Operating System and System Programming

(SEng2034)

Overview of Operating Systems

1.1 What is Operating Systems

- An Operating System is a software that acts as an interface between the computer user and the hardware .
- There are various resources present in the system and managing such resources manually is a difficult task. So, we make the use of the Operating System to manage all the resources present in the system.
- It is a kind of **resource manager** that manages both hardware and software resources of a system.
- Apart from resource management, the other thing that the Operating System does is, it provides a platform where other application programs can be installed.
- It also allows you to communicate with the computer without knowing how to speak the computer's language.
- Without an operating system, a computer is useless. In other words, it’s the backbone of your computer.

,,, Cont'd

- An operating system can be installed on various hardware platforms, from desktop computers to mobile devices.
- To run other programs, every computer must have at least one operating system installed.
- Operating systems usually come pre-loaded on any computer you buy.
- Most people use the operating system that comes with their computer, but it is possible to upgrade or even change operating systems.
- The three most common operating systems for personal computers are **Microsoft Windows, macOS, and Linux.**
- Modern operating systems exist today has graphical user interface, or GUI.
- A GUI lets you use your mouse to click icons, buttons, and menus, and everything is clearly displayed on the screen using a combination of graphics and texts.

1.2 The Purpose of Operating System

- Operating System provides a number of functionalities. They are:
- **Processor Management:** Operating system responsible for managing processes running on your computer. This includes starting and stopping programs, allocating resources, and managing memory usage.
- **Memory Management: -** For the execution of a program, the whole program is put into the main memory and the program is executed and after the execution of the program , the memory is freed and that memory can be used for other program.
- So, it is the duty of the Operating System to manage the memory by allocating and de allocating the memory for the process.
- **I/O Device Management:** There are various I/O devices that are present in a system. Various processes require access to these resources and the process should not directly access these devices.
- So, it is the duty of the Operating System to allow the use of I/O devices by the various process that are requiring these resources.

,,, Cont'd

- **File Management.** Operating systems are responsible for managing the files and folders on a computer.
- This includes creating, opening, closing, and deleting files.
- The operating system is also responsible for organizing the files on the disk.
- **Provides Security:** The operating system provides various techniques which assure the integrity and confidentiality of user data.
- It prevents unauthorized access to programs and data by means of password and other similar techniques.
- **Error Detection:** The operating system checks the system for any external threat or malicious software activity.
- It also checks the hardware for any type of damage. This process displays several alerts to the user so that the appropriate action can be taken against any damage caused to the system.

1.3 Types of Operating system

- There are several types of operating systems. The most widely used are:-
- **Windows**: is the popular Microsoft brand preferred by most personal users.
- **Unix/Linux**
- **UNIX**: It is well known for its stability and often used more as a server than a workstation.
- **Linux**: was designed based on the UNIX system, with the source code being a part of GNU open-source project.
- **Macintosh:** Recent versions OS, including the Mac OS X, follow the secure architecture of UNIX.
- It is efficient and easy to use, but can only function on Apple branded hardware.

- **Type of Operating system based on the:-**
- Number of users they can serve
- Number of programs the operating system handle/ support on a time, and
- The time required to respond the given request/input
- **Single-user, single tasking :**
- This kind of OS allows only one user to access the system at a time. i.e. It runs only one program at a time.
- Examples: MS-DOS (Micro Soft Disk Operating System) produced by Microsoft.
- **Single-user, multi-tasking:**
- In this operating system, all resources are dedicated to only one user. But, the user can use more than one program
- It overcomes the limitation single tasking operating system.
- Single user have a permission to run multiple programs at the same time.
- Example Microsoft's Windows (98, 2000, XP, Vista etc) and Apple's MacOS X.
- For example we can play MP3 music while edit documents in Microsoft Word.

,,, Cont'd

- **Multi-user , Multi tasking:**
- It is mostly used on the networked environment.
- More than one user can utilized the resources available on the computer like hardware's and soft wares.
- Example: Linux, UNIX, etc
- **Real-time operating system (RTOS):**
- This kind of Os is used **when time requirements are very strict** like missile system, robots, scientific instruments and other industrial systems.
- This Os has very small response time.
- i.e. The time interval required to process and response the input is very small.

,,, Cont'd

- **Multi-programming operating system**
- In multi-programming system there are one or more program loaded in main memory which are ready to execute.
- Only one program at a time get the CPU for executing the instruction,while the remaining program waits their turn.
- If the current running process is waiting for I/O task, the operating system interrupt that process and , give the control to one of the other in main memory program that are ready to execute.
- The main goal of multi-programming is to maximize CPU utilization.

Chapter Two

Process Management

What is a Process?

- When we execute a program, the program is loaded from secondary memory to main memory.
- So, in main memory, program is in execution and becomes a process.
- Thus, we can say that when program in execution, it is called a process. or it is a running program.
- An entity that is executed by the processor.
- We emphasize that a program by itself is not a process,
- A program is a **passive entity**, such as a file containing a list of instructions stored on disk (often called an executable file), whereas a process is an **active entity**.

Cont…

- A program becomes a process when an **executable file** is loaded into memory.
- There can be one or more processes associated with the same program.
- For example, when we write a program in C or C++ and **compile** it, the compiler creates **binary code**.
- The **original code** and **binary code** are both programs. When we actually **run the binary code, it becomes a process**.
- Two common techniques for loading executable files are:
- Double-clicking an icon representing the executable file and
- Entering the name of the executable file on the command line.

Cont…

There are two types of process.

1**. System process**: Processes which are created by operating system and run inside the operating system is called system processes.

2. **User process:** Processes which are created by user and interacts with the operating system are called user process.

- Web browser, torrent is example of user processes because these are installed by user.

Process Structure in Memory

- **Process in memory** is divided into **four sections** for efficient working: -
- **Stack:** a process stores temporary data onto the stack. Temporary data such as local variables, parameters of a function, a return address etc.
- **Heap:** Heap is a memory allocated dynamically to the process during execution.
- **Data:** the data section stores the global variable.
- **Text:** the text section includes the program code that has to be executed.

Process States

- The process, from its creation to completion, passes through various states. The minimum number of states is five.
- **New State:** It is the initial stage of each process. At this stage the process is being created.
- A process is said to be in new state when a program present in the secondary memory is initiated for execution.
- **Ready State:** The process is ready to be assigned to the processor.
- Whenever a process is created, it directly enters in the ready state, in which, it waits for the CPU to be assigned.

Cont…

- The OS picks the new processes from the secondary memory and put all of them in the main memory.
- The processes which are ready for the execution and reside in the main memory are called ready state processes. There can be many processes present in the ready state.
- **Running State:** When the processor is consistently executing the instructions. The process is in a running state or Instructions are being executed.
- One of the processes from the ready state will be chosen by the OS depending upon the scheduling algorithm.

Cont…

- Hence, if we have only one CPU in our system, the number of running processes for a particular time will always be one.
- If we have n processors in the system then we can have n processes running simultaneously.
- **Blocked (or waiting)** State: the process waiting for some event to happen such that as an I/O completion or reception of a signal.
- From the Running state, a process can make the transition to the block or wait state depending upon the scheduling algorithm or the intrinsic behavior of the process.

Cont…

- **Terminated state**: When a process finishes its execution, it comes in the termination state.
- All the context of the process (Process Control Block) will also be deleted the process will be terminated by the Operating system.
- Only one process can be running on any processor at any instant, although many process may be ready and waiting.

Cont…

new
admitted
interrupt
exit
terminated
ready
running
scheduler dispatch
I/O or event completion
I/O or event wait
waiting

Process Control Block (PCB)

- There is a Process Control Block for each process, enclosing all the information about the process.
- It is also known as the **task control block**. It is a data structure, which contains the following:
- **Process ID**: When a process is created, a unique id is assigned to the process which is used for unique identification of the process in the system.
- **Process State:** The Process, from its creation to the completion, goes through various states which are new, ready, running and waiting.

Cont…

- **CPU Register:** The information in the **program counter** alone is not enough to resume the process after serving an interrupt.
- Thus, we need CPU registers to hold the information. The processor uses this information along with the program counter to resume its execution.
- **CPU Scheduling Information**: This information specifies the priority of the process. It also specifies the pointer pointing to the scheduling queues.

Cont…

- **Memory-management Information:** This information specifies the value of base and limit registers. It also includes information about the page table.
- **Accounting Information**: This information specifies the number of resources used by the system. Such as CPU time of a process, time limits and process numbers.
- **I/O Status: -** This information specifies the list of I/O devices that are allocated to a process for its execution. / A process may require accepting data from the input devices before writing the data to the output devices.
- Thus, the information contained by each PCB varies from process to process.

