

### CS703 Final Term 2015 Paper Solution

#### Question 1: Solve the following Mcqs.

- 1) In page table entries the \_\_\_\_\_ bits control which operations are allowed.
  - **Protection**
  - Valid
  - Referenced
  - Modified
- 2) The working set of process is used to model the dynamic \_\_\_\_\_ of its memory usage.
  - Identity
  - Attribute
  - **Locality**
  - State
- 3) We can use \_\_\_\_\_ memory to allow processes to data using direct memory reference.
  - **Shared**
  - Primary
  - Secondary
  - Main
- 4) Fast file system used a minimum of \_\_\_\_\_ size disk block.
  - 8974
  - 4089
  - 3095
  - **4096**
- 5) Hardware layer of the I/O system perform \_\_\_\_\_ operation.
  - **I/O**
  - Naming
  - Setup register
  - Spooling
- 6) In direct I/O device controller's register and internal memory is accessible via special \_\_\_\_\_ in the assembly language instruction.
  - Driver
  - Block
  - Code
  - **Instruction.**
- 7) Device independent I/O software usually interfaces to device drivers through a standard \_\_\_\_\_.
  - **Interface**
  - Driver
  - Address
  - Buffer.
- 8) Interrupt handler is more \_\_\_\_\_ than exception.
  - **Complex**
  - Simple
  - Weak
  - Strong
- 9) Each timer contains a \_\_\_\_\_ that indicates how far in the future the timer should expire.
  - **Field**
  - ID
  - Row
  - Column
- 10) In loadable kernel module sections of kernel code can be compiled loaded, and unloaded independently of the rest of the \_\_\_\_\_.
  - **Kernel**
  - Code
  - Compilation
  - Memory

- 11) \_\_\_\_\_ is when the system spends most of its time servicing page faults, little time doing useful work.
- **Thrashing**
  - Segment
  - Block
  - Chunk
- 12) Module Management Supports loading modules into memory and letting them talk to the rest of the \_\_\_\_\_.
- **kernel**
  - Code
  - Segment
  - Page
- 13) \_\_\_\_\_ FS contains an index to a particular field of each record in a file
- **Indexed access**
  - Direct access
  - Sequential access
  - Record access
- 14) Device controller converts \_\_\_\_\_ bit stream to block of bytes.
- Parallel
  - **Serial**
  - Both
  - None
- 15) Using Fast File system (FFS) multiple file systems with different \_\_\_\_\_ sizes can co-reside.
- **Block**
  - Chunk
  - Segment
  - None
- 16) The purpose of a \_\_\_\_\_ system is to prevent accidental or intentional misuse of a system.
- **Protection**
  - Memory Management
  - Paging
  - Thrashing
- 17) \_\_\_\_\_ is placed on the number of currently active dynamic timers.
- 0
  - 1
  - 2
  - **No limit**
- 18) In \_\_\_\_\_ device controller's registers and internal memory are directly mapped into the processor's address space.
- **Memory mapped I/O**
  - I/O device
  - Polled I/O
  - DMA
- 19) In page table entries the ..... bits control which operations are allowed.
- Protection
  - Valid
  - Referenced
  - Modified.
- 20) In demand paging pages are only brought into \_\_\_\_\_ when they only referenced only the code data that is needed by process needs to be loaded.
- Cache
  - Page Table
  - Segment Table
  - **Main Memory**

21) Each Process has a separate pool of pages a page fault in one process can only replace one of this process \_\_\_\_\_

- Pages
- Segments
- ID
- **Frames**

22) While conservatively moving a file there is a rule reset old pointer to \_\_\_\_\_ before a new pointer has been set.

- Attribute
- Value
- Node
- **Object**

23) In virtual file system \_\_\_\_\_ object represents a specific directory entry.

- Superblock
- Inode
- File
- **Dentry**

24) \_\_\_\_\_ timers may be dynamically created and destroyed.

