**FIT2100 OPERATING SYSTEMS SAMPLE EXAM**

**http://www.tommysailing.com/content/COMP3231\_80Questions.pdf**

**SHORT ANSWER**

**QUESTION 1**

*Describe the two general roles of an operating system, and elaborate why these roles are important.*

A computer is a set of resources for the movement, storage, and processing of data. An operating system is a program that acts as an interface between applications and hardware. It provides a layer of abstraction for software to run on, without knowledge of hardware-specific details. This is required by an OS to reduce complication for software developers and basic users.  
The OS is the software responsible for managing resources and controlling the computer’s basic functions. The OS manages resources and controls the execution of programs by providing instructions to the processor and directing it in terms of the use of other system resources and the timing of execution for other programs. This ensure progress and efficient usage of computing resources.

The three main objectives of an operating system are to provide convenience, efficiency, and the ability to evolve. An OS must make applications and programs simple to execute for a user while being responsible for the movement, storage, and processing of data. It must hide the complexity of hardware from the user. A major OS will evolve over time for a number of reasons such as hardware upgrades, new types of hardware, new services and fixes. An OS must be able to evolve to account for these changes, otherwise they may become obsolete.

**QUESTION 2**

*Describe the three-state process model, describe what transitions are valid between the three states, and describe an event that might cause such a transition.*

The three-state process model describes the state of a process in terms of READY, RUNNING and BLOCKED.  
READY means that a process that is queuing and is prepared to be executed when given the opportunity.  
RUNNING means that the process is currently being executed.  
BLOCKED means that a process cannot execute until some event occurs, such as the completion of an I/O operation.

State transitions include:  
RUNNING to READY: Time slice process management dictates when a process is ready.  
BLOCKED to READY: The process has the required input.  
READY to RUNNING: An appropriate scheduler decides it should run.  
RUNNING to BLOCKED: The process requires some input to run.

The system must initially go into the READY state before they enter a RUNNING state.  
Processes normally leave the system from the RUNNING state.

**QUESTION 3**

*What is a process? What are the attributes of a process?*

A process can be defined as:  
- A program in execution.  
- An instance of a running program.  
- The entity that can be assigned to, and executed on, a processor.  
- A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources.

The attributes of a process include the identifier, state, priority, program counter, memory pointers, context data, I/O status information and accounting information.

**QUESTION 4**

*What is the function of the ready queue?*

The ready queue is a queue of processes that is ready for execution in the three-state process model. A process will enter the READY queue when it had received its required inputs and is ready to be executed. The queue establishes a scheduling mechanism for the processes order of execution.

**QUESTION 5**

*What is the relationship (or differences) between threads and process?*

Each process provides the resources needed to execute a program. A process has a virtual address space, executable code, open handles to system objects, a security context, a unique process identifier, environment variables, a priority class, min and max working set sizes, and at least one thread of execution. Each process is started with a single thread but additional threads can be created.

A thread of execution is the smallest sequence of programmer instructions that can be managed independently by a scheduler. It is an entity within a process that can be scheduled for execution.

**QUESTION 6**

*Name the advantages and disadvantages between threads and processes?*

An advantage of threads is that each thread can also have its own global variables, stack, program counter and allotted registers. Threads can be executed concurrently. A process can be made up of multiple threads of execution that execute instructions concurrently. A disadvantage is that threads are best used for smaller tasks.

An advantage of processes is that they can be used for heavyweight tasks such as the execution of applications. A disadvantage of processes is that they run in main memory, and so disappear if the machine is rebooted. Multiple process can be executed in parallel in a multiprocessor system, but on a uni-processor system, process scheduling must be applied, yielding the illusion of concurrency. But thus, not achieving true parallelism

**QUESTION 7**

*Describe the process control block and the details of information it maintains?*

A process control block is a data structure which contains the process elements (i.e. the information needed to manage a particular process).  
Process elements include:  
- Program code: May be shared with other processes that are executing the same program.  
- A set of data associated with the program code.  
The process control block is created and managed by the OS and allows for a process to be resumed later upon interruption of a running process.

