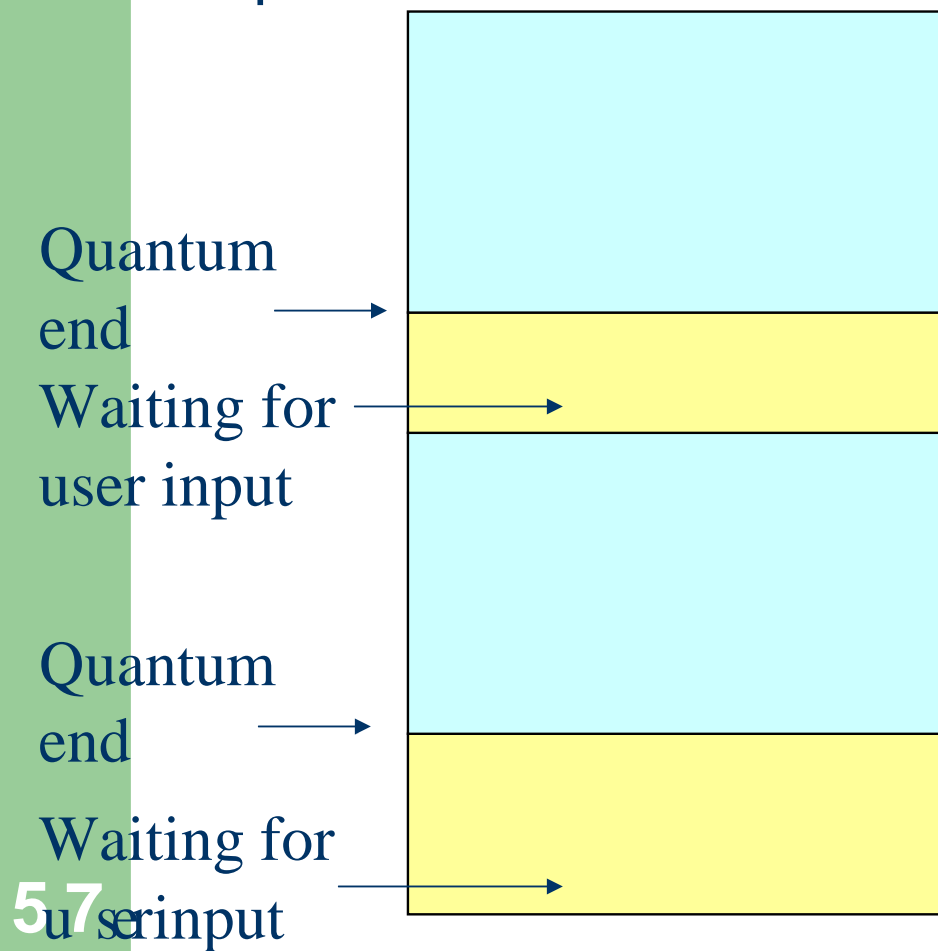
Program B





2. Operating System Case Study: Linux

- Program A and B will be at the running state alternatively, depends on the quantum size and the availability of the required resource





2. Operating System Case Study: Linux

Terminal pts/0 has the editor **vi** running

Terminal pts/1 is executing **ps** to see the processes of both terminals

The processes of a system can be seen by using the command

ps

```
dlun@enpkun.polyu.edu.hk: /home/dlun/Desktop
File Edit Settings Help
It is a test.
It is another test.

dlun@enpkun.polyu.edu.hk: /home/dlun
File Edit Settings Help
[dlun@enpkun dlun]$ ps -t pts/0,pts/1 a
  PID TTY          STAT       TIME COMMAND
 14748 pts/1        S           0:00 -bash
 14795 pts/0        S           0:00 -bash
 14874 pts/0        S           0:00 vi test1.txt
 14876 pts/1        R           0:00 ps -t pts/0,pts/1
[dlun@enpkun dlun]$
```



2. Operating System Case Study: Linux

PID	TTY	STAT	TIME	COMMAND
14748	pts/1	S	0:00	-bash
14795	pts/0	S	0:00	-bash
14974	pts/0	S	0:00	vi test1.txt
14876	pts/1	R	0:00	ps ...

Process ID

Terminal
name

State:

S – Sleeping
(waiting for input)

R – Running

How much time the
process is continuously
executing



- For the example above, both bash processes, which are the shell of both terminals, are waiting for the input of user. They must be in the **sleeping state**
- The vi process, which is an editor, is also waiting for the input of user. Hence it is also in **sleeping state**
- When ps reporting the processes in the system, it is the only process that is running. Hence it is in **running state**



- A process can be forced to terminate by using the command **kill -9 PID**

```
dlun@enpkun.polyu.edu.hk: /home/dlun
File Edit Settings Help

[dlun@enpkun dlun]$ ps -t pts/0,pts/1 a
  PID TTY          STAT       TIME COMMAND
 14748 pts/1        S           0:00 -bash
 14795 pts/0        S           0:00 -bash
 14874 pts/0        S           0:00 vi test1.txt
 14876 pts/1        R           0:00 ps -t pts/0,pts/1 a
[dlun@enpkun dlun]$
[dlun@enpkun dlun]$
[dlun@enpkun dlun]$ kill -9 14874
[dlun@enpkun dlun]$ ps -t pts/0,pts/1 a
  PID TTY          STAT       TIME COMMAND
 14748 pts/1        S           0:00 -bash
 14795 pts/0        S           0:00 -bash
 14891 pts/1        R           0:00 ps -t pts/0,pts/1 a
[dlun@enpkun dlun]$
```

The vi process is terminated by using the command **kill -9 14874**

Dr. Mahesh Bunde
B.E., M.E., Ph.D.
Director
Peernima College of Engineering
ISI-0, FIICO Institutional Area
Sitapura, JAIPUR