Threads

- **Thread** is a smallest unit of process. It always exist within a process.
- The modern operating system allows a process to have multiple threads. That allows a process to perform multiple tasks at the same time.
- A thread is also known as lightweight process. The idea is to achieve parallelism by dividing a process into multiple threads. For example, in a browser, multiple tabs can be different threads.
- MS Word uses multiple threads: one thread to format the text, another thread to process inputs. So, a single application may need to perform several similar tasks parallelly.
- Thread is used to improve the application performance through parallelism.
- Single threaded or multi-threaded process can be exist.

What is Process Scheduling?

- The act of determining which process is in the **ready state**, and should be moved to the **running state** is known as **Process Scheduling**.
- The aim of the process scheduling system is to **keep the CPU busy all the time** and to deliver **minimum response time** for all programs.
- For achieving this, the **scheduler** must apply appropriate rules for **swapping processes** IN and OUT of CPU.
- What are **Scheduling Queues**?
- All processes, upon entering into the system, are stored in the **Job Queue.**
- Processes in the Ready state are placed in the **Ready Queue.**
- Processes waiting for a device to become available are placed in **Device Queues.**

Context Switching

- Most modern computer systems allow more than one process to be executed simultaneously.

  This is called **Multitasking systems**.

- **Context switching** means switching the CPU to different processes.
- It requires saving of the state of the current process into the PCB and load the saved PCB of the previous or new process.
- **How does CPU switch between different processes?**
- Dispatcher (also called Short Term Scheduler/ CPU Scheduler): The inner-most portion of the OS that runs processes without interference:
- 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, decide which process to execute next.

Cont….

- **CPU Scheduler supports and facilitates the following activities:**
  - Run process for a while
  - Save its state
  - Load state of another process
  - Run the new process and after some time it reload the suspended /previous process.
- When more than one process loaded , the operating system must decide which one should first run.
- The part of the operating system concerned with this decision is called **the scheduler**, and algorithm it uses is called the **scheduling algorithm**.

Cont….

- In **multitasking and uniprocessor system** scheduling is needed because more than one process is in ready and waiting state.
- A certain scheduling algorithm is used to get all the processes to run correctly.
- The processes should be in such a manner that **no processes** must be made to **wait for a long time.**

Types of CPU Scheduling:

- The Scheduling algorithms can be divided into **two** categories

1. Non-preemptive Scheduling:

- A scheduling discipline is non-preemptive , **if once the CPU has been given to the process , the CPU cannot be taken away from that process** ,until the assigned process completes its execution.
- In Non-Preemptive scheduling, a running process can’t be interrupted by any other process.
- Any other process which wants get the CPU has to wait, until the existing process finishes its CPU cycle.
- Non-preemptive scheduling is **rigid**.

Cont…

- Following are some characteristics of **non-preemptive scheduling**.
- In non-preemptive system, **short jobs** are made to wait by **longer jobs**.
- In non-preemptive system, **response times** are more predictable, **maximum**, because incoming high priority jobs cannot displace waiting jobs.
- In non-preemptive scheduling, a scheduler execute other jobs in the following two situations.
- When a process switches from **running state to the waiting state**.
- When a process **terminates.**

Preemptive Scheduling:

- The strategy of making running process temporarily suspended is called **Preemptive Scheduling** and it is contrast to the "**run to completion" method**.
- In preemptive scheduling, a running process can be interrupted, if a high priority process request to CPU.
- Current process switches from **running queue to ready queue**.
- If the process is of **low priority**, it has to **starve for CPU resources**, because in preemptive scheduling high priority process served first.
- The preemptive scheduling is flexible as it allows any high priority process.
- CPU Utilization is more compared to non-preemptive scheduling.

Objectives of CPU scheduling

- **CPU utilization:** To make the CPU busy most of the time. CPU should not be idle.
- **Throughput**: Number of processes that complete their execution per time unit.
- **Turnaround Time**: The Amount of time spent by the process from its arrival to its completion.
- **Waiting time:** Time spent in the ready queue for getting the CPU.
- **Completion/Finish Time (CT):** The time at which the process completes its execution.
- **Arrival Time (AT):** The time at which a process enters the ready queue.
- **Burst /Execution Time (BT):** The amount of time required by a Process to complete execution.
- **Response/Start Time:-** The time at which the process gets the CPU for the first time.
- **TAT = CT – AT**
- **WT = TAT – BT**

CPU Scheduling Algorithms

- CPU Scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU.
- **The following are some scheduling algorithms:**
  - First Come First Served (FCFS) Scheduling.
  - Shortest Job First (SJF) Scheduling.
  - Round Robin Scheduling.
  - Shortest Remaining Time (SRT) Scheduling.
  - Priority Scheduling.

First Come First Served (FCFS) Scheduling

- First-Come-First-Served algorithm is the simplest scheduling algorithm.
- Processes are **dispatched** according to their arrival time on the ready queue.
- Being a non-preemptive discipline, once a process gets a CPU, it runs to completion.
- The FCFS scheduling is fair in the formal sense or human sense of fairness,
- But ,it is unfair in the sense that **long jobs make short jobs wait a long period**.
- One of the major drawbacks of this scheme is that the **average waiting time** is often quite **long.**

Cont…

- **Example 1:** Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Process	Burst Time	Arrival Time
P1	2	0
P2	1	0
P3	6	0
P4	5	0

- Based on the given above **calculate Average waiting time, Turnaround time for each process ?**

Cont…

Gantt chart

- **Explanation:-** Arrival time of all processes is 0.
- So, all processes will be present in ready queue (in the order P1, P2, P3, and P4).
- At Arrival time = 0, P1 process is picked from the ready queue.
- So, P1 will be executed first. It will take 2 unit time for complete execution.
- After completion of P1 process, P2 will be executed. It takes 1 unit time for complete execution. The completion time for P2 will be 3 (= 2+1).
- Now, P3 will be executed. It will take 6 unit time. So, completion time of P3 is 9 (= 3 + 6).
- Now, P4 will be executed for 5 unit time. So, completion time for P4 is 14 (= 9 + 5).

Cont…

Now, let's calculate turnaround time and waiting time.

Turnaround time = (Completion/Finish time – Arrival time)

P1 = 2-0 = 2,

P2 = 3-0 = 3,

P3 = 9-0 = 9,

P4 = 14-0 = 14

Waiting time = (Turnaround time – Burst time) or Start time/Response time – Arrival time

P1 = 2-2 = 0,

P2 = 3-1 = 2,

P3 = 9-6 = 3,

P4 = 14 – 5 = 9

Average waiting time = (Total waiting time) / (number of processes) = (0+2+3+9) / 4 = 14/4 = 3.25 ms

Convoy Effect

- When a process with high **burst time arrives first**, **it takes lot of time to finish** due to which all the other processes will be **blocked**. This is called **convoy effect** in operating system.
- It is the disadvantage of FCFS scheduling algorithm.
- A real-life example of convoy effect is **a big truck passing through the road, because of this small vehicle following it gets blocked.**

Example to Demonstrate Convoy Effect

- Let's assume we have **three processes** P1, P2 and P3 with burst time and arrival time as shown below.

PID	Burst Time	Arrival Time
P1	40	0
P2	2	1
P3	1	1

Cont…

- ✓ Now, calculate average waiting time for above processes

Gantt chart:

- ✓ Here, P1 process is executed first. All other process can get access to CPU only after its completion.
- ✓ Now, **P1 process has high burst time.**
- ✓ So, **Processes P2 and P3 have to wait for longer period to get access to CPU**.
- ✓ So, P2, P3 process suffers from **starvation**.
- ✓ **So, Average waiting time=?**

Shortest-Job-First (SJF) Scheduling

- Out of all available process it selects the process with **the smallest burst time** to execute next.
- In other words, when CPU is available, it is assigned to the process that has smallest next CPU burst.
- The SJF algorithm favors for short jobs (or processes) at the expense of longer ones.
- SJF scheduling algorithm gives the **minimum average time** for a given set of processes, it is probably optimal.

Process	Arrival Time	Burst Time
P_1	0	6
P_2	0	8
P_3	0	7
P_4	0	3

Cont…

TAT =CT-AT

P1=9-0=9

P2=24-0=24

P3=16-0=16

P4=3-0=3

Average TAT =(9+24+16+3)/4=13ms

WT =TAT-BT

P1=9-6=3

P2=24-8=16

P3=16-7=9

P4= 3-3=0

Average WT=(3+16+9+0)/4=7ms

Example 2: Cont…

✓ Let's assume we have **five** processes **P1, P2 , P3,P4, and P5 ,** with burst time and arrival time as shown below. **Then calculate the Complete/Finish time, Turnaround time and Waiting Time for each process?**

Process	Burst Time	Arrival Time	CT/FT	TAT	WT
P1	10	0			
P2	5	0			
P3	3	2			
P4	20	5			
P5	2	10			

Gantt chart

Round Robin Scheduling

- ✓ Round robin uses **time slice** (fixed time period) for execution of the process, called time **quantum**.
- ✓ Round robin scheduling algorithm is one of the important scheduling algorithms in job scheduling. It is the preemptive scheduling algorithm
- ✓ This scheduling algorithm is used in time sharing system.
- ✓ Once a process is executed for a given time period, the process is preempted and the next process execution starts for the given time period.
- ✓ Round robin scheduling uses context switching to save states of preempted process.