- I/O
- **Dynamic**
- State
- Memory

25) \_\_\_\_\_ modules allow a linux system to be set up with a standard minimal kernel without any extra device driver built in

- Code
- **Kernel**
- Memory
- Compilation

26) IN \_\_\_\_\_ security mechanism system know who the user & user enters a name and password or other info.

- Access control
- Biometric
- Security
- Physical security

### Question 2:

Write down detail at least five file operation and their corresponding UNIX/LINUX system calls.

Unix	NT
create(name)	CreateFile(name, CREATE)
open(name, mode)	CreateFile(name, OPEN)
read(fd, buf, len)	ReadFile(handle, ...)
write(fd, buf, len)	WriteFile(handle, ...)
sync(fd)	FlushFileBuffers(handle, ...)
seek(fd, pos)	SetFilePointer(handle, ...)
close(fd)	CloseHandle(handle, ...)
unlink(name)	DeleteFile(name)
rename(old, new)	CopyFile(name)
	MoveFile(name)

**Question 3:**

What are five major activities of an operating system in regard to file management?

**Answer:**

Operating systems perform the vital function of being the bridge between a computer's hardware and software. They provide an environment where software can be written without the need to cater to the specifics of the underlying hardware, which was necessary in the earlier days of computing. There are several widely-used operating systems which differ from each other in many respects. However, they each perform a number of similar functions including executing basic instructions, either compiled or interpreted; and also managing processes, memory, input and output, storage, network operations, and file and folder/directory operations. There are five major file management functions that an operating system controls.

**Creating and Deleting Files:**

File creation and deletion are fundamental to computer operations. In the former, data can't be stored in an efficient manner unless arranged in some form of file structure. In the latter, permanent storage would quickly fill up if files were not deleted and the space occupied by them reallocated to new files.

**Creating and Deleting Directories:**

As a corollary to the need to store data in files, files themselves need to be arranged in directories or folders in order to allow their efficient storage and retrieval. This is particularly so in the case of personal computers where the user needs to navigate to one or more specific files to access them. Without some form of compartmentalization, this would prove an onerous if not impossible task. Much like file deletion, unnecessary directories or folders need to be removed in order to keep the system uncluttered.

**File Manipulation Instructions:**

Since operating systems allow application software to perform file manipulation using symbolic instructions, the operating system itself needs to have a machine-level instruction set in order to interface with the hardware directly. The application's symbolic instructions need to be translated into the machine-level instructions either by an interpreter or by compiling the application code. The operating system contains provisions to manage this machine-level file manipulation.

**Mapping to Permanent Storage:**

Operating systems need to be able to map files and folders to their physical location on permanent storage in order to be able to store and retrieve them. This will be recorded in some form of disk directory which varies according to the file system or systems that the operating system uses. The operating system will include a mechanism to locate the separate file segments where it has divided a file.

**Backing Up Files:**

Files represent a considerable investment in time, intellectual effort and often money as well, thus their loss can have a severe impact. Computer's permanent storage devices generally contain a number of mechanical devices which can fail, and the storage media itself may degrade. A function of operating systems is to obviate the risk of data loss by backing files up on additional secure and stable media in a redundant system.

**Question 4:**

How block devices are different from character devices?

**Answer:**

- Character Device Vs. Block Device
  - ✓ A Character ('c') Device is one with which the Driver communicates by sending and receiving single characters (bytes, octets).
  - ✓ A Block ('b') Device is one with which the Driver communicates by sending entire blocks of data.
  - ✓ Examples for Character Devices: serial ports, parallel ports, sound cards.
  - ✓ Examples for Block Devices: hard disks, USB cameras, Disk-On-Key.
  - ✓ For the user, the type of the Device (block or character) does not matter - you just care that this is a hard disk partition or a sound card.
  - ✓ Driver programmers, however, do care, but that's beyond our scope.

Block devices are those that can be read by blocks even though some programs may want to read char by char. These include tapes and hard drives. Character devices are those only read char by char such as keyboards and serial ports.