**QUESTION 8**

*Describe the sequence of steps that occur when a timer interrupt occurs that eventually results in a context switch to another application.*

1. System signals an interrupt.
2. The current instruction is completed.
3. The contents of the internal registers are stored in the stack, including the program counter to indicate where the process is up to.
4. The scheduler executes the next process.
5. The internal register contents are popped from the stack.
6. The original process is continued from the program counter.

**QUESTION 9**

*What is a race condition? Give an example.*

A race condition is a situation that occurs in concurrency in which multiple threads of processes read and write a shared data item and the final result depends on the relative timing of their execution. Data inconsistency arises as a result. In order to manage this, mechanisms are required to ensure the orderly execution of cooperating processes.

Example: Two processes simultaneously update a counter. Process A reads the variable, but Process B reads and writes to it. Control is passed back to Process A, who writes to it. This results in both processes being written to at the same time, thus data inconsistency arises.

**QUESTION 10**

*What is a critical region? How do they relate to controlling access to shared resources?*

A critical region (critical section) is a region of code within a process that requires access to shared resources, and that must not be executed while another process is in a corresponding section of code. In order to avoid data inconsistency, deadlocks and starvation, it must be ensured that when one process is executing in its critical section, not other process is allowed in its critical section. Mutual exclusion must be enforced to avoid these issues.

**QUESTION 11**

*What are the three requirements of any solution to the critical sections problem? Why are the requirements needed?*

1. Mutual exclusion must be enforced. This is the requirement that when one process is in a critical section that accesses shared resources, no other process may be in a critical section that accesses any of those shared resources.
2. A process remains inside its critical section for a finite time only. If a critical section was stuck an in infinite loop, no other process will be able to access their critical section.
3. A process must not be denied access to a critical section when no other process is using the shared resources. If this is the case, processes may never be able to access their critical section and thus the entire operation is affected.

**QUESTION 12**

*What is the producer consumer problem? Give an example of its occurrence in operating systems.*

Producers and consumers refer to processes in a multi-process synchronization problem.

The producer/consumer model is where:  
- One or more producers are generating data and placing these in a buffer.  
- A single consumer is taking items out of the buffer one at a time.  
- Only one producer or consumer may access the buffer at any one time.  
The producer/consumer problem ensures that the producer cannot add data into a full buffer, and the consumer cannot remove data from an empty buffer.  
Concurrency problems arise when we need to keep track of the number of items in the buffer, which has a fixed limit on how many items can be inside at any one time.  
The action of performing buffering on

**QUESTION 13**

*What is a deadlock? What is starvation? How do they differ from each other?*

A deadlock occurs when a set of processes are awaiting an event that can only be triggered by another block process in the set. For example, if one process relies on the result of another process, but the process is stuck in an infinite loop and therefore does not produce a result, a deadlock occurs.

Starvation occurs when a runnable process is overlooked indefinitely by the schedule. In other words, it is never chosen by the schedule although it is ready to proceed.

**QUESTION 14**

*What are the four conditions required for deadlock to occur?*

1. Mutual exclusion: Only one process may use a resource at any one time.
2. Hold-and-wait: A process may hold allocated resources while awaiting assignment of others.
3. No pre-emption: No resource can be forcibly removed form a process holding it.
4. Circular wait: A closed chain of processes exists, such that each process holds at least one resource needed by the next process in the chain.

**QUESTION 15**

*Describe general strategies for dealing with deadlocks.*

Ignorance: If deadlocks are not liable to happen, the effort required to deal with them outweighs the problem of deadlocks occurring.

Prevention: This involves adopting a policy that eliminates one of the conditions. This works by limiting resources by imposing restrictions on processes.

Avoidance: This involves making the appropriate dynamic choices based on the current state of resource allocation. Works by settings safe states in which process completion is always guaranteed. There are two main approaches:  
1. Do not start a process if its demands might lead to deadlock.  
2. Do not grant an incremental resource request to a process if this allocation might lead to a deadlock.

Detection: Keep a log of resource ownership and requests. If no progress is made, recover from the deadlock by using pre-emption, rollback, or killing the process stuck in the deadlock cycle.

**QUESTION 16**

*For single unit resources, we can model resource allocation and requests as a directed graph connecting processes and resources. Given such a graph, what is involved in deadlock detection?*

Deadlocks can be detected in such a graph by finding closed loops in the graph, i.e. where two or more processes requesting resources are held by other processes.

