<table>
<tr><th>Q</th><th></th><th>Answer</th><th>M</th></tr>
<tr><td>1</td><td>a)</td><td>What are the various objectives and functions of Operating Systems?</td><td>5</td></tr>
<tr><td></td><td>Ans:</td><td>
• Operating systems (OS) are an essential part of any computer system.
• They act as a communication bridge between the user and computer hardware, providing a platform on which a user can execute programs conveniently and efficiently.
• The primary goal of an operating system is to make the computer environment more convenient to use, and the secondary goal is to use the resources most efficiently.
• The following are the main objectives of an operating system:
- To make the computer system convenient to use in an efficient manner.
- To hide the details of the hardware resources from the users.
- To provide users a convenient interface to use the computer system.
- To act as an intermediary between the hardware and its users, making it easier for the users to access and use other resources.
- To manage the resources of a computer system.

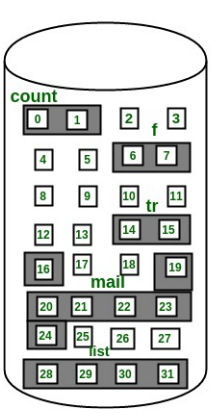
• The functions of an operating system include:
i. Memory management: managing the allocation and deallocation of memory to various processes and ensuring that the other process does not consume the memory allocated to one process.
ii. Processor management: allocating the processor to different processes and ensuring that the processor is used efficiently.
iii. Device management: managing the allocation of devices to different processes and ensuring that the devices are used efficiently.
iv. File management: managing the creation, deletion, and modification of files and directories.
v. Security: ensuring that the system is secure from unauthorized access and malicious software.
</td><td></td></tr>
<tr><td></td><td>b)</td><td>Differentiate between process and threads.</td><td>5</td></tr>
<tr><td></td><td>Ans:</td><td>
<table>
<tr><th>Process</th><th>Thread</th></tr>
<tr><td>A process is an instance of a program in execution.</td><td>A thread is a sequence of instructions within a process.</td></tr>
<tr><td>A process has its own address space, code, data, and resources.</td><td>A thread shares the address space, code, data, and resources of its process.</td></tr>
<tr><td>Processes are isolated from each other and cannot directly communicate.</td><td>Threads can communicate with each other through shared memory and synchronization mechanisms.</td></tr>
<tr><td>Creating and switching between processes is expensive in terms of time and resources.</td><td>Creating and switching between threads is cheaper and faster than processes.</td></tr>
<tr><td>Processes can have multiple threads running concurrently within them.</td><td>Threads can only run within a process and cannot exist independently.</td></tr>
</table>
</td><td></td></tr>
</table>

	c)	Explain about Resource Allocation Graph (RAG).	5
	Ans:	• A Resource Allocation Graph (RAG) is a graphical representation of the state of a system in terms of processes and resources. • It shows which resources are held by which processes and which processes are waiting for which resources. • It consists of two types of vertices: process vertices (circles) and resource vertices (squares). • It also consists of two types of edges: request edges (from processes to resources) and assignment edges (from resources to processes). • It can be used to detect and avoid deadlock situations by identifying cycles in the graph. • A deadlock occurs when a set of processes are waiting for each other in a circular chain, and none of them can proceed. • To avoid deadlock, the system can use some strategies to prevent the formation of cycles in the RAG, such as resource ordering or the banker's algorithm.	
	d)	Explain about file attributes, file operations, and file types.	5
	Ans:	File attributes, file operations, and file types are important concepts in operating systems that deal with how files are stored, accessed, and manipulated. Here is a brief explanation of each concept: • File attributes are properties of a file that provide additional information about the file. Examples of file attributes include file name, file size, file type, modification date, file permissions, owner, and group. These attributes can be used to organize and manage files more efficiently. File attributes can be displayed and modified by using commands or graphical interfaces. Some file attributes are platform-specific, meaning they only apply to certain operating systems or file systems. • File operations are actions that can be performed on files, such as creating, opening, reading, writing, closing, deleting, copying, moving, renaming, and appending. File operations are usually implemented by system calls or application programming interfaces (APIs) that allow programs to interact with the file system. File operations may have different effects depending on the file attributes and the file system. For example, deleting a file may not actually remove the file data from the disk, but only mark the file as deleted in the file system. • File types are categories of files that indicate the format and content of the file data. File types are usually determined by the file extension, which is a suffix added to the file name after a dot. For example, .txt indicates a text file, .jpg indicates a JPEG image file, and .exe indicates an executable file. File types help the operating system and the users to identify the appropriate programs to open and process the files. Some file types are universal, meaning they can be recognized and used by different operating systems and applications, while some	