A block device is one that is designed to operate in terms of the block I/O supported by Digital UNIX. It is accessed through the buffer cache. A block device has an associated block device driver that performs I/O by using file system block-sized buffers from a buffer cache supplied by the kernel. Block device drivers are particularly well-suited for disk drives, the most common block devices.

A character device is any device that can have streams of characters read from or written to it. A character device has a character device driver associated with it that can be used for a device such as a line printer that handles one character at a time. However, character drivers are not limited to performing I/O a single character at a time (despite the name "character" driver). For example, tape drivers frequently perform I/O in 10K chunks. A character device driver can also be used where it is necessary to copy data directly to or from a user process. Because of their flexibility in handling I/O, many drivers are character drivers. Line printers, interactive terminals, and graphics displays are examples of devices that require character device drivers.

**Question 5:**

Consider the I/O scenario in single user PC.

- a. A mouse used with a graphical user interface
- b. A tape drive on a multitasking operating system (assume no device preallocation is available)
- c. A disk drive containing user files
- d. A graphics card with direct bus connection, accessible through memory-mapped I/O

For each of these I/O scenarios, would you design the operating system to use buffering, spooling, caching, or a combination? Would you use polled I/O, or interrupt-driven I/O? Give reasons for your choices.

**Answer:**

- a) A mouse used with a graphical user interface Buffering may be needed to record mouse movement during times when higher-priority operations are taking place. Spooling and caching are inappropriate. Interrupt driven I/O is most appropriate.

- b) A tape drive on a multitasking operating system (assume no device pre-allocation is available). Buffering may be needed to manage throughput difference between the tape drive and the source or destination of the I/O, Caching can be used to hold copies of data that resides on the tape, for faster access. Spooling could be used to stage data to the device when multiple users desire to read from or write to it. Interrupt driven I/O is likely to allow the best performance.
- c) A disk drive containing user files. Buffering can be used to hold data while in transit from user space to the disk, and vice versa. Caching can be used to hold disk-resident data for improved performance. Spooling is not necessary because disks are shared-access devices. Interrupt-driven I/O is best for devices such as disks that transfer data at slow rates.
- d) A graphics card with direct bus connection, accessible through memory-mapped I/O. Buffering may be needed to control multiple access and for performance (double-buffering can be used to hold the next screen image while displaying the current one). Caching and spooling are not necessary due to the fast and shared-access natures of the device. Polling and interrupts are only useful for input and for I/O completion detection, neither of which is needed for a memory-mapped device.

**Question 6:**

Consider a system that sport 5000 users, suppose that you want to allow 4990 of these users to be able to access one file.

- How would you specify this protection scheme in UNIX?
- Could you suggest another scheme that can be used more affectively for this purpose then the scheme provided by UNIX?

**Answer:**

- You can add each of the 4,990 users to a group, and then use the allow access to the file by that group. Alternately, on some Unix systems, you can create an Access Control List (ACL) and assign access according to that list.
- In this case, it would be easier to be able to specify a group of users that cannot access the file and allow any user not on the list to have access. Another alternative is to password the file and give the password to users that you deem need access to the file.

**Question 7:**

Write the functionality of two lines

- ✓ `put(blk, address)`
- ✓ `get(address),,,,blk`

**Answer:**

- ✓ `put(blk, address)` : writes data in blk on disk at address
- ✓ `get(address) -> blk` : returns blk at given disk address

**Question 8:**

If there is no page frame free in page table. You want to page replacement. What step you taken for page replacement.

**Answer:**

- When there are no available free frames to handle a fault we must find a page to replace.
- How it does this is determined by the page replacement algorithm?
- The goal of the replacement algorithm is to reduce the fault rate by selecting the best victim page to remove.
- Write the victim frame to disk. Change all related page tables to indicate that this page is no longer in memory.

**Question 9:**

Briefly explain major security issues.

