**D.Y. PATIL COLLEGE OF ENGINEERING & TECHNOLOGY**

**KASABA BAWADA KOLHAPUR-416006**

**(An Autonomous Institute)**

S. Y. B. Tech (CSE)

**END SEMESTER EXAMINATION (ESE), May -June. – 2023**

Course Name: **Operating System,** Course Code: **201CSL211**

**Day and Date: Thursday, 01.06.2023**

**Time: 1.00 am to 12.00 pm Max. Marks- 50**

**Key Answer**

**Unit No. 1, 2 & 3**

**Set-: II**

**Q. Paper Code:**

**22SYCS211402**

**Q. 1 Solve the following questions,**

|  |  |  |
| --- | --- | --- |
| **Q. 1** | **Solve the following questions,** | |
|  | a) | List the types of Operating System. Explain any 3 types of Operating System.  **Explanation of each type[3X2]**  Types of Operating System.  1. Batch Operating System  2. Multiprogrammed Operating System  3. Multitasking Operating System  4. Real Time OS  5. Distributed  6. Clustered  7. Embedded OS   1. Batch Operating System     The users of a batch operating system do not interact with the computer directly. Each user prepares his job on an off-line device like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group. The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.  This grouping of batches is done with the help of the operator. When we have the batches ready, the execution is done one by one for the batches which means batch-wise.  The problems with Batch Systems are as follows −  • Lack of interaction between the user and the job.  • CPU is often idle, because the speed of the mechanical I/O devices is slower than the CPU.  • If a job fails, the other jobs will have to wait for an unknown time.  • Batch systems are sometimes costly.  • Difficult to debug.   1. Multiprogramming Operating System   Multiprogramming is an extension to batch processing where the CPU is always kept busy. Each process needs two types of system time: CPU time and IO time.  In a multiprogramming environment, when a process does its I/O, The CPU can start the execution of other processes. Therefore, multiprogramming improves the efficiency of the system.     1. Time-Sharing Operating Systems – Each task is given some time to execute so that all the tasks work smoothly. Each user gets the time of CPU as they use a single system. These systems are also known as Multitasking Systems. The task can be from a single user or different users also. After this time interval is over OS switches over to the next task. The multitasking operating system is a logical extension of a multiprogramming system that enables multiple programs simultaneously. It allows a user to perform more than one computer task at the same time.     **Explanation of each type[3X2]** |
|  | b) | Describe the different models used for inter-process communication. Suggest which model is better and why?.  **MS–Expanation of each model[0],Suggestion[01]** Inter Process Communication (IPC) IPC helps achieve these things:   * 1. Computational Speedup (all the tasks are made to run concurrently and small tasks communicate with each other)   2. Modularity ( a big system is broken into small system to perform in less amount of time and efficiently)   3. Information and data sharing (can share files, printer,scanner etc)   4. Convenience (it helps user interact with other process) Approaches for Inter Process Communication      1. Shared Memory 2. Message Passing 3. Message Queue 4. Pipes 5. Direct communication 6. Indirect Communication 7. FIFO 8. Shared Memory  * Multiple processes can access a common shared memory by cooperating process. * Multiple processes communicate by shared memory, where one process makes changes at a time and then others view the change. * Shared memory does not use kernel. * Processes can exchange information by reading and writing data to shared memory.     Let the two cooperating processes P1 and P2. Both the processes P1 and P2, have their different address spaces. Now let us assume, P1 wants to share some data with P2.  So, P1 and P2 will have to perform the following steps −  Step 1 − Process P1 has some data to share with process P2. First P1 takes initiative and establishes a shared memory region in its own address space and stores the data or information to be shared in its shared memory region.  Step 2 − Now, P2 requires the information stored in the shared segment of P1. So, process P2 needs to attach itself to the shared address space of P1. Now, P2 can read out the data from there.  Step 3 − The two processes can exchange information by reading and writing data in the shared segment of the process.   1. Message Passing  * It is a type of mechanism that allows processes to synchronize and communicate with each other. * In message passing model communication takes place by means of messages exchanged between the cooperating processes. * Usually, the inter-process communication mechanism provides two operations that are as follows: * send (message) * received (message) * Size of the message can be fixed or variable   Disadvantage of Message Passing Model  The message passing model has slower communication than the shared memory model because the connection setup takes time.   1. Message Queue   Communication using message queues can happen in the following ways −   * + Message QueueWriting into the shared memory by one process and reading from the shared memory by another process. As we are aware, reading can be done with multiple processes as well.   + Multiple Message QueueWriting into the shared memory by one process with different data packets and reading from it by multiple processes, i.e., as per message type.   Having seen certain information on message queues, now it is time to check for the system call (System V) which supports the message queues.  To perform communication using message queues, following are the steps −  Step 1 − Create a message queue or connect to an already existing message queue (msgget())  Step 2 − Write into message queue (msgsnd())  Step 3 − Read from the message queue (msgrcv())  Step 4 − Perform control operations on the message queue (msgctl())   1. Pipes      * A pipe is a mechanism by which the output of one process is directed into the input of another process. Thus it provides one way flow of data between two related processes. * Although pipe can be accessed like an ordinary file, the system actually manages it   as FIFO queue. A pipe file is created using the pipe system call. A pipe has an input end and an output end. One can write into a pipe from input end and read from the output end. A pipe descriptor, has an array that stores two pointers, one pointer is for its input end and the other pointer is for its output end.   * Suppose two processes, Process A and Process B, need to communicate. In such a case, it is important that the process which writes, closes its read end of the pipe and the process which reads, closes its write end of a pipe. Essentially, for a communication from Process A to Process B the following should happen. * Process A should keep its write end open and close the read end of the pipe. * Process B should keep its read end open and close its write end. When a pipe is created, it is given a fixed size in bytes.  1. Direct Communication  * The process which wants to communicate must explicitly name the recipient or sender of the communication.   e.g. send(p1, message) means send the message to p1.  Similarly, receive(p2, message) means to receive the message from p2.   * In this method of communication, the communication link gets established automatically, which can be either unidirectional or bidirectional, but one link can be used between one pair of the sender and receiver and one pair of sender and receiver should not possess more than one pair of links. The problem with this method of communication is that if the name of one process changes, this method will not work.  1. Indirect Communication  * processes use mailboxes (also referred to as ports) for sending and receiving messages. Each mailbox has a unique id and processes can communicate only if they share a mailbox. * Link established only if processes share a common mailbox and a single link can be associated with many processes. * Each pair of processes can share several communication links and these links may be unidirectional or bi-directional. * Suppose two processes want to communicate through Indirect message passing, the required operations are: create a mailbox, use this mailbox for sending and receiving messages, then destroy the mailbox.. * There is a problem with this mailbox implementation. Suppose there are more than two processes sharing the same mailbox and suppose the process p1 sends a message to the mailbox, which process will be the receiver? This can be solved by either enforcing that only two processes can share a single mailbox or enforcing that only one process is allowed to execute the receive at a given time or select any process randomly and notify the sender about the receiver. * A mailbox can be made private to a single sender/receiver pair and can also be shared between multiple sender/receiver pairs.  Which approach is better? Ans: Shared Memory The advantages of Shared Memory are as follows −   * + Shared memory is a faster inter process communication system.   + It allows cooperating processes to access the same pieces of data concurrently.   + It speeds up the computation power of the system and divides long tasks into smaller sub-tasks and can be executed in parallel.   + Modularity is achieved in a shared memory system.   + Users can perform multiple tasks at a time.   **MS–Expanation of each model[06],Suggestion[01]** |
|  | c) | What are the different methods for handling deadlock? Explain Banker’s Algorithm.  There are four approaches to dealing with deadlocks.  1. Deadlock Prevention  2. Deadlock avoidance (Banker's Algorithm)  3. Deadlock detection & recovery  4. Deadlock Ignorance (Ostrich Method)  Safety Algorithm  It is a safety algorithm used to check whether or not a system is in a safe state or follows the safe sequence in a banker's algorithm:  1. There are two vectors **Wok** and **Finish** of length m and n in a safety algorithm.  Initialize: Work = Available Finish[i] = false; for I = 0, 1, 2, 3, 4… n - 1.   1. Check the availability status for each type of resources [i], such as:   Need[i]<=Work Finish[i]==false If the i does not exist, go to step 4.  3. Work = Work +Allocation(i) // to get new resource allocation  Finish[i] = true  Go to step 2 to check the status of resource availability for the next process.   * 1. If Finish[i] == true; it means that the system is safe for all processes.   **MS – Each method [02], Bankers Algorithm[5]** |
|  |  | **Unit No. 04** |
|  |  | **Q.2: All Questions are compulsory** |