		file types are proprietary, meaning they can only be used by specific programs or platforms.	
	e)	What is virtual memory? Mention its advantages.	5
	Ans:	• Virtual memory is a technique that allows a computer to use more memory than is physically available by using a part of the secondary storage, such as a hard disk or a solid-state drive, as an extension of the main memory, such as RAM. • Virtual memory creates an illusion that the computer has more memory than it actually does and enables the execution of larger and more complex programs. Some of the advantages of virtual memory are: • It allows more processes to be maintained in the main memory, as only the required pages or segments of each process need to be loaded into memory. This leads to more efficient utilization of the processor and faster switching between processes. • It provides memory isolation and protection, as each process operates in its own virtual address space and cannot access or modify the memory of other processes. This enhances the security and reliability of the system. • It simplifies the programming and linking of applications, as the virtual address space is consistent and independent of the physical memory layout. This also enables the sharing of common libraries and code segments among different processes. • It allows the system to handle larger workloads and run more applications at once, as the virtual memory can be dynamically allocated and deallocated according to the demand. This also reduces the need for additional memory modules when the physical memory runs out.	
2	a)	Explain file allocation methods in detail with proper diagram.	10
	Ans:	• File allocation methods are different ways of storing files on a disk, such that they can be accessed efficiently and securely by the operating system and the users. • There are three main file allocation methods: contiguous, linked, and indexed. i. Contiguous file allocation: o Each file occupies a contiguous (adjacent) set of blocks on the disk. o The directory entry for a file contains the address of the starting block and the length of the file (in terms of blocks). o It supports both sequential and direct access, as the address of any block of the file can be easily calculated by adding the block number to the starting address. o It suffers from both internal and external fragmentation, as it is difficult to find a contiguous chunk of free blocks for a	

file, and the file size cannot be easily increased or decreased.

- It wastes disk space, as the file may not occupy the entire last block allocated to it.

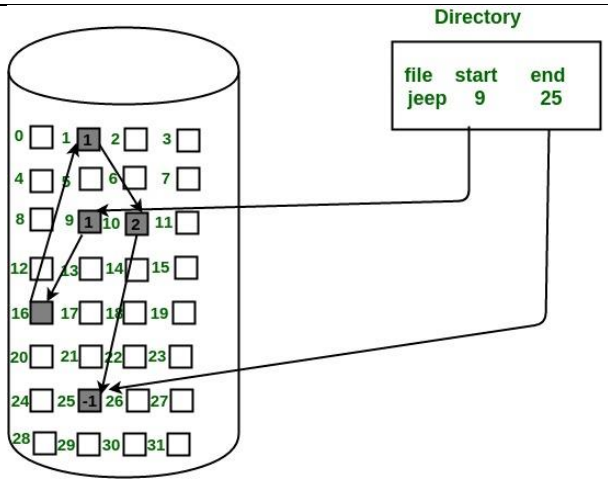


Directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

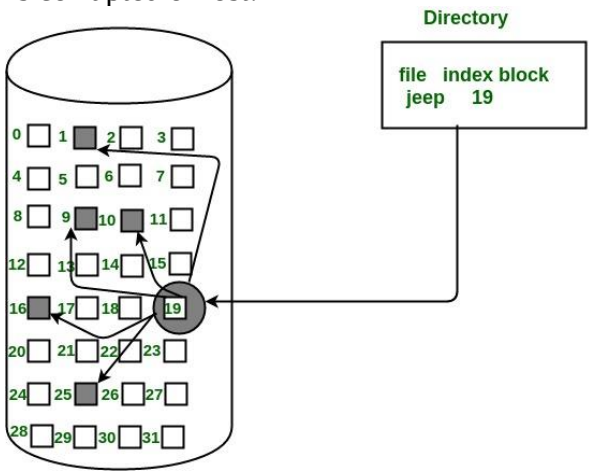
ii. Linked file allocation:

- Each file is a linked list of disk blocks, which need not be contiguous. The disk blocks can be scattered anywhere on the disk.
- The directory entry for a file contains a pointer to the first and the last block of the file. Each block contains a pointer to the next block occupied by the file. The last block contains a null pointer, indicating the end of the file.
- It is flexible in terms of file size, as the file can grow or shrink by adding or removing blocks from the linked list.
- It avoids external fragmentation, as any free block can be used to store a file block.
- It does not support direct access, as the blocks of the file have to be accessed sequentially by following the pointers.
- It incurs extra overhead for storing the pointers in each block and may cause disk failure if a pointer is corrupted or lost.



iii. Indexed file allocation:

- Each file has a special block, called the index block, that contains the pointers to all the blocks occupied by the file. The index block can be either a single block or a linked list of blocks, depending on the size of the file.
- The directory entry for a file contains a pointer to the index block of the file.
- It combines the advantages of both contiguous and linked file allocation, as it supports direct access, avoids external fragmentation, and allows dynamic file size.
- It also has some drawbacks, such as the extra overhead for storing the index block, the limitation on the maximum file size (depending on the number of pointers that can fit in the index block), and the possibility of disk failure if the index block is corrupted or lost.



	b)	Consider the following set of processes indicated as (process name, Arrival time, burst time) for the following. (P1,0,6), (P2,1,4), (P3,3,5), (P4, 5, 3). Draw the Gantt charts illustrating the execution of these processes using pre-emptive and non-pre-emptive SJF and FCFS. Calculate average turnaround time, average waiting time in each case.	10
	Ans:		
3	a)	Give the explanation of necessary conditions for deadlock. Explain how a resource allocation graph determines a deadlock.	10
	Ans:	• The necessary conditions for deadlock are four conditions that must hold simultaneously for a deadlock situation to occur. These conditions are also known as Coffman conditions, and they are: ○ **Mutual exclusion:** This condition requires that at least one resource be held in a non-shareable mode, which means that only one process can use the resource at any given time. If another process requests the same resource, it has to wait until the resource is released by the current holder. ○ **Hold and wait:** This condition specifies that a process must be holding at least one resource while waiting for other resources that are currently held by other processes. This means that a process cannot release its allocated resources until it gets all the resources it needs. ○ **No pre-emption:** This condition states that a resource cannot be forcibly taken away from a process that is holding it. A process can only release a resource voluntarily after completing its task. This means that a process cannot be interrupted by another process that needs the same resource. ○ **Circular wait:** This condition implies that there exists a set of processes {P1, P2, ..., Pn} such that P1 is waiting for a resource held by P2, P2 is waiting for a resource held by P3, ..., Pn is waiting for a resource held by P1. This forms a circular chain of processes that are waiting for each other, and none of them can proceed. • A resource allocation graph (RAG) is a graphical representation of the state of a system in terms of processes and resources. It shows which resources are held by which processes and which processes are waiting for which resources. It consists of two types of vertices: process vertices (circles) and resource vertices (squares). It also consists of two types of edges: request edges (from processes to resources) and assignment edges (from resources to processes). • A resource allocation graph can be used to determine whether a deadlock exists in the system or not. The rules are: ○ If the RAG has no cycles, then there is no deadlock. ○ If the RAG has a cycle, then there may or may not be a deadlock, depending on the type of resources involved.	