**Answer:**

- Isolation
  - ✓ Separate processes execute in separate memory space
  - ✓ Process can only manipulate allocated pages
- Authentication
  - ✓ Who can access the system. Involves proving identities to the system
- Access control
  - ✓ When can process create or access a file?
  - ✓ Create or read/write to socket?
  - ✓ Make a specific system call?
- Protection problem
  - ✓ Ensure that each object is accessed correctly and only by those processes that are allowed to do so
- Comparison between different operating systems
  - ✓ Compare protection models: which model supports least privilege most effectively?
  - ✓ Which system best enforces its protection model?

**Question 10:**

How authentication can achieve by biometric?

**Answer:**

Authentication of a person based on a physiological or behavioral characteristic. A number of biometric methods have been introduced over the years, but few have gained wide acceptance.

- Signature dynamics: Based on an individual's signature, but considered unforgeable because what is recorded isn't the final image but how it is produced -- i.e., differences in pressure and writing speed at various points in the signature.
- Typing patterns: Similar to signature dynamics but extended to the keyboard, recognizing not just a password that is typed in but the intervals between characters and the overall speeds and pattern. This is akin to the way World War II intelligence analysts could recognize a specific covert agent's radio transmissions by his "hand" -- the way he used the telegraph key.

- Eye scans: This favorite of spy movies and novels presents its own problems. The hardware is expensive and specialized, and using it is slow and inconvenient and may make users uneasy.
- In fact, two parts of the eye can be scanned, using different technologies: the retina and the iris.
- Fingerprint recognition: Everyone knows fingerprints are unique. They are also readily accessible and require little physical space either for the reading hardware or the stored data.
- Hand or palm geometry: We're used to fingerprints but seldom think of an entire hand as an individual identifier. This method relies on devices that measure the length and angles of individual fingers. Although more user-friendly than retinal scans, it's still cumbersome.
- Voice recognition: This is different from speech recognition. The idea is to verify the individual speaker against a stored voice pattern, not to understand what is being said.
- Facial recognition: Uses distinctive facial features, including upper outlines of eye sockets, areas around cheekbones, the sides of the mouth and the location of the nose and eyes. Most technologies avoid areas of the face near the hairline so that hairstyle changes won't affect recognition.

**Question 11:**

Describe in details file sharing system. 5 marks

**Answer:**

File sharing is the public or private sharing of computer data or space in a network with various levels of access privilege. While files can easily be shared outside a network (for example, simply by handing or mailing someone your file on a diskette), the term file sharing almost always means sharing files in a network, even if in a small local area network. File sharing allows a number of people to use the same file or file by some combination of being able to read or view it, write to or modify it, copy it, or print it. Typically, a file sharing system has one or more administrators. Users may all have the same or may have different levels of access privilege. File sharing can also mean having an allocated amount of personal file storage in a common file system.

File sharing can be done using several methods. The most common techniques for file storage, distribution and transmission include the following:

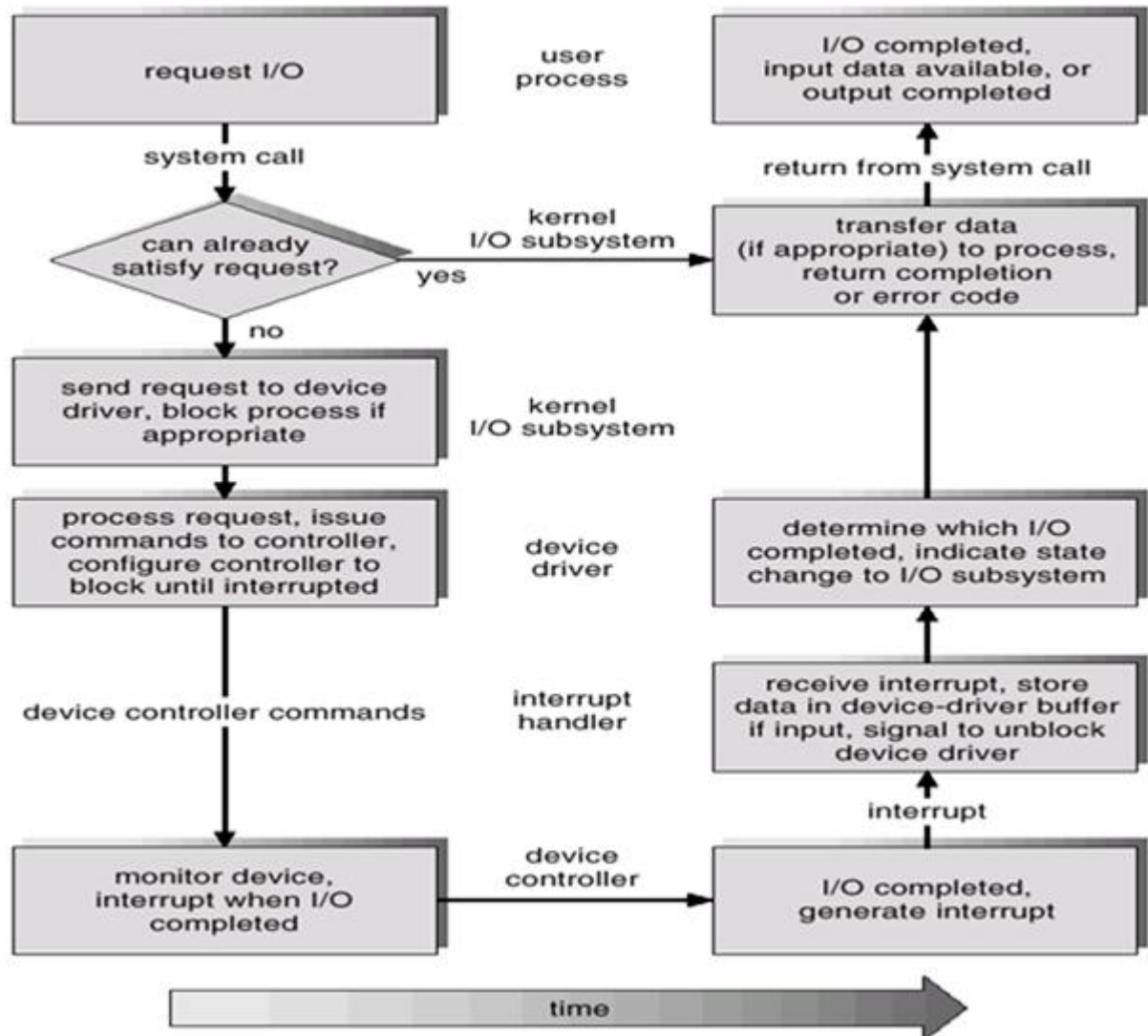
- Removable storage devices
- Centralized file hosting server installations on networks
- World Wide Web-oriented hyperlinked documents
- Distributed peer-to-peer networks



**Question 12:**

Draw diagram of complete life cycle of an I/O request.

**Answer:**



**Question 13:**

What are the blocking I/O and three non-blocking I/O. Why non-blocking I/O is useful for busy wait 10 marks

**Answer:**

- Blocking - process suspended until I/O completed
  - ✓ Easy to use and understand
  - ✓ Insufficient for some needs
- Non-blocking - I/O call returns as much as available
  - ✓ User interface, data copy (buffered I/O)
  - ✓ Implemented via multi-threading inside the kernel
  - ✓ Returns quickly with count of bytes read or written
- Asynchronous - process runs while I/O executes
  - ✓ Difficult to use
  - ✓ I/O subsystem signals process when I/O completed

**Question 14:**

When a program enters into an infinite loop and never returns control back to CPU. Explain how timer interrupts help to terminate this condition? 10 Marks

**Answer:**

An infinite loop (also known as an endless loop or unproductive loop) is a sequence of instructions in a computer program which loops endlessly, either due to the loop having no terminating condition, having one that can never be met, or one that causes the loop to start over. In older operating systems with cooperative multitasking, infinite loops normally caused the entire system to become unresponsive. With the now-prevalent preemptive multitasking model, infinite loops usually cause the program to consume all available processor time, but can usually be terminated by the user. Busy wait loops are also sometimes called "infinite loops". One possible cause of a computer "freezing" is an infinite loop; others include thrashing, deadlock, and access violations.

