MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

L 1

LECTURE HANDOUTS

CSE **II/IV**

Course Name with Code	**: Operating Systems & 16CSD05**
Course Faculty	**: Mr.S.Pragadeeswaran**
Unit	**: I Introduction**

Date of Lecture:

Topic of Lecture: What Operating System Do , Operating System Structure

Introduction : (Maximum 5 sentences)

- A program that acts as an intermediary between a user of a computer and the computer hardware

Operating system goals:

- Execute user programs and make solving user problems easier
- Make the computer system convenient to use
- Use the computer hardware in an efficient manner

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:
What is an Operating System?

A program that acts as an intermediary between a user of a computer and the computer hardware

Operating system goals:

Execute user programs and make solving user problems easier
Make the computer system convenient to use
Use the computer hardware in an efficient manner

Computer System Structure
Computer system can be divided into four components

Hardware – provides basic computing resources CPU, memory, I/O devices
Operating system-Controls and coordinates use of hardware among various applications and users
Application programs – define the ways in which the system resources are used to solve the

computing problems of the users
-Word processors, compilers, web browsers, database systems, video games

Users
- People, machines, other computers

Process Management

- ✓ A process is a program in execution. It is a unit of work within the system. Program is a passive entity, process is an active entity.
- ✓ Process needs resources to accomplish its task
- ✓ CPU, memory, I/O, files
- ✓ Initialization data
- ✓ Process termination requires reclaim of any reusable resources
- ✓ Single-threaded process has one program counter specifying location of next instruction to execute
- ✓ Process executes instructions sequentially, one at a time, until completion
- ✓ Multi-threaded process has one program counter per thread
- ✓ Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
- ✓ Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

- ✓ The operating system is responsible for the following activities in connection with process management:
- ✓ Creating and deleting both user and system processes
- ✓ Suspending and resuming processes
- ✓ Providing mechanisms for process synchronization
- ✓ Providing mechanisms for process communication
- ✓ Providing mechanisms for deadlock handling

Memory Management

- ✓ All data in memory before and after processing
- ✓ All instructions in memory in order to execute
- ✓ Memory management determines what is in memory when
- ✓ Optimizing CPU utilization and computer response to users

Memory management activities

- ✓ Keeping track of which parts of memory are currently being used and by whom
- ✓ Deciding which processes (or parts thereof) and data to move into and out of memory
- ✓ Allocating and deal locating memory space as needed

Storage Management

- ✓ OS provides uniform, logical view of information storage
- ✓ Abstracts physical properties to logical storage unit - **file**
- ✓ Each medium is controlled by device (i.e., disk drive, tape drive)
- ✓ Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- ✓ File-System management
- ✓ Files usually organized into directories
- ✓ Access control on most systems to determine who can access what

OS activities include

- ✓ Creating and deleting files and directories
- ✓ Primitives to manipulate files and dirs
- ✓ Mapping files onto secondary storage
- ✓ Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- ✓ Usually disks used to store data that does not fit in main memory or data that must be kept for a "long" period of time
- ✓ Proper management is of central importance
- ✓ Entire speed of computer operation hinges on disk subsystem and its algorithms

MASS STORAGE activities

- ✓ Free-space management
- ✓ Storage allocation
- ✓ Disk scheduling
- ✓ Some storage need not be fast
- ✓ Tertiary storage includes optical storage, magnetic tape
- ✓ Still must be managed
- ✓ Varies between WORM (write-once, read-many-times) and RW (read-write)

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(04-54).

Course Teacher

Verified by HOD

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **I Introduction**

Date of Lecture:

Topic of Lecture: Operating system Operations , Operating System Components: Process Management

Introduction : (Maximum 5 sentences)

Operating-System Operations

- ✓ Modern operating systems are interrupt driven. If there are no processes to execute, no I/O devices to service, and no users to whom to respond, an operating system will sit quietly, waiting for something to happen. Events are almost always signaled by the occurrence of an interrupt or a trap
- ✓ A trap (or an exception) is a software-generated interrupt caused
- ✓ The interrupt-driven nature of an operating system defines that system's general structure. For each type of interrupt, separate segments of code in the operating system determine what action should be taken. An interrupt service routine is provided that is responsible for dealing with the interrupt.
- ✓ The operating system and the users share the hardware and software resources of the computer system, we need to make sure that an error in a user program could cause problems only for the one program that was running. With sharing, many processes could be adversely affected by a bug in one program. For example, if a process gets stuck in an infinite loop, this loop could prevent the correct operation of many other processes.
- ✓ Without protection against these sorts of errors, either the computer must execute only one process at a time or all output must be suspect.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Operating-System Operations

- ✓ Modern operating systems are interrupt driven**.** If there are no processes to execute, no I/O devices to service, and no users to whom to respond, an operating system will sit quietly, waiting for something to happen. Events are almost always signaled by the occurrence of an interrupt or a trap

- ✓ A trap (or an exception) is a software-generated interrupt caused
- ✓ The interrupt-driven nature of an operating system defines that system's general structure. For each type of interrupt, separate segments of code in the operating system determine what action should be taken. An interrupt service routine is provided that is responsible for dealing with the interrupt.
- ✓ The operating system and the users share the hardware and software resources of the computer system, we need to make sure that an error in a user program could cause problems only for the one program that was running. With sharing, many processes could be adversely affected by a bug in one program. For example, if a process gets stuck in an infinite loop, this loop could prevent the correct operation of many other processes.
- ✓ Without protection against these sorts of errors, either the computer must execute only one process at a time or all output must be suspect.

Dual-Mode Operation

- o **Dual-mode** operation allows OS to protect itself and other system components **User mode** and **kernel mode**
- o **Mode bit** provided by hardware Provides ability to distinguish when system is running user code or kernel code Some instructions designated as **privileged**, only executable in kernel mode System call changes mode to kernel, return from call resets it to user

Transition from User to Kernel Mode

- ✓ Timer to prevent infinite loop / process hogging resources Set interrupt after specific period
- ✓ Operating system decrements counter
- ✓ When counter zero generate an interrupt
- ✓ Set up before scheduling process to regain control or terminate program that exceeds allotted time

Micro kernels

- o The kernel became large and difficult to manage. In the mid-1980s, researchers at Carnegie Mellon University developed an operating system called **Mach** that modularized the kernel using the **microkernel** approach.
- o This method structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs. The result is a smaller kernel. microkernel provide minimal process and memory management, in addition to a communication facility.
- o The main function of the microkernel is to provide a communication facility between the client program and the various services that are also running in user space.
- o One benefit of the microkernel approach is ease of extending the operating system. All new services are added to user space and consequently do not require modification of the kernel. When the kernel does have to be modified, the changes tend to be fewer, because the microkernel is a smaller kernel.
- o The resulting operating system is easier to port from one hardware design to another. The microkernel also provides more security and reliability, since most services are running as user rather than kernel processes. If a service fails, the rest of the operating system remains untouched.

Layered Approach

- ✓ The operating system can then retain much greater control over the computer and over the applications that make use of that computer.
- ✓ Implementers have more freedom in changing the inner workings of the system and in creating modular operating systems. Under the top down approach, the overall functionality and features are determined and are separated into components.

- ✓ Information hiding is also important, because it leaves programmers free to implement the low-level routines as they see fit, provided that the external interface of the routine stays unchanged and that the routine itself performs the advertised task.

- ✓ A system can be made modular in many ways. One method is the **layered approach,** in which the operating system is broken up into a number of layers (levels). The bottom layer (layer 0) is the hardware; the highest (layer N) is the user interface.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.1

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(04-54).

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

LECTURE HANDOUTS

L 3

CSE **II/IV**

Course Name with Code : Operating Systems & 16CSD05

Course Faculty : Mr.S.Pragadeeswaran

Unit : I Introduction

Date of Lecture:

Topic of Lecture: Memory Management , Storage Management , I/O Management
Introduction : (Maximum 5 sentences) **Memory Management** ✓ All data in memory before and after processing ✓ All instructions in memory in order to execute ✓ Memory management determines what is in memory when ✓ Optimizing CPU utilization and computer response to users **Memory management activities** ✓ Keeping track of which parts of memory are currently being used and by whom ✓ Deciding which processes (or parts thereof) and data to move into and out of memory ✓ Allocating and deal locating memory space as needed
Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics) • Operating Systems • Hardware • Software • Process

Detailed content of the Lecture:

What is an Operating System?

- ✓ A program that acts as an intermediary between a user of a computer and the computer hardware

Operating system goals:

- ✓ Execute user programs and make solving user problems easier
- ✓ Make the computer system convenient to use
- ✓ Use the computer hardware in an efficient manner

Computer System Structure

Computer system can be divided into four components

Hardware – provides basic computing resources CPU, memory, I/O devices
Operating system-Controls and coordinates use of hardware among various applications and users
Application programs – define the ways in which the system resources are used to solve the computing problems of the users
-Word processors, compilers, web browsers, database systems, video games
Users
- People, machines, other computers

Operating-System Structure

Simple Structure

- ✓ Many commercial systems do not have well-defined structures. Frequently, such operating systems started as small, simple, and limited systems and then grew beyond their original scope.
- ✓ MS-DOS is an example of such a system. It was written to provide the most functionality in the least space, so it was not divided into modules carefully.
- ✓ In MS-DOS, the interfaces and levels of functionality are not well separated. For instance, application programs are able to access the basic I/O routines to write directly to the display and disk drives.
- ✓ Such freedom leaves MS-DOS vulnerable to errant (or malicious) programs, causing entire system crashes when user programs fail. Of course, MS-DOS was also limited by the hardware of its era.
- ✓ Another example of limited structuring is the original UNIX operating system. UNIX is another system that initially was limited by hardware functionality.

Process Management Activities

- ✓ The operating system is responsible for the following activities in connection with process management:
- ✓ Creating and deleting both user and system processes
- ✓ Suspending and resuming processes
- ✓ Providing mechanisms for process synchronization
- ✓ Providing mechanisms for process communication
- ✓ Providing mechanisms for deadlock handling

Memory Management

- ✓ All data in memory before and after processing
- ✓ All instructions in memory in order to execute
- ✓ Memory management determines what is in memory when
- ✓ Optimizing CPU utilization and computer response to users

Memory management activities

- ✓ Keeping track of which parts of memory are currently being used and by whom
- ✓ Deciding which processes (or parts thereof) and data to move into and out of memory
- ✓ Allocating and deal locating memory space as needed

Storage Management

- ✓ OS provides uniform, logical view of information storage
- ✓ Abstracts physical properties to logical storage unit - **file**
- ✓ Each medium is controlled by device (i.e., disk drive, tape drive)
- ✓ Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- ✓ File-System management
- ✓ Files usually organized into directories
- ✓ Access control on most systems to determine who can access what

Mass-Storage Management

- ✓ Usually disks used to store data that does not fit in main memory or data that must be kept for a "long" period of time
- ✓ Proper management is of central importance
- ✓ Entire speed of computer operation hinges on disk subsystem and its algorithms

MASS STORAGE activities

- ✓ Free-space management
- ✓ Storage allocation
- ✓ Disk scheduling
- ✓ Some storage need not be fast
- ✓ Tertiary storage includes optical storage, magnetic tape
- ✓ Still must be managed
- ✓ Varies between WORM (write-once, read-many-times) and RW (read-write)

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concep,-9th 2012.12.

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(04-54).

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L 4

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **I Introduction**

Date of Lecture:

Topic of Lecture: Network Management , Protection and Security

Introduction : (Maximum 5 sentences)

Protection and security

- ✓ **Protection** is any mechanism for controlling the access of processes or users to the resources defined by a computer system. This mechanism must provide means for specification of the controls to be imposed and means for enforcement.
- ✓ Protection can improve reliability by detecting latent errors at the interfaces between component subsystems.
- ✓ Early detection of interface errors can often prevent contamination of a healthy subsystem by another subsystem that is malfunctioning.
- ✓ An unprotected resource cannot defend against use (or misuse) by an unauthorized or incompetent user.
- ✓ A protection-oriented system provides a means to distinguish between authorized and unauthorized usage, A system can have adequate protection but still
- ✓ be prone to failure and allow inappropriate access

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Security

The Security Problem

- ✓ Protection dealt with protecting files and other resources from accidental misuse by cooperating users sharing a system, generally using the computer for normal purposes.

- ✓ Security deals with protecting systems from deliberate attacks, either internal or external, from individuals intentionally attempting to steal information, damage information, or otherwise deliberately wreak havoc in some manner.

Some of the most common types of **violations** include:

- ✓ **Breach of Confidentiality -** Theft of private or confidential information, such as credit-card numbers, trade secrets, patents, secret formulas, manufacturing
- ✓ procedures, medical information, financial information, etc.

Breach of Integrity - Unauthorized **modification** of data, which may have serious indirect consequences. For example a popular game or other program's source code could be modified to open up security holes on users systems before being released to the public.

Breach of Availability - Unauthorized **destruction** of data, often just for the "fun" of causing havoc and for bragging rites. Vandalism of web sites is a common form of this violation.

Theft of Service - Unauthorized use of resources, such as theft of CPU cycles, installation of daemons running an unauthorized file server, or tapping into the target's telephone or networking services.

Denial of Service, DOS - Preventing legitimate users from using the system, often by overloading and overwhelming the system with an excess of requests for service.

Program Threats

There are many common threats to modern systems. Only a few are discussed here.

Trojan Horse

A **Trojan Horse** is a program that secretly performs some maliciousness in addition to its visible actions.

- ✓ Some Trojan horses are deliberately written as such, and others are the result of legitimate programs that have become infected with **viruses,** (see below.)
- ✓ One dangerous opening for Trojan horses is long search paths, and in particular paths which include the current directory (“.”) as part of the path. If a dangerous program having the same name as a legitimate program (or a common mis-spelling, such as "sl" instead of "ls”) is placed anywhere on the path, then an unsuspecting user may be fooled into running the wrong program by mistake.

- ✓ Another classic Trojan Horse is a login emulator, which records a users account name and password, issues a "password incorrect" message, and then logs off the system. The user then tries again (with a proper login prompt), logs in successfully, and doesn't realize that their information has been stolen.
- ✓ Two solutions to Trojan Horses are to have the system print usage statistics on logouts, and to require the typing of non-trappable key sequences such as Control-Alt-Delete in order to log in. (This is why modern Windows systems require the Control-Alt-Delete sequence to commence logging in, which cannot be emulated or caught by ordinary programs. I.e. that key sequence always transfers control over to the operating system.)

Spy ware is a version of a Trojan Horse that is often included in "free" software downloaded off the Internet. Spy ware programs generate pop-up browser windows, and may also accumulate information about the user and deliver it to some central site. (This is an example of **covert channels,** in which surreptitious communications occur.) Another common task of spyware is to send out spam e-mail messages, which then purportedly come from the infected user.

Trap Door

A **Trap Door** is when a designer or a programmer (or hacker) deliberately inserts a security hole that they can use later to access the system.

Because of the possibility of trap doors, once a system has been in an untrustworthy state, that system can never be trusted again. Even the backup tapes may contain a copy of some cleverly hidden back door.

A clever trap door could be inserted into a compiler, so that any programs compiled with that compiler would contain a security hole. This is especially dangerous, because inspection of the code being compiled would not reveal any problems.

Logic Bomb

A **Logic Bomb** is code that is not designed to cause havoc all the time, but only when a certain set of circumstances occurs, such as when a particular date or time is reached or some other noticeable event.

A classic example is the **Dead-Man Switch**, which is designed to check whether a certain person (e.g. the author) is logging in every day, and if they don't log in for a long time (presumably because they've been fired), then the logic bomb goes off and either opens up security holes or causes other problems.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts-9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(04-54).

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L 5

CSE

II/IV

Course Name with Code : Operating Systems & 16CSD05

Course Faculty : Mr.S.Pragadeeswaran

Unit : I Introduction

Date of Lecture:

Topic of Lecture: Network Management , Protection and Security

Introduction : (Maximum 5 sentences)

The Security Problem

- ✓ Protection dealt with protecting files and other resources from accidental misuse by cooperating users sharing a system, generally using the computer for normal purposes.
- ✓ Security deals with protecting systems from deliberate attacks, either internal or external, from individuals intentionally attempting to steal information, damage information, or otherwise deliberately wreak havoc in some manner.

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:
Security

The Security Problem

- ✓ Protection dealt with protecting files and other resources from accidental misuse by cooperating users sharing a system, generally using the computer for normal purposes.
- ✓ Security deals with protecting systems from deliberate attacks, either internal or external, from individuals intentionally attempting to steal information, damage information, or otherwise deliberately wreak havoc in some manner.

Some of the most common types of **violations** include:

Breach of Confidentiality - Theft of private or confidential information, such as credit-card numbers, trade secrets, patents, secret formulas, manufacturing procedures, medical information, financial information, etc.

Breach of Inegrity - Unauthorized **modification** of data, which may have serious indirect consequences. For example a popular game or other program's source code could be modified to open up security holes on users systems before being released to the public.

Breach of Availability - Unauthorized **destruction** of data, often just for the "fun" of causing havoc and for bragging rites. Vandalism of web sites is a common form of this violation.

Theft of Service - Unauthorized use of resources, such as theft of CPU cycles, installation of daemons running an unauthorized file server, or tapping into the target's telephone or networking services.

Denial of Service, DOS - Preventing legitimate users from using the system, often by overloading and overwhelming the system with an excess of requests for service.

Program Threats

There are many common threats to modern systems. Only a few are discussed here.

Trojan Horse

A **Trojan Horse** is a program that secretly performs some maliciousness in addition to its visible actions.

- ✓ Some Trojan horses are deliberately written as such, and others are the result of legitimate programs that have become infected with **viruses,** (see below.)
- ✓ One dangerous opening for Trojan horses is long search paths, and in particular paths which include the current directory (".") as part of the path. If a dangerous program having the same name as a legitimate program (or a common mis-spelling, such as "sl" instead of "ls") is placed anywhere on the path, then an unsuspecting user may be fooled into running the wrong program by mistake.

- ✓ Another classic Trojan Horse is a login emulator, which records a users account name and password, issues a "password incorrect" message, and then logs off the system. The user then tries again (with a proper login prompt), logs in successfully, and doesn't realize that their information has been stolen.
- ✓ Two solutions to Trojan Horses are to have the system print usage statistics on logouts, and to require the typing of non-trappable key sequences such as Control-Alt-Delete in order to log in. (This is why modern Windows systems require the Control-Alt-Delete sequence to commence logging in, which cannot be emulated or caught by ordinary programs. I.e. that key sequence always transfers control over to the operating system.)
- ✓ **Spy ware** is a version of a Trojan Horse that is often included in "free" software downloaded off the Internet. Spy ware programs generate pop-up browser windows, and may also accumulate information about the user and deliver it to some central site. (This is an example of **covert channels,** in which surreptitious communications occur.) Another common task of spyware is to send out spam e-mail messages, which then purportedly come from the infected user.

Trap Door

A **Trap Door** is when a designer or a programmer (or hacker) deliberately inserts a security hole that they can use later to access the system.

Because of the possibility of trap doors, once a system has been in an untrustworthy state, that system can never be trusted again. Even the backup tapes may contain a copy of some cleverly hidden back door.

A clever trap door could be inserted into a compiler, so that any programs compiled with that compiler would contain a security hole. This is especially dangerous, because inspection of the code being compiled would not reveal any problems.

Logic Bomb

A **Logic Bomb** is code that is not designed to cause havoc all the time, but only when a certain set of circumstances occurs, such as when a particular date or time is reached or some other noticeable event.

A classic example is the **Dead-Man Switch**, which is designed to check whether a certain person (e.g. the author) is logging in every day, and if they don't log in for a long time (presumably because they've been fired), then the logic bomb goes off and either opens up security holes or causes other problem

Viruses

A virus is a fragment of code embedded in an otherwise legitimate program, designed to replicate itself (by infecting other programs), and (eventually) wreaking havoc.

Viruses are more likely to infect PCs than UNIX or other multi-user systems, because programs in the latter systems have limited authority to modify other programs or to access critical system structures (such as the boot block.)

Viruses are delivered to systems in a **virus dropper,** usually some form of a Trojan Horse, and usually via e-mail or unsafe downloads.

Viruses take many forms

✓ **Polymorphic** viruses change every time they spread - Not their underlying functionality, but just their **signature,** by which virus checkers recognize them.

✓ **Encrypted** viruses travel in encrypted form to escape detection. In practice they are self-decrypting, which then allows them to infect other files.

Stealth viruses try to avoid detection by modifying parts of the system that could be used to detect it. For example the read ( ) system call could be modified so that if an infected file is read the infected part gets skipped and the reader would see the original unadulterated file.

In 2004 a virus exploited three bugs in Microsoft products to infect hundreds of Windows servers (including many trusted sites) running Microsoft Internet Information Server, which in turn infected any Microsoft Internet Explorer web browser that visited any of the infected server sites.

One of the back-door programs it installed was a **keystroke logger,** which records user’s keystrokes, including passwords and other sensitive information.There is some debate in the computing community as to whether a **monoculture,** in which nearly all systems run the same hardware, operating system, and applications, increases the threat of viruses and the potential for harm caused by them.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(04-54).