**QUESTION 17**

*Assuming the operating system detects the system is deadlocked, what can the operating system do to recover from the deadlock?*

The OS can recover from a deadlock by:  
- Terminating the deadlocked process.  
- Rolling back to a previously saved version.  
- Pre-emption: Forcibly steal a resource from another process.

**QUESTION 18**

*What must the banker’s algorithm know prior in order to prevent a deadlock?*

The banker’s algorithm must calculate whether giving a process a resource will lead to a safe state or not. It must know how many resources the process may be granted at any one time, and how many resources the OS is able to give.   
To check if the state is safe, the algorithm checks to see if enough resources exist to satisfy a process. If so, the resources loaned are assumed to be repaid.

A safe state is one in which the is at least one sequence of resource allocations to processes that does not result in a deadlock. An unsafe state is a state that is not safe.

**QUESTION 19**

*Describe the general strategy behind deadlock prevention, and give an example of a practical deadlock prevention method.*

Deadlock prevention involves adopting a policy that eliminates one of the conditions that can cause a deadlock.  
Preventing no pre-emption: If a process holding certain resources is denied a further request, that process must release its original resources and request them again. The OS may pre-empt the second process and require it to release its resources to the first process.  
Preventing Circular Wait: Define a linear ordering of resource types.  
Prevent Hold-And-Wait: Adopt of policy of not letting a process wait for one resource while holding another. This can be done either by requiring each process to hold only one resource at a time, or to request all of the resources it needs simultaneously.

**QUESTION 20**

*Explain short term and medium term scheduling policy.*

Short term scheduling is a type of processor scheduling in which the decision as to which available process will be executed next is made by the processor.

Medium term scheduling is a type of processor scheduling which makes the decision to add the number of processes that are partially or fully in main memory. The medium term scheduler removes processes from the memory and is in-charge of handling the swapped out processes.

**QUESTION 21**

*What are four general characteristics of processor scheduling policies?*

Selection function: Determines which process, among ready processes, is selected next for execution. This can be based on priority, resource requirements, or execution characteristics of the process.

Decision mode: Specifies the instants in time at which the selection function is exercised. Can be:  
- Non pre-emptive: Once a process is in the running state, it will continue until it terminates or blocks itself for I/O.  
- Pre-emptive: Currently running process may be interrupted and moved to ready state by the OS. May occur when a new process arrives, on an interrupt, or periodically.

Throughput: A measure of the number of processes completed per unit time.

Response time: The amount of time it takes from when a request was submitted until the first time the process is executed.

Overhead: Any combination of excess or indirect computation time, memory, bandwidth, or other resources that are required to perform a specific task.

**QUESTION 22**

*Define turnaround time and normalized turnaround time. Why are these useful for measuring the performance of a scheduling algorithm?*

The turnaround time is the total amount of time it takes to complete a process. That is, the total time taken between the submission of a process for execution and the return of a complete output to the customer/user. Normalized turnaround time is the turnaround time divided by the response time.

**QUESTION 23**

*List and describe the four memory allocation algorithms. Which two of the four are more commonly used in practice?*

First-fit: OS scans the linked list of available memory addresses from the beginning and chooses the first available block that is large enough. This is the simplest, best and fastest method. It is aimed to minimise the amount of searching required, but leads to external fragmentation later on.

Best-fit: OS chooses the block that is the closest in size to the request. This is the worst in terms of performance as it searches through the memory space for the smallest block.

Next-fit: OS scans the linked list of available memory addresses from the location of the last successful memory allocation and chooses the next available block that is large enough. This can produce slightly worse results than first-fit as the largest block of free memory at the end of the memory space is broken up into small fragments. Can cause external fragmentation at the beginning of the memory.

Worst-fit: Traverses through the memory and gives the partitions as large as space as possible to leave usable fragments left over. This is a poor performer as it needs to search the complete list of memory addresses.

**QUESTION 24**

*Describe the difference between external and internal fragmentation. Indicate which of the two are most likely to be an issue on:  
a) A simple memory management machine using static partitioning.*

Static partitioning: Memory is divided into a fixed number of static partitions at system generation time. A process may be loaded into a partition of equal or greater size.