So long as the system is responsive, infinite loops can often be interrupted by sending a signal to the process (such as SIGINT in Unix), or an interrupt to the processor, causing the current process to be aborted. This can be done in a task manager, in a terminal with the Control-C command, or by using the kill command or system call. However, this does not always work, as the process may not be responding to signals or the processor may be in an uninterruptible state, such as in the Cyrix coma bug (caused by overlapping uninterruptible instructions in an instruction pipeline). In some cases other signals such as SIGKILL can work, as they do not require the process to be responsive, while in other cases the loop cannot be terminated short of system shutdown.

**Question 15:**

Why Rotational latency is not considered in disk scheduling? How FCFS, SSTF, SCAN and C-SCAN are modified for rotational latency optimization? 10 Marks

**Answer:**

Most disks do not export their rotational position information to the host. Even if they did, the time for this information to reach the scheduler would be subject to imprecision and the time consumed by the scheduler is variable, so the rotational position information would become incorrect. Further, the disk requests are usually given in terms of logical block numbers, and the mapping between logical blocks and physical locations is very complex

**Question 16:**

Briefly explain major security issues? 10 Marks

**Answer:**

- Isolation
  - ✓ Separate processes execute in separate memory space
  - ✓ Process can only manipulate allocated pages
- Authentication
  - ✓ Who can access the system. Involves proving identities to the system
- Access control
  - ✓ When can process create or access a file?
  - ✓ Create or read/write to socket?
  - ✓ Make a specific system call?
- Protection problem
  - ✓ Ensure that each object is accessed correctly and only by those processes that are allowed to do so
- Comparison between different operating systems
  - ✓ Compare protection models: which model supports least privilege most effectively?
  - ✓ Which system best enforces its protection model?

**Question 17:**

Assume we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds. Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds?

**Answer:**

$$0.2 \mu\text{sec} = (1 - P) \times 0.1 \mu\text{sec} + (0.3P) \times 8 \text{ millisecc} + (0.7P) \times 20 \text{ millisecc}$$

$$0.1 = -0.1P + 2400 P + 14000 P$$

$$0.1 \approx 16,400 P$$

$$P \approx 0.000006$$

**Question18:**

The electronic component is the device controller may be able to handle multiple devices, what are the main tasks of a device controller?

**Answer:**

- I/O devices have components:

- mechanical component
- electronic component
- The electronic component is the device controller
  - may be able to handle multiple devices
- Controller's tasks
  - convert serial bit stream to block of bytes
  - perform error correction as necessary
  - make available to main memory

**Question 19:**

What are difficulties with working set? Describe in detail & also provide examples? (10marks)

**Answer:**

- T is magic
  - ✓ what if T too small? Too large?
  - ✓ How did we pick it? Usually "try and see"
  - ✓ Fortunately, system's aren't too sensitive
- What processes should be in the balance set?
  - ✓ Large ones so that they exit faster?
  - ✓ Small ones since more can run at once?
- How do we compute working set for shared pages?

**Question 20:**

In a multiprogramming and time sharing environment, several users share the system simultaneously. This situation in various security problems. What are two such problem. Can we ensure the same degree of security in a time shared machine as we have in a dedicated machine. Explain ur answer.

**Answer:**

- Stealing or copying a user's files; writing over another program's (belonging to another user or to the OS) area in memory; using system resources (CPU, disk space) without proper accounting; causing the printer to mix output by sending data while some other user's file is printing.
- Probably not, since any protection scheme devised by a human can also be broken -- and the more complex the scheme is, the more difficult it is to be confident of its correct implementation.

Q11. Consider a virtual memory in which a process does not have enough pages therefore the page fault rate is very high. This leads to low CPU and I/O devices utilization, whereas disk utilization is high. Write down the name of the activity that leads to said problems and how can you overcome these problems. Threshing

Q: disk scheduling, other than FCFS is useful in case of single user environment

i Node and pointer 10 marks

WWW.VUMULTAN.COM