- Example 1: Assume there are 5 processes with burst time given below.
- **Time quantum: 2**
- Assume that all process arrives at 0.
- **Calculate average TAT and WT ?**

Process	Burst Time
P1	**6**
P2	**5**
P3	**2**
P4	**3**
P5	**7**

Gantt Chart:

P1	P2	P3	P4	P5	P1	P2	P4	P5	P1	P2	P5

0 2 4 6 8 10 12 14 15 17 19 20 23

Cont…

- **Turn Around Time = CT-AT**

P1 = 19 – 0 = 19

P2 = 20 – 0 = 20

P3 = 6 – 0 = 6

P4 = 15 – 0 = 15

P5 = 23 – 0 = 23

- **Average TAT = (19+20+6+15+23) / 5 = 64.6 ms**

Waiting time = TAT– BT

P1 = 19 – 6 = 13

P2 = 20 – 5 = 15

P3 = 6 – 2 = 4

P4 = 15 – 3 = 12

P5 = 23 – 7 = 16

Average WT = (13+15+4+12+16) / 5 = 12 ms

- **Example 2**
- **Assume here are 6 processes burst time and arrival time as shown below.**
- **Time quantum = 2**

Process	Arrival Time	Burst Time
P1	0	4
P2	1	5
P3	2	2
P4	3	1
P5	4	6
P6	6	3

- **Now, we will calculate average waiting time, completion time, turnaround time for each processes' execution.**

Grant Chart

P1	P2	P3	P1	P4	P5	P2	P6	P5	P2	P6	P5

0 2 4 6 8 9 11 13 15 17 18 19 21

- **Turn Around time:**

P1 = 8 – 0 = 8

P2 = 18 -1 = 17

P3 = 6 – 2 = 4

P4 = 9 – 3 = 6

P5 = 21 – 4 =17

P6 = 19 – 6 = 13

- **Waiting time:**

P1 = 8 – 4 = 4,

P2 = 17 – 5 = 12

P3 = 4 – 2 = 2

P4 = 6 – 1 = 5

P5 = 17 – 6 = 11

P6= 13 – 3 = 10

Cont…

Note: In Round Robin scheduling algorithm, as the **time quantum** decreases **context switching** increases.

- The **increase in time quantum** value results **starvation** which may put many processes on hold.
- So, time quantum should neither be large nor be small.
- If time quantum becomes **infinity**, Round Robin scheduling algorithm gradually become **FCFS scheduling algorithm.**
- **Advantages of Round Robin Scheduling**
- In RR all the processes have **equal priority** because of fixed time quantum.
- **Starvation** will never occur, because each process in every RR cycle will be schedule for a **fixed time slice or time quantum.**
- **Disadvantages of Round Robin Scheduling**
- In RR, **throughput** depends on the **time quantum**. if time quantum is increased, the throughput will be decreased.

Shortest Remaining Time First Scheduling Algorithm

- As the name suggests, it selects the process for execution which has the **smallest amount of time remaining until completion.**
- Shortest Remaining Time First Scheduling is a **preempted version of SJF (Shortest Job First).**
- In SJF, once a process begins execution, it runs till completion.
- In SRTF a running process may be **preempted** by a user process with a smallest estimated run time.
- Once all the process available in the **ready queue no preemption will be done** and the algorithm will work **as non-preemptive SJF Scheduling**.

Cont…

Example: Suppose we have the following 3 processes with process ID's P1, P2, and P3 and they arrive into the CPU in the following manner:

Process	Arrival Time	Burst Time
P1	0	8
P2	1	2
P3	4	3

Gant Chart:

0 P1	1 P2	2 P2	3 P1	4 P3	5 P3	6 P3	7 P1	8 P1	9 P1	10 P1	11 P1	12 P1 13

Cont…

- Total Turnaround Time = 13 + 2 + 3= 18 milliseconds
- Average Turnaround Time= Total Turnaround Time / Total No. of Processes

  = **18 / 3= 6 milliseconds**
- Total Waiting Time = 5 + 0 + 0= 5 milliseconds
- Average Waiting Time = Total Waiting Time / Total No. of Processes

  = 5 / 3= 1.67 milliseconds

Exercise : Cont…

✓ Let's assume we have **five** processes **P1, P2 ,P3,P4 and P5 ,**with burst time and arrival time as shown below. **Then calculate Complete/Finish time, Turnaround time and Waiting Time for each process?**

Process	Arrival Time	Brust Time	CT	TAT	WT
P1	1	2			
P2	2	6			
P3	0	3			
P4	4	1			
P5	2	4			

Priority Scheduling Algorithm

- In Priority Scheduling, priority is assigned to each process and the CPU is allocated to the process with the highest priorities.
- In some systems, lower priority number means higher priority (Example P1 with priority 1 and P2 with priority 4, in this case P1 will have higher priority).
- However, in some system higher priority number means higher priority. If two processes have equal priorities those process are scheduled in First Come First Serve order.
- It can be either preemptive or non-preemptive algorithm.
- **Example 1:-** Consider the following set of process, Assumed to have arrived at time 0 in the order P1, P2, P3, P4, and P5 with the length of CPU burst given in milliseconds (**Non-preemptive**).

Cont…

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Gantt chart

Calculate Average TAT and Average WT?

Cont…

Example 2: Consider the set of 5 processes whose arrival time and burst time are given below

Process Id	Arrival time	Burst time	Priority
P1	0	10	5
P2	0	5	2
P3	2	3	1
P4	5	20	4
P5	10	2	3

Gantt chart

Cont…

Process Id	Exit time	Turn Around time	Waiting time
P1	40	40 – 0 = 40	40 – 10 = 30
P2	5	5 – 0 = 5	5 – 5 = 0
P3	8	8 – 2 = 6	6 – 3= 3
P4	28	28 – 5 = 23	23 – 20 = 3
P5	30	30 – 10 = 20	20 – 2 = 18

Average Turn Around time = (40 + 5 + 6 + 23 + 20) / 5 = 114 / 5 = **22.8 unit**

Average waiting time = (30 + 0 + 3 + 3+ 18) / 5 = 54 / 5 = **10.8 unit**

Example 2:- Consider the following set of processes, Assumed all processes arrived at different times in the order P0, P1, P2, and P3, with the length of CPU burst given in milliseconds (**preemptive**). **Then calculate the average TAT and average WT**.

Process Id	Arrival time	Burst time	Priority
P0	0	10	5
P1	1	6	4
P2	3	2	2
P3	5	4	0

Interprocess Communication (IPC)

- **IPC** is the way in which different process communicate with each other, or a mechanism that involves communication of one process with other process.
- Process executing concurrently in the operating system may be either **independent or cooperative process.**
- **Independent Process: -** They cannot affect or be affected by other process executing in the system , and this process does not share data with other process.
- **Cooperating Process: -** They can affect or be affected by other process executing in the system. This is because cooperating processes share data with the other processes.

Cont…

- Some advantages of the cooperating system are as follows:

1. Information Sharing

- Cooperating processes can be used to share information between various processes. It could involve having access to the same files.

2. Modularity

- Modularity refers to the division of complex tasks into smaller subtasks. Different cooperating processes can complete these smaller subtasks.
- As a result, the required tasks are completed more quickly and efficiently.

Cont…

- Some advantages of the cooperating system are as follows:

1. Information Sharing

- Cooperating processes can be used to share information between various processes. It could involve having access to the same files.

2. Modularity

- Modularity refers to the division of complex tasks into smaller subtasks. Different cooperating processes can complete these smaller subtasks.
- As a result, the required tasks are completed more quickly and efficiently.

Cont…

- **3. Computation Speedup**
- If we want a particular task to run faster, we must break into subtasks, each of which will be **executing in parallel** with others.
- It **improves computation speed** by allowing the task to be accomplished faster. But, it is only possible if the system contains **several processing elements**.
- **4. Convenience**
- There are multiple tasks that a user requires to perform, such as printing, compiling, editing, etc.
- It is more convenient if these activities may be managed through cooperating process.

Cont…

- **3. Computation Speedup**
- If we want a particular task to run faster, we must break into subtasks, each of which will be **executing in parallel** with others.
- It **improves computation speed** by allowing the task to be accomplished faster. But, it is only possible if the system contains **several processing elements**.
- **4. Convenience**
- There are multiple tasks that a user requires to perform, such as printing, compiling, editing, etc.
- It is more convenient if these activities may be managed through cooperating process.