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

L 6

LECTURE HANDOUTS

CSE **II/IV**

Course Name with Code : Operating Systems & 16CSD05

Course Faculty : Mr.S.Pragadeeswaran

Unit : I Introduction

Date of Lecture:

Topic of Lecture: Multiprocessor Systems ,Desktop Systems, Distributed Systems

Introduction : (Maximum 5 sentences)

Multiprocessor Systems

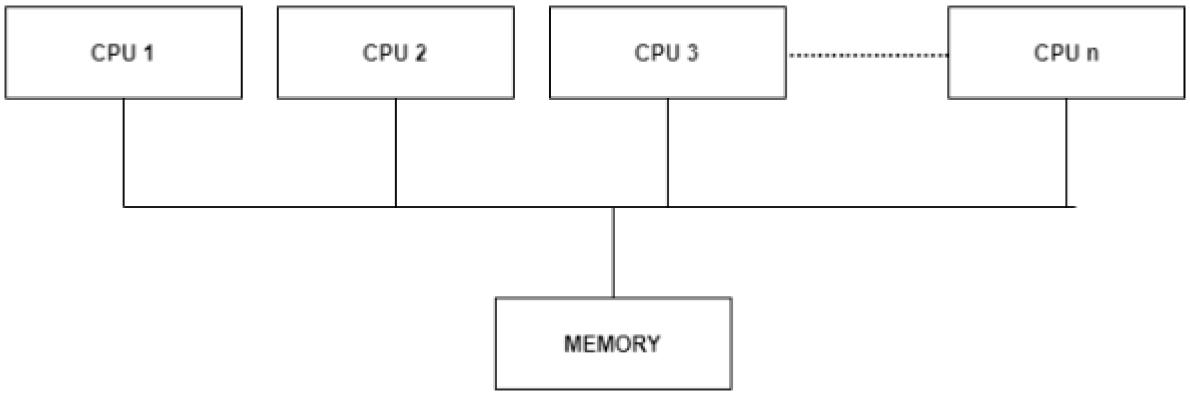
- ✓ Most computer systems are single processor systems i.e they only have one processor. However, multiprocessor or parallel systems are increasing in importance nowadays.
- ✓ These systems have multiple processors working in parallel that share the computer clock, memory, bus, peripheral devices etc.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Most computer systems are single processor systems i.e they only have one processor. However, multiprocessor or parallel systems are increasing in importance nowadays. These systems have multiple processors working in parallel that share the computer clock, memory, bus, peripheral devices etc.



Multiprocessing Architecture

Types of Multiprocessors

There are mainly two types of multiprocessors i.e. symmetric and asymmetric multiprocessors. Details about them are as follows −

Symmetric Multiprocessors

In these types of systems, each processor contains a similar copy of the operating system and they all communicate with each other. All the processors are in a peer to peer relationship i.e. no master - slave relationship exists between them.

An example of the symmetric multiprocessing system is the Encore version of Unix for the Multimax Computer.

Asymmetric Multiprocessors

In asymmetric systems, each processor is given a predefined task. There is a master processor that gives instruction to all the other processors. Asymmetric multiprocessor system contains a master slave relationship.

Asymmetric multiprocessor was the only type of multiprocessor available before symmetric multiprocessors were created. Now also, this is the cheaper option.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(04-54).

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 7

CSE

II/IV

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **I Introduction**

Date of Lecture:

Topic of Lecture: Clustered Systems ,Real-Time Systems ,Handheld Systems , Open Source Operating Systems

Introduction : (Maximum 5 sentences)

- ✓ Clustered systems are similar to parallel systems as they both have multiple CPUs.
- ✓ However a major difference is that clustered systems are created by two or more individual computer systems merged together.
- ✓ Basically, they have independent computer systems with a common storage and the systems work together.

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Types of Clustered Systems

There are primarily two types of clustered systems i.e. asymmetric clustering system and symmetric clustering system. Details about these are given as follows −

Asymmetric Clustering System

In this system, one of the nodes in the clustered system is in hot standby mode and all the others run the required applications. The hot standby mode is a failsafe in which a hot standby node is part of the system . The hot standby node continuously monitors the server and if it fails, the hot standby node takes its place.

Symmetric Clustering System

In symmetric clustering system two or more nodes all run applications as well as monitor each other. This is more efficient than asymmetric system as it uses all the hardware and doesn't keep a node merely as a hot standby.

Attributes of Clustered Systems

There are many different purposes that a clustered system can be used for. Some of these can be scientific calculations, web support etc. The clustering systems that embody some major attributes are

Load Balancing Clusters

In this type of clusters, the nodes in the system share the workload to provide a better performance. For example: A web based cluster may assign different web queries to different nodes so that the system performance is optimized. Some clustered systems use a round robin mechanism to assign requests to different nodes in the system.

High Availability Clusters

These clusters improve the availability of the clustered system. They have extra nodes which are only used if some of the system components fail. So, high availability clusters remove single points of failure i.e. nodes whose failure leads to the failure of the system. These types of clusters are also known as failover clusters or HA clusters.

Benefits of Clustered Systems

The difference benefits of clustered systems are as follows −

- **Performance**

Clustered systems result in high performance as they contain two or more individual computer systems merged together. These work as a parallel unit and result in much better performance for the system.

- **Fault Tolerance**

Clustered systems are quite fault tolerant and the loss of one node does not result in the loss of the system. They may even contain one or more nodes in hot standby mode which allows them to take the place of failed nodes.

- **Scalability**

Clustered systems are quite scalable as it is easy to add a new node to the system. There is no need to take the entire cluster down to add a new node.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(04-54).

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L 8

CSE

II/IV

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **I Introduction**

Date of Lecture:

Topic of Lecture: Operating System Structures**:** Operating System Services

Introduction : (Maximum 5 sentences)

An Operating System provides services to both the users and to the programs.

- It provides programs an environment to execute.

  It provides users the services to execute the programs in a convenient manner

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

What is an Operating System?

An Operating System provides services to both the users and to the programs.

- It provides programs an environment to execute.
- It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system −

- Program execution
- I/O operations
- File System manipulation
- Communication
- Error Detection
- Resource Allocation

- Protection

Program execution

Operating systems handle many kinds of activities from user programs to system programs like printer spooler, name servers, file server, etc. Each of these activities is encapsulated as a process.

A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

I/O Operation

An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users.

An Operating System manages the communication between user and device drivers.

- I/O operation means read or write operation with any file or any specific I/O device.
- Operating system provides the access to the required I/O device when required.

File system manipulation

A file represents a collection of related information. Computers can store files on the disk (secondary storage), for long-term storage purpose. Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, data transfer rate and data access methods.

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management −

- Program needs to read a file or write a file.
- The operating system gives the permission to the program for operation on file.
- Permission varies from read-only, read-write, denied and so on.
- Operating System provides an interface to the user to create/delete files.
- Operating System provides an interface to the user to create/delete directories.
- Operating System provides an interface to create the backup of file system.

Communication

In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network.The OS handles routing and connection strategies, and the problems of contention and security.

Following are the major activities of an operating system with respect to communication −

- Two processes often require data to be transferred between them
- Both the processes can be on one computer or on different computers, but are connected through a computer network.
- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

Error handling

Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling −

- The OS constantly checks for possible errors.
- The OS takes an appropriate action to ensure correct and consistent computing.

Resource Management

In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management −

- The OS manages all kinds of resources using schedulers.
- CPU scheduling algorithms are used for better utilization of CPU.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(04-54).

Course Teacher

Verified by HOD

L 9

LECTURE HANDOUTS

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **I Introduction**

Date of Lecture:

Topic of Lecture: User and Operating System Interface ,System Calls,Types of System Calls

Introduction : (Maximum 5 sentences)

The Role Of Operating System In The Computer

An operating system (OS) is a set of programs which ensures the interoperability of the hardware and software in your computer. The operating system enables, among other things,

- the identification and activation of devices connected to the computer,
- the installation and use of programs, and
- the handling of files.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

OPERATING SYSTEM AND USER INTERFACE
As already mentioned, in addition to the hardware, a computer also needs a set of programs—an operating system—to control the devices. This page will discuss the following:

- **There are different kinds of operating systems**: such as Windows, Linux and Mac OS
- **There are also different versions of these operating systems,** e.g. Windows 7, 8 and 10
- **Operating systems can be used with different user interfaces (UI)**: text user interfaces (TUI) and graphical user interfaces (GUI) as examples
- **Graphical user interfaces have many similarities in different operating systems**: such as the start menu, desktop etc.

When you can recognize the typical parts of each operating system's user interface, you will mostly be able to use both Windows and Linux as well as e.g. Mac OS.

The Role Of Operating System In The Computer

An operating system (OS) is a set of programs which ensures the interoperability of the hardware and software in your computer. The operating system enables, among other things,

- the identification and activation of devices connected to the computer,
- the installation and use of programs, and
- the handling of files.

Windows

The name of the Windows OS comes from the fact that programs are run in "windows": each program has its own window, and you can have several programs open at the same time. Windows is the most popular OS for home computers, and there are several versions of it. The newest version is Windows 10.

Linux And Unix

Linux is an open-source OS, which means that its program code is freely available to software developers. This is why thousands of programmers around the world have developed Linux, and it is considered the most tested OS in the world. Linux has been very much influenced by the commercial Unix OS.

In addition to servers, Linux is widely used in home computers, since there are a great number of free programs for it (for text and image processing, spreadsheets, publishing, etc.). Over the years, many different versions of Linux have become available for distribution, most of which are free for the user (such as Ubuntu, Fedora and Mint, to name a few).

Mac Os X

Apple's Mac computers have their own operating system, OS X. Most of the programs that are available for PCs are also available for Macs running under OS X, but these two types of computers cannot use the exact same programs: for example, you cannot install the Mac version of the Microsoft Office suite on a Windows computer. You can install other operating systems on Mac computers, but the OS X is only available for computers made by Apple. Apple's lighter portable devices (iPads, iPhones) use a light version of the same operating system, called iOS.

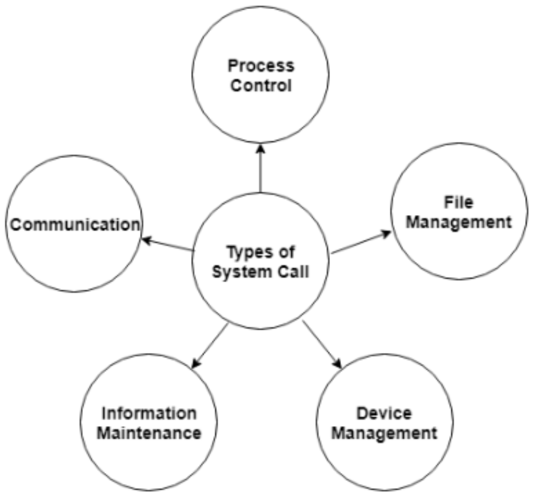
Mac computers are popular because OS X is considered fast, easy to learn and very stable and Apple's devices are considered well-designed—though rather expensive. See the additional reading material for more information on OS X.

Android

Android is an operating system designed for phones and other mobile devices. Android is not available for desktop computers, but in mobile devices it is extremely popular: more than a half of all mobile devices in the world run on Android.

Types of System Calls

There are mainly five types of system calls. These are explained in detail as follows −



Process Control

These system calls deal with processes such as process creation, process termination etc.

File Management

These system calls are responsible for file manipulation such as creating a file, reading a file, writing into a file etc.

Device Management

These system calls are responsible for device manipulation such as reading from device buffers, writing into device buffers etc.

Information Maintenance

These system calls handle information and its transfer between the operating system and the user program.

Communication

These system calls are useful for interprocess communication. They also deal with creating and deleting a communication connection.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(04-55).

Course Teacher

Verified by HOD

Estd. 2000

LECTURE HANDOUTS **L1**

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **Process Management and Threading**

Date of Lecture:

Topic of Lecture: Processes**:** Process concept ,Process scheduling,

Introduction : (Maximum 5 sentences)

- The OS maintains all PCBs in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue.
- The state of a process is changed; its PCB is unlinked from its current queue and moved to its new state queue.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Hardware
- Software
- Operating Systems
- Operating System Components

Detailed content of the Lecture:

Definition

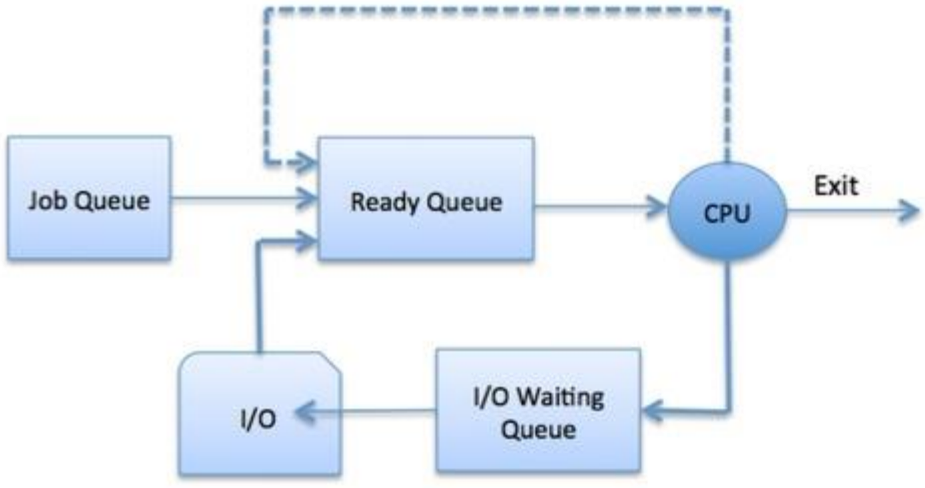
- The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.
- Process scheduling is an essential part of a Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

Process Scheduling Queues

The OS maintains all PCBs in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues −

- **Job queue** − This queue keeps all the processes in the system.
- **Ready queue** − This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** − The processes which are blocked due to unavailability of an I/O device constitute this queue.



Schedulers

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

Long Term Scheduler

✓ It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

✓ The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

✓ On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler

✓ It is also called as CPU scheduler. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

- ✓ Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler

- ➢ Medium-term scheduling is a part of swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.
- ➢ A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/operating_system/os_process_scheduling
https://mail.google.com/mail/u/0/#inbox/FMfcgxwKjxGMxnZznwlTpLFcdnWdqFsw?projector=1&messagePartId=0.2

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, "Operating System Concepts", John Wiley & Sons (ASIA) Pvt. Ltd, 9th Edition. 2015

Course Teacher

Verified by HOD

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **Process Management and Threading**

Date of Lecture:

Topic of Lecture: Operation on Processes

Introduction : (Maximum 5 sentences)

- ✓ A process may create several new processes, via a create-process system call, during the course of execution.
- ✓ The creating process is called a parent process, and the new processes are called the children of that process.
- ✓ Each of these new processes may in turn create other processes, forming a tree of processes.

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Hardware
- Software
- Operating Systems
- Operating System Components

Detailed content of the Lecture:

- ➢ A process may create several new processes, via a create-process system call, during the course of execution. The creating process is called a parent process, and the new processes are called the children of that process. Each of these new processes may in turn create other processes, forming a tree of processes.
- ➢ Most operating systems identify processes according to a unique process identifier (or pid), which is typically an integer number. These processes are responsible for managing memory and file systems. The shed process also creates the init process, which serves as the root parent process for all user processes.

- ➢ When a process creates a new process, two possibilities exist in terms of execution:
- ➢ The parent continues to execute concurrently with its children.
- ➢ The parent waits until some or all of its children have terminated.

There are also two possibilities in terms of the address space of the new process:

The child process is a duplicate of the parent process (it has the same program and data as the parent).

The child process has a new program loaded into it.

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
int main()
{
pid-t pid;
/* fork a child process */
pid = fork();
if (pid < 0) {/* error occurred */
fprintf(stderr, "Fork Failed");
exit (-1) ;
}
else if (pid == 0} {/* child process */
execlpf"/bin/Is","Is",NULL);
}
else {/* parent process */
/* parent will wait for the child to complete */
wait(NULL);
printf("Child Complete");
exit (0) ;