- If all the resources in the cycle are single-instance resources, then there is a deadlock. This is because each process in the cycle is holding one resource and waiting for another resource that is held by another process in the cycle, and none of them can proceed.
- If some of the resources in the cycle are multi-instance resources, then there may or may not be a deadlock. This is because some processes in the cycle may be able to obtain the resources, they need from the available instances of the multi-instance resources and break the cycle. To confirm whether there is a deadlock or not, a more detailed analysis, such as the banker's algorithm, is required.

b) What is Internal fragmentation? Explain static partitioned allocation with partition sizes 400,180, 100,300,45. Assuming First fit and Best fit method indicate the memory status after memory request for sizes 95, 180, 285, 380, 30. — 10

Ans:

- Internal fragmentation is a phenomenon that occurs when a process is allocated to a memory block whose size is more than the size of that process, and due to which some part of the memory is left unused. This unused memory is called internal fragmentation.
- Static partitioned allocation is a memory allocation technique in which the memory is divided into fixed-sized partitions, and each partition is assigned to a process. The partition sizes are fixed and do not change during the execution of the program. The partitions can be allocated to the processes in two ways:
  i. First Fit
  ii. Best Fit.
- In First Fit, the memory is allocated to the first partition that is large enough to hold the process. In contrast, in Best Fit, the memory is allocated to the partition that is closest in size to the process but still larger than the process.

Let's assume we have the following partitions: 400, 180, 100, 300, and 45. We will now allocate memory to the processes of sizes 95, 180, 285, 380, and 30 using both First Fit and Best Fit methods.

First Fit Method:

Process Size	Partition Allocated	Memory Status
95	100	5

180	180	0
285	400	115
380	400	35
30	45	15

Best Fit Method:

Process Size	Partition Allocated	Memory Status
95	100	5
180	180	0
285	300	15
380	400	20
30	45	15

4	a)	What is a thread? How multithreading is beneficial? Compare and contrast different multithreading models.	10
	Ans:	**Thread:** • A thread is a lightweight process that can run concurrently with other threads within a process. • Threads share the same memory space and can access the same data, making them more efficient than processes. **Multithreading**:	

		• Multithreading is beneficial because it allows multiple threads to execute concurrently, which can improve the performance of an application. • By dividing a program into multiple threads, the program can take advantage of the available CPU resources and execute tasks in parallel. • This can lead to faster execution times and improved responsiveness. • There are several multithreading models, including many-to-one, one-to-one, and many-to-many models. • In the many-to-one model, multiple user-level threads are mapped to a single kernel thread. • In contrast, in the one-to-one model, each user-level thread is mapped to a kernel thread. Finally, in the many-to-many model, multiple user-level threads are mapped to multiple kernel threads. • Each model has its own advantages and disadvantages. The many-to-one model is efficient because thread management is done in user space, but the entire process will block if a thread makes a blocking system call. • The one-to-one model provides more concurrency than the many-to-one model, but it can be less efficient because of the overhead associated with creating kernel threads. Finally, the many-to-many model provides the best concurrency, but it can be complex to implement and manage.	
	b)	Explain paging in detail. Describe how logical address is converted into physical address.	10
	Ans:	• Paging is a memory management technique that allows a computer to use virtual memory. • In paging, the logical address space of a process is divided into fixed-size blocks called pages, and the physical memory is divided into blocks of the same size called page frames. • The operating system maps pages to page frames, allowing the process to access memory that is not physically present in the main memory. • To convert a logical address to a physical address in paging, the operating system uses a page table. • The page table contains information about the mapping between logical addresses and physical addresses. • The page table is indexed using the page number of the logical address, which is obtained by dividing the logical address by the size of the page. The remainder of this division is the page offset. Here are the steps to convert a logical address to a physical address: 1. Divide the logical address by the size of the page to obtain the page number. 2. Use the page number to index the page table to obtain the corresponding page frame number. 3. Multiply the page frame number by the size of the page to obtain the base address of the page frame. 4. Add the page offset to the base address to obtain the physical address.	