Process synchronization

- The task of coordinating the execution of a process in such a way that **no two process can have access to the same shared data and resources.**
- It is especially needed, in **a multiprocessing system** when multiple process are running together, and more than one process are trying to gain access 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.

What is Race Condition

- **Race Condition** occurs when more than one process tries to access and **modify the same shared data or resources.**
- Because many processes try to modify the shared data or resources there are huge chances of a process getting the wrong result or data this situation is called **Race Condition.**
- Now the question arises that how to handle a race condition.
- We can tackle this problem by implementing logic in the **critical section**.
- The part of the process, where **the code for accessing the shared resources is written,** that part or section is the **critical section (CS**) of that process.

What is Deadlock

- Every process needs some resources to complete its execution such as central processing unit (CPU), file storage, input/output devices, etc.
- Once the execution is finished, the process **releases the resource** it was holding.
- **Deadlock** is defined as a situation where **set of processes are blocked** ,because each process holding a resource and waiting to acquire a resource held by another process.
- In this situation, none of the process gets executed, since the resource it needs, is held by some other process ,which is also waiting for some other resource to be released.

Cont…

Resource 1
holds
needs
Process 1
Process 2
needs
holds
Resource 2

Cont…

- In the above figure, there are two processes and two resources. Process 1 holds "Resource 1" and needs "Resource 2" while Process 2 holds "Resource 2" and requires "Resource 1".
- This creates a situation of deadlock, because none of the two processes can be executed.
- Since the resources are non-shareable, they can only be used by one process at a time (**Mutual Exclusion**).

Necessary Conditions for Deadlock

- A deadlock situation can arise if the following 4 conditions hold simultaneously in a system.
- **Mutual Exclusion**: A process should work in mutual exclusive way. **It means a resource is non-sharable**. At a time only one process can use the resource.
- **Hold and Wait**: A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.
- **No Preemption**: Once a process is holding a resource, then that resource cannot be taken away from the process until the process voluntarily releases it.
- **Circular Wait:** Assume that there are four processes called P1, P2, P3, and P4. P1 is waiting for a resource held by P2. P2 is waiting for a resource held by P3. P3 is waiting for a resource held by P4 and P4 is waiting for a process held by P1. We call this kind of situation a circular wait.

Deadlock Prevention

- Deadlock happens only when Mutual Exclusion, hold and wait, No preemption and circular wait holds simultaneously.
- If it is possible to violate one of the four conditions at any time then the deadlock can never occur in the system.

Reading Assignments

1. Discuss more about critical section problem and methods for solving critical section problem in detail.
2. Discuss more about the difference between deadlock and starvation and different way of solving deadlocks and Starvation problem.

Thank You!!!

Chapter Three

Memory Management

INTRODUCTION

- Memory is one of the most important resources of the computer system, that is used to store data and programs temporarily.
  - Part of the operating system that manages memory is called the **Memory Manager (MM).**
- The main functions of the Memory Manager (MM) are the following:
  - Keeping track of which part of memory are **in use** and which parts are **free**.
  - Allocating and deallocating memory to processes.
  - Manage **swapping** of process between memory and disk, when memory is not big enough to hold all the processes.

SWAPPING

Process P1
Process P2
P1 goes for I/O wait
Process P1
Swap out
Swap in
Process P3
P1 comes back after I/O
Process P3
Swap out
Swap in
Process P1
Process P3
Process P4
Process Pn

CONT'D

- A good memory manager has the following attributes :
  - It allows all processes to run.
  - Its memory space overhead, used for the management activity must be reasonable.
  - Its time overhead, time required for the management activity is reasonable.

Memory Management Schemes

- They can be broadly classified into two:

- **Mono-programming** – Only one process will be in the memory in addition to the operating system.

- **Multi-programming** – multiple programs are loaded in the main memory which are ready to execute.

- **Advantage**:- CPU utilization increased (CPU never idle).

- **Disadvantage**:- Less Response time.

Memory Management Schemes

- Multiprogramming, multitasking, and multi-threading are execute on a single CPU or multiple CPUs. But Multiprocessing only execute on multiple CPU's.

Memory Partitioning

- There are two types of Partitioning
  - Fixed Partitioning
  - Dynamic Partitioning
- Fixed Partitioning:- Partitioning is done before the processes comes to memory.
- Dynamic Partitioning :-Partitioning is done when processes request memory space.

Fixed Partitioning

- It is the simplest and oldest partitioning technique.
- It divides the memory into fixed number of equal or unequal sized partitions.
- The partitioning can be done by the OS or the operator, when the system is started.
- Variations of fixed partitioning
  - **Equal sized fixed partitions**
  - **Unequal sized fixed partitions**

Equal sized fixed partitions

- The memory is partitioned into equal-sized partitions.
- Any process whose size is less than or equal to the partition size can be loaded into any available partition.
- If all partitions are occupied with processes that are not ready to run, then one of these processes is **swapped-out** and a new process is loaded or, a suspended process is **swapped in** and loaded.

Cont…

- E.g Consider a main memory size of 30MB.

Equal Size Partitions

Problems of Equal sized fixed partitions

- A program may be too big to fit into a partition.
- Inefficient use of memory due to internal fragmentation.
- **Internal fragmentation** is the phenomenon, in which there is internal space wastage to a partition ,due to the fact that the process loaded is smaller than the partition.

Problems of Equal sized fixed partitions

- Suppose, Main memory has five partitions with each size 6MB and five incoming processes (P1 to P5) with sizes 2MB, 6MB, 3MB, 4MB, 6MB respectively.

Unequal Sized Fixed Partitioning

- Memory is partitioned into unequal-sized fixed partitions.
- E.g Consider a main memory size of 30MB.
- In this case there are two memory schemes:
  - Using Single input queues.
  - Using Multiple input queues.

Using Single Queue

- The smallest available partition that can hold the process is selected.
- Whenever a partition becomes free, the process closest to the front of the queue that fits in it could be loaded into the empty partition.
- It can cause internal fragmentation,.
- Advantage: Partitions are used all the time.
- Disadvantage: When best fit is searched, it will not give best service for small jobs.

Using multiple queues

- Each partition has its own queue.
- A process is assigned to the smallest partition which it will fit , and it can only be loaded into the partition it belongs.
- **Advantage**: Minimize internal fragmentation.
- **Disadvantage**: Some partitions may remain unused, even though processes are waiting in some other queues.

CONT..

Single
Input
Queue
Partition - 4
Partition - 3
Partition - 2
Partition - 1
Operating
System

Multiple
Input
Queues
Partition - 4
Partition - 3
Partition - 2
Partition - 1
Operating
System

Dynamic Partitioning

- Partitions are created dynamically during the allocation of memory.
- The size and number of partitions vary throughout of the operation of the system.
- Processes will be allocated exactly as much memory they require.
- Dynamic partitioning leads to a situation in which there are a lot of small useless holes in memory this phenomenon is called **External fragmentation**.

Solution to External Fragmentation:

- **Compaction** – Combining the multiple useless holes in to one big hole.
- With fixed partitions, the space given to process is may be larger than it asked, and hence the process can expand if it needs.
- In dynamic partitions ,a process is allocated exactly as much memory it requires, so, it creates an overhead of moving and swapping processes, whenever the process grows dynamically.
- It is expected that most processes will grow as they run, so it is a good idea to allocate a little extra memory whenever a process is loaded into the memory.

Cont'd

- If a process runs, out of the extra space allocated to it, either
  - It will have to be moved to a hole/partition with enough space or
  - Swapped out of memory until a large enough hole can be created or
  - Kill the process.

Example of Dynamic Partitioning and Compaction

Memory Allocation algorithms (Placement algorithms)

- When a new process is created or swapped in, the OS must decide which free bock to allocate.
- Placement algorithms are :-
    - **First-Fit**
    - **Next Fit**
    - **Best Fit**
    - **Worst Fit**

First-Fit

- It scans the memory from the beginning , and chooses the first available block that is large enough to hold the process.
- It is the simplest and fastest algorithm.
- Disadvantage: Memory loss may occur, as very large hole may be selected for small process.

Cont…

- Example: Process of 212K,417K,112K and 426K arrives in order. So how to allocate these process in these memory using First-Fit ?

- Here process of 426K will not get any partition for allocation.

Next- Fit

- Next fit is a modified version of first fit.
- It starts scanning the memory ,from the **location of the last placement**, and chooses the **next** available block that is large enough to hold the process.
- Advantage: Search time is smaller, because every time this algorithm will not start searching at the beginning.
- Disadvantage: It may cause the last section of memory to be quickly fragmented.

Cont…

- Example: Process of 212K,417K,112K and 426K arrives in order. So how to allocate these process in these memory using Next-Fit ?

- Here process of 426K will not get any partition for allocation.

Best-Fit

- Entire memory is searched.
- The smallest hole which is large enough to hold the process is selected for allocation.
- Advantage: Memory utilization is much better.
- Disadvantage: Search time is slower because it scans the whole list.