}

}
```

- ✓ Each process is identified by its process identifier, which is a unique integer. A new process is created by the fork() system call. The new process consists of a copy of the address space of the original process.
- ✓ This mechanism allows the parent process to communicate easily with its child process.
- ✓ Both processes (the parent and the child) continue execution at the instruction after the f ork(), with one difference: The return code for the fork() is zero for the new (child) process, whereas the (nonzero) process identifier of the child is returned to the parent. the exec() system call is used after a fork() system call by one of the two processes to replace the process's memory space with a new program. The exec () system call loads a binary file into memory (destroying the memory image of the program containing the execO system call) and starts its execution

Process Termination

- ✓ A process terminates when it finishes executing its final statement and asks the operating system to delete it by using the exit () system call. At that point, the process may return a status value (typically an integer) to its parent process (via the wait() system call). All the resources of the process—including physical and virtual memory, open files, and I/O buffers—are deal located by the operating system.
    - o Termination can occur in other circumstances as well. A process can cause the termination of another process via an appropriate system call (for example, TerminateProcessO in Win32). Usually, such a system call can be invoked only by the parent of the process that is to be terminated.
    - o A parent may terminate the execution of one of its children for a variety of reasons, such as these:
- ✓ The child has exceeded its usage of some of the resources that it has been allocated.
- ✓ The task assigned to the child is no longer required.
- ✓ The parent is exiting, and the operating system does not allow a child to

continue if its parent terminates.

- Consider that, in UNIX, we can terminate a process by using the exit() system call; its parent process may wait for the termination of a child process by using the wait() system call. The wait () system call returns the process identifier of a terminated child so that the parent can tell which of its possibly many children has terminated.
- If the parent terminates, however, all its children have assigned as their new parent the init process.

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/operating_system/os_process_scheduling
https://mail.google.com/mail/u/0/#inbox/FMfcgxwKjxGMxnZznwlTpLFcdnWdqFsw?projector=1&messagePartId=0.2

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, “Operating System Concepts”, John Wiley & Sons (ASIA) Pvt. Ltd, 9th Edition. 2015

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS **L**

CSE **II/IV**

Course Name with Code	**: Operating Systems & 16CSD05**
Course Faculty	**: Mr.S.Pragadeeswaran**
Unit	**: Process Management and Threading**

Date of Lecture:

Topic of Lecture: Inter-process Communication: Shared Memory Systems

Introduction : (Maximum 5 sentences)

- Processes executing concurrently in the operating system may be either independent processes or cooperating processes. A process is independent if it cannot affect or be affected by the other processes executing in the system.
- Any process that does not share data with any other process is independent. A process is cooperating if it can affect or be affected by the other processes executing in the system.

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Hardware
- Software
- Operating Systems
- Operating System Components

Detailed content of the Lecture:

communication is done via this shared memory where changes made by one process can be viewed by another process.

The problem with pipes, fifo and message queue – is that for two process to exchange information. The information has to go through the kernel.

- Server reads from the input file.
- The server writes this data in a message using either a pipe, fifo or message queue.
- The client reads the data from the IPC channel,again requiring the data to be copied from kernel’s IPC buffer to the client’s buffer.
- Finally the data is copied from the client’s buffer.

✓ A total of four copies of data are required (2 read and 2 write). So, shared memory provides a way by letting two or more processes share a memory segment. With Shared Memory the data is only copied twice – from input file into shared memory and from shared memory to the output file.

✓ Shared-Memory Systems Interprocess communication using shared memory requires communicating processes to establish a region of shared memory.

✓ Typically, a shared-memory region resides in the address space of the process creating the shared-memory segment. Other processes that wish to communicate using this shared-memory segment must attach it to their address space.

- ✓ The operating system tries to prevent one process from accessing another process's memory. Shared memory requires that two or more processes agree to remove this restriction. They can then exchange information by reading and writing data in the shared areas.

- ✓ The form of the data and the location are determined by these processes and are not under the operating system's control. The processes are also responsible for ensuring that they are not writing to the same location simultaneously.
    - o Message-Passing Systems The scheme requires that these processes share a region of memory and that the code for accessing and manipulating the shared memory be written explicitly by the application programmer.
    - o Another way to achieve the same effect is for the operating system to provide the means for cooperating processes to communicate with each other via a message-passing facility.
    - o Message passing provides a mechanism to allow processes to communicate and to synchronize their actions without sharing the same address space and is particularly useful in a distributed environment, where the communicating processes may reside on different computers connected by a network.

A message-passing facility provides at least two operations: send(message) and receive(message). Messages sent by a process can be of either fixed or variable size. If only fixed-sized messages can be sent, the system-level implementation is straightforward. This restriction, however, makes the task of programming more difficult. Conversely, variable-sized messages require a more complex system-level implementation, but the programming task becomes simpler. This is a common kind of tradeoff seen throughout operating system design.

Naming

Processes that want to communicate must have a way to refer to each other. They can use either direct or indirect communication.
Under direct communication, each process that wants to communicate must explicitly name the recipient or sender of the communication. In this scheme, the send.0 and receive() primitives are defined as:

send(P, message)—Send a message to process P.
receive (Q, message)—Receive a message from process Q.

A link is established automatically between every pair of processes that want to communicate. The processes need to know only each other's identity to communicate.
A link is associated with exactly two processes.
Between each pair of processes, there exists exactly one link.

The disadvantage in both of these schemes (symmetric and asymmetric) is the limited modularity of the resulting process definitions. Changing the identifier of a process may necessitate examining all other process definitions.

Synchronization

Communication between processes takes place through calls to send() and receive () primitives. There are different design options for implementing each primitive. Message passing may be either blocking or nonblocking— also known as synchronous and asynchronous.

Blocking send- The sending process is blocked until the message is received by the receiving process or by the mailbox.

Nonblocking send- The sending process sends the message and resumes operation.

Blocking receive- The receiver blocks until a message is available.

Nonblocking receive- The receiver retrieves either a valid message or a null.

Buffering

Whether communication is direct or indirect, messages exchanged by communicating processes reside in a temporary queue. Basically, such queues can be implemented in three ways:

• **Zero capacity-** The queue has a maximum length of zero; thus, the link cannot have any messages waiting in it. In this case, the sender must block until the recipient receives the message.

Bounded capacity- The queue has finite length n; thus, at most n messages can reside in it. If the queue is not full when a new message is sent, the message is placed in the queue (either the message is copied or a pointer to the message is kept), and the sender can continue execution without waiting. The links capacity is finite, however. If the link is full, the sender must block until space is available in the queue.

Unbounded capacity- The queues length is potentially infinite; thus, any number of messages can wait in it. The sender never blocks.

An Example: POSIX Shared Memory

Several IPC mechanisms are available for POSIX systems, including shared memory and message passing. A process must first create a shared memory segment using the shmget () system call (shmget () is derived from SHared Memory GET).

The following example illustrates the use of shmget ():

- ✓ segment_id = shmget(IPCJPRIVATE, size, SJRUSR | SJVVUSR) ;

- ➢ This first parameter specifies the key (or identifier) of the shared-memory segment. If this is set to IPC-PRIVATE, a new shared-memory segment is created.
- ➢ The second parameter specifies the size (in bytes) of the shared memory segment.
- ➢ Finally, the third parameter identifies the mode, which indicates how the shared-memory segment is to be used—that is, for reading, writing, or both.

By setting the mode to SJRUSR | SJVVUSR, we are indicating that the owner may read or write to the shared memory segment.

- ✓ Processes that wish to access a shared-memory segment must attach it to their address space using the shmat () (SHared Memory ATtach) system call.
- ✓ The call to shmat () expects three parameters as well.
- ➢ The first is the integer identifier of the shared-memory segment being attached, and the second is a pointer location in memory indicating where the shared memory will be attached.
- ➢ If we pass a value of NULL, the operating system selects the location on the user's behalf.
- The third parameter identifies a flag that allows the shared memory region to be attached in read-only or read-write mode; by passing a parameter of 0, we allow both reads and writes to the shared region.
- The third parameter identifies a mode flag. If set, the mode flag allows the shared-memory region to be attached in read-only mode; if set to 0, the flag allows both reads and writes to the shared region

We attach a region of shared memory using shmat () as follows:

shared_memory = (char *) shmat(id, NULL, 0);

If successful, shmat () returns a pointer to the beginning location in memory where the shared-memory region has been attached.

Video Content / Details of website for further learning (if any): https://www.tutorialspoint.com/operating_system/os_process_scheduling https://mail.google.com/mail/u/0/#inbox/FMfcgxwKjxGMxnZznwlTpLFcdnWdqFsw?projector=1&messagePartId=0.2
Important Books/Journals for further learning including the page nos.: **Book:** Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, "Operating System Concepts", John Wiley & Sons (ASIA) Pvt. Ltd, 9th Edition. 2015

Course Teacher

Verified by HOD

LECTURE HANDOUTS **L4**

CSE **II/IV**

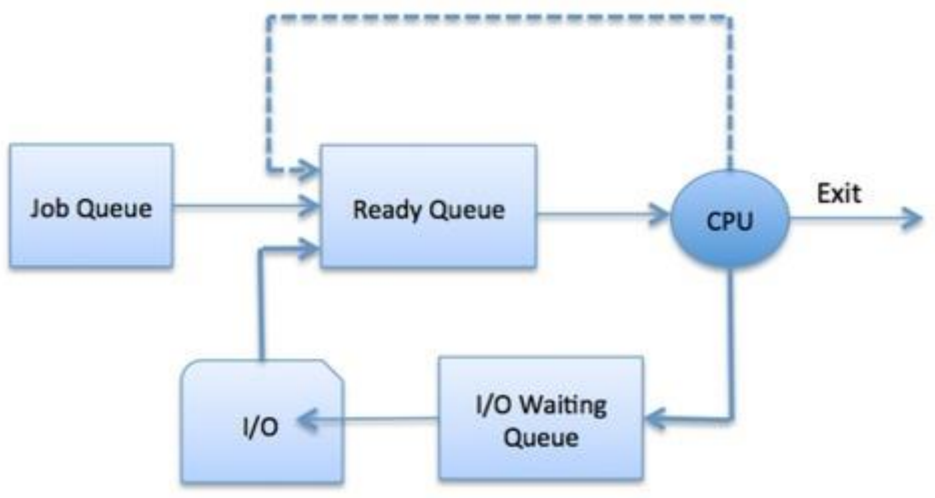
Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **Process Management and Threading**

Date of Lecture:

Topic of Lecture: Process Scheduling: Basic Concepts – Scheduling Criteria
Introduction : (Maximum 5 sentences) • A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms.
Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics) • Hardware • Software • Operating Systems • Operating System Components
Details of website for further learning (if any): Process Scheduling Queues The OS maintains all PCBs in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue. The Operating System maintains the following important process scheduling queues − • **Job queue** − This queue keeps all the processes in the system. • **Ready queue** − This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue. • **Device queues** − The processes which are blocked due to unavailability of an I/O device constitute this queue.



The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

Two-State Process Model

Two-state process model refers to running and non-running states which are described below −

S.N.	State & Description
1	**Running** When a new process is created, it enters into the system as in the running state.
2	**Not Running** Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Schedulers

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

Long Term Scheduler

- It is also called a **job scheduler**. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler

- It is also called as **CPU scheduler**. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler

- Medium-term scheduling is a part of **swapping**. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

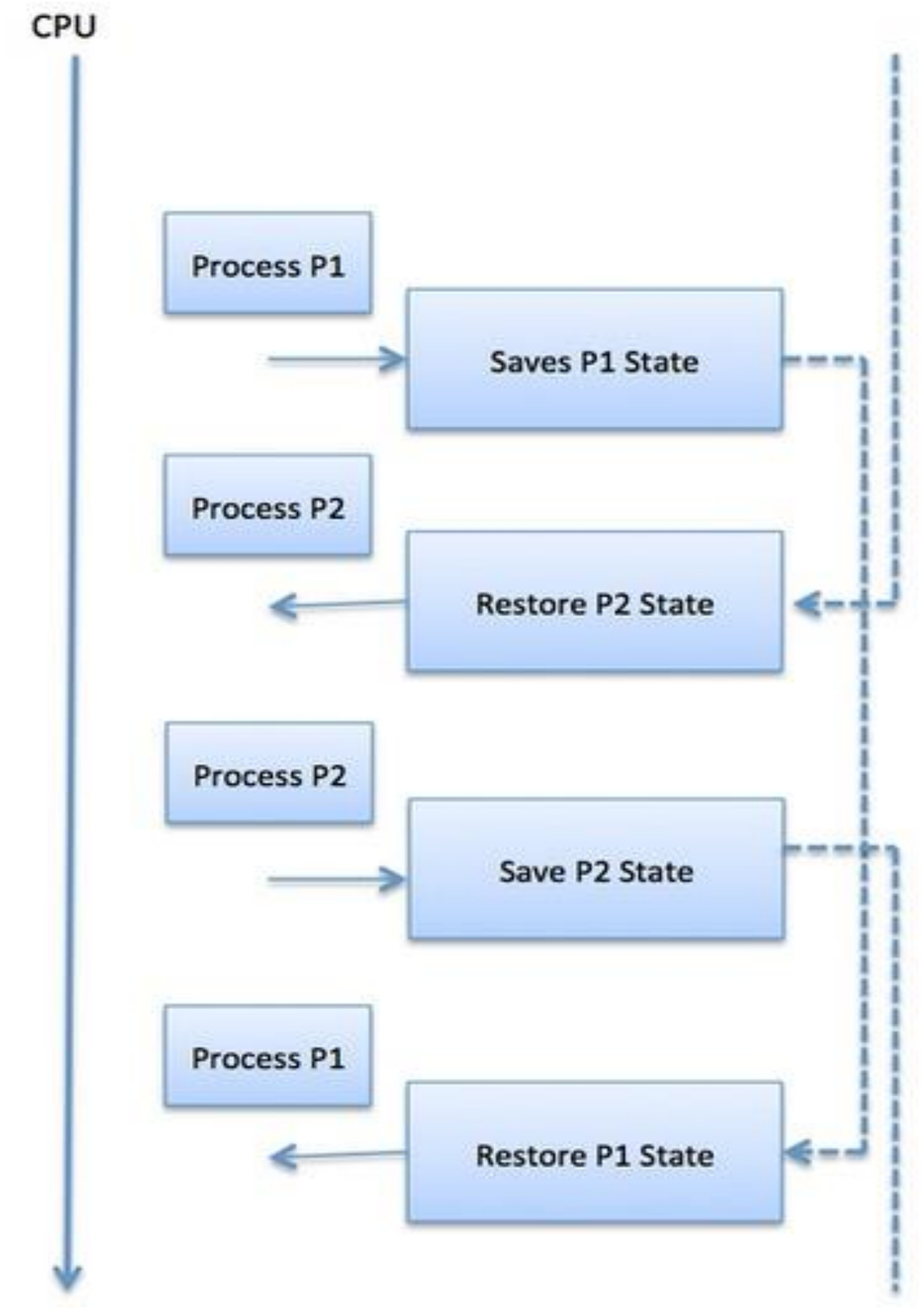
A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called **swapping**, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

Comparison among Scheduler

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Context Switch

- A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time.
- Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.
- When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block.
- After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc. At that point, the second process can start executing.



Context switches are computationally intensive since register and memory state must be saved and restored. To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers. When the process is switched, The following information is stored for later use.

- Program Counter
- Scheduling information
- Base and limit register value
- Currently used register
- Changed State
- I/O State information
- Accounting information

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/operating_system/os_process_scheduling
https://mail.google.com/mail/u/0/#inbox/FMfcgxwKjxGMxnZznwlTpLFcdnWdqFsw?projector=1&messagePartId=0.2

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, "Operating System Concepts", John Wiley & Sons (ASIA) Pvt. Ltd, 9th Edition. 2015

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS **L5**

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **Process Management and Threading**

Date of Lecture:

Topic of Lecture: Scheduling Algorithms: First-Come, First-Served – Priority

Introduction : (Maximum 5 sentences)

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority.
- Process with highest priority is to be executed first and so on. Processes with same priority are executed on first come first served basis.

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Hardware
- Software
- Operating Systems
- Operating System Components

Detailed content of the Lecture:

A proceess Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms which we are going to discuss in this chapter −

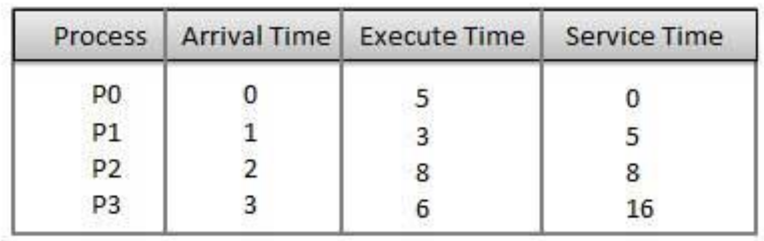
- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

✓ These algorithms are either non-preemptive or preemptive.

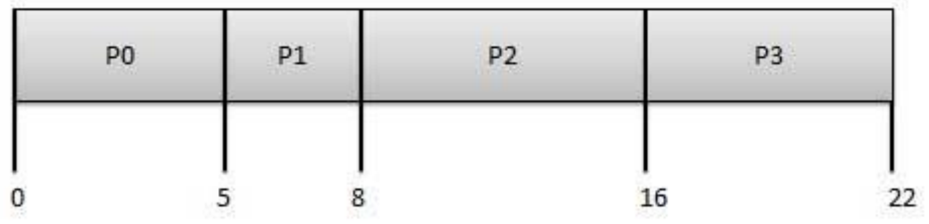
✓ Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.



Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



Wait time of each process is as follows −

Process	**Wait Time : Service Time - Arrival Time**
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

- This is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.

- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

Given: Table of processes, and their Arrival time, Execution time

Process	Arrival Time	Execution Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	14
P3	3	6	8

Waiting time of each process is as follows −

Process	Waiting Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	14 - 2 = 12
P3	8 - 3 = 5

Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

Priority Based Scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 is the lowest priority.

Process	Arrival Time	Execution Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

Waiting time of each process is as follows −

Process	Waiting Time
P0	0 - 0 = 0
P1	11 - 1 = 10
P2	14 - 2 = 12
P3	5 - 3 = 2

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

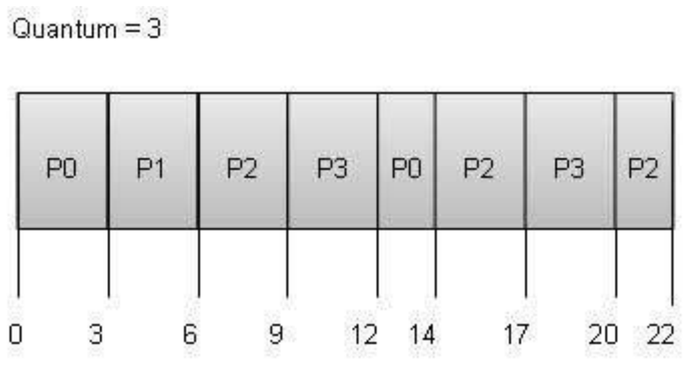
Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.

- Context switching is used to save states of preempted processes.



Wait time of each process is as follows −

Process	**Wait Time : Service Time - Arrival Time**
P0	(0 - 0) + (12 - 3) = 9
P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

Multiple-Level Queues Scheduling

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

- Multiple queues are maintained for processes with common characteristics.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue.

For example, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs in another queue. The Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.

➢ The OS maintains all PCBs in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

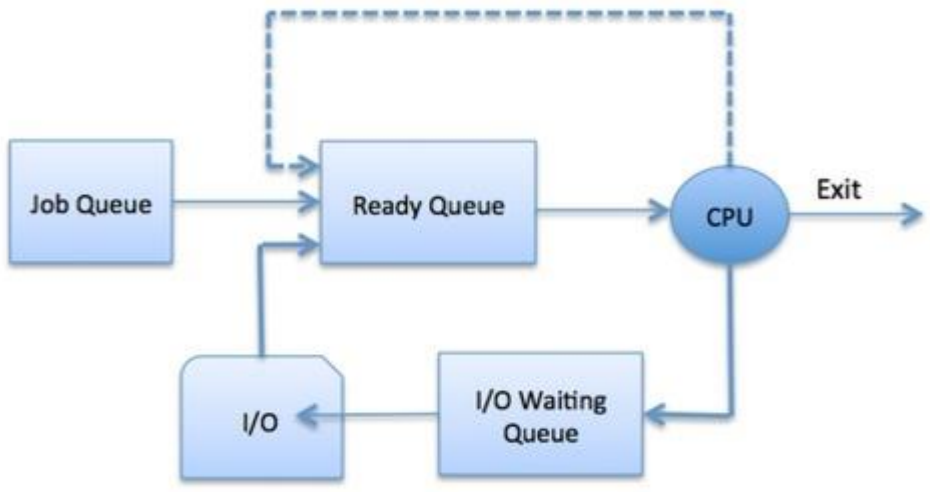
➢ The OS maintains all PCBs in Process Scheduling Queues. The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues −

- **Job queue** − This queue keeps all the processes in the system.
- **Ready queue** − This queue keeps a set of all processes residing in main memory, ready and

waiting to execute. A new process is always put in this queue.

- **Device queues** − The processes which are blocked due to unavailability of an I/O device constitute this queue.



The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

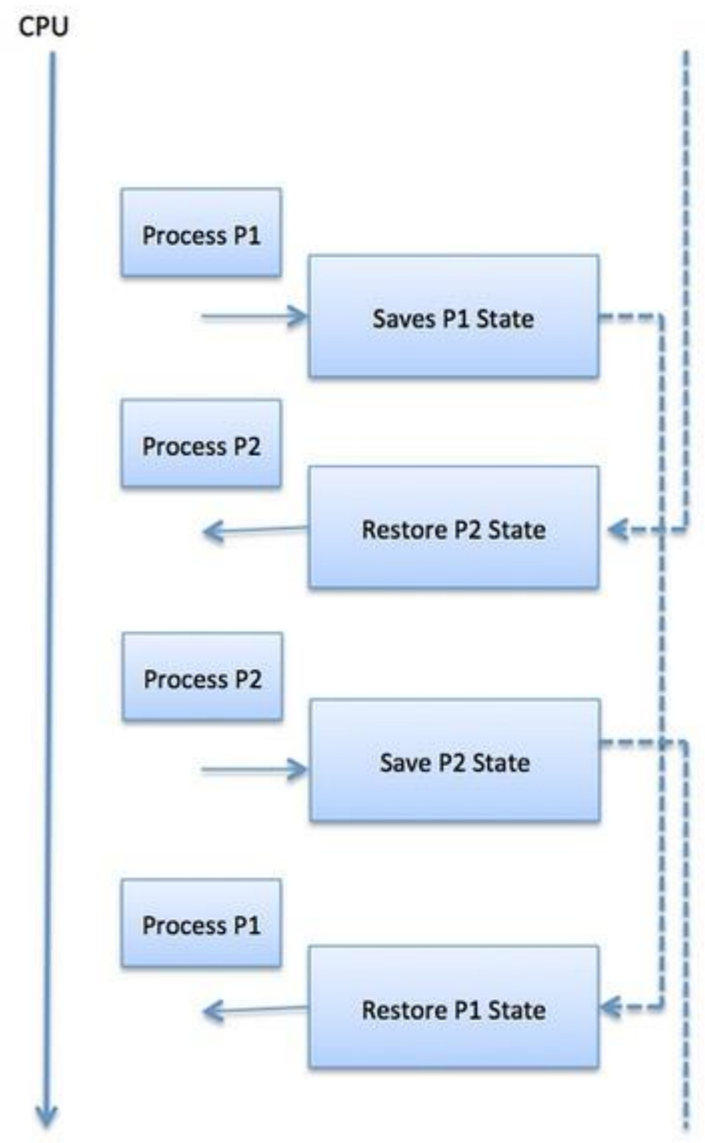
Two-State Process Model

Two-state process model refers to running and non-running states which are described below −

S.N.	State & Description
1	**Running** When a new process is created, it enters into the system as in the running state.
2	**Not Running** Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

Schedulers

Schedulers are special system software which handle process



Context switches are computationally intensive since register and memory state must be saved and restored. To avoid the amount of context switching time, some hardware systems employ two or more sets of processor registers. When the process is switched, the following information is stored for later use.