		There are two types of paging: i. static paging ii. dynamic paging. In static paging, the size of the page is fixed and determined at compile time. In contrast, in dynamic paging, the size of the page can be changed at runtime.	
5	a)	What is semaphore and its types? How the classic synchronization problem -Dining philosopher is solved using semaphores?	10
	Ans:	• **Semaphore** is a synchronization object that controls access to a shared resource in a concurrent system. • It is a non-negative integer variable that is used to signal between processes or threads. • Semaphores are of different types, including general semaphores, binary semaphores, counting semaphores, and strong semaphores. • The **Dining Philosopher Problem** is a classic synchronization problem that involves a group of philosophers who sit around a circular table and share chopsticks. • The problem is to design a solution that allows each philosopher to eat without causing a deadlock or a starvation. • One solution to this problem is to use semaphores to represent the chopsticks. • In this solution, each fork is represented by a semaphore, and a philosopher must acquire both the semaphore for the fork to their left and the semaphore for the fork to their right before they can eat. • This solution ensures that no two philosophers will be able to pick up the same chopstick at the same time, thus avoiding a deadlock. OR • Semaphore is a synchronization object that controls access to a shared resource in a concurrent system. • Semaphores are of different types, including general semaphores, binary semaphores, counting semaphores, and strong semaphores. • The Dining Philosopher Problem is a classic synchronization problem that involves a group of philosophers who sit around a circular table and share chopsticks. • One solution to this problem is to use semaphores to represent the chopsticks. • In this solution, each fork is represented by a semaphore, and a philosopher must acquire both the semaphore for the fork to their left and the semaphore for the fork to their right before they can eat. • This solution ensures that no two philosophers will be able to pick up the same chopstick at the same time, thus avoiding a deadlock. • Semaphore is an essential synchronization mechanism for concurrent systems. • Semaphores can be used to control access to shared resources, coordinate the execution of multiple threads or processes, and avoid race conditions and deadlocks.	

		• There are several types of semaphores, each with its own advantages and disadvantages. • The choice of semaphore type depends on the specific requirements of the application.	
	b)	Explain RAID Level in Details	10
	Ans:	• RAID (Redundant Array of Independent Disks) is a technique that makes use of a combination of multiple disks instead of using a single disk for increased performance, data redundancy, or both. • RAID levels are used to describe the different structural approaches and functions of the RAID. • The most used RAID levels are RAID 0, RAID 1, RAID 5, RAID 6, and RAID 10. Here is a brief description of each RAID level: **i. RAID 0 (Striping):** o Blocks are "stripped" across disks. o It is easy to implement and utilizes the storage capacity in a better way. o However, it is not a good choice for a critical system as a single drive loss can result in the complete failure of the system. **ii. RAID 1 (Mirroring):** o More than one copy of each block is stored in a separate disk. Thus, every block has two (or more) copies, lying on different disks. o It is capable of reliability, but it is not an economical option as it requires twice the number of disks. **iii. RAID 5 (Block-Level Stripping with Distributed Parity):** o It distributes parity among all the disks in the array. o It is a good choice for systems that require high read performance and moderate write performance. o It is also cost-effective as it requires only one disk for parity. **iv. RAID 6 (Block-Level Stripping with two Parity Bits):** o It is similar to RAID 5, but it uses two parity bits instead of one. o It can tolerate the failure of two disks simultaneously, making it more reliable than RAID 5. However, it requires more disk space for parity information. **v. RAID 10 (Striping of Mirrors):** o It is a combination of RAID 0 and RAID 1. o It provides both performance and redundancy by striping data across mirrored pairs. It requires at least four disks and is more expensive than other RAID levels.	
6	a)	Consider the page reference string 1,2,3,5,2,4,5,6,2,1,2,3,7,6,3,2,1,2,3,6. Calculate the Page fault using 1. Optimal 2.LRU 3. FIFO algorithms for a memory with three frames.	10

Ans:

i. FIFO algorithm

- Total frames: 3
- Algorithm: FIFO
- Reference string length: 20 references
- String: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

Solution visualization

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ref		**1**	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
f		1	2	3	4	4	1	5	6	2	1	1	3	7	6	6	2	1	1	3	6
f			1	2	3	3	4	1	5	6	2	2	1	3	7	7	6	2	2	1	3
f				1	2	2	3	4	1	5	6	6	2	1	3	3	7	6	6	2	1
hit		✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✓	✗	✗
v					1		2	3	4	1	5		6	2	1		3	7		6	2

- Total references: 20
- Total distinct references: 7
- Hits: 4
- Faults: 16
- **Hit rate:** 4/20 = **20**%
- **Fault rate:** 16/20 = **80%**

ii. Optimal Page Replacement algorithm

- Total frames: 3
- Algorithm: OPT
- Reference string length: 20 references
- String: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