Cont…

- Example: Process of 212K,417K,112K and 426K arrives in order. So how to allocate these process in these memory using Best-Fit ?

- Here each and every process gets allocation.

WORST FIT

- Entire memory is searched here also.
- The largest available hole, which is largest enough to hold the process , is selected for allocation.
- Advantage : Reduces the rate of production of small gaps.
- Disadvantage: It is as slow as best fit.

Cont…

- Example: Process of 212K,417K,112K and 426K arrives in order. So how to allocate these process in these memory using Worst-Fit ?

- Here process of 426K will not get any partition for allocation.

Memory Usage Management

- There are two ways to keep track of memory usage:
  - Bitmap
  - Linked list

Bitmaps

- The memory is divided into **fixed size** allocation units.
- Each allocation unit is represented with a **bit** in the bit map. If the bit is 0, it means the allocation unit is free and if it is 1, it means it is occupied.
- **Example**: Look at the following portion of a memory. Let the size of the allocation units is 5kB as shown in the diagram.

- The corresponding bitmap for the above memory can be as follows:

11110011
11111001
11100000
...

Cont'd

- The size of the allocation unit is an important design issue.
- The smaller the allocation unit size, the larger the bitmap size, which will result to Memory and CPU overheads.
- Example:- Consider size of allocation unit=4bits(1/2bytes)
- **Size of bitmaps=Total main memory/size of allocation unit in bit + bit required for bitmap.**
- So, size of bitmap =1/5 of total main memory.
- The number of bit in the bitmap always equal to the number of allocation unit in main memory.

Cont'd

- Searching will take time.
- But, If we increase size allocation unit size of bitmap will be reduced, but, memory wastage (internal fragmentation) will occur.
- Because two process never take one allocation unit, but more than one allocation unit is allowed for one process depends on the size of the process.

Cont'd

- Assume the allocation unit is 5kb and the total memory size is 120kb ,So how to calculate size of bitmap.
- Size of bitmap =120x1024Bytes/5x1024Bytes=122,800Bytes/5120Bytes=122,800x8/5120x8
- =982,400/ 40960 +1= 982,400/40961 =24 bits= 0.003kb
- Percentage of memory used for bitmap= (0.003KB/120KB)X100=0.0025%.

Linked Lists

- A linked list of allocated and free memory segments is used.
- Each segment contains either a process or a hole between two processes, and contains a number of allocation units.
- Each entry in the list consists of
  - Segment type: P/H (1bit)
  - The address at which it starts
  - The length of the segment
  - A pointer to the next entry

Example: The memory in the bitmap example can be represented using linked list as follows:

Logical Address

- A logical address is an address that is generated by the CPU during program execution.
- The logical address is a virtual address as it does not exist physically, and therefore, it is also known as a **Virtual Address**.
- The set of all logical addresses generated by a program is referred to as **Logical Address Space.**
- The logical address is a virtual address viewed by the user. But, The user can't view the physical address directly.

Physical Address

- The physical address identifies the physical location of required data in memory.
- Generated by the **Memory management unit**.
- The user never directly deals with the physical address. but can access it by its corresponding logical address.
- The user program generates the logical address, and thinks ,it is running in it, but the program needs physical memory for its execution.
- Therefore, the logical address must be mapped to the physical address by **MMU** before they are used.

- The set of all physical addresses is referred to as a **Physical Address Space.**

Cont…

- For example, **J1** is a program written by a user. The size of the program is **100KB**.
- But , the program is loaded in the main memory at base address of **2100 KB**, this actual loaded address in the main memory is called **Physical address**.
- Here in this example from 1 to 100KB is the logical address space, Therefore:

Physical address space = Logical address space + Contents of the relocation register.

100 + 2100 =220KB

- So, from 2100 to 2200 KB is the physical address space.

Base and Limit Register

- **Base/Relocation register**:- The starting address of the program.
- **Limit register**:- the length or the size of the program/process.
- For every memory access the base address

  is added to the address. Then,
- The result is compared to the limit register.
- Only OS can modify the base and limit register

during execution..

RELOCATION AND PROTECTION

- Multiprogramming introduces two problems that must be solved.
- **Protection**
- **Relocation**

PROTECTION

- Programs in one process should not refer the **memory locations** of another process, for reading or writing purposes without permission.
- A user program should not to be address a memory area that is not in its partition.
- A user process must not access any portion of the OS.
  - Usually a program in one process can not branch to, an instruction in another process.
  - Without permission, a program in one process cannot access the data area of another process.

RELOCATION

- The location of a program in the main memory is unpredictable due to loading, swapping, and compaction.

STATIC RELOCATION

- In Relocation each and every program executes its own instruction.
- **Static relocation** converts a logical address/instruction address into a physical address ,before executing the program. Once the conversion completes the execution starts.
- a program is loaded at base address **16384** in the previous example, so, the constant **16384** is added to every **program address** during the load process.
- Whenever a program is loaded into memory, before starting execution, each and every address is converted into a physical address.
- Once the program starts its execution, MMU will not convert any address during run time.

DYNAMIC RELOCATION

- Step for showing how **dynamic relocation** works during the execution of a program.

i. Hardware adds **relocation register to virtual address** to get physical address.

ii. Hardware compares the physical address with the limit register address , and must be less than or equal to limit register.

iii. If the test fails, the processor takes an address trap, and ignores the physical address.

Virtual Memory

- A process may need more memory than the available main memory so, there is a need of additional option is called **virtual memory**.
- In the early days **overlays** were used to solve this problem.
- The programmer will divide the program into modules and make switching the modules in and out of memory as needed.
- **Drawback**: The programmer must know how much memory is available, and it wastes the programmer time.

Cont'd

- In virtual memory , the OS keeps those parts of the program **currently in use**, **in main memory** and the rest **on the disk**.
- Piece of programs are swapped between disk and memory as needed.
- If a process encounters, a logical address that is not in main memory, it generates an interrupt indicating a memory access fault.
- Then the OS puts the process in blocking state, until it brings the piece that contains the logical address that caused the access fault.

Cont'd

- There are two virtual memory implementation techniques:
  - Paging
  - Segmentation

PAGING

- The virtual address space is divided into equal-sized block called **pages.**
- The memory is divided into equal-sized units called **page frames.**
- Pages and page frames should have the same size.
- Transfer between disk and memory are always in units of **pages**.
- The operation of dividing a process into pages is done by the **compiler or MM**.
- To map each and each page into frames we need a special data structure called **page table**.

Page 1
Page 2
Page 3
Page 4
Page 5
1 -> 3
2 -> 5
3 -> 1
4 -> 2
5 -> 4
Page Table
Frame 1 Page 3
Frame 2 Page 4
Frame 3 Page 1
Frame 4 Page 5
Frame 5 Page 2
Address space
RAM

CONT…

- We have a computer generated 16-bit addresses, from 0 up to 44K. These are the virtual addresses.

CONT…

- Virtual Address = (4 bits page number, 12-bits offset)
- Virtual Address Space= 44K
- Maximum Number of pages = 11 pages for the logical address space

and 6 pages for physical address space.

Page Tables

- The OS maintains page table for mapping virtual addresses to physical addresses.
- Page tables are stored in main memory.
- The OS also maintains all **free-frames** list of all frames in main memory that are currently unoccupied.
- Each page table entry contains:
- Present-bit.
- Modified-bit,
- Frame number.
- Other bit.

Cont'd

- **Present-bit**: Whether the page is present in main memory or not.
- **Modified-bit**: whether the contents of the corresponding page have been altered since the page was last loaded into main memory.
- If it is not altered, then it is not necessary to write the page out when it is to replace the page in the frame it occupies.
- **Frame number**: Indicates the corresponding page frame number in memory if its present-bit is 1, otherwise it is null.
- **Other Bits:** Other control bits like protection and sharing bits.

Cont'd

- If an instruction in a process tries to access an address in unmapped page:
  - The MMU causes the CPU to generate an interrupt, called **page fault**
  - The OS puts the interrupted process in a blocking state and takes control.
  - The OS chooses a page (page frame) to remove from memory and updates .
  - The OS issues a disk I/O read request to fetch the page just referenced into the page frame just freed ,and the OS can dispatch another process to run while the disk I/O is performed.

Cont'd

- Once the desired page has been loaded into main memory, and I/O interrupt is issued, giving control back to the OS, which places the affected process back into a ready state.
- When the process is dispatched, the trapped instruction is restarted.
- Page fault:- the page referenced or required by CPU is not available in the main memory
- If the required page is not available in main memory we need to load the page into memory, but

  if no free frame in memory we need apply page replacement algorithm.

PAGE SIZE AND MAPPING SPEED

- **There are two issues:-**
- The smaller the page size, the smaller number of internal fragmentations which optimizes the use of main memory.
- The smaller the page size, the greater the number of pages and thus the larger the page table.
- Every virtual memory can cause two physical memory access:
  - One to fetch the appropriate page table entry.
  - The other to fetch the desired data.