- Program Counter
- Scheduling information
- Base and limit register value
- Currently used register
- Changed State
- I/O State information
- Accounting information

scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

Long Term Scheduler

- It is also called a **job scheduler**. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.
- The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.
- On some systems, the long-term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler

- It is also called as **CPU scheduler**. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.
- Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler

- Medium-term scheduling is a part of **swapping**. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.
- A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called **swapping**, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

Comparison among Scheduler

S.N.	Long-Term Scheduler	Short-Term Scheduler	Medium-Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.

4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Context Switch

- A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.
- When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block. After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc. At that point, the second process can start executing.

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/operating_system/os_process_scheduling
https://mail.google.com/mail/u/0/#inbox/FMfcgxwKjxGMxnZznwlTpLFcdnWdqFsw?projector=1&messagePartId=0.2

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, “Operating System Concepts”, John Wiley & Sons (ASIA) Pvt. Ltd, 9th Edition. 2015

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS **L6**

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **Process Management and Threading**

Date of Lecture:

Topic of Lecture: Round-Robin,Multilevel Queue

Introduction : (Maximum 5 sentences)

- Each process gets a small unit of CPU time (time quantum), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are n processes in the ready queue and the time quantum is q, then each process gets 1/n of the CPU time in chunks of at most q time units at once. No process waits more than (n-1)q time units Performance

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Hardware
- Software
- Operating Systems
- Operating System Components

Detailed content of the Lecture:

<u>Round Robin (RR)</u>

Each process gets a small unit of CPU time (time quantum), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.

If there are n processes in the ready queue and the time quantum is q, then each process gets 1/n of the CPU time in chunks of at most q time units at once. No process waits more than (n-1)q time units.

Performance

q large FIFO

q small q must be large with respect to context switch, otherwise overhead is too high

<u>Example of RR with Time Quantum = 20</u>

Process	Burst Time
P1	53
P_2	17
P_3	68

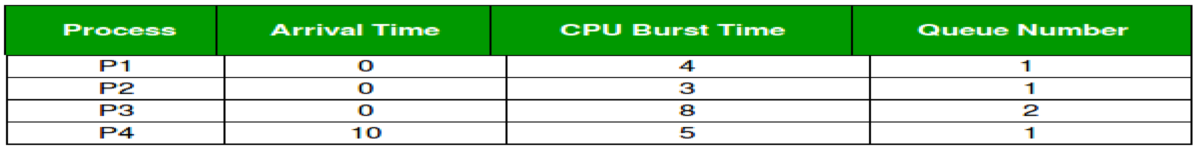
Multilevel Queue (MLQ) CPU Scheduling

Prerequisite: CPU Scheduling . It may happen that processes in the ready queue can be divided into different classes where each class has its own scheduling needs. For example, a common division is a **foreground (interactive)** process and **background (batch)** processes.These two classes have different scheduling needs. For this kind of situation Multilevel Queue Scheduling is used.Now, let us see how it works.
Ready Queue is divided into separate queues for each class of processes. For example, let us take three different types of process System processes, Interactive processes and Batch Processes.

Scheduling among the queues : What will happen if all the queues have some processes? Which process should get the cpu? To determine this Scheduling among the queues is necessary. There are two ways to do so –
Fixed priority preemptive scheduling method – Each queue has absolute priority over lower priority queue. Let us consider following priority order **queue 1 > queue 2 > queue 3**.According to this algorithm no process in the batch queue(queue 3) can run unless queue 1 and 2 are empty. If any batch process (queue 3) is running and any system (queue 1) or Interactive process(queue 2) entered the ready queue the batch process is preempted.

1. **Time slicing** – In this method each queue gets certain portion of CPU time and can use it to schedule its own processes.For instance, queue 1 takes 50 percent of CPU time queue 2 takes 30 percent and queue 3 gets 20 percent of CPU time.

ExampleProblem
Consider below table of four processes under Multilevel queue scheduling.Queue number denotes the queue of the process.



Process	Arrival Time	CPU Burst Time	Queue Number
P1	0	4	1
P2	0	3	1
P3	0	8	2
P4	10	5	1

Priority of queue 1 is greater than queue 2. queue 1 uses Round Robin (Time Quantum = 2) and queue 2 uses FCFS. .

Advantages:
- The processes are permanently assigned to the queue, so it has advantage of low scheduling overhead.

Disadvantages:
- Some processes may starve for CPU if some higher priority queues are never becoming empty.
- It is inflexible in nature.

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/operating_system/os_process_scheduling
https://mail.google.com/mail/u/0/#inbox/FMfcgxwKjxGMxnZznwlTpLFcdnWdqFsw?projector=1 &messagePartId=0.2

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, "Operating System Concepts", John Wiley & Sons (ASIA) Pvt. Ltd, 9th Edition. 2015

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L7

CSE

II/IV

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **Process Management and Threading**

Date of Lecture:

Topic of Lecture: Multilevel Feedback Queue, Threads: Overview

Introduction : (Maximum 5 sentences)

- In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system. Processes do not move between queues.
- This setup has the advantage of low scheduling overhead, but the disadvantage of being inflexible.

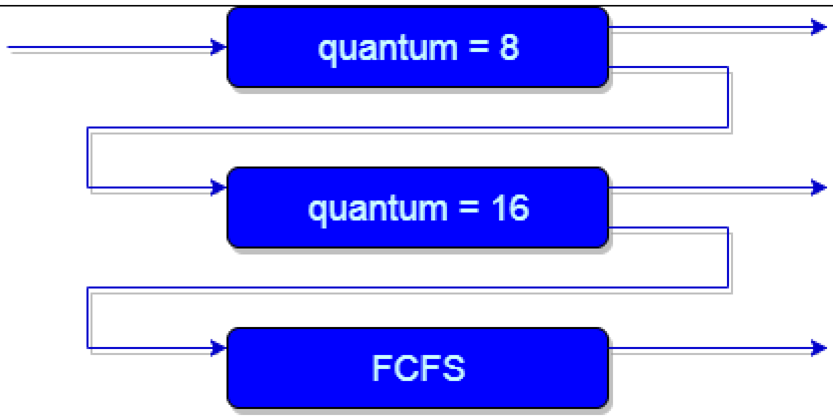
Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Hardware
- Software
- Operating Systems
- Operating System Components

Detailed content of the Lecture:

➢ In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system. Processes do not move between queues. This setup has the advantage of low scheduling overhead, but the disadvantage of being inflexible.

➢ Multilevel feedback queue scheduling, however, allows a process to move between queues. The idea is to separate processes with different CPU-burst characteristics. If a process uses too much CPU time, it will be moved to a lower-priority queue. Similarly, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. This form of aging prevents starvation.



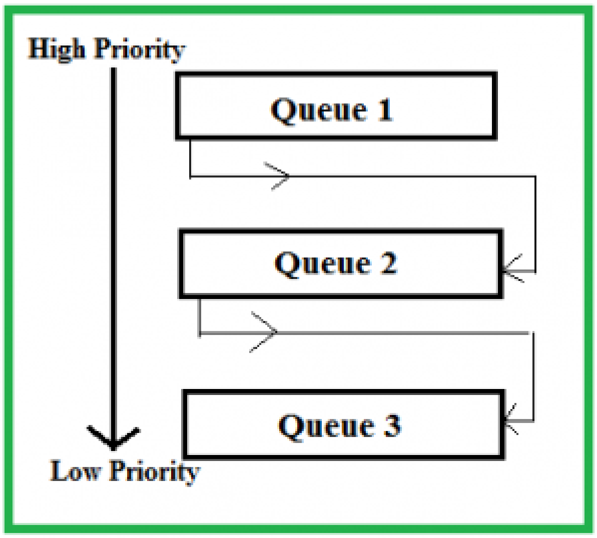
An example of a multilevel feedback queue can be seen in the below figure.

In general, a multilevel feedback queue scheduler is defined by the following parameters:

- The number of queues.
- The scheduling algorithm for each queue.
- The method used to determine when to upgrade a process to a higher-priority queue.
- The method used to determine when to demote a process to a lower-priority queue.
- The method used to determine which queue a process will enter when that process needs service.
- The definition of a multilevel feedback queue scheduler makes it the most general CPU-scheduling algorithm. It can be configured to match a specific system under design. Unfortunately, it also requires some means of selecting values for all the parameters to define the best scheduler. Although a multilevel feedback queue is the **most general scheme**, it is also the **most complex**.

Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling

This Scheduling is like Multilevel Queue(MLQ) Scheduling but in this process can move between the queues. Multilevel Feedback Queue Scheduling (MLFQ) keep analyzing the behavior (time of execution) of processes and according to which it changes its priority.Now, look at the diagram and explanation below to understand it properly.



Now let us suppose that queue 1 and 2 follow round robin with time quantum 4 and 8 respectively and queue 3 follow FCFS.One implementation of MFQS is given below –

1. When a process starts executing then it first enters queue 1.
2. In queue 1 process executes for 4 unit and if it completes in this 4 unit or it gives CPU for I/O operation in this 4 unit than the priority of this process does not change and if it again comes in the ready queue than it again starts its execution in Queue 1.
3. If a process in queue 1 does not complete in 4 unit then its priority gets reduced and it shifted to queue 2.
4. Above points 2 and 3 are also true for queue 2 processes but the time quantum is 8 unit.In a general case if a process does not complete in a time quantum than it is shifted to the lower priority queue.
5. In the last queue, processes are scheduled in FCFS manner.
6. A process in lower priority queue can only execute only when higher priority queues are empty.
7. A process running in the lower priority queue is interrupted by a process arriving in the higher priority queue.

Well, above implementation may differ for example the last queue can also follow Round-robin Scheduling.

Problems in the above implementation – A process in the lower priority queue can suffer from starvation due to some short processes taking all the CPU time.
Solution – A simple solution can be to boost the priority of all the process after regular intervals and place them all in the highest priority queue.

What is the need of such complex Scheduling?

- Firstly, it is more flexible than the multilevel queue scheduling.
- To optimize turnaround time algorithms like SJF is needed which require the running time of processes to schedule them. But the running time of the process is not known in advance. MFQS runs a process for a time quantum and then it can change its priority(if it is a long process). Thus it learns from past behavior of the process and then predicts its future behavior.This way it tries to run shorter process first thus optimizing turnaround time.
- MFQS also reduces the response time.

Example –
Consider a system which has a CPU bound process, which requires the burst time of 40 seconds.The multilevel Feed Back Queue scheduling algorithm is used and the queue time quantum ‘2’ seconds and in each level it is incremented by ‘5’ seconds.Then how many times the process will be interrupted and on which queue the process will terminate the execution?
Solution –
Process P needs 40 Seconds for total execution.
At Queue 1 it is executed for 2 seconds and then interrupted and shifted to queue 2.
At Queue 2 it is executed for 7 seconds and then interrupted and shifted to queue 3.
At Queue 3 it is executed for 12 seconds and then interrupted and shifted to queue 4.
At Queue 4 it is executed for 17 seconds and then interrupted and shifted to queue 5.
At Queue 5 it executes for 2 seconds and then it completes.
Hence the process is interrupted 4 times and completes on queue 5.
Advantages:
1. It is more flexible.
2. It allows different processes to move between different queues.
3. It prevents starvation by moving a process that waits too long for lower priority queue to the higher priority queue.

Disadvantages:
1. For the selection of the best scheduler, it require some other means to select the values.
2. It produces more CPU overheads.
3. It is most complex algorithm.

Video Content / Details of website for further learning (if any): https://www.tutorialspoint.com/operating_system/os_process_scheduling https://mail.google.com/mail/u/0/#inbox/FMfcgxwKjxGMxnZznwlTpLFcdnWdqFsw?projector=1&messagePartId=0.2
Important Books/Journals for further learning including the page nos.: **Book:** Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, "Operating System Concepts", John Wiley & Sons (ASIA) Pvt. Ltd, 9th Edition. 2015

Course Teacher

Verified by HOD

LECTURE HANDOUTS **L8**

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **Process Management and Threading**

Date of Lecture:

Topic of Lecture: Multithreading models

Introduction : (Maximum 5 sentences)

- A Thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Hardware
- Software
- Operating Systems
- Operating System Components

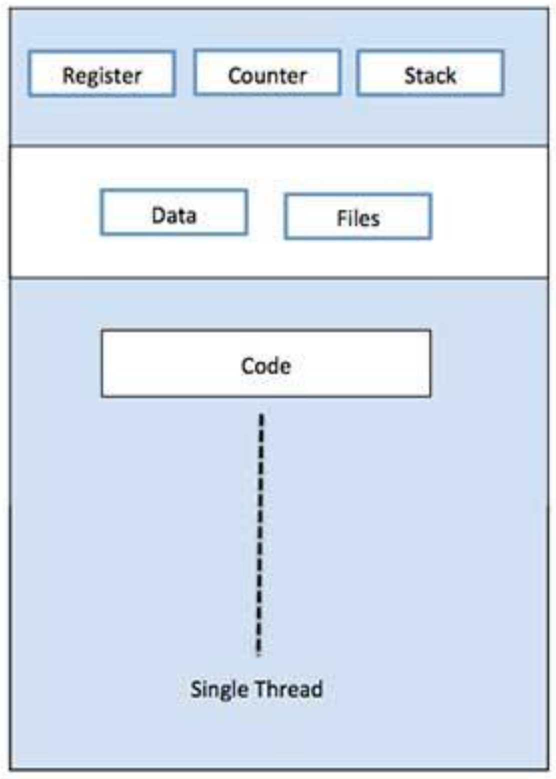
Detailed content of the Lecture:

A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

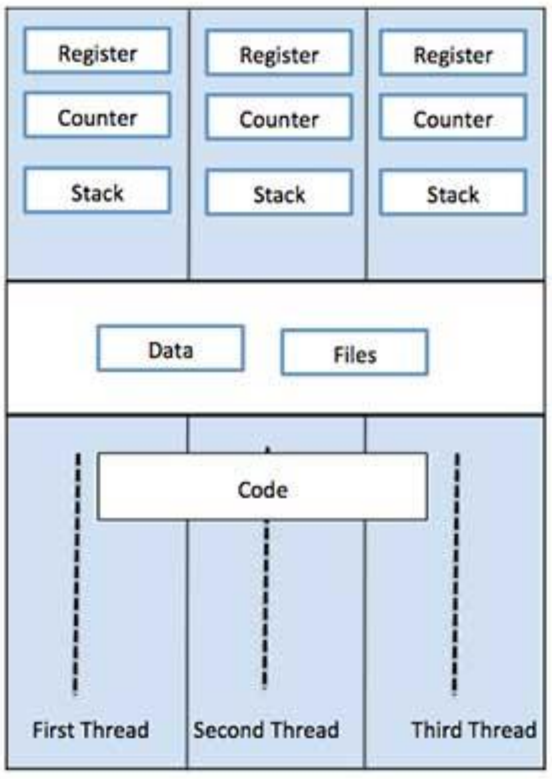
A thread shares with its peer threads few information like code segment, data segment and open files. When one thread alters a code segment memory item, all other threads see that.

A thread is also called a **lightweight process**. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. The following figure shows the working of a single-threaded and a multithreaded process.



Single Process P with single thread



Single Process P with three threads

Difference between Process and Thread

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Advantages of Thread

- Threads minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to create and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

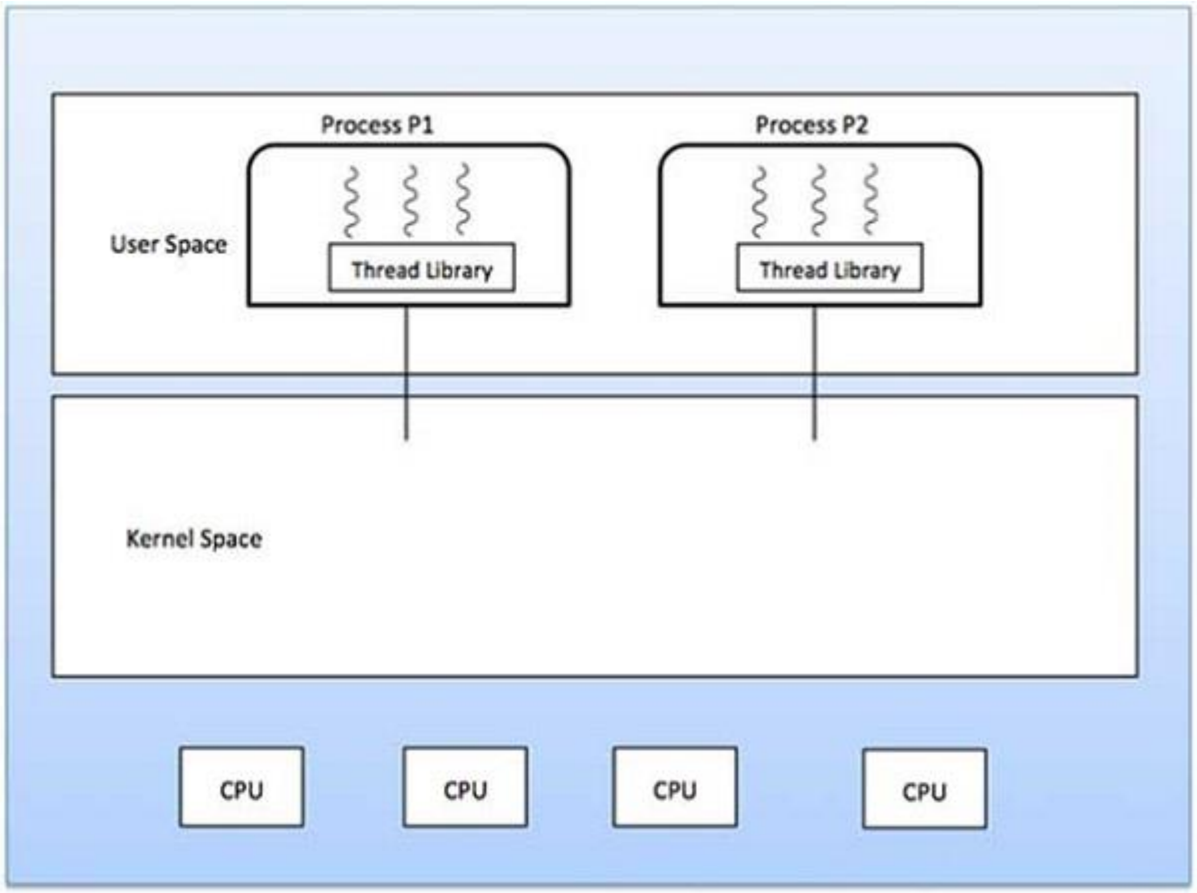
Types of Thread

Threads are implemented in following two ways −

- **User Level Threads** − User managed threads.
- **Kernel Level Threads** − Operating System managed threads acting on kernel, an operating system core.

User Level Threads

In this case, the thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.



Advantages

- Thread switching does not require Kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.

Disadvantages

- In a typical operating system, most system calls are blocking.
- Multithreaded application cannot take advantage of multiprocessing.

Kernel Level Threads

Thread management is done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

The Kernel maintains context information for the process as a whole and for individuals threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.

Advantages

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
- Kernel routines themselves can be multithreaded.

Disadvantages

- Kernel threads are generally slower to create and manage than the user threads.
- Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.

Multithreading Models

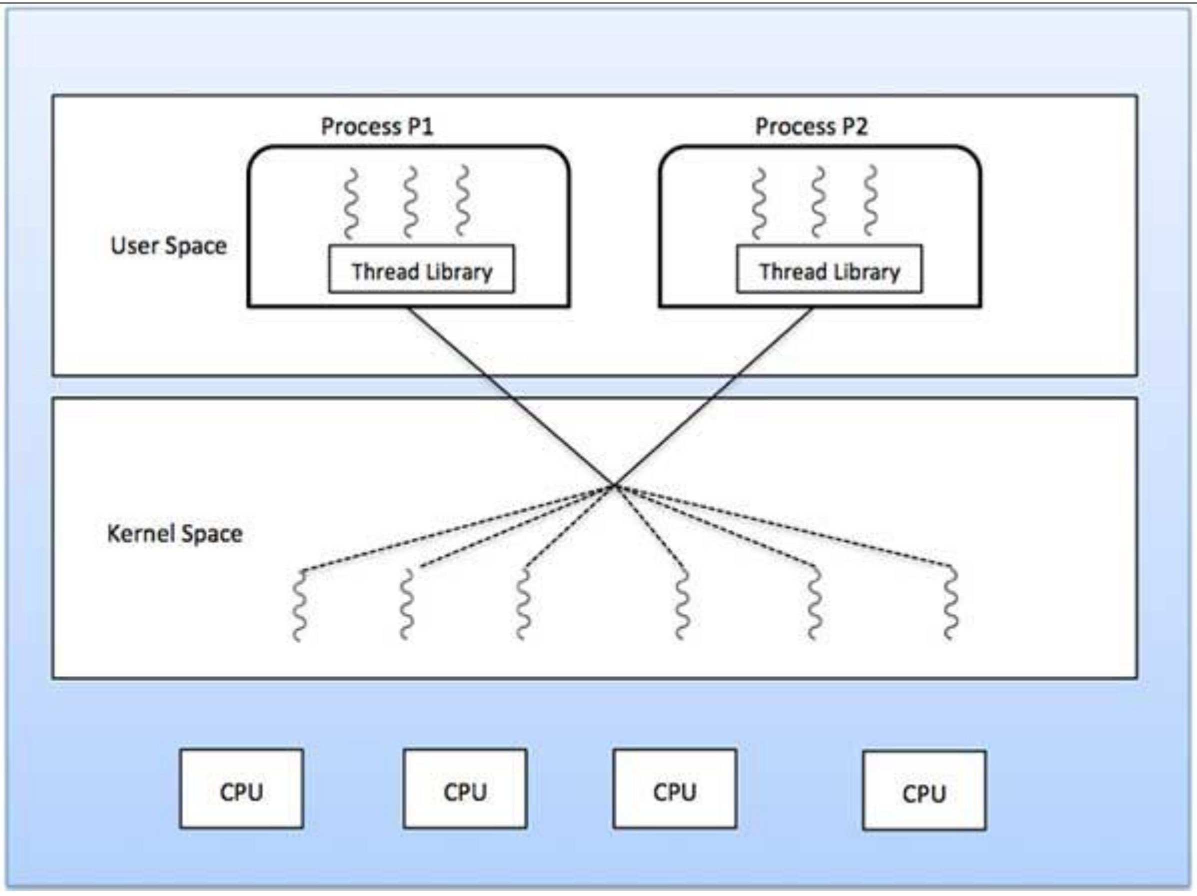
Some operating system provide a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not block the entire process. Multithreading models are three types

- Many to many relationship.
- Many to one relationship.
- One to one relationship.

Many to Many Model

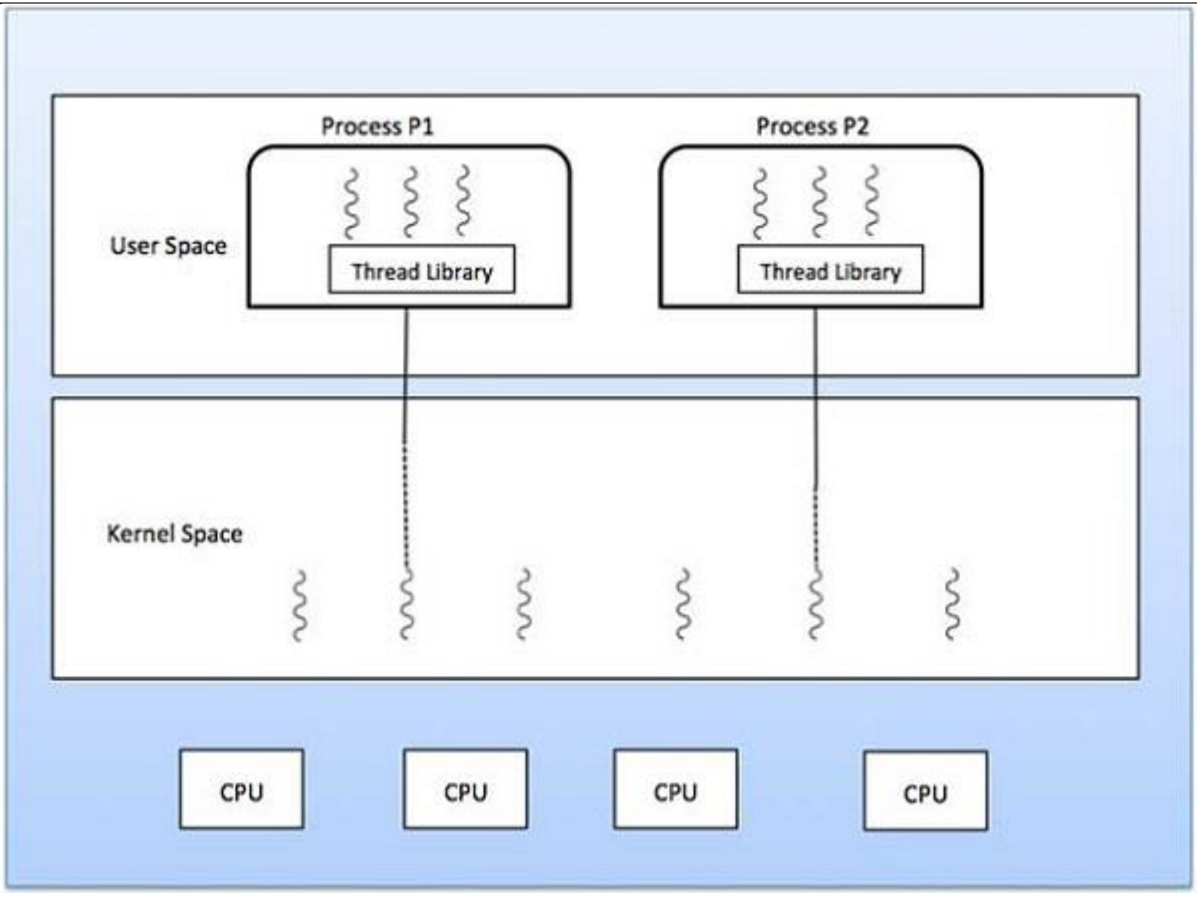
The many-to-many model multiplexes any number of user threads onto an equal or smaller number of kernel threads.

- ➢ In this model, developers can create as many user threads as necessary and the corresponding Kernel threads can run in parallel on a multiprocessor machine.
- ➢ This model provides the best accuracy on concurrency and when a thread performs a blocking system call, the kernel can schedule another thread for execution.



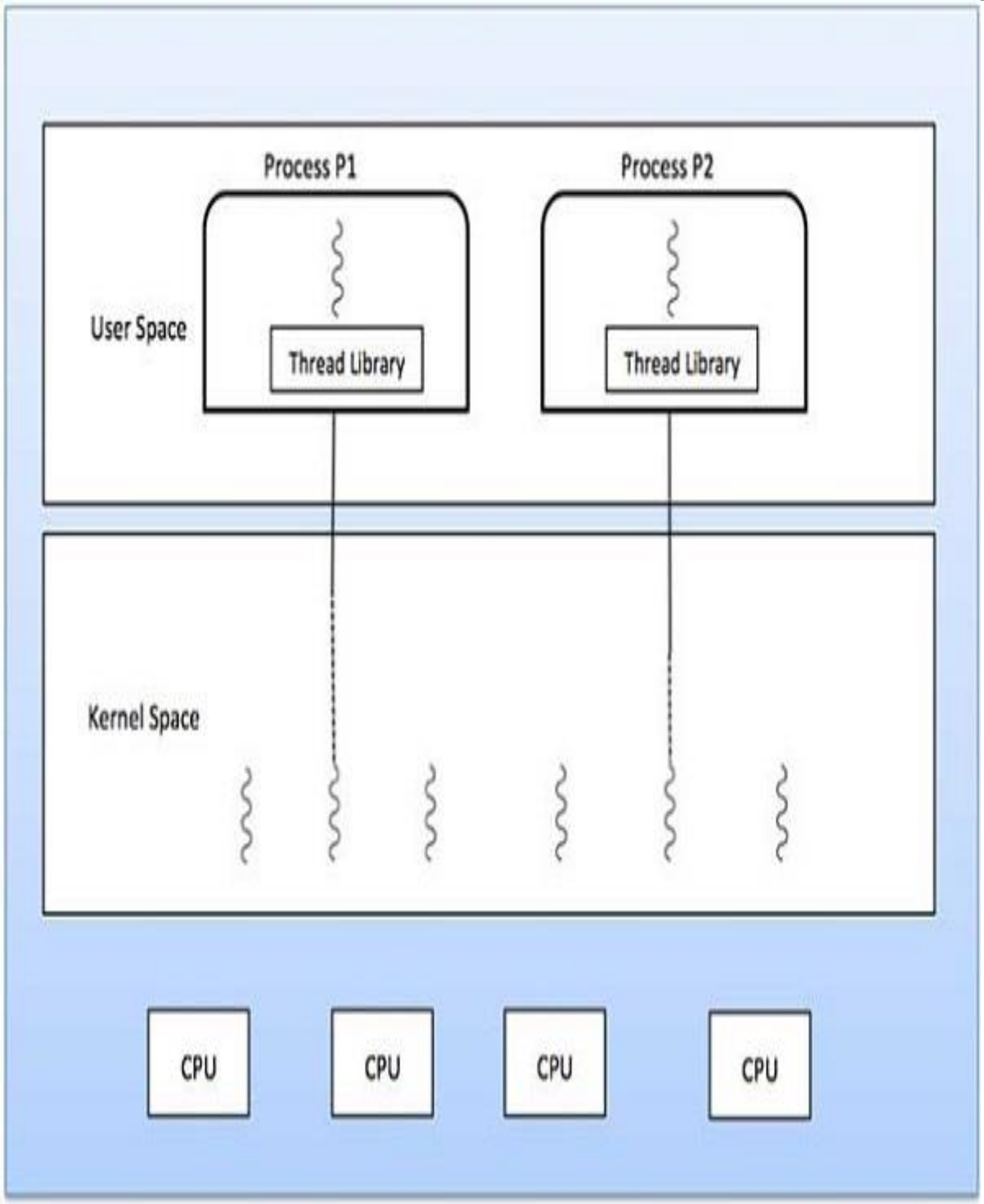
Many to One Model

- Many-to-one model maps many user level threads to one Kernel-level thread. Thread management is done in user space by the thread library. When thread makes a blocking system call, the entire process will be blocked. Only one thread can access the Kernel at a time, so multiple threads are unable to run in parallel on multiprocessors.
- If the user-level thread libraries are implemented in the operating system in such a way that the system does not support them, then the Kernel threads use the many-to-one relationship modes.



One to One Model

- There is one-to-one relationship of user-level thread to the kernel-level thread. This model provides more concurrency than the many-to-one model. It also allows another thread to run when a thread makes a blocking system call. It supports multiple threads to execute in parallel on microprocessors.
- Disadvantage of this model is that creating user thread requires the corresponding Kernel thread. OS/2, windows NT and windows 2000 use one to one relationship



Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/operating_system/os_process_scheduling
https://mail.google.com/mail/u/0/#inbox/FMfcgxwKjxGMxnZznwlTpLFcdnWdqFsw?projector=1&messagePartId=0.2

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, "Operating System Concepts", John Wiley & Sons (ASIA) Pvt. Ltd, 9th Edition. 2015

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L9

CSE

II/IV

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **Process Management and Threading**

Date of Lecture:

Topic of Lecture: Threading issues

Introduction : (Maximum 5 sentences)

- A **thread** is a basic unit of CPU utilization, consisting of a program counter, a stack, and a set of registers, (and a thread ID.)
- Traditional (heavyweight) processes have a single thread of control - There is one program counter, and one sequence of instructions that can be carried out at any given time.
- As shown in Figure 4.1, multi-threaded applications have multiple threads within a single process, each having their own program counter, stack and set of registers, but sharing common code, data, and certain structures such as open files

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Hardware
- Software
- Operating Systems
- Operating System Components

Detailed content of the Lecture:

Overview

- A **thread** is a basic unit of CPU utilization, consisting of a program counter, a stack, and a set of registers, (and a thread ID.)
- Traditional (heavyweight) processes have a single thread of control - There is one program counter, and one sequence of instructions that can be carried out at any given time.
- As shown in Figure 4.1, multi-threaded applications have multiple threads within a single process, each having their own program counter, stack and set of registers, but sharing common code, data, and certain structures such as open files

Benefits

- There are four major categories of benefits to multi-threading:
    1. Responsiveness - One thread may provide rapid response while other threads are blocked or slowed down doing intensive calculations.
    2. Resource sharing - By default threads share common code, data, and other resources, which allows multiple tasks to be performed simultaneously in a single address space.
    3. Economy - Creating and managing threads (and context switches between them) is much faster than performing the same tasks for processes.
