PES University, Bangalore

(Established under Karnataka Act No. 16 of 2013)

UE18CS302

NOVEMBER 2020: IN SEMESTER ASSESSMENT B Tech 5 SEMESTER

TEST-2

UE18CS302 (4 Credits) – OPERATING SYSTEMS

Time: 80 Minutes **Answer All Questions** **Max Marks: 40**

1.	a)	Consider a system having a file system with a multilevel inode data structure to track data blocks of a file and secondary storage of size 2 TB, with 512-byte sized blocks. The inode has 64 bytes of space available to store pointers to data blocks, including a single indirect block, a double indirect block, and several direct blocks. (i) What is the largest possible file that can be supported with this design? (ii) If the same file system supports an additional triple-indirect pointer, what is the largest file that can be supported? **Note**: Show the answer as an expression and calculate the final numeric value for the largest possible file **(Give proportional marks for the steps, expression and the largest possible file size)** **Ref: Section 11.4.3, Combined Scheme** (i) Number of data blocks = 2^41/2^9 = 2^32, so 32 bits or 4 bytes are required to store the number of a data block. Number of data block pointers in the inode = 64/4 = 16, of which 14 are direct blocks. The single indirect block stores pointers to 512/4 = 128 data blocks. The double indirect block points to 128 single indirect blocks, which in turn point to 128 data blocks each. So, the total number of data blocks in a file can be 14 + 128 + 128*128 = 16526, and the maximum file size is 16526*512 bytes i.e ~8 MB (ii) 13 + 128 + 128*128 + 128*128*128 = 2113677*512 bytes i.e ~1 GB	4 (3 +1)
	b)	(i) Explain how the VFS layer allows an operating system to support multiple types of file systems easily. **Ref: Section 11.2.3** VFS introduces a layer of indirection in the file system implementation. In many ways, it is similar to object-oriented programming techniques. System calls can be made generically (independent of file system type). Each file system type provides its function calls and data structures to the VFS layer. A system call is translated into the proper specific functions for the target file system at the VFS layer. The calling program has no file-system-specific code, and the upper levels of the system call structures likewise are file system-independent. The translation at the VFS layer turns these generic calls into file-system-specific operations. (ii) What is the advantage of contiguous-allocation technique for allocating disk blocks and what is the benefit you get when this technique is combined with indexed allocation? **Ref: Section 11.4.1 & 11.4.4** Contiguous-allocation is fast if file is usually accessed sequentially and if file is relatively small. It requires only one access to get a disk block Combining both techniques we can use contiguous allocation for small files and automatically switch to an indexed allocation if the file grows large.	4 (2+2)
	c)	Under what circumstance would RAID 1 achieve better performance for read requests than RAID 0? **Ref: Section 12.7.3** RAID Level 1 organization could achieve better performance for read requests. When a read operation is performed, a RAID Level 1 system can decide which of the two copies of the block should be accessed to satisfy the request. This choice could be based on the current location of	2