Page Replacement Algorithms (PRA)

- **First In First Out PRA:-** Remove the oldest page.
- **Optimal PRA :** It says replace the page that will not be referenced/used soon.
  - **Drawback** – impossible to know whether the page will be referenced or not in the future.
- **Least Recently Used PRA**
  - Remove pages that have less used in the past.
  - It assumes that pages that have been heavily used in the last few instructions will probably be heavily used again in the future.

Segmentation

- The virtual space is divided into units called segments.
- Segments can have unequal and dynamic size.
- Each segment has its own virtual address space.
- Each segment consists of a linear sequence of addresses, form 0 to some maximum
- A process is loaded in partially (Only some of its segments).
- The OS maintains a segment table for each process and also a free-segment list.
- It simplifies the handling of growing data structures.
- When segmentation is used there is no internal fragmentation.

Segmentation

CONT…

- **Reading Assignment**
- Discuss on working set and thrashing concept, how thrashing occur and methods of handling thrashing.
- What is cache memory, and why and when CPU use cache memory ?.

THANK YOU

Chapter Four

Input/Output Management

INTRODUCTION

- I/O is important for the communication between the user and the computer.
- The following are functions of the operating system as I/O manager:
  - Control all the computer I/O devices.
  - Issue commands to I/O devices, catch interrupts and handle errors.
  - Provide an interface between the device and the rest of the system.

Principles of I/O hardware

- There are two types of I/O devices:
  - **Block devices**: work on data of fixed size blocks,(complete block of data is transferred) with each block having a unique address, hence, are randomly accessible.
  - Example: Disk
  - **Character devices**: Process one characters(byte) at a time without any block structure.
  - Example Printers , NIC, mouse, and keyboards.

Cont'd

- I/O devices have two major components:-
- **Electronic Component:-**device controller/circuit present inside the devices.
- **Mechanical Component:-**the moving parts/ the devices itself.
- A controller has an **interface** into which a cable that connects its device is plugged in.
- Each controller has a few **registers** for communication with the operating system.

- Besides registers, some devices have data **buffers** on which programs and the operating system can read/write data.
- For the CPU to communicate with device controllers, each device must have a unique ID.

PRINCIPLES OF I/O SOFTWARE

- **Device Independence**: It is possible to write programs that can read files on a floppy disk, on hard disk, or on a CD-ROM, without having to modify the program for each different device types.

- It is up to the operating system to take care of the problems caused by the fact that these devices are really different.

CONT…

- **Uniform Naming:** the name of a file or a device should simply be a string or an integer, and not be dependent on the device in any way.
- **Error handling:** errors should be handled as close to the hardware as possible (at the device controller level).

CONT…

- **Transfer:** There are two types of transfer modes – **Synchronous (Blocking)** and **Asynchronous (interrupt–driven)**.
- In the case of **synchronous** transfer, the program requesting I/O transfer/request will be suspended, until the transfer is completed.
- In the case of **Asynchronous** transfer, the CPU starts the transfer and goes off to do something, until the interrupt that shows the completion of the transfer arrives.

LAYERS OF I/O SOFTWARE

The following are the I/O software layers

- Interrupt handler (bottom)
- Device driver
- Device-independent I/O software
- User-space I/O software (top)

Interrupt Handler

- Interrupts should be hidden.
- The best way to hide them is to have every process starting an I/O operation block until the I/O has completed and the interrupt occurs.
- When the interrupt happens, the interrupt procedure does whatever it has to do in order to unblock the process that started it.

DEVICE DRIVER

- All device dependent code goes in the device driver.
- Each device driver handles one device type, or at most, one class of closely related devices.
- The device driver issues the commands and check that they are carried out properly.
- The job of a device driver is to accept requests from the **device-independent software** above it and sees to it that the request is executed.

DEVICE INDEPENDENT I/O SOFTWARE

Functions of the device-independent I/O software

1. **Uniform interfacing for device drivers** – Perform I/O function common to all drives.
2. **Device Naming** – Responsible for mapping symbolic devices names onto the proper drives.
3. **Device protection** – Prevent users from accessing devices that they are not entitled to access.
4. **Buffering** – a user process write half a block, the OS will normally keep the data in buffer until the rest of the data are written. Keyboard inputs that arrive before it is needed also require buffering.

Cont'd

5. **Storage allocation on block devices:** when a file is created and filled with data, new disk blocks have to be allocated to the file. To perform this allocation, the OS needs a **list of free blocks** and used some algorithms for allocation

6. **Allocating and releasing dedicated devices**: It is up to the OS to examine requests for devices usage and accept or reject them.

7. **Error reporting:** Most errors are device dependent. After the driver tries to read the block a certain number of times, it gives up and informs the device-independent software. It then reports to the caller.

USER SPACE I/O SOFTWARE

- A small portion of the I/O software is outside the OS.
- System calls, including the I/O system calls, are normally made by library procedures.
- Formatting of input and output is done by library procedures.
- The following figure shows the I/O system layers and the main functions of each layer.

User Processes	Make I/O call; format I/O; Spooling
Device-independent software	Naming, protection, blocking, buffering, allocation
Device Drivers	Setup devices registers; check status
Interrupt Handlers	Wakeup driver when I/O completed
Hardware	Perform I/O operation

STRUCTURE OF HARD DISK DRIVE

- A **platter** is a circular magnetic plate that is used for storing data in a hard disk.
- It is often made of aluminum or glass substrate.
- A hard disk drive contains several platters.
- The platters rotate when the hard disk is performing read/write operations;
- the rotations per minute depending on the hard disk model.
- Each platter has two working surfaces.

Cont…

- These surface platters hold the recorded information.
- A typical HDD consists of a **spindle** which is a motor that holds a platter.
- Each working surface of the platter is divided into a number of concentric circles which is called **tracks**.
- Each of the tracks then will be divided into **sectors** (an equal number of sectors or a different number of sectors.
- Collection of all tracks that are the same distance from the edge of the platter is called **cylinder.**
- The data on the hard drive platter is read by **read-write heads**, of read-write arm, the read-write arm also known as **Actuator**.

Cont…

track t
spindle
arm assembly
sector s
cylinder c
read-write
head
platter
arm
rotation

DISK ACCESS TIME

- The time required to read or write a disk block mainly determined by the following **three** parameters:
- **1. Seek time: -**The time taken by read and write head to reach the desired track from the current position.
- **2. Rotational delay/Latency**: **-**The time taken by read and write head to reach the desired sector.
- **3. Transfer time:-**The time to transfer data.
- It is depends on the rotation speed of the disk and the number of bytes to be transferred.
- For most disks seek time is the most dominant time.
- Reducing the seek time can improve system performance.
- **Disk Access Time= Seek time + Rotational delay/Latency+ Transfer time**
- **Disk response time:- Time that a request waits to perform I/O operation.**

DISK ARM SCHEDULING ALGORITHM

- Some possible disk scheduling algorithms to reduce the average seek time are:
- **First-Come, First-Served (FCFS):** The disk requests are served in the order they arrive.
- The response time is poor.
- **Shortest Seek Time First (SSTF):** the next cylinder to be accessed is the nearest to the current position of the head. In this scheduling Requests having shortest seek time served first.
- It minimize the disk movement by serving the request closest to the current position of the head.
- This algorithm minimizes average seek time, but cylinders at the extreme locations suffer long waits.
- Since most requests tend to cylinders near the middle.
- Thus, minimal response time compromises fairness.

Cont'd

- **Example:-** Consider an imaginary disk with 51 cylinders. a request come in , to read a block on cylinder 11. while the seek to cylinder 11 is in progress , new requests come in for cylinders 1,36,16,34, 9 and 12 in that order.

- Starting from the current head position, what is the total distance (in cylinder) that the disk arm move to satisfy all the pending request in all disk scheduling algorithm ?

FCFS ALGORITHM EXAMPLE

1,
36,
16,
34,
9,
12

- Disk movement will be 11, 1, 36, 16, 34, 9 and 12.
- Total cylinder movement: (11-1) + (36-1) + (36-16) + (34-16) + (34-9) + (9-12) = 111

SSTF ALGORITHM EXAMPLE :

1,
36,
16,
34,
9,
12

- Disk movement will be 11, 12, 9, 16, 1, 34, 36.
- Total cylinder movement: (12-11) + (12-9) + (16-9) + (16-1) + (34-1) + (36-34) = 61

ELEVATOR ALGORITHM:

- The next cylinder to be accessed is determined by the direction of search.
- If the current direction is toward the outer cylinders, the next cylinder is one with the next higher number.
- If the direction is downward, the next cylinder is one with the next smaller number.
- The search direction is reversed, when there are no more requests in the current direction.
- In general, the average seek time of the elevator algorithm is higher than that of SSF, but has better fairness.