    4. Scalability, i.e. Utilization of multiprocessor architectures - A single threaded process can only run on one CPU, no matter how many may be available, whereas the execution of a multi-threaded application may be split amongst available processors. (Note that single threaded processes can still benefit from multi-processor architectures when there are multiple processes contending for the CPU, i.e. when the load average is above some certain threshold.)

Multicore Programming

- A recent trend in computer architecture is to produce chips with multiple **cores**, or CPUs on a single chip.
- A multi-threaded application running on a traditional single-core chip would have to interleave the threads, as shown in Figure 4.3. On a multi-core chip, however, the threads could be spread across the available cores, allowing true parallel processing,

Multithreading Models

- There are two types of threads to be managed in a modern system: User threads and kernel threads.
- User threads are supported above the kernel, without kernel support. These are the threads that application programmers would put into their programs.
- Kernel threads are supported within the kernel of the OS itself. All modern OSes support kernel level threads, allowing the kernel to perform multiple simultaneous tasks and/or to service multiple kernel system calls simultaneously.
- In a specific implementation, the user threads must be mapped to kernel threads, using one of the following strategies.

- Thread libraries provide programmers with an API for creating and managing threads.
- Thread libraries may be implemented either in user space or in kernel space. The former involves API functions implemented solely within user space, with no kernel support. The latter involves system calls, and requires a kernel with thread library support.
- There are three main thread libraries in use today:
    1. POSIX Pthreads - may be provided as either a user or kernel library, as an extension to the POSIX standard.
    2. Win32 threads - provided as a kernel-level library on Windows systems.
    3. Java threads - Since Java generally runs on a Java Virtual Machine, the implementation of threads is based upon whatever OS and hardware the JVM is running on, i.e. either

Pthreads or Win32 threads depending on the system.

Threading Issues

Following threading issues are:

- The fork() and exec() system call
- Signal handling
- Thread cancelation
- Thread local storage
- Scheduler activation

The fork() and exec() system call

In case if a thread fork is the complete process copied or is the new process single threaded? The answer is here it depends on the system and in case of if new process execs immediately then there is no need to copy all the other thread and if it does not create new process then the whole process should be copied.

Signal Handling

1. Signal deliver to the thread to which the signal applies.
2. Signal deliver to each and every thread in the process.
3. Signal deliver to some of the threads in the process.
4. Assign a particular thread to receive all the signals in a process.

Thread Cancellation

Threads that are no-longer required can be cancelled by another thread in one of two techniques:

1. Asynchronies cancellation
2. Deferred cancellation

Asynchronies Cancellation

It means cancellation of thread immediately. Allocation of resources and inter thread data transfer may be challenging for asynchronies cancellation.

Deferred Cancellation

In this method a flag is sets that indicating the thread should cancel itself when it is feasible. It's upon the cancelled thread to check this flag intermittently and exit nicely when it sees the set flag.

Thread Local Storage

The benefit of using threads in the first place is that Most data is shared among the threads but, sometimes threads also need thread explicit data. Major libraries of threads are pThreads, Win32 and java which provide support for thread specific which is called as TLS thread local storage.

Scheduler Activation

Numerous implementation of threads provides a virtual processor as an interface b/w user and kernel thread specifically for two tier model. The virtual processor is called as low weight process (LWP).

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/operating_system/os_process_scheduling
https://mail.google.com/mail/u/0/#inbox/FMfcgxwKjxGMxnZznwlTpLFcdnWdqFsw?projector=1&messagePartId=0.2

Important Books/Journals for further learning including the page nos.:
Book:

Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, "Operating System Concepts", John Wiley & Sons (ASIA) Pvt. Ltd, 9th Edition. 2015

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L19

CSE

II/IV

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **III Process Synchronization And Deadlocks**

Date of Lecture:

Topic of Lecture: Process Synchronization: Background - The critical-section problem

Introduction : (Maximum 5 sentences)

- Process Synchronization is the task of coordinating the execution of processes in a way that no two processes can have access to the same shared data and resources.
- It is specially needed in a multi-process system when multiple processes are running together, and more than one processes try to gain access to the same shared resource or data at the same time.

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

What is Process Synchronization?

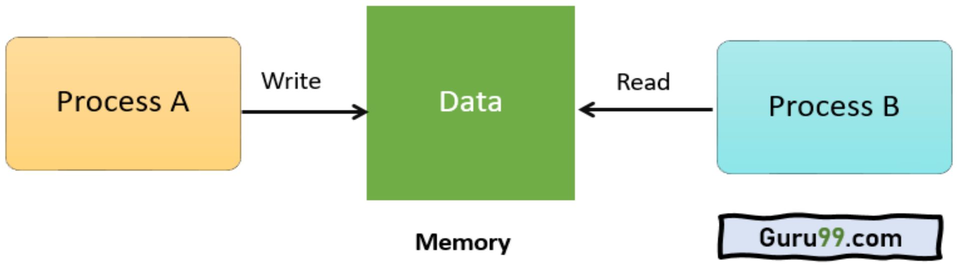
- Process Synchronization is the task of coordinating the execution of processes in a way that no two processes can have access to the same shared data and resources.
- It is specially needed in a multi-process system when multiple processes are running together, and more than one processes try to gain access to the same shared resource or data at the same time.
- This can lead to the inconsistency of shared data. So the change made by one process not necessarily reflected when other processes accessed the same shared data. To avoid this type of inconsistency of data, the processes need to be synchronized with each other.

In this operating system tutorial, you will learn:

- What is Process Synchronization?
- How Process Synchronization Works?
- Sections of a Program
- What is Critical Section Problem?
- Rules for Critical Section
- Solutions To The Critical Section

How Process Synchronization Works?

For Example, process A changing the data in a memory location while another process B is trying to read the data from the same memory location. There is a high probability that data read by the second process will be erroneous.



Sections of a Program

Here, are four essential elements of the critical section:

- **Entry Section:** It is part of the process which decides the entry of a particular process.
- **Critical Section:** This part allows one process to enter and modify the shared variable.
- **Exit Section:** Exit section allows the other process that are waiting in the Entry Section, to enter into the Critical Sections. It also checks that a process that finished its execution should be removed through this Section.
- **Remainder Section:** All other parts of the Code, which is not in Critical, Entry, and Exit Section, are known as the Remainder Section.

What is Critical Section Problem?

A critical section is a segment of code which can be accessed by a signal process at a specific point of time. The section consists of shared data resources that required to be accessed by other processes.

- The entry to the critical section is handled by the wait() function, and it is represented as P().
- The exit from a critical section is controlled by the signal() function, represented as V().

In the critical section, only a single process can be executed. Other processes, waiting to execute their critical section, need to wait until the current process completes its execution.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(203-260).

Course Teacher

Verified by HOD

LECTURE HANDOUTS **L20**

CSE **II/IV**

Course Name with Code	**: Operating Systems & 16CSD05**
Course Faculty	**: Mr.S.Pragadeeswaran**
Unit	**: III Process Synchronization And Deadlocks**

Date of Lecture:

Topic of Lecture: Semaphores – Classic Problems of Synchronization, – Monitors

Introduction : (Maximum 5 sentences)

- A large class of concurrency-control problems.
- In our solutions to the problems, we use semaphores for synchronization, since that is the traditional way to present such solutions.
- However, actual implementations of these solutions could use mutex locks in place of binary semaphores.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

- ✓ A large class of concurrency-control problems. In our solutions to the problems, we use semaphores for synchronization, since that is the traditional way to present such solutions.
- ✓ However, actual implementations of these solutions could use mutex locks in place of binary semaphores.
- ✓ These problems are used for testing nearly every newly proposed synchronization scheme. The following problems of synchronization are considered as classical problems:

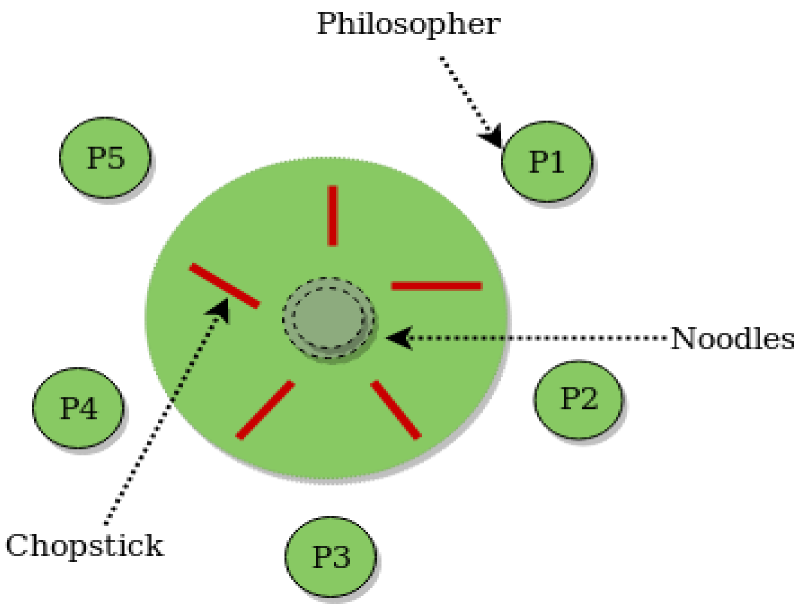
1. Bounded-buffer (or Producer-Consumer) Problem,
2. Dining-Philosphers Problem,
3. Readers and Writers Problem,
4. Sleeping Barber Problem

These are summarized, for detailed explanation, you can view the linked articles for each.

1. **Bounded-buffer (or Producer-Consumer) Problem:**
   Bounded Buffer problem is also called producer consumer problem.
2. This problem is generalized in terms of the Producer-Consumer problem. Solution to this

problem is, creating two counting semaphores “full” and “empty” to keep track of the current number of full and empty buffers respectively. Producers produce a product and consumers consume the product, but both use of one of the containers each time.

1. **Dining-Philosphers Problem:**
   The Dining Philosopher Problem states that K philosophers seated around a circular table with one chopstick between each pair of philosophers.
2. There is one chopstick between each philosopher.
3. A philosopher may eat if he can pickup the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both. This problem involves the allocation of limited resources to a group of processes in a deadlock-free and starvation-free manner.

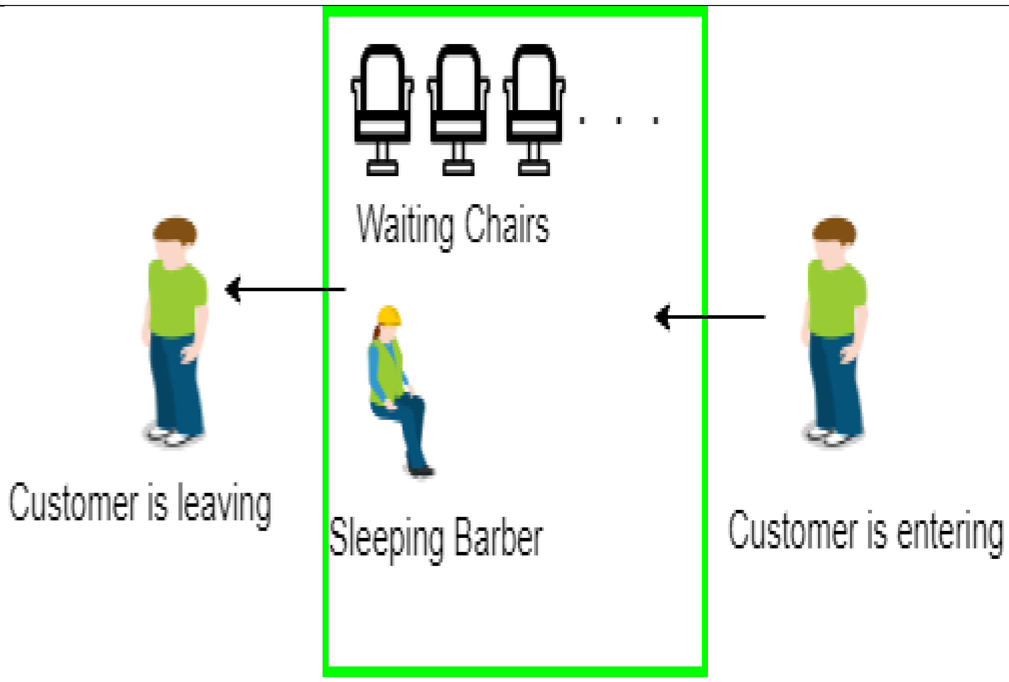


1. **Readers and Writers Problem:**
   Suppose that a database is to be shared among several concurrent processes. Some of these processes may want only to read the database, whereas others may want to update (that is, to read and write) the database.
2. We distinguish between these two types of processes by referring to the former as readers and to the latter as writers. Precisely in OS we call this situation as the readers-writers problem

Problem parameters:

- One set of data is shared among a number of processes.
- Once a writer is ready, it performs its write. Only one writer may write at a time.
- If a process is writing, no other process can read it.
- If at least one reader is reading, no other process can write.
- Readers may not write and only read.

3. **Sleeping Barber Problem:**
   Barber shop with one barber, one barber chair and N chairs to wait in. When no customers the barber goes to sleep in barber chair and must be woken when a customer comes in. When barber is cutting hair new custmers take empty seats to wait, or leave if no vacancy.



Video Content / Details of website for further learning (if any): http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12
Important Books/Journals for further learning including the page nos.: **Book:** Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(203-260).

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L21

CSE

II/IV

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **III Process Synchronization And Deadlocks**

Date of Lecture:

Topic of Lecture: Deadlocks: System model

Introduction : (Maximum 5 sentences)

- **Deadlock** is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.
- Consider an example when two trains are coming toward each other on the same track and there is only one track, none of the trains can move once they are in front of each other.
- A similar situation occurs in operating systems when there are two or more processes that hold some resources and wait for resources held by other(s)

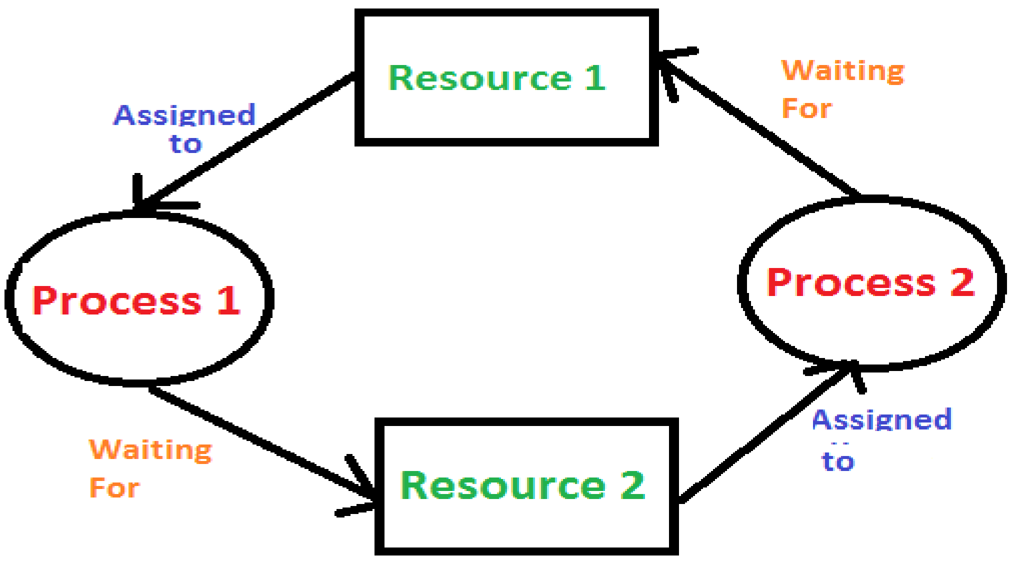
Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

A process in operating systems uses different resources and uses resources in the following way.
1) Requests a resource
2) Use the resource
3) Releases the resource

- ✓ **Deadlock** is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.
- ✓ Consider an example when two trains are coming toward each other on the same track and there is only one track, none of the trains can move once they are in front of each other.
- ✓ A similar situation occurs in operating systems when there are two or more processes that hold some resources and wait for resources held by other(s). For example, in the below diagram, Process 1 is holding Resource 1 and waiting for resource 2 which is acquired by process 2, and process 2 is waiting for resource 1.



Deadlock can arise if the **following four conditions hold simultaneously (Necessary Conditions)**
Mutual Exclusion: One or more than one resource are non-shareable (Only one process can use at a time)
Hold and Wait: A process is holding at least one resource and waiting for resources.
No Preemption: A resource cannot be taken from a process unless the process releases the resource.
Circular Wait: A set of processes are waiting for each other in circular form.
Methods for handling deadlock
There are three ways to handle deadlock
1) Deadlock prevention or avoidance: The idea is to not let the system into a deadlock state.
One can zoom into each category individually, Prevention is done by negating one of above mentioned necessary conditions for deadlock.
Avoidance is kind of futuristic in nature. By using strategy of "Avoidance", we have to make an assumption. We need to ensure that all information about resources which process will need are known to us prior to execution of the process. We use Banker's algorithm (Which is in-turn a gift from Dijkstra) in order to avoid deadlock.
2) Deadlock detection and recovery: Let deadlock occur, then do preemption to handle it once occurred.

3) Ignore the problem altogether: If deadlock is very rare, then let it happen and reboot the system. This is the approach that both Windows and UNIX take.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(203-260).

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L22

CSE

II/IV

Course Name with Code : Operating Systems & 16CSD05

Course Faculty : Mr.S.Pragadeeswaran

Unit : III Process Synchronization And Deadlocks

Date of Lecture:

Topic of Lecture: Deadlock Characterization

Introduction : (Maximum 5 sentences)

- ✓ A deadlock happens in operating system when two or more processes need some resource to complete their execution that is held by the other process.
- ✓ A deadlock occurs if the four Coffman conditions hold true. But these conditions are not mutually exclusive

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

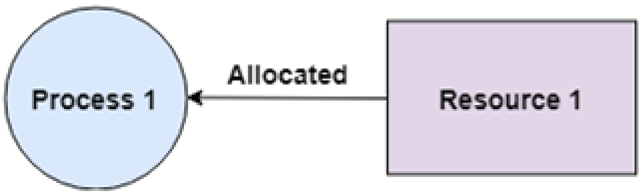
- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

- ✓ A deadlock happens in operating system when two or more processes need some resource to complete their execution that is held by the other process.
- ✓ A deadlock occurs if the four Coffman conditions hold true. But these conditions are not mutually exclusive. They are given as follows –

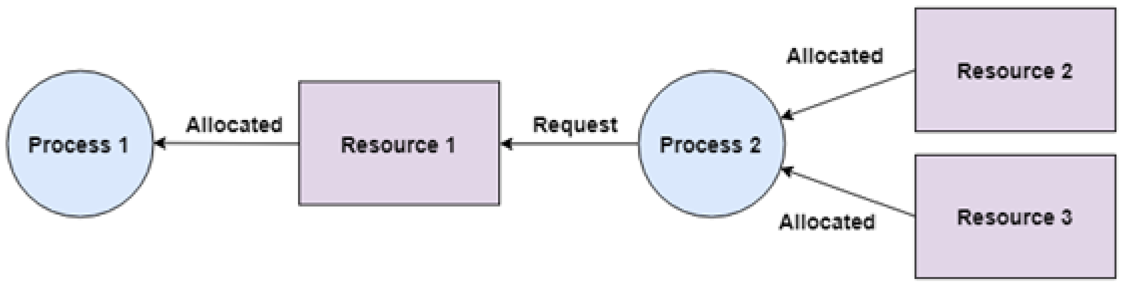
Mutual Exclusion

There should be a resource that can only be held by one process at a time. In the diagram below, there is a single instance of Resource 1 and it is held by Process 1 only.



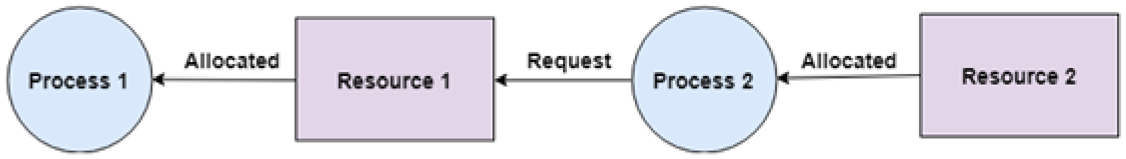
Hold and Wait

A process can hold multiple resources and still request more resources from other processes which are holding them. In the diagram given below, Process 2 holds Resource 2 and Resource 3 and is requesting the Resource 1 which is held by Process 1.



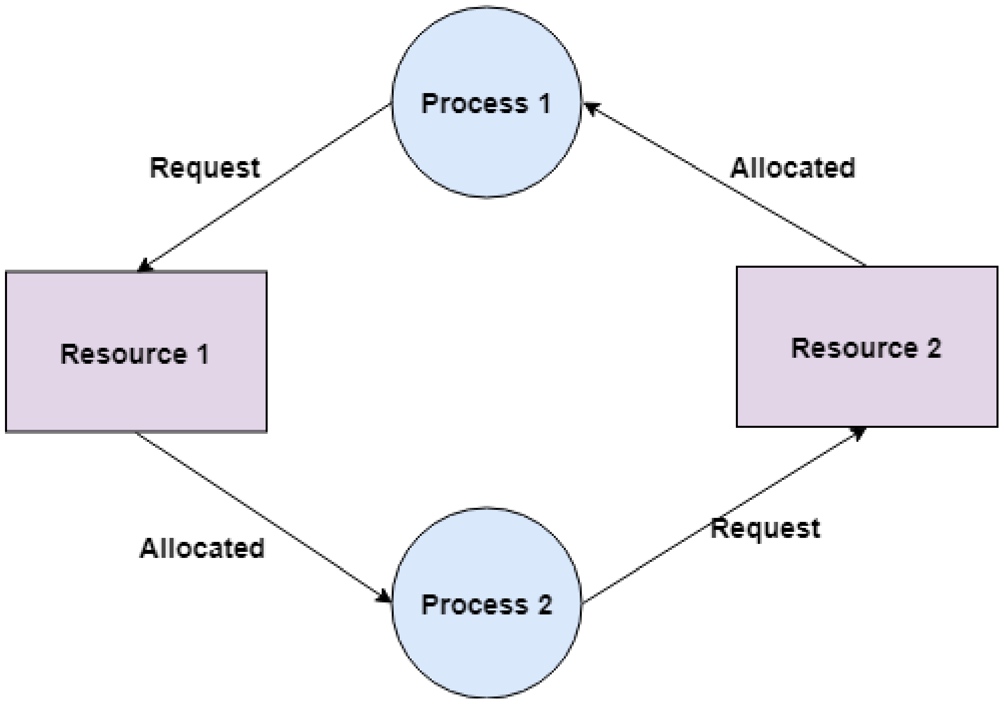
No Preemption

A resource cannot be preempted from a process by force. A process can only release a resource voluntarily. In the diagram below, Process 2 cannot preempt Resource 1 from Process 1. It will only be released when Process 1 relinquishes it voluntarily after its execution is complete.



Circular Wait

A process is waiting for the resource held by the second process, which is waiting for the resource held by the third process and so on, till the last process is waiting for a resource held by the first process. This forms a circular chain. For example: Process 1 is allocated Resource2 and it is requesting Resource 1. Similarly, Process 2 is allocated Resource 1 and it is requesting Resource 2. This forms a circular wait loop.



Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(203-260).

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L23

CSE

II/IV

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **III Process Synchronization And Deadlocks**

Date of Lecture:

Topic of Lecture: Methods for Handling Deadlocks

Introduction : (Maximum 5 sentences)

- Deadlock detection, deadlock prevention and deadlock avoidance are the main methods for handling deadlocks. Details about these are given as follows −

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Deadlock detection, deadlock prevention and deadlock avoidance are the main methods for handling deadlocks. Details about these are given as follows −

Deadlock Detection

Deadlock can be detected by the resource scheduler as it keeps track of all the resources that are allocated to different processes. After a deadlock is detected, it can be handed using the given methods −

- All the processes that are involved in the deadlock are terminated. This approach is not that useful as all the progress made by the processes is destroyed.
- Resources can be preempted from some processes and given to others until the deadlock situation is resolved.

Deadlock Prevention

It is important to prevent a deadlock before it can occur. So, the system checks each transaction before it is executed to make sure it does not lead to deadlock. If there is even a slight possibility that a transaction may lead to deadlock, it is never allowed to execute.

Some deadlock prevention schemes that use timestamps in order to make sure that a deadlock does not occur are given as follows −

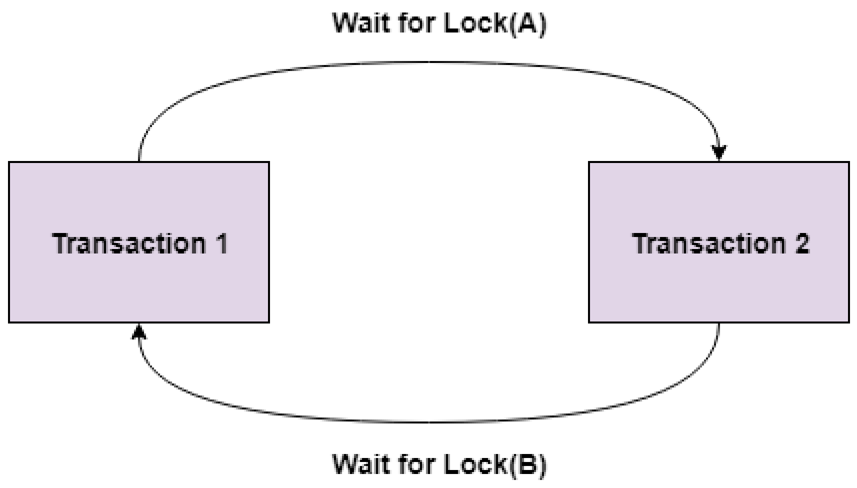
- **Wait - Die Scheme**
- In the wait - die scheme, if a transaction T1 requests for a resource that is held by transaction T2, one of the following two scenarios may occur −
    - TS(T1) < TS(T2) - If T1 is older than T2 i.e T1 came in the system earlier than T2, then it is allowed to wait for the resource which will be free when T2 has completed its execution.