|  | a) | What is Segmentation? Explain the basic segmentation method.  **MS – Definition[01],Explanation [03]**  In Operating Systems, Segmentation is a memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process. |
|  | b) | Draw and Explain schematic view of Swapping.    A process can be swapped temporarily out of memory to a backing  store, and then brought back into memory for continued execution  Total physical memory space of processes can exceed physical  memory  Backing store – fast disk large enough to accommodate copies of all  memory images for all users; must provide direct access to these  memory images  Roll out, roll in – swapping variant used for priority-based scheduling  algorithms; lower-priority process is swapped out so higher-priority  process can be loaded and executed  Major part of swap time is transfer time; total transfer time is directly  proportional to the amount of memory swapped  System maintains a ready queue of ready-to-run processes which have  memory images on disk    **MS - Sketch [02],Explanation[04]**  **OR**    Now, let's understand the procedure of page fault handling in the OS:   1. Firstly, an internal table for this process to assess whether the reference was valid or invalid memory access. 2. If the reference becomes invalid, the system process would be terminated. Otherwise, the page will be paged in. 3. After that, the free-frame list finds the free frame in the system. 4. Now, the disk operation would be scheduled to get the required page from the disk. 5. When the I/O operation is completed, the process's page table will be updated with a new frame number, and the invalid bit will be changed. Now, it is a valid page reference. 6. If any page fault is found, restart these steps from starting.   **MS - Sketch [02],Explanation[04]** |
|  |  | **Unit No. 05** |
|  |  | **Q.3: All Questions are compulsory** |
|  | a) | Write short note on the following terms:   * Direct File Access method   Direct Access –  Another method is direct access method also known as relative access method. A filed-length logical record that allows the program to read and write record rapidly. in no particular order. The direct access is based on the disk model of a file since disk allows random access to any file block. For direct access, the file is viewed as a numbered sequence of block or record. Thus, we may read block 14 then block 59, and then we can write block 17. There is no restriction on the order of reading and writing for a direct access file. A block number provided by the user to the operating system is normally a relative block number, the first relative block of the file is 0 and then 1 and so on.  **MS –Explanation of eac method [1 x 4**] |
|  | b) | Draw and Explain the interrupt – driven I/O cycle with Polling and Interrupt concepts.    **MS –Sketch[02], Explanation [04]**  **OR**  What are file attributes and Explain the minimal set of requirements needed for the file operations?  Attributes of the File  **1.Name**  Every file carries a name by which the file is recognized in the file system. One directory cannot have two files with the same name.  **2.Identifier**  Along with the name, Each File has its own extension which identifies the type of the file. For example, a text file has the extension **.txt,** A video file can have the extension **.mp4.**  **3.Type**  In a File System, the Files are classified in different types such as video files, audio files, text files, executable files, etc.  **4.Location**  In the File System, there are several locations on which, the files can be stored. Each file carries its location as its attribute.  **5.Size**  The Size of the File is one of its most important attribute. By size of the file, we mean the number of bytes acquired by the file in the memory.  **6.Protection**  The Admin of the computer may want the different protections for the different files. Therefore each file carries its own set of permissions to the different group of Users.  **7.Time and Date**  Every file carries a time stamp which contains the time and date on which the file is last modified.  **.Create operation:**  This operation is used to create a file in the file system. It is the most widely used operation performed on the file system. To create a new file of a particular type the associated application program calls the file system. This file system allocates space to the file. As the file system knows the format of directory structure, so entry of this new file is made into the appropriate directory.  **2. Open operation:**  This operation is the common operation performed on the file. Once the file is created, it must be opened before performing the file processing operations. When the user wants to open a file, it provides a file name to open the particular file in the file system. It tells the operating system to invoke the open system call and passes the file name to the file system.  **3. Write operation:**  This operation is used to write the information into a file. A system call write is issued that specifies the name of the file and the length of the data has to be written to the file. Whenever the file length is increased by specified value and the file pointer is repositioned after the last byte written.  **4. Read operation:**  This operation reads the contents from a file. A Read pointer is maintained by the OS, pointing to the position up to which the data has been read.  **5. Re-position or Seek operation:**  The seek system call re-positions the file pointers from the current position to a specific place in the file i.e. forward or backward depending upon the user's requirement. This operation is generally performed with those file management systems that support direct access files.  **6. Delete operation:**  Deleting the file will not only delete all the data stored inside the file it is also used so that disk space occupied by it is freed. In order to delete the specified file the directory is searched. When the directory entry is located, all the associated file space and the directory entry is released.  **7. Truncate operation:**  Truncating is simply deleting the file except deleting attributes. The file is not completely deleted although the information stored inside the file gets replaced.  **8. Close operation:**  When the processing of the file is complete, it should be closed so that all the changes made permanent and all the resources occupied should be released. On closing it deallocates all the internal descriptors that were created when the file was opened.  **9. Append operation:**  This operation adds data to the end of the file.  **10. Rename operation:**  This operation is used to rename the existing file.    **MS –File attributes & File operations [06]** |
|  |  | **Unit No. 06** |
|  |  | **Q.4: Attempt any two out of three questions** |
|  | a) | With a neat diagram explain Architecture of the UNIX operating system **?**  **1. The Kernel**   * The Kernel is the heart of the Operating System. * It interface between Shell and Hardware. * It performs Low Level Task. * Eg: Device Management, Memory Management etc.   **2. The Shell**   * The Shell is a collection of UNIX Commands. * The Shell acts as an interface between the user and the kernel. * The Shell is a Command Line Interpreter(CLI)–Translates the commands * provided by the user and converts it into a language that is understood by the * Kernel. * Only one Kernel running on the system, but several shells in action-one for each user who is logged in. * Eg: C Shell, Bourne Shell, Korn Shell etc.   **3. Files and Processes**   * A File is an array of bytes and can contain anything. * All the data in UNIX is organized into files. * A Process is a program file under execution. * Files and Processes belongs to a separate hierarchical structure.   **4. System Calls**   * UNIX is written in C. * Thousands of commands in the system uses a handful of functions called System Calls to communicate with the kernel.   **Architecture with Explanation [05]** |
|  | b) | Write a note on Network based Operating Systems ?  Network Operating Systems extend the facilities and services provided by computer operating systems to support a set of computers, connected by a network. The environment managed by a network operating system consists of an interconnected group of machines that are loosely connected  . By loosely connected, we mean that such computers possess no hardware connections at the CPU – memory bus level, but are connected by external interfaces that run under the control of software. Each computer in this group run an autonomous operating system, yet cooperate with each other to allow a variety of facilities including  file sharing, data sharing, peripheral sharing, remote execution and  cooperative computation  . Network operating systems are autonomous operating systems that support such cooperation. The group of machines comprising the management domain of the network operating system is called a  distributed system  . A close cousin of the network operating system is the distributed operating system  . A distributed operating system is an extension of the network operating system that supports even higher levels of cooperation and integration of the machines on the network  **Explanation [05]** |
|  | c) | Explain System calls for the file system in Unix?  It starts with system calls for accessing existing files, such as open, read, write, /seek, and close, then presents system calls to create new files, namely, creat and mknod. It then examines the system calls that manipulate the inode or that maneuver through the file system: chdir, chroot, chown, chmod, stat, and fstat. It investigates more advanced system calls: pipe and dup are important for the implementation of pipes in the shell; mount and unmount extend the file system tree visible to users; link and unlink change the structure of the file system hierarchy. Then, it presents the notion of file system abstractions, allowing the support of various file systems as long as they conform to standard interfaces. Figure 4.1 shows the relationship between the system calls and the algorithms.  It classifies the system calls into several categories, although some system calls appear in more than one category:   * System calls that return file descriptors for use in other system calls * System calls that use the namei algorithm to parse a path name * System calls that assign and free inodes, using algorithms ialloc and ifree * System calls that set or change the attributes of a file System calls that do I/O to and from a process, using algorithms alloc, free, and the buffer allocation algorithms * System calls that change the structure of the file system * System calls that allow a process to change its view of the file system tree   **Explanation [05]** |