Elevator Algorithm Example

- Disk movement will be 11, 12, 16, 34, 36, 9, 1.
- Total cylinder movement: (12-11) + (16-12) + (34-16) + (36-34) + (36-9) + (9-1)=60

BUFFERING STRATEGIES

- Buffers are executed in **three** capabilities .

- **Zero capacity buffer** – the memory size of buffer is zero.

- **Bounded capacity buffer** – the memory size of the buffer is limited, so it Blocks the sender when it is full.

- **Unbounded capacity buffer** –memory size is of the buffer unlimited and it accepts any amount of data from the sender.

DIRECT MEMORY ACCESS (DMA)

- Direct memory access (DMA)

1. Transfers data without the intervention of CPU.

2 Used for high-speed devices (disk).

- Many computers avoid burdening the CPU with programmed I/O by offloading some of this work to a special purpose processor. This type of processor is called, a Direct Memory Access (DMA) controller.

DIRECT MEMORY ACCESS (DMA)

- A special control unit is used to transfer block of data directly between an external device and the main memory, without intervention by the processor. This approach is called **Direct Memory Access (DMA).**
- DMA can be used with either polling or interrupt software.
- DMA is particularly useful on devices like disks, where many bytes of information can be transferred in single I/O operations.
- When used with an interrupt, the CPU is notified only after the entire block of data has been transferred.

Cont'd

- For each byte or word transferred, it must provide the memory address and all the bus signals controlling the data transfer.
- Interaction with a device controller is managed through a device driver.
- Handshaking is a process between the DMA controller and the device controller.

SIX STEP PROCESS TO PERFORM DMA TRANSFER

Thank You!

Chapter Five

File System Management

Introduction

- A file is a collection of related information recorded on secondary storage.
- For example, files containing student information, files containing employee information, files containing C source code, and so on.
- Information in files could be program code or data in numeric, alphanumeric, alphabetic, or binary form either formatted or in free form.
- A file is therefore a collection of records if it is a data file or a collection of bits/bytes/lines if it is code.

Cont'd

- We used several files to store information on your computer as the information in them can later be retrieved and used.
- It is important that the files are managed properly, so that they can be located and accessed easily without consuming time.
- To organize and manage the files and directories on your computer, OS uses an application called **a file manager.**
- A file manager enables you to create, delete, copy, move, rename and view files and create and manage directories (folders).
- The way in which an os organizes, names, protects, accesses, and uses files is called a **file management system.**

Files are required for the following reasons.

- To store data permanently (even after the termination of the process that creates them).
- To store large data that is beyond the size of a process's address space.
- To enable multiple processes to access data simultaneously as in the case of databases.
- Part of the operating system that deals with files are known as the **file system.**
- Users and designers of file systems have different views of files.

Cont…

- From the users' point of view, the most important issue is the interface (naming, protection, and operations allowed.)
- From the designers' point of view, the choice between the use of linked lists versus bit maps and the number of tracks per block has more importance.

Files and Directories – Interface

- **Files and File Naming**
- Name is used to identify a file with an abstract way. A file name consists of strings.
- The maximum allowed length of the string and case sensitivity of names vary from one operating system to the other.
- Most operating systems have some characters, called file extensions, as part of the file name following a period (.).
- In some operating systems (such as Microsoft Windows), the file extension is used to associate the file with a program.
- Some characters, e.g. /, \ >, have special meaning in the file system so that they cannot be used as part of the file name.

File Access

- **Sequential access**: the bytes or records of the file are read from beginning to end sequentially.
- **Random access/direct accesss**: the bytes or records of the file can be read out of order.

FILE ATTRIBUTES

- A file has a number of attributes. The available attributes of files differ from one operating system to the other.
- The most common attributes are:
- Attributes for ownership, access permission, and password.
- Flags (bits) for archive, hidden, system, read-only, temporary, etc.
- File size.
- Time of creation, last access, and last modification.

Cont'd

Product inventory.accdt Properties
General
Security
Details
Product inventory.accdt
Type of file:
Microsoft Access Template (.accdt)
Opens with:
Access
Change...
Location:
C:\Users\Jon\Documents
Size:
3.79 MB (3,984,799 bytes)
Size on disk:
3.80 MB (3,985,408 bytes)
Created:
Thursday, July 12, 2018, 6:13:39 AM
Modified:
Thursday, July 12, 2018, 6:13:39 AM
Accessed:
Thursday, July 12, 2018, 6:13:39 AM
Attributes:
Read-only
Hidden
Advanced...
OK
Cancel
Apply

DMS Log Files Properties
General
Customize
DMS Log Files
Type:
File Folder
Location:
C:\Users\Jon\Documents
Size:
0 bytes
Size on disk:
0 bytes
Contains:
0 Files, 0 Folders
Created:
Friday, July 7, 2017, 4:57:01 PM
Attributes:
Read-only (Only applies to files in folder)
Hidden
Advanced...
OK
Cancel
Apply

File Operations

- In order to provide the basic store and retrieval operations of a file and other properties of files outlined above, operating systems have some common file operations.
- Some of them are: Create, delete, open, close, read, write, append, seek, get attributes, set attributes, and rename.

Seek:

For random access file, a method is needed to specify from where to take the data.

It repositions the pointer from the current position to a specific place in the file.

Truncate:

The user can erase the contents of a file. But can keep its attributes.

This operation allows all the attributes to remain unchanged except for file length. Thus the length of file is reset to 0 and its space is released.

Append:

This call is restricted form of **Write**. It can only add data to the end of the file.

Directories

- To keep track of files, file systems normally have directories (folders).
- In most operating systems directories themselves are files.
  - Single Level Directory Systems
  - Two Level Directory Systems
  - Hierarchical Directory Systems

Single Level Directory Systems

- There is only one directory (the root directory) in the system.
- The advantage of this scheme is its implementation simplicity and ease of locating files.
- The main disadvantage is the need to provide distinct name for each file.
- This kind of directory organization was used in early operating systems.
- Here f1, f2, f3, f4, f5 represent the five different files.
- Practically it can be thought of as a structure where all the files are stored in the same folder.

Two Level Directory Systems

- If two different users wanted to create a file with the same name (say report.doc), it was not allowed in a single-level directory.
- In a two-level directory structure, there is a master node that has a separate directory for each user.
- Each user can store the files in that directory.
- It can be practically thought of as a folder that contains many folders, each user can store files in the allocated directory just like a single-level directory.

Cont…

- For every user, there is a separate directory. At the next level, every directory stores the files just like a single-level directory.

Hierarchical Directory Systems

- This type of directory is used in our PCs.
- The biggest disadvantage of a two-level directory was that one could not create sub-directories in a directory.
- The tree-structured directory solved this problem.
- In a tree-structured directory, there is a root directory at the peak.
- The root directory contains directories for each user.
- The users can, however, create subdirectories inside their directory and also store the files.

Hierarchical Directory Systems

Directory Operations

- The type of operations on directories varies from one operating system to the other.
- The most common operations are: create, delete, opendir, closedir, and rename.

Files and Directories – Implementation

- **Implementing Files**
- The most important issue of file implementation is to keep track of which disk blocks go to which file. There are several alternative ways of allocating disk space for files.

Contiguous Allocation

- A file is stored in consecutive disk blocks.
- **Advantage:**
- Simple to implement: we need to keep track of first block of a file and the number of blocks used by the file.
- High performance: when accessing a file, the only delay is while searching the first block of the file.
- **Disadvantage:** The file size must be specified at the time of file creation.
- The disk would eventually have many holes (unused blocks) as files are deleted.
- The disk may be full, while there are holes at many locations which are small to contain a file of a specified length.

Cont…

(A)Contiguous allocation of disk space for seven files (b) state of disk space after files D and F have been removed.

Linked List Allocation

- Files are implemented as a linked list of disk blocks.
- The first word of a block is used to point to the next block and the rest of the block is used to store data.
- **Advantage:**
- Disk usage is effective; there is no fear of losing disk blocks to holes.
- It is enough to store only the address of the first block of a file in the directory.

Cont..

- **Disadvantage:**
- Random access of a file is slow since file access always begins at the first block.
- The amount of data stored in a block is not a power of two as some space in each block is allocated for the pointer to the next block.
- Since, many programs read/write data with size of a power of two, it may be necessary to read two consecutive blocks and concatenate the data.

Cont…

Figure 5.2 Implementation of two files (File A and File B) with linked list of disk blocks

Linked List Allocation Using a Table in Memory

- In this method, instead of a pointer, a table in memory called file allocation table is used.
- Take the pointer word from each block and put it in a table in memory.
- The main advantage of this method is random access becomes faster.
- The main disadvantage is much memory is required to store the table. Moreover, the size of the table grows linearly with disk size.