    - TS(T1) > TS(T2) - If T1 is younger than T2 i.e T1 came in the system after T2, then T1 is killed. It is restarted later with the same timestamp.
- **Wound - Wait Scheme**
- In the wound - wait scheme, if a transaction T1 requests for a resource that is held by transaction T2, one of the following two possibilities may occur −
    - TS(T1) < TS(T2) - If T1 is older than T2 i.e T1 came in the system earlier than T2, then it is allowed to roll back T2 or wound T2. Then T1 takes the resource and completes its execution. T2 is later restarted with the same timestamp.
    - TS(T1) > TS(T2) - If T1 is younger than T2 i.e T1 came in the system after T2, then it is allowed to wait for the resource which will be free when T2 has completed its execution.

Deadlock Avoidance

It is better to avoid a deadlock rather than take measures after the deadlock has occurred. The wait for graph can be used for deadlock avoidance. This is however only useful for smaller databases as it can get quite complex in larger databases.

Wait for graph

The wait for graph shows the relationship between the resources and transactions. If a transaction requests a resource or if it already holds a resource, it is visible as an edge on the wait for graph. If the wait for graph contains a cycle, then there may be a deadlock in the system, otherwise not.



Ostrich Algorithm

- ✓ The ostrich algorithm means that the deadlock is simply ignored and it is assumed that it will never occur. This is done because in some systems the cost of handling the deadlock is much higher than simply ignoring it as it occurs very rarely.
- ✓ So, it is simply assumed that the deadlock will never occur and the system is rebooted if it occurs by any chance.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,**Page no**(203-260).

Course Teacher

Verified by HOD

LECTURE HANDOUTS **L24**

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **III Process Synchronization And Deadlocks**

Date of Lecture:

Topic of Lecture: Deadlock Prevention

Introduction : (Maximum 5 sentences)

Deadlock Characteristics
As discussed in the previous post, deadlock has following characteristics.

1. Mutual Exclusion
2. Hold and Wait
3. No preemption
4. Circular wait

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Deadlock Characteristics
As discussed in the previous post, deadlock has following characteristics.

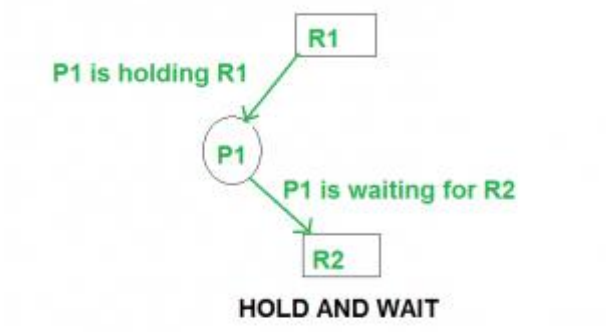
1. Mutual Exclusion
2. Hold and Wait
3. No preemption
4. Circular wait

Deadlock Prevention
We can prevent Deadlock by eliminating any of the above four conditions.

Eliminate Mutual Exclusion
It is not possible to dis-satisfy the mutual exclusion because some resources, such as the tape drive and printer, are inherently non-shareable.

Eliminate Hold and wait

1. Allocate all required resources to the process before the start of its execution, this way hold and wait condition is eliminated but it will lead to low device utilization. for example, if a process requires printer at a later time and we have allocated printer before the start of its execution printer will remain blocked till it has completed its execution.
2. The process will make a new request for resources after releasing the current set of resources. This solution may lead to starvation.



Eliminate No Preemption
Preempt resources from the process when resources required by other high priority processes.
Deadlock Avoidance
Deadlock avoidance can be done with Banker's Algorithm.

Banker's Algorithm
Bankers's Algorithm is resource allocation and deadlock avoidance algorithm which test all the request made by processes for resources, it checks for the safe state, if after granting request system remains in the safe state it allows the request and if there is no safe state it doesn't allow the request made by the process.

Inputs to Banker's Algorithm:

1. Max need of resources by each process.
2. Currently allocated resources by each process.
3. Max free available resources in the system.

The request will only be granted under the below condition:

1. If the request made by the process is less than equal to max need to that process.
2. If the request made by the process is less than equal to the freely available resource in the system.

Example:
Total resources in system:

A B C D

6 5 7 6

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(203-260).

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS **L25**

CSE **II/IV**

Course Name with Code	**: Operating Systems & 16CSD05**
Course Faculty	**: Mr.S.Pragadeeswaran**
Unit	**: III Process Synchronization And Deadlocks**

Date of Lecture:

Topic of Lecture: Deadlock Avoidance

Introduction : (Maximum 5 sentences)

- ✓ In deadlock avoidance, the request for any resource will be granted if the resulting state of the system doesn't cause deadlock in the system.
- ✓ The state of the system will continuously be checked for safe and unsafe states.In order to avoid deadlocks, the process must tell OS, the maximum number of resources a process can request to complete its execution.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Deadlock avoidance

- In deadlock avoidance, the request for any resource will be granted if the resulting state of the system doesn't cause deadlock in the system. The state of the system will continuously be checked for safe and unsafe states.In order to avoid deadlocks, the process must tell OS, the maximum number of resources a process can request to complete its execution.
- The simplest and most useful approach states that the process should declare the maximum number of resources of each type it may ever need. The Deadlock avoidance algorithm examines the resource allocations so that there can never be a circular wait condition.

Safe and Unsafe States

- The resource allocation state of a system can be defined by the instances of available and allocated resources, and the maximum instance of the resources demanded by the processes.

- Above tables and vector E, P and A describes the resource allocation state of a system. There are 4 processes and 4 types of the resources in a system. Table 1 shows the instances of each resource assigned to each process.
- The instances of the resources, each process still needs. Vector E is the representation of total instances of each resource in the system.Vector P represents the instances of resources that have been assigned to processes. Vector A represents the number of resources that are not in use.
- A state of the system is called safe if the system can allocate all the resources requested by all the processes without entering into deadlock.If the system cannot fulfill the request of all processes then the state of the system is called unsafe.The key of Deadlock avoidance approach is when the request is made for resources then the request must only be approved in the case if the resulting state is also a safe state.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(203-260).

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS **L26**

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **III Process Synchronization And Deadlocks**

Date of Lecture:

Topic of Lecture: Deadlock Detection

Introduction : (Maximum 5 sentences)

- Deadlock Prevention and Avoidance.
- Deadlock Detection and Recovery

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

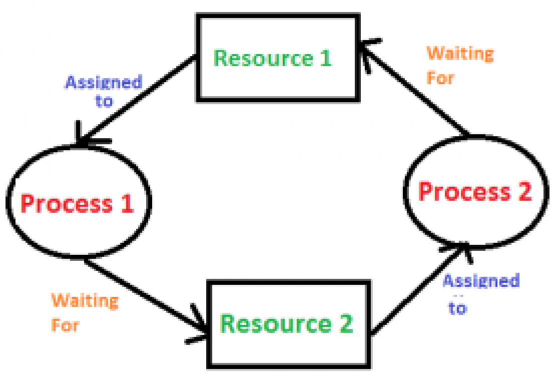
- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Deadlock Detection And Recovery

Deadlock Detection

1. If resources have single instance:
   In this case for Deadlock detection we can run an algorithm to check for cycle in the Resource Allocation Graph. Presence of cycle in the graph is the sufficient condition for deadlock.



   In the above diagram, resource 1 and resource 2 have single instances. There is a cycle R1 → P1 → R2 → P2. So, Deadlock is Confirmed.

2. If there are multiple instances of resources:

3. Detection of the cycle is necessary but not sufficient condition for deadlock detection, in this case, the system may or may not be in deadlock varies according to different situations.

Deadlock Recovery

A traditional operating system such as Windows doesn't deal with deadlock recovery as it is time and space consuming process. Real-time operating systems use Deadlock recovery.

Recovery method

- **Killing the process:** killing all the process involved in the deadlock. Killing process one by one. After killing each process check for deadlock again keep repeating the process till system recover from deadlock.
- **Resource Preemption:** Resources are preempted from the processes involved in the deadlock, preempted resources are allocated to other processes so that there is a possibility of recovering the system from deadlock. In this case, the system goes into starvation.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(203-260).

Course Teacher

Verified by HOD

LECTURE HANDOUTS **L27**

CSE **II/IV**

Course Name with Code : **Operating Systems & 19ITC09**

Course Faculty : **P.Bhuvaneshwari**

Unit : **III Process Synchronization And Deadlocks**

Date of Lecture:

Topic of Lecture: Recovery from Deadlocks.

Introduction : (Maximum 5 sentences)

- Prerequisite – Deadlock Detection And Recovery
  When a Deadlock Detection Algorithm determines that a deadlock has occurred in the system, the system must recover from that deadlock

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

1. Process Termination:

To eliminate the deadlock, we can simply kill one or more processes. For this, we use two methods:

- **(a). Abort all the Deadlocked Processes:**
  Aborting all the processes will certainly break the deadlock, but with a great expenses. The deadlocked processes may have computed for a long time and the result of those partial computations must be discarded and there is a probability to recalculate them later.
- **(b). Abort one process at a time untill deadlock is eliminated:**
  Abort one deadlocked process at a time, untill deadlock cycle is eliminated from the system. Due to this method, there may be considerable overhead, because after aborting each process, we have to run deadlock detection algorithm to check whether any processes are still deadlocked.

2. Resource Preemption:

To eliminate deadlocks using resource preemption, we preepmt some resources from processes and give those resources to other processes. This method will raise three issues –

- **(a). Selecting a victim:**
  We must determine which resources and which processes are to be preempted and also the order to minimize the cost.
- **(b). Rollback:**
  We must determine what should be done with the process from which resources are preempted.

One simple idea is total rollback. That means abort the process and restart it.

- **(c). Starvation:**
  In a system, it may happen that same process is always picked as a victim. As a result, that process will never complete its designated task. This situation is called **Starvation** and must be avoided.

Deadlock Recovery

A traditional operating system such as Windows doesn't deal with deadlock recovery as it is time and space consuming process. Real-time operating systems use Deadlock recovery.

Recovery method

1. **Killing the process:** killing all the process involved in the deadlock. Killing process one by one. After killing each process check for deadlock again keep repeating the process till system recover from deadlock.
2. **Resource Preemption:** Resources are preempted from the processes involved in the deadlock, preempted resources are allocated to other processes so that there is a possibility of recovering the system from deadlock. In this case, the system goes into starvation.

Video Content / Details of website for further learning (if any):

http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:

Book:

Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(203-260).

Course Teacher

Verified by HOD

LECTURE HANDOUTS | **L28**

CSE | **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **IV Memory Management**

Date of Lecture:

Topic of Lecture: Management Strategies: Background , Swapping

Introduction : (Maximum 5 sentences)

- 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.

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

- Memory management is concerned with managing the primary memory. Memory consists of array of bytes or words each with their own address. The instructions are fetched from the memory by the CPU based on the value program counter.

Functions of memory management:

✓ Keeping track of status of each memory location.
✓ Determining the allocation policy.
✓ Memory allocation technique.
✓ De-allocation technique.

Address Binding:

- ✓ Programs are stored on the secondary storage disks as binary executable files. When the programs are to be executed they are brought in to the main memory and placed within a process.
- ✓ The collection of processes on the disk waiting to enter the main memory forms the input queue.
- ✓ One of the processes which are to be executed is fetched from the queue and placed in the main memory.
- ✓ During the execution it fetches instruction and data from main memory. After the process terminates it returns back the memory space.
- ✓ During execution the process will go through different steps and in each step the address is represented in different ways.
- ✓ In source program the address is symbolic.
- ✓ The compiler converts the symbolic address to re-locatable address. The loader will convert this re-locatable address to absolute address.
- ✓ Binding of instructions and data can be done at any step along the way:
- ✓ Compile time:-If we know whether the process resides in memory then absolute code can be
- ✓ generated. If the static address changes then it is necessary to re-compile the code from the beginning.
- ✓ Load time:-If the compiler doesn't know whether the process resides in memory then it generates
- ✓ the re-locatable code. In this the binding is delayed until the load time.
- ✓ Execution time:-If the process is moved during its execution from one memory segment to
- ✓ another then the binding is delayed until run time. Special hardware is used for this. Most of the general
- ✓ purpose operating system uses this method.

Dynamic Loading:

- For a process to be executed it should be loaded in to the physical memory. The size of the process is limited to the size of the physical memory.
- Dynamic loading is used to obtain better memory utilization. In dynamic loading the routine or procedure will not be loaded until it is called.
- Whenever a routine is called, the calling routine first checks whether the called routine is already loaded or not.
- If it is not loaded it cause the loader to load the desired program in to the memory and updates the programs address table to indicate the change and control is passed to newly called routine.

Video Content / Details of website for further learning (if any): http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12
Important Books/Journals for further learning including the page nos.: **Book:**Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(351-466**).**

Course Teacher

Verified by HOD

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **IV Memory Management**

Date of Lecture:

Topic of Lecture: Contiguous Memory Allocation –Non- Contiguous Memory Allocation

Introduction : (Maximum 5 sentences)

- Contiguous memory allocation is basically a method in which a single contiguous section/part of memory is allocated to a process or file needing it.
- We can implement/achieve contiguous memory allocation by dividing the memory partitions into fixed size partitions

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

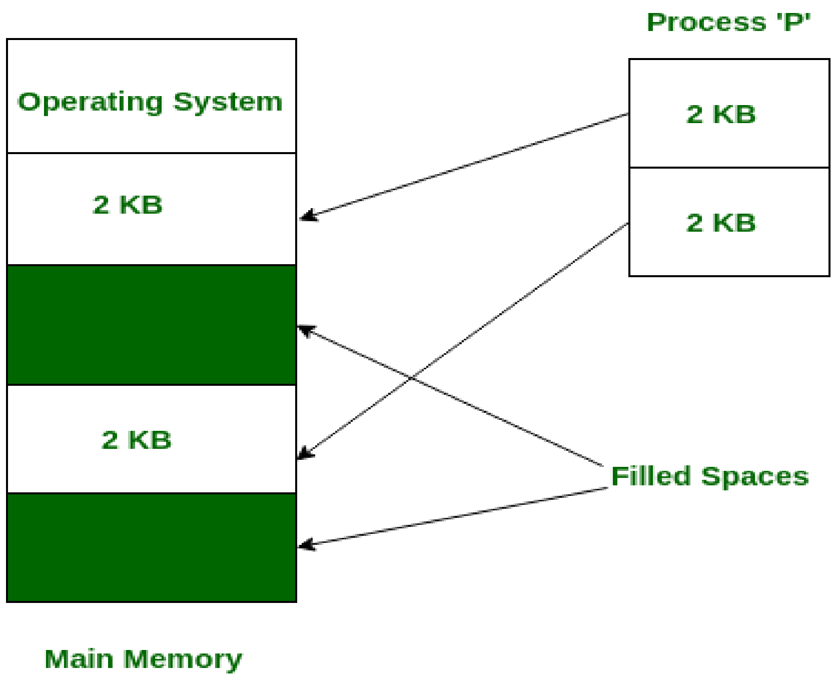
- In non-contiguous allocation, Operating system needs to maintain the table which is called **Page Table** for each process which contains the base address of the each block which is acquired by the process in memory space.
- In non-contiguous memory allocation, different parts of a process is allocated different places in Main Memory. Spanning is allowed which is not possible in other techniques like Dynamic or Static Contiguous memory allocation.
- That's why paging is needed to ensure effective memory allocation. Paging is done to remove External Fragmentation.

Working:

Here a process can be spanned across different spaces in main memory in non-consecutive manner. Suppose process P of size 4KB. Consider main memory have two empty slots each of size 2KB. Hence total free space is, 2*2= 4 KB. In contiguous memory allocation, process P cannot be accommodated as spanning is not allowed.

- In contiguous allocation, space in memory should be allocated to whole process. If not, then that space remains unallocated
- But in Non-Contiguous allocation, process can be divided into different parts and hence filling the space in main memory.
- In this example, process P can be divided into two parts of equal size – 2KB. Hence one

part of process P can be allocated to first 2KB space of main memory and other part of processP can be allocated to second 2KB space of main memory.



- But, in what manner we divide a process to allocate them into main memory is very important to understand. Process is divided after analysing the number of empty spaces and their size in main memory.
- Then only we divide our process. It is very time consuming process.
- Their number as well as their sizes changing every time due to execution of already present processes in main memory.
- In order to avoid this time consuming process, we divide our process in secondary memory in advance before reaching the main memory for its execution.
- Every process is divided into various parts of equal size called Pages. We also divide our main memory into different parts of equal size called Frames. It is important to understand that

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(351-466).

Course Teacher

Verified by HOD

LECTURE HANDOUTS **L30**

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **IV Memory Management**

Date of Lecture:

Topic of Lecture: Segmentation – Paging

Introduction : (Maximum 5 sentences)

- Most users do not think memory as a linear array of bytes rather the users thinks memory as a collection of variable sized segments which are dedicated to a particular use such as code, data, stack, heap etc.
- A logical address is a collection of segments. Each segment has a name and length. The address specifies both the segment name and the offset within the segments.
- The users specify address by using two quantities: a segment name and an offset. For simplicity the segments are numbered and referred by a segment number

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Hardware support:

- We must define an implementation to map 2D user defined address in to 1D physical address. This mapping is affected by a segment table. Each entry in the segment table has a segment base and segment limit. The segment base contains the starting physical address where the segment resides and limit specifies the length of the segment.
- The use of segment table is shown in the above figure: Logical address consists of two parts: segment number‘s’ and an offset‘d’ to that segment. The segment number is used as an index to segment table. The offset’d’ must be in between 0 and limit, if not an error is reported to OS. If legal the offset is added to the base to generate the actual physical address. The segment table is an array of base limit register pairs.

Protection and Sharing:

- A particular advantage of segmentation is the association of protection with the segments.
- The memory mapping hardware will check the protection bits associated with each segment table entry to prevent illegal access to memory like attempts to write in to read-only segment.
- Another advantage of segmentation involves the sharing of code or data. Each process has a segment table associated with it.
- Segments are shared when the entries in the segment tables of two different processes points to same physical location.
- Sharing occurs at the segment table. Any information can be shared at the segment level. Several segments can be shared so a program consisting of several segments can be shared.

Advantages: Eliminates fragmentation. x Provides virtual growth. Allows dynamic segment growth.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(351-466).

Course Teacher

Verified by HOD

Estd. 2000

LECTURE HANDOUTS **L31**

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **IV Memory Management**

Date of Lecture:

opic of Lecture: Segmentation with Paging

Introduction : (Maximum 5 sentences)

- Paging is a memory management scheme that permits the physical address space of a process to be non-contiguous.
- Support for paging is handled by hardware. It is used to avoid external fragmentation. Paging avoids the considerable problem of fitting the varying sized memory chunks on to the backing store.
- When some code or date residing in main memory need to be swapped out, space must be found on backing store.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Basic Method:

- Physical memory is broken in to fixed sized blocks called frames (f). Logical memory is broken in to blocks of same size called pages (p).
- When a process is to be executed its pages are loaded in to available frames from backing store. The blocking store is also divided in to fixed-sized blocks of same size as memory frames. The following figure shows paging hardware

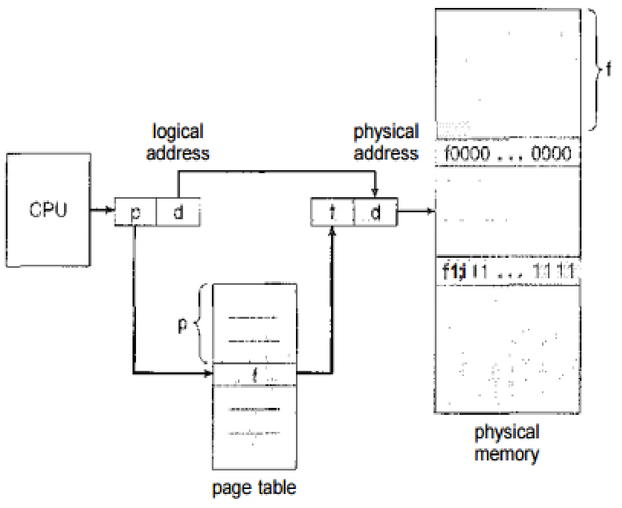


Figure 8.7 Paging hardware.

- Logical address generated by the CPU is divided in to two parts: page number (p) and page offset (d). The page number (p) is used as index to the page table.
- The page table contains base address of each page in physical memory.
- This base address is combined with the page offset to define the physical memory i.e., sent to the memory unit.
- The page size is defined by the hardware. The size of a power of 2, varying between 512 bytes and 10Mb per page. If the size of logical address space is 2^m address unit and page size is 2^n, then high order m-n designates the page number and n low order bits represents page offset.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(351-466).

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L32

CSE

II/IV

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **IV Memory Management**

Date of Lecture:

Topic of Lecture: Structure of the Page Table

Introduction : (Maximum 5 sentences)

- A Page Table is the data structure used by a virtual memory system in a computer operating system to store the mapping between virtual addresses and physical addresses.
- Common Techniques used for structuring the page table are :

✓ Hierarchical paging
✓ Hashed page tables
✓ Inverted page tables

- Hierarchical Paging:

✓ It is also known as multilevel paging.
✓ The page table might be too big to fit in a contiguous space , so we may have a hierarchy with several levels.
✓ So , we break up the logical address space into multiple page tables.
✓ For this a simple techniques we can use are: Two level page table Three level page table

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

a. Hierarchical paging:

- Recent computer system support a large logical address apace from 2^32 to 2^64. In this system the page table becomes large. So it is very difficult to allocate contiguous main memory for page table. One simple solution to this problem is to divide page table in to smaller pieces. There are several ways to accomplish this division.

- One way is to use two-level paging algorithm in which the page table itself is also paged.
- Eg:-In a 32 bitmachine with page size of 4kb. A logical address is divided in to a page number consisting of 20 bits and a page offset of 12 bit. The page table is further divided since the page table is paged, the page number is further divided in to 10 bit page number and a 10 bit offset

b. Hashed page table:

✓ Hashed page table handles the address space larger than 32 bit. The virtual page number is used as hashed value. Linked list is used in the hash table which contains a list of elements that hash to the same location.

✓ Each element in the hash table contains the following three fields: Virtual page number x Mapped page frame value x Pointer to the next element in the linked list

Working:

✓ Virtual page number is taken from virtual address. Virtual page number is hashed in to hash table.

✓ Virtual page number is compared with the first element of linked list. Both the values are matched, that value is (page frame) used for calculating the physical address. If not match then entire linked list is searched for matching virtual page number. Clustered pages are similar to hash table

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(351-466).

Course Teacher

Verified by HOD

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **IV Memory Management**

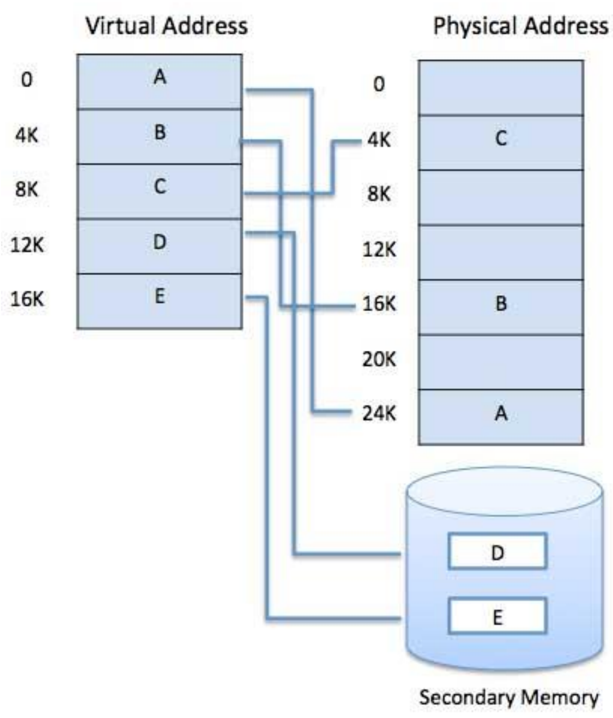
Date of Lecture:

Topic of Lecture: Virtual Memory: Background
Introduction : (Maximum 5 sentences) ✓ A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to emulate the computer's RAM. ✓ The main visible advantage of this scheme is that programs can be larger than physical memory. ✓ Virtual memory serves two purposes. ✓ First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.
Prerequisite knowledge for Complete understanding and learning of Topic: **(Max. Four important topics)** • Operating Systems • Hardware • Software • Process
Detailed content of the Lecture: ✓ Preceding sections talked about how to avoid memory fragmentation by breaking process memory requirements down into smaller bites (pages), and storing the pages non-contiguously in memory. ✓ However the entire process still had to be stored in memory somewhere. In practice, most real processes do not need all their pages, or at least not all at once, for several reasons: o Error handling code is not needed unless that specific error occurs, some of which are quite rare. o Arrays are often over-sized for worst-case scenarios, and only a small fraction of the arrays are actually used in practice. o Certain features of certain programs are rarely used, such as the routine to balance the federal budget. :-) The ability to load only the portions of processes that were actually needed (and only when they were needed) has several benefits: o Programs could be written for a much larger address space (virtual memory space) than physically exists on the computer. o Because each process is only using a fraction of their total address space, there is more memory left for other programs, improving CPU utilization and system

throughput.

o Less I/O is needed for swapping processes in and out of RAM, speeding things up.

Figure below shows the general layout of **virtual memory**, which can be much larger than physical memory:



Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(351-466).

Course Teacher

Verified by HOD

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **IV Memory Management**

Date of Lecture:

Topic of Lecture: Demand Paging – Page Replacement

Introduction : (Maximum 5 sentences)

- ✓ The basic idea behind demand paging is that when a process is swapped in, its pages are not swapped in all at once.
- ✓ Rather they are swapped in only when the process needs them. (on demand.) This is termed a lazy swapper, although a pager is a more accurate term.
- ✓ The operating system copies a disk page into physical memory only if an attempt is made to access it and that page is not already in memory (i.e., if a page fault occurs).

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Basic Concepts

- ✓ The basic idea behind paging is that when a process is swapped in, the pager only loads into memory those pages that it expects the process to need (right away.)
- ✓ Pages that are not loaded into memory are marked as invalid in the page table, using the invalid bit. (The rest of the page table entry may either be blank or contain information about where to find the swapped-out page on the hard drive.)
- ✓ If the process only ever accesses pages that are loaded in memory (memory resident pages), then the process runs exactly as if all the pages were loaded in to memory

- On the other hand, if a page is needed that was not originally loaded up, then a page fault trap is generated, which must be handled in a series of steps:
  - o The memory address requested is first checked, to make sure it was a valid memory request.
  - o If the reference was invalid, the process is terminated. Otherwise, the page must be paged in.

A free frame is located, possibly from a free-frame list.

A disk operation is scheduled to bring in the necessary page from disk.(This will usually block the process on an I/O wait,allowing some other process to use the CPU in the meantime.)

When the I/O operation is complete, the process's page table is updated with the new frame number, and the invalid bit is changed to indicate that this is now a valid page reference.

The instruction that caused the page fault must now be restarted from the beginning,

(as soon as this process gets another turn on the CPU.)

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(351-466).

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS **L35**

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **IV Memory Management**

Date of Lecture:

Topic of Lecture: Allocation of Frames

Introduction : (Maximum 5 sentences)

- An important aspect of operating systems, virtual memory is implemented using demand paging. Demand paging necessitates the development of a page-replacement algorithm and a **frame allocation algorithm**.
- Frame allocation algorithms are used if you have multiple processes; it helps decide how many frames to allocate to each process.

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Equal allocation:

In a system with x frames and y processes, each process gets equal number of frames, i.e. x/y. For instance, if the system has 48 frames and 9 processes, each process will get 5 frames. The three frames which are not allocated to any process can be used as a free-frame buffer pool.

Disadvantage:

In systems with processes of varying sizes, it does not make much sense to give each process equal frames. Allocation of a large number of frames to a small process will eventually lead to the wastage of a large number of allocated unused frames.

Proportional allocation:

Frames are allocated to each process according to the process size.

For a process p_i of size s_i, the number of allocated frames is $\mathbf{a_i = (s_i/S)*m}$, where S is the sum of the sizes of all the processes and m is the number of frames in the system. For instance, in a system with 62 frames, if there is a process of 10KB and another process of 127KB, then the first process will be allocated (10/137)*62 = 4 frames and the other process will get (127/137)*62 = 57 frames.

Advantage:

All the processes share the available frames according to their needs, rather than equally.

Global vs Local Allocation –

The number of frames allocated to a process can also dynamically change depending on whether you have used **global replacement** or **local replacement** for replacing pages in case of a page fault.

Local replacement: When a process needs a page which is not in the memory, it can bring in the new page and allocate it a frame from its own set of allocated frames only.

Advantage: The pages in memory for a particular process and the page fault ratio is affected by the paging behavior of only that process.

Disadvantage: A low priority process may hinder a high priority process by not making its frames available to the high priority process.

Global replacement: When a process needs a page which is not in the memory, it can bring in the new page and allocate it a frame from the set of all frames, even if that frame is currently allocated to some other process; that is, one process can take a frame from another.

Advantage: Does not hinder the performance of processes and hence results in greater system throughput.

Disadvantage: The page fault ratio of a process can not be solely controlled by the process itself. The pages in memory for a process depends on the paging behavior of other processes as wel

Global replacement allows a process to select a replacement frame from the set of all frames, even if that frame is currently allocated to some other process; one process can take a frame from another.

Local replacement requires that each process selects from only its own set of allocated frames.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(351-466).

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L36

CSE

II/IV

Course Name with Code	**: Operating Systems & 16CSD05**
Course Faculty	**: Mr.S.Pragadeeswaran**
Unit	**: IV Memory Management**

Date of Lecture:

Topic of Lecture: Thrashing

Introduction : (Maximum 5 sentences)

- If a process cannot maintain its minimum required number of frames, then it must be swapped out, freeing up frames for other processes. This is an intermediate level of CPU scheduling.
- But what about a process that can keep its minimum, but cannot keep all of the frames that it is currently using on a regular basis? In this case it is forced to page out pages that it will need again in the very near future, leading to large numbers of page faults.
- A process that is spending more time paging than executing is said to be thrashing.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

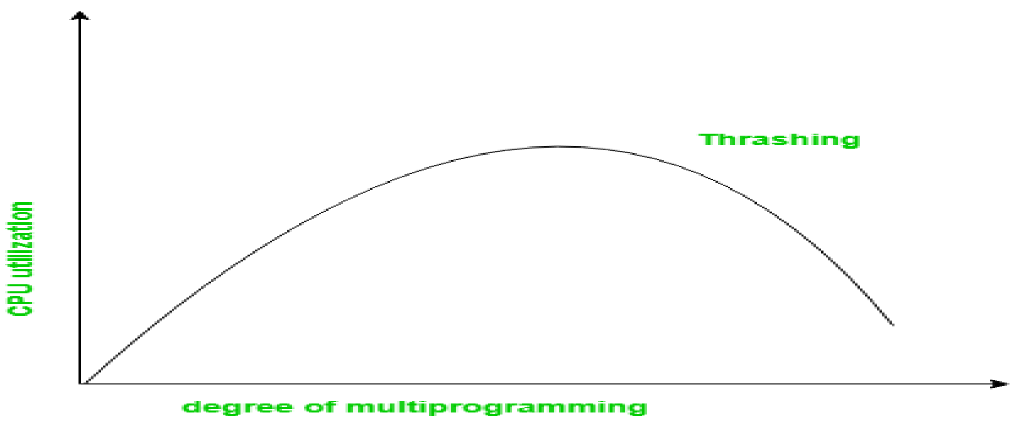
Detailed content of the Lecture:

Cause of Thrashing

✓ Early process scheduling schemes would control the level of multiprogramming allowed based on CPU utilization, adding in more processes when CPU utilization was low.

✓ The problem is that when memory filled up and processes started spending lots of time waiting for their pages to page in, then CPU utilization would lower, causing the schedule to add in even more processes and exacerbating the problem! Eventually the system would essentially grind to a halt.

✓ Local page replacement policies can prevent one thrashing process from taking pages away from other processes, but it still tends to clog up the I/O queue, thereby slowing



- To prevent thrashing we must provide processes with as many frames as they really need "right now", but how do we know what that is?

- The locality model notes that processes typically access memory references in a given locality, making lots of references to the same general area of memory before moving periodically to a new locality, as shown in Figure 9.19 below. If we could just keep as many frames as are involved in the current locality, then page faulting would occur primarily on switches from one locality to another. (E.g. when one function exits and another is called.)

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(351-466).

Course Teacher

Verified by HOD

LECTURE HANDOUTS **L37**

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **V File System and Storage Management**

Date of Lecture:

Topic of Lecture: File System Interface: File Concept , Access Methods

Introduction : (Maximum 5 sentences)

File Concept

Computers can store information on various storage media, such as magnetic disks, magnetic tapes, and optical disks. So that the computer system will be convenient to use, the operating system provides a uniform logical view of information storage. The operating system abstracts from the physical properties of its

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

File

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

File Structure

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.

File Type

File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. Many operating systems support many types of files. Operating system like MS-DOS and UNIX have the following types of files −

Ordinary files

- These are the files that contain user information.
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the entire file.

Directory files

- These files contain list of file names and other information related to these files.

Special files

- These files are also known as device files.
- These files represent physical device like disks, terminals, printers, networks, tape drive etc.

These files are of two types −

- **Character special files** − data is handled character by character as in case of terminals or printers.
- **Block special files** − data is handled in blocks as in the case of disks and tapes.

File Access Mechanisms

File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files −

- Sequential access
- Direct/Random access
- Indexed sequential access

Sequential access

A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

Direct/Random access

- Random access file organization provides, accessing the records directly.
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
- The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

Indexed sequential access

- This mechanism is built up on base of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(467-624).

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L38

CSE

II/IV

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **V File System and Storage Management**

Date of Lecture:

Topic of Lecture: Directory and Disk Structure – Protection

Introduction : (Maximum 5 sentences)

- Directory can be defined as the listing of the related files on the disk. The directory may store some or the entire file attributes.
- To get the benefit of different file systems on the different operating systems, A hard disk can be divided into the number of partitions of different sizes.