		the disk head and could therefore result in performance optimizations by choosing a disk head that is closer to the target data.	
2.	a)	Consider a disk with 512 cylinders, and the disk is currently at cylinder 110 (and has previously just processed a request for cylinder 105) and the disk queue contains read/write requests for sectors on cylinders 84, 302, 103, 96, 407 and 113. What is the total head movement to satisfy the requests in the queue using (i) **LOOK** scheduling (ii)**C-LOOK** scheduling? In each case, show a diagram or the steps leading to the result. **Ref: Section 12.4.5** **1 Mark for diagram, 1 Mark for the steps and 1 Mark for the final answer** LOOK 110-113-302-407-103-96-84 (3+189+105+304+7+12) = 620 cylinders C-LOOK 110-113-302-407-84-96-103 (3+189+105+323+12+7) = 639 cylinders	6 (3+3)
	b)	Highlight the fundamental problem that exists in each of these disk scheduling algorithms: FCFS, SSTF, SCAN and C-SCAN. **Ref: Section 12.4** FCFS is fair but has large head movement and generally provides the slowest service SSTF can cause starvation SCAN moves the disk arm across the full width of the disk instead of moving the arm only as far as the final request in each direction. When the head reverses direction, heavier density of requests at the other end of the disk have to wait the longest. C-SCAN moves the disk arm across the full width of the disk. More seek movements are caused in C-SCAN compared to SCAN Algorithm. Even if there are no requests left to be serviced the head will still travel to the end of the disk unlike SCAN algorithm.	4
3.	a)	Suppose you have an Operating system with the ring-protection scheme like in MULTICS. If you were to implement the system calls of this system and store them in a segment associated with ring 0, (i) what should be the values stored in the ring field of the segment descriptor? (ii) What should you do to allow a process executing in a higher-numbered ring to invoke a procedure in ring 0? **Ref: Section 14.3.3** The ring should be associated with an access bracket (b1, b2), a limit value b3, and a set of designated entry points. The processes that are allowed to invoke any code stored in segment 0 in an unconstrained manner are those processes that are currently executing in ring i where $b1 \le i \le b2$. Any other process executing within ring $b2 < i \le b3$ is allowed to invoke only those functions that are designated entry points. This implies that we should have b1=0 and set b2 to be the highest ring number that comprises of system code that is allowed to invoke the code in segment 0 in an unconstrained fashion. We should also store only the system call functions as designated entry points and we should set b3 to be the ring number associated with user code so that user code can invoke the system calls.	5 (3 +2)
	b)	(i) What are the main differences between capability lists and access lists? **Ref: Section 14.5** An access list is a list for each object consisting of the domains with a nonempty set of access rights for that object. A capability list is a list of objects and the operations allowed on those objects for each domain. (ii) Capability lists are usually kept within the address space of the user. How does the system ensure that the user cannot modify the contents of the list? A capability list is considered a “protected object” and is accessed only indirectly by the user. The operating system ensures the user cannot access the capability list directly	3 (2+1)
	c)	Explain how buffer-overflow attacks can be avoided by adopting a better program-ming methodology and/or by using special hardware support.	2

		Ref: Section 15.2.4 One form of hardware support that guarantees that a buffer-overflow attack does not take place is to prevent the execution of code that is located in the stack segment of a process's address space. The buffer-overflow attacks are performed by overflowing the buffer on a stack frame, overwriting the return address of the function, thereby jumping to another portion of the stack frame that contains malicious executable code, which had been placed there as a result of the buffer overflow. By preventing the execution of code from the stack segment, this problem is eliminated. Approaches that use a better programming methodology are typically built around the use of bounds-checking to guard against buffer-overflows. Buffer overflows do not occur in languages like Java where every array access is guaranteed to be within bounds through a software check. Such approaches require no hardware support butresult in runtime costs associated with performing bounds-checking.	
4.	a)	Explain with a diagram the life cycle of an I/O request by clearly showing all the steps how the operating system connects an application request to the device hardware for a blocking read() system call. **2 Marks for diagram, 3 Marks for the steps** Galvin 9th Edition 2016 India Edition, Chapter 13, **Sec 13.5, Fig 13.13** and the 10 steps in Page 600-601	5
	b)	(i) How is the NTFS namespace organized in Windows? **Ref: Section 17.5.1.1** The NTFS namespace is organized as a hierarchy of directories where each directory uses a B+ tree data structure to store an index of the file names in that directory. The index root of a directory contains the top level of the B+ tree. Each entry in the directory contains the name and file reference of the file as well as the update timestamp and file size (ii) How does NTFS handle data structures and how does NTFS recover from a system crash? **Ref: Section 17.5.2** In NTFS, all file-system data structure updates are performed inside transactions. Before a data structure is altered, the transaction writes a log record that contains redo and undo information. A commit record is written to the log after a transaction has succeeded. After a crash the file system can be restored to a consistent state by processing the log records, first redoing operations for committed transactions and undoing operations for transactions that did not successfully commit.	3 (1+2)
	c)	Explain how Direct Memory Access (DMA) technique improves disk I/O performance **Ref: Section 13.2.3** With DMA, the CPU can process other tasks while data transfer is being performed. The transfer of data is first initiated by the CPU. During the transfer of data between the DMA channel and the I/O device, the CPU performs other tasks. When the data transfer is completed, the CPU receives an interrupt request from the DMA controller.	2