Implementation of Linked List Allocation Using a Table in Memory

I-NODES

- An i-node (index node) stores the addresses of disk blocks and attributes of a file.
- The advantage of i-nodes is that only the i-nodes of opened files are needed to be in memory.
- If the size of a file is too large to be stored in a disk space pointed by a single i-node.
- It is possible to use some of the last i-node entries to point to a block with more disk block addresses.

An example of an i-node for a file

Thank You!

Chapter Six

Protection and Security

Goals of Protection

- Computer system consists of a collection of objects, hardware or software.
- Each object has a unique name and can be accessed through a well-defined set of operations.
- Protection problem - ensure that each object is accessed correctly and only by those processes that are allowed to do so.

Domain Structure

- A domain is a collection of sets of <object, access-right> pairs.
- Access-right = valid operations on an object
  - Rights-set is a subset of all valid operations that can be performed on the object.
- Domain = set of <object, access-right> pairs.

- AR1 = <file_A,{Read, Write}> AR8 = <file_A,{Read}>

Access Matrix

- It is a row and column representation of a domain.
- Rows – domains.
- Columns – objects.
- Each entry – set of access rights.

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Use of Access Matrix

- If a process in Domain D tries to do "op" on object O, then "op" must be in the access matrix.
- Special access rights:
  - Owner of O.
  - Copy op from O to O (denoted by “*”).
  - Control – D can modify D access rights.
  - Transfer – switch from domain D to D.

Cont…

- Access matrix design separates mechanism from policy.
- **Mechanism** - operating system provides Access-matrix + rules.
- It ensures that the matrix is only manipulated by authorized agents and that rules are strictly enforced.
- **Policy** - user dictates policy.
- Who can access what object and in what mode.

Implementation of Access Matrix

- **Access List:-** For every object, the access list will maintain the domain name and the access right for the object in ordered pair.
- Domain 1 = Read, Write
- Domain 2 = Read
- Domain 3 = Read
- **Capability list:-** When we are using a capability list to implement an access matrix the capability list contains domain-wise information.
  - Every domain contains a set of objects and operations allowable in the object.
- Object 1 - Read
- Object 4 - Read, Write, Execute.
- Object 5 - Read, Write, Delete, Copy.

Revocation of Access Rights

- **Access List -** Delete access rights from the access list.
  - Simple
  - Immediate
- **Capability List –** The scheme required locating capability in the system before capability can be revoked.
  - Reacquisition: periodically delete the capability from each domain.

Security

- Security must consider external environment of the system, and protect it from:
  - Unauthorized access.
  - Malicious modification or destruction.
  - Accidental introduction of inconsistency.
- Easier to protect against accidental than malicious misuse.

Authentication

- User identity most often established through passwords can be considered a special case of either keys or capabilities.
- Passwords must be kept secret.
  - Frequent change of passwords.
  - Use of “non-guessable”" passwords (not in dictionary or username).
  - Log all invalid access attempts.

Program Threats

- **Trojan Horse**
  - Code segment that misuses its environment.
  - Exploits mechanisms for allowing programs written by users to be executed by other users.
- **Trap Door**
  - Specific user identifier or password that circumvents normal security procedures.
  - Could be included in a compiler or the kernel.

System Threats

- Worms – it is a standalone malicious program that can self-replicate and propagate independently.
- Viruses - fragments of code embedded in a legitimate program.
  - Mainly affect microcomputer systems.
  - Downloading viral programs from public bulletin boards or exchanging USBs containing an infection.

Threat Monitoring

- **Check for suspicious patterns of activity -** i.e., several incorrect password attempts may signal password guessing.
- **Audit log -** records the time, user, and type of all accesses to an object; useful for recovery from a violation and developing better security measures.

Cont…

- Scan the system periodically for security holes; when the computer is relatively unused.
- Check for:
  - Short or easy-to-guess passwords
  - Unauthorized programs in system directories
  - Unexpected long-running processes
  - Improper directory protections
  - Improper protection of system data files
  - Dangerous entries in the program search path (Trojan horse).

Encryption

- Encrypt the clear text into cipher text.
- Properties of good encryption technique:
  - Relatively simple for authorized users to encrypt and decrypt data.
  - The encryption scheme depends not on the secrecy of the algorithm but on a parameter of the algorithm called the **encryption key**.
  - Extremely difficult for an intruder to determine the encryption key.
- Encryption substitutes characters and rearranges their order on the basis of an encryption key, provided to authorized users via a secure mechanism.

Encryption

- Public-key encryption based on each user having two keys:
  - Public key – public key used to encrypt data.
  - Private key –private key known only to individual user used to decrypt data.
- Must be an encryption scheme that can be made public without making it easy to figure out the decryption scheme.

Thank You!

Debre Markos University Technology Institute

Software Engineering Academic Program

Operating System Sample Questions for Exit Exam

Choose the Best Answer from the Given Alternatives

1. Which one of the following statements is **False** about operating systems? [**Choose two**]
   - **A.** It provides a platform on which application software can be installed on the bottom.
   - **B.** It controls the overall functionality of the computer system.
   - **C.** It manages both the hardware and software resources.
   - **D.** We can use a computer, even if a computer has no installed operating system.
   - **E.** None
2. Which one of the following statements is **False** about process state transition?
   - **A.** If the given quantum time expired a running process can be interrupted and goes to the waiting queue.
   - **B.** When a process is created it goes to a ready state immediately.
   - **C.** When a process successfully finished its task, it goes to a terminated state
   - **D.** If a running process needs to perform an I/O operation before finishing its execution it goes to ready queue.
   - **E.** None
3. Which one of the following statements is **True**? [**Choose Two**]
   - **A.** With the help of an operating system, a user can communicate with the computer hardware without knowing what is happening in the background.
   - **B.** In a non-preemptive scheduling algorithm, the scheduler cannot remove the process from the processor even if, the allotted Burst time expired.
   - **C.** Computers we are using right now are an example of a multiuser, multitasking operating system.
   - **D.** The main goal of CPU scheduling is to make the CPU busy at all times.
   - **E.** When a program needs to be executed, it is loaded from the main memory to the secondary memory.
4. Which one of the following operating systems is Open source?

   A. UNIX **B**. Windows **C**. Linux **D**. MacOS **E**. None
5. Which one of the following CPU scheduling algorithms is both preemptive and non-preemptive

   A. Round Robin **B**. Priority **C.** FCFS **D**. SJF **E**. None
6. Assume Mr. Daniel is running different applications at the same time like browsing the internet while preparing his assignment, so which type of operating system he used

   A. ROTS **B**. Multi-tasking **C**. Multi-user **D**. DOS **E**. None
7. A process stack does not contain _____

   A. Function parameters **C.** Return addresses

   B. Local variables **D.** PID of the child process **E**. None

8. The number of processes that complete their execution per unit of time is known as _____

A. Turnaround time
B. Throughput
C. Efficiency
D. Response time
E. None

9. Which of the following is **True** about Process?

A. A process is basically a program in execution
B. The execution of a process must progress in a sequential fashion.
C. A process is defined as an entity that represents the basic unit of work to be implemented in the system.
D. All of the above
E. Except C

10. In PCB, which one specifies the number of resources used by the system. Such as the CPU time of a process, Time limits, and process numbers.

A. IO status information
B. CPU Scheduling Information
C. CPU registers
D. Accounting information

11. which one of the following statements are **False**. [**choose two**]

A. Context switching means switching the CPU from one process to other processes.
B. In non-preemptive scheduling algorithm, the response times of the process is better than preemptive.
C. Preemptive scheduling algorithm is flexible as it allows any high priority process to run first
D. Preemptive scheduling algorithm can be also called run to completion" method.
E. None

12. Which one of the following CPU scheduling Algorithm used in timesharing system.

A. SRTF **B**. Round Robin **C**. FCFS **D**. Priority **E**. SJF

13. Which one of the following is **not** necessary condition for deadlock to occurred.

A. Hold and Wait
C **C** No- preemption
B. Mutual inclusion
D. Circular wait
E. None

14. Let's assume we have five processes P1, P2, P3, P4, and P5 with burst time and arrival time as shown below. So, by applying **SJF CPU Scheduling algorithm** what will be the average Turnaround time and average Waiting Time respectively

Process	Burst Time	Arrival Time
P1	10	0
P2	8	0
P3	3	4
P4	15	5
P5	2	10

A. 7.2, 15

B. 14.8, 7.2

C. 15,7.2

D. 7.2,14.8

15. Assume there are 6 processes with burst time and arrival time as shown below in the table. Then by applying **Round Robin CPU scheduling algorithm**, what will be the average turnaround time and average waiting time respectively. Consider time quantum for each process is 3ms.

Process	Arrival Time	Burst Time
P1	0	4
P2	1	5
P3	2	3
P4	3	3
P5	4	6
P6	6	3

A. 12.7, 9.1

B. 10, 14

C. 9.66,13.66

D. 13.66,9.66