- The partitions are also called volumes or mini disks.
- Each partition must have at least one directory in which, all the files of the partition can be listed.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Single-level directory –

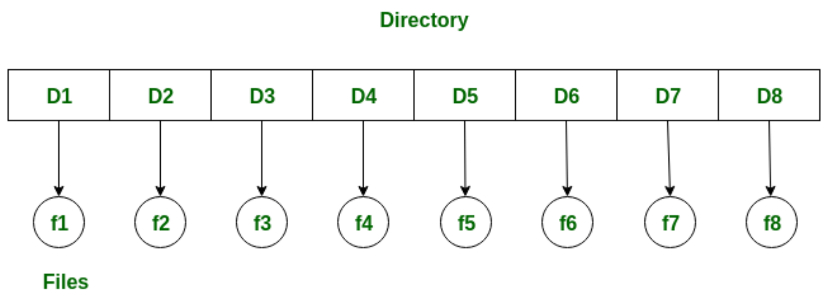
✓ Single level directory is simplest directory structure.In it all files are contained in same directory which make it easy to support and understand.

✓ 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.

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.

Two-level directory –

- ✓ 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 id=s logged in. The MFD is indexed by username or account number, and each entry points to the UFD for that user.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(467-624).

Course Teacher

Verified by HOD

LECTURE HANDOUTS **L39**

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **V File System and Storage Management**

Date of Lecture:

Topic of Lecture: File System Implementation: File System Structure

Introduction : (Maximum 5 sentences)

✓ A file is a collection of related information. The file system resides on secondary storage and provides efficient and convenient access to the disk by allowing data to be stored, located, and retrieved.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

- **I/O Control level –**
  Device drivers acts as interface between devices and Os, they help to transfer data between disk and main memory. It takes block number a input and as output it gives low level hardware specific instruction.
  /li>
- **Basic file system –**
  It Issues general commands to device driver to read and write physical blocks on disk.It manages the memory buffers and caches. A block in buffer can hold the contents of the disk block and cache stores frequently used file system metadata.
- **File organization Module –**
  It has information about files, location of files and their logical and physical blocks.Physical blocks do not match with logical numbers of logical block numbered from 0 to N. It also has a free space which tracks unallocated blocks.
- **Logical file system –**

It manages metadata information about a file i.e includes all details about a file except the actual contents of file. It also maintains via file control blocks. File control block (FCB) has information about a file – owner, size, permissions, location of file contents.

Advantages:

Duplication of code is minimized.

Each file system can have its own logical file system.

Disadvantages :

If we access many files at same time then it results in low performance.

A boot control block usually contains the information required by the system for booting an operating system from that volume. When the disks do not contain any operating system, this block can be treated as empty. This is typically the first chunk of a volume. In UFS, this is termed as the boot block; in NTFS, it is the partition boot sector.

A volume control block holds volume or the partition details, such as the number of blocks in the partition, size of the blocks or chunks, free-block count along with free-block pointers. In UFS, it is termed as superblock; in NTFS, it is stored in the master file table.

Video Content / Details of website for further learning (if any):

http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:

Book:

Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(467-624).

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L40

CSE

II/IV

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **V File System and Storage Management**

Date of Lecture:

Topic of Lecture: File System Implementation, Directory Implementation

Introduction : (Maximum 5 sentences)

A directory can be designed in two ways. In a simple design a directory consists of a list of fixed-size entries,one per file, containing a (fixed-length) file name, a structure of the file attributes, and one or more disk addresses telling where the disk blocks are.

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

File Systems are stored on disks. The above figure depicts a possible File-System Layout.

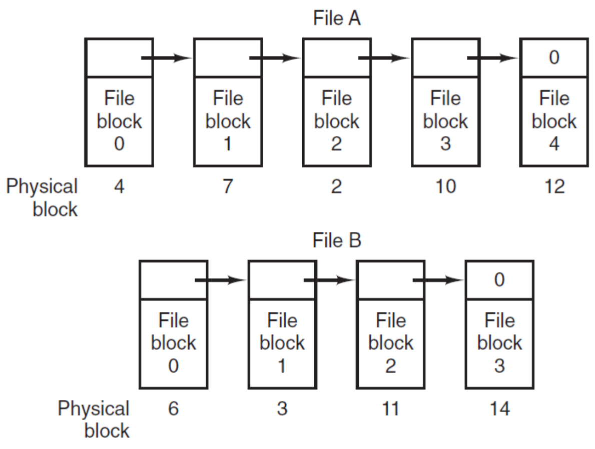
- **MBR:** Master Boot Record is used to boot the computer
- **Partition Table:** Partition table is present at the end of MBR. This table gives the starting and ending addresses of each partition.
- **Boot Block:** When the computer is booted, the BIOS reads in and executes the MBR. The first thing the MBR program does is locate the active partition, read in its first block, which is called the boot block, and execute it. The program in the boot block loads the operating system contained in that partition. Every partition contains a boot block at the beginning though it does not contain a bootable operating system.
- **Super Block:** It contains all the key parameters about the file system and

is read into memory when the computer is booted or the file system is first touched.

Implementing Files

- **Contiguous Allocation:**
  Each file is stored as a contiguous run of disk blocks.
  Example: On a disk with 1KB blocks, a 50KB file would be allocated 50 consecutive blocks. With 2KB blocks it would be 25 consecutive blocks. Each file begins at the start of a new block, so that if file A is occupying 3½ blocks, some space is wasted at the end of the last block.

- **Linked List Allocation:**
  The second method for storing files is to keep each one as a linked list of disk blocks. The first word of each block is used as a pointer to the next one. The rest of the block is for data. Unlike Contiguous allocation no space is lost in disk fragmentation.



Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(467-624).

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L41

CSE

II/IV

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **V File System and Storage Management**

Date of Lecture:

Topic of Lecture: Allocation Methods , Free Space Management.
Introduction : (Maximum 5 sentences) ✓ The system keeps tracks of the free disk blocks for allocating space to files when they are created. Also, to reuse the space released from deleting the files, free space management becomes crucial. ✓ The system maintains a free space list which keeps track of the disk blocks that are not allocated to some file or directory.
Prerequisite knowledge for Complete understanding and learning of Topic: **(Max. Four important topics)** • Operating Systems • Hardware • Software • Process
Detailed content of the Lecture: **Bitmap or Bit vector –** A Bitmap or Bit Vector is series or collection of bits where each bit corresponds to a disk block. The bit can take two values: 0 and 1: *0 indicates that the block is allocated* and 1 indicates a free block. The given instance of disk blocks on the disk in *Figure 1* (where green blocks are allocated) can be represented by a bitmap of 16 bits as: **0000111000000110**.

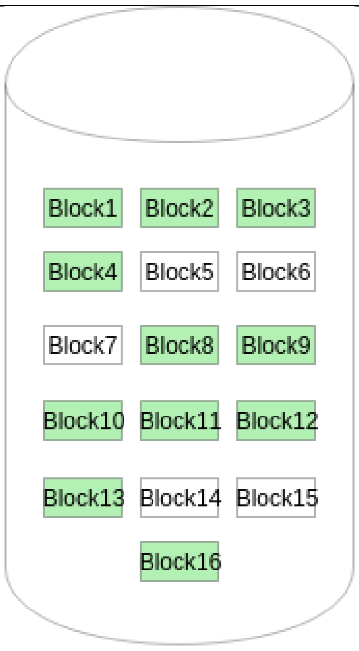


Figure - 1

Advantages –

- Simple to understand.
- Finding the first free block is efficient. It requires scanning the words (a group of 8 bits) in a bitmap for a non-zero word. (A 0-valued word has all bits 0). The first free block is then found by scanning for the first 1 bit in the non-zero word.

The block number can be calculated as:
(number of bits per word) *(number of 0-values words) + offset of bit first bit 1 in the non-zero word .

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(467-624).

Course Teacher

Verified by HOD

LECTURE HANDOUTS **L42**

CSE **II/IV**

Course Name with Code	**: Operating Systems & 16CSD05**
Course Faculty	**: Mr.S.Pragadeeswaran**
Unit	**: V File System and Storage Management**

Date of Lecture:

Topic of Lecture: Mass Storage Structure: Overview of Mass Storage Structure

Introduction : (Maximum 5 sentences)

- ✓ Each modern disk contains concentric tracks and each track is divided into multiple sectors. The disks are usually arranged as a one dimensional array of blocks, where blocks are the smallest storage unit.
- ✓ Blocks can also be called as sectors. For each surface of the disk, there is aread/write desk available. The same tracks on all the surfaces is known as a cylinder.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Disk Scheduling

There are many disk scheduling algorithms that provide the total head movement for various requests to the disk. Here are the types of disk scheduling algorithms

First Come First Serve Scheduling (FCFS)

In first come first served scheduling, the requests are serviced in their coming order. This algorithm is fair as it allows all requests a chance but it does not provide the fastest possible service. An example of FCFS scheduling is given below −

SCAN Scheduling

In this scheduling algorithm, the head moves towards one direction while servicing all the requests in that direction until it reaches the end of the disk. After that it starts moving towards the other direction. In this way, the head continuously scans back and forth across the disk. An example of SCAN scheduling

LOOK Scheduling

LOOK scheduling algorithm is similar to SCAN scheduling but is its practical version. In this algorithm, the head moves towards one direction while servicing all the requests in that direction until it reaches the last request. After that it starts moving towards the other direction. An example of LOOK scheduling

- ✓ A head crash normally cannot be repaired; the entire disk must be replaced. A disk can be removable, allowing different disks to be mounted as needed.
- ✓ Removable magnetic disks generally consist of one platter, held in a plastic case to prevent damage while not in the disk drive.
- ✓ Floppy disks are inexpensive removable magnetic disks that have a soft plastic case containing a flexible platter. The head of a floppy-disk drive generally sits directly on the disk surface, so the drive is designed to rotate more slowly than a hard-disk drive

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:

Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(467-624).

Course Teacher

Verified by HOD

LECTURE HANDOUTS **L43**

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **V File System and Storage Management**

Date of Lecture:

Topic of Lecture: Disk Structure – Disk Scheduling

Introduction : (Maximum 5 sentences)

- ✓ Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.
- ✓ Disk scheduling is important because: Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Disk scheduling is done by operating systems to schedule I/O requests arriving for the disk. Disk scheduling is also known as I/O scheduling.
Disk scheduling is important because:

- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
- Two or more request may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

There are many Disk Scheduling Algorithms but before discussing them let's have a quick look at some of the important terms:

- **Seek Time:**Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
- **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.

- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
- **Disk Access Time:** Disk Access Time is:

- **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. Average Response time is the response time of the all requests. Variance Response Time is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:
Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ,” Operating system concepts”,2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(467-624).

Course Teacher

Verified by HOD

LECTURE HANDOUTS **L44**

CSE **II/IV**

Course Name with Code : **Operating Systems & 16CSD05**

Course Faculty : **Mr.S.Pragadeeswaran**

Unit : **V File System and Storage Management**

Date of Lecture:

Topic of Lecture: Disk Management - Swap Space Management

Introduction : (Maximum 5 sentences)

- ✓ Swapping is a memory management technique used in multi-programming to increase the number of process sharing the CPU.
- ✓ It is a technique of removing a process from main memory and storing it into secondary memory, and then bringing it back into main memory for continued execution.
- ✓ This action of moving a process out from main memory to secondary memory is called Swap Out and the action of moving a process out from secondary memory to main memory is called Swap In.

Prerequisite knowledge for Complete understanding and learning of Topic:
(Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

Swap-Space :

The area on the disk where the swapped out processes are stored is called swap space.

Swap-Space Management :

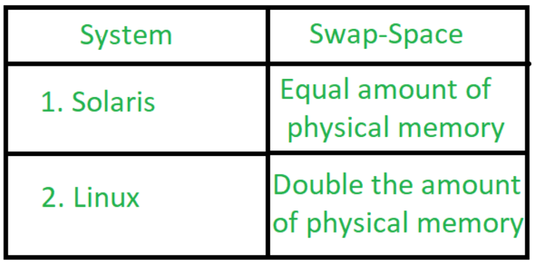
- ✓ Swap-Swap management is another low-level task pf the operating system. Disk space is used as an extension of main memory by the virtual memory.
- ✓ The fact that disk access is much slower than memory access, In the swap-space management we are using disk space, so it will significantly decreases system performance.
- ✓ Basically, in all our systems we require the best throughput, so the goal of this swap-space implementation is to provide the virtual memory the best throughput. In these article, we are going to discuss how swap space is used, where swap space is located on disk, and how swap space is managed.

Swap-Space Use :

- ✓ Swap-space is used by the different operating-system in various ways.
- ✓ The systems which are implementing swapping may use swap space to hold the entire process which may include image, code and data segments.
- ✓ Paging systems may simply store pages that have been pushed out of the main memory.

- ✓ The need of swap space on a system can vary from a megabytes to gigabytes but it also depends on the amount of physical memory, the virtual memory it is backing and the way in which it is using the virtual memory.
- ✓ It is safer to overestimate than to underestimate the amount of swap space required, because if a system runs out of swap space it may be forced to abort the processes or may crash entirely. Overestimation wastes disk space that could otherwise be used for files, but it does not harm other.

Following table shows different system using amount of swap space:



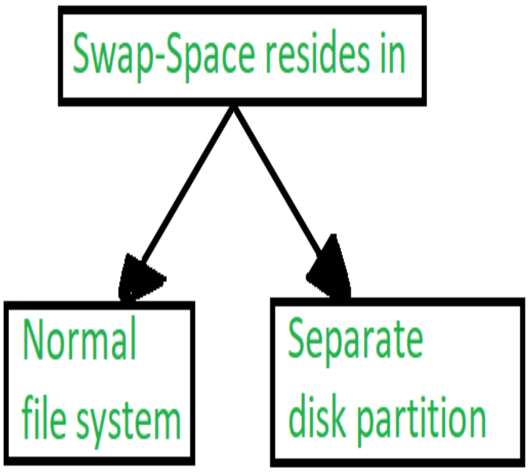
System	Swap-Space
1. Solaris	Equal amount of physical memory
2. Linux	Double the amount of physical memory

Figure – Different systems using amount of swap-space

Explanation of above table :

- Solaris, setting swap space equal to the amount by which virtual memory exceeds page-able physical memory.
- In the past Linux has suggested setting swap space to double the amount of physical memory.
- Today, this limitation is gone, and most Linux systems use considerably less swap space.
- Including Linux, some operating systems; allow the use of multiple swap spaces, including both files and dedicated swap partitions.
- The swap spaces are placed on the disk so the load which is on the I/O by the paging and swapping will spread over the system's bandwidth.

Swap-Space Location :



Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:

Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(467-624).

Course Teacher

Verified by HOD

LECTURE HANDOUTS **L45**

CSE **II/IV**

Course Name with Code	**: Operating Systems & 16CSD05**
Course Faculty	**: Mr.S.Pragadeeswaran**
Unit	**: V File System and Storage Management**

Date of Lecture:

Topic of Lecture: Linux and Android operating Systems

Introduction : (Maximum 5 sentences)

- Linux OS, precisely speaking Linux kernel is the most popular OS while Android is a framework built on top of Linux kernel.
- So every Android device is running Linux kernel as well but every Linux device doesn't have Android. We can think of Linux kernel as foundation on which Android is built

Prerequisite knowledge for Complete understanding and learning of Topic: (Max. Four important topics)

- Operating Systems
- Hardware
- Software
- Process

Detailed content of the Lecture:

- Many folks consider that both of these as operating systems. Linux OS is for Desktop & Server while Android OS is for Mobiles. Is that completely correct?
- Linux OS, precisely speaking Linux kernel is the most popular OS while Android is a framework built on top of Linux kernel. So every Android device is running Linux kernel as well but every Linux device doesn’t have Android. We can think of Linux kernel as foundation on which Android is built. Also Android isn’t limited to Mobiles only. Android runs on other devices such as TV, Camera, Watch and even Cars!
- Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets.
- Android is developed by a consortium of developers known as the Open Handset Alliance and commercially sponsored by Google. It was unveiled in November 2007, with the first commercial Android device launched in September 2008.
- It is free and open source software; its source code is known as Android Open Source Project (AOSP), which is primarily licensed under the Apache License.
- However most Android devices ship with additional proprietary software pre-installed, most notably Google Mobile Services (GMS) which includes core apps such as Google Chrome, the digital distribution platform Google Play and associated Google Play Services development platform.

- About 70 percent of Android smartphones run Google's ecosystem; competing Android ecosystems and forks include Fire OS (developed by Amazon) or LineageOS.
- However the "Android" name and logo are trademarks of Google which impose standards to restrict "uncertified" devices outside their ecosystem to use Android branding.

✓ Distributions include the Linux kernel and supporting system software and libraries, many of which are provided by the GNU Project.

✓ Many Linux distributions use the word "Linux" in their name, but the Free Software Foundation uses the name GNU/Linux to emphasize the importance of GNU software, causing some controversy.

✓ Popular Linux distributions include Debian, Fedora, and Ubuntu. Commercial distributions include Red Hat Enterprise Linux and SUSE Linux Enterprise Server. Desktop Linux distributions include a windowing system such as X11 or Wayland, and a desktop environment such as GNOME or KDE Plasma.

✓ Distributions intended for servers may omit graphics altogether, or include a solution stack such as LAMP. Because Linux is freely redistributable, anyone may create a distribution for any purpose.

Video Content / Details of website for further learning (if any):
http://www.cs.nthu.edu.tw/~ychung/slides/CSC3150/Abraham-Silberschatz-Operating-System-Concepts---9th2012.12

Important Books/Journals for further learning including the page nos.:
Book:

Abraham Silberschatz,Peter Bear Galvin and Greg Gagne, ," Operating system concepts",2015, John Wiley & Sons (ASIA) ,9th Edition,Page no(467-624).

Course Teacher

Verified by HOD