Solution visualization

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ref		**1**	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
f		1	2	3	4	4	4	5	6	6	6	6	3	7	7	7	2	1	1	1	6
f			1	2	2	2	2	2	2	2	2	2	6	3	3	3	3	2	2	2	2
f				1	1	1	1	1	1	1	1	1	2	6	6	6	6	3	3	3	3
hit		✗	✗	✗	✗	✓	✓	✗	✗	✓	✓	✓	✗	✗	✓	✓	✗	✗	✓	✓	✗
v					3			4	5				1	2			7	6			1

- Total references: 20

- Total distinct references: 7
- Hits: 9
- Faults: 11
- **Hit rate:** 9/20 = **45**%
- **Fault rate:** 11/20 = **55**%

iii. LRU algorithm

- Total frames: 3
- Algorithm: LRU
- Reference string length: 20 references
- String: 1 2 3 4 2 1 5 6 2 1 2 3 7 6 3 2 1 2 3 6

Solution visualization

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ref		**1**	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
f		1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
f			1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3
f				1	2	3	4	2	1	5	6	6	1	2	3	7	6	3	3	1	2
hit		✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✓	✓	✗
v					1		3	4	2	1	5		6	1	2		7	6			1

- Total references: 20
- Total distinct references: 7
- Hits: 5
- Faults: 15
- **Hit rate:** 5/20 = **25**%
- **Fault rate:** 15/20 = **75**%

b) What is open-source operating system? What are the design issues of Mobile operating system and Real time operating system? 10

Ans:

1. An open-source operating system is a type of operating system whose source code is publicly available for anyone to view, modify, use, and share under the terms of open-source licenses such as MIT, GNU Public License, and Apache 2.0.
2. Linux is one of the most popular open-source operating systems. It provides the core of these operating systems, the kernel that interacts with a computer's hardware, and was developed by Linus Torvalds in 1991.
3. Operating system makers then build tools that plug into this Linux kernel to create an operating system. These tools range from the windowing systems that power graphical desktops to the systems managing services running in the background.

4. While Linux underpins most open-source operating systems, there are OSes built around other kernels. One notable alternative is FreeBSD, a free operating system whose lineage dates back to the Berkeley Unix operating system of the 1970s but that isn't built around the Linux kernel.

Here are some design issues of mobile operating systems:

1. **Multitasking**:
    - Mobile operating systems are not true multitasking systems.
    - Only one app can be active at a time, and when another app is started, or the app is interrupted by another app (for example, a phone call), the app that was running gets put in the background.
    - It remains in the background until the user specifically accesses it again.
    - If it remains in the background too long, or if available memory gets too low, the operating system may kill it.
2. **Battery life**:
    - Mobile devices are powered by batteries, so battery life is a critical issue. Mobile operating systems must be designed to minimize power consumption.
    - This is achieved by using power-efficient hardware, optimizing software, and providing users with tools to manage power consumption .
3. **Screen size and resolution**:
    - Mobile devices have smaller screens than desktop computers, so mobile operating systems must be designed to work with smaller screens.
    - This means that the user interface must be optimized for small screens, and the operating system must be able to handle different screen resolutions.
4. **Security**:
    - Mobile devices are often used to access sensitive information, so mobile operating systems must be designed with security in mind.

- This includes features such as encryption, secure boot, and secure storage.

Here are some design issues of real-time operating systems:

1. **Determinism**:
    - Real-time operating systems must be deterministic, meaning that they must be able to guarantee that a task will be completed within a certain amount of time.
    - This is critical for real-time applications such as industrial control systems, where timing is critical.
2. **Scheduling**:
    - Real-time operating systems must be able to schedule tasks in a way that ensures that the most critical tasks are completed first.
    - This requires sophisticated scheduling algorithms that take into account the priority of each task and the resources required to complete it.
3. **Interrupt handling**:
    - Real-time operating systems must be able to handle interrupts quickly and efficiently.
    - This is critical for real-time applications, where an interrupt can indicate a critical event that requires immediate attention.
4. **Memory management**:
    - Real-time operating systems must be able to manage memory efficiently.
    - This includes features such as memory protection, virtual memory, and memory allocation algorithms that minimize fragmentation.