Internal fragmentation can arise as pages may not always end on the page boundary, thus causing wasted space.

*b) A similar machine using dynamic partitioning.*

Dynamic partitioning: The partitions of space allocation are of variable length and number. When a process is brought into main memory, it is allocated as much memory as it requires.

Through this, external fragmentation may arise as the varying segment sizes may cause wasted space between segments.

**QUESTION 25**

*What is thrashing? How might it be deceased? How might one recover from it once detected?*

Thrashing occurs when a computers virtual memory is in a constant state of paging. This rapid exchanging of data from memory to disk causes the performance of the computer to decrease or collapse. In other words, thrashing occurs due to resources being exhausted and too limited to perform needed operations, causing the number of page faults to increase. When this happens, the operating system attempts to steal resources from other processes, thus causing the computer to crash.

This may be prevented by only allowing the system to run a specified number of programs at a time, writing programs that use memory more efficiently, or adding extra RAM to the system.

Thrashing may be detected by monitoring the swapping levels compared to real CPU utilization. To recover from this, the number of processes currently in the running/ready queue must be reduced by suspending processes (push them into the blocked/waiting queue) so that the pressure on physical memory is reduced.

**QUESTION 26**

*Enumerate some pros and cons for increasing the page size.*

Pros:   
- Reduce total page table size which frees some memory.  
- Increases translation lookaside buffer coverage.  
- Increase swapping I/O throughput.

Cons:  
- Increases page fault latency as there are more pages to search through.  
- Increase internal fragmentation of pages.

Page sizes affect the working set size as more memory can be wasted on irrelevant data, which increases the working set size for no reason. Smaller page sizes more accurately reflect the current memory usage.

**QUESTION 27**

*Describe two virtual memory page fetch policies. Which is less common in practice? Why?*

Demand paging: Relevant pages are loaded as page faults occur. In other words, it is a type of swapping in which pages of data are not copied from disk to RAM until they are needed.

Pre-paging: A technique in which the virtual memory multitasking environment loads all pages of a process’s working set into a memory before the process is restarted. That is, tries to load pages for processes before they are accessed. This is less common in practice as bandwidth is wasted if the pages loaded are incorrect.

The selection of pages is done based on the common access patterns.

**QUESTION 28**

*What is the maximum file size supported by a file system with 16 direct blocks, single, double, and triple indirection? The block size 512 bytes. Disk block numbers can be stored in 4 bytes.*

Number of blocks:  
- Direct blocks = 16 blocks.  
- Single indirect blocks = 512 / 4 = 128  
- Double indirect blocks = 128 \* 128 = 16384 blocks  
- Triple indirect blocks = 128 \* 128 \* 128 = 2097152 blocks.

Total number of blocks = direct + single + double + triple = 16 + 128 + 16384 + 2097152 = 2113680.  
Total number of bytes = 2113680 \* 512 = 1.08220416 E 9 = 1.08 GB

**QUESTION 29**

*What are temporal locality and spatial locality?*

Temporal locality is the allocation of data into virtual memory based on the time they are used. This assume the user may want to use recently used data sometime soon in the future.

Spatial locality refers to the allocation and fetching of data into and from virtual memory based on the location of the data. This assumes that the user is likely to use data in neighbouring address spaces.

**QUESTION 30**

*Explain the following basic algorithms that are used for the selection of a page as replacement algorithms: clock policy, first-in-first-out (FIFO), least recently used (LRU), and optimal policy?*

Clock policy: Each page is marked with a ‘usage’ bit. When a page is first loaded in memory or referenced, the use bit is set to 1. The set of frames is allocated in a circular buffer with a pointer allocated, and any frame with a use bit of 1 is passed over by the algorithm.

First-in-first-out (FIFO): A strict queueing scheme where the first page to arrive in the queue is the first element to be executed. Performs better for long processes than short once as it is non-pre-emptive.

Least-recently-used (LRU): Each page is marked with a timestamp and the page that has been most recently accessed gets removed. It replaces the page that has not been reference for the longest time.

Optimal Policy: Selects the replacement page based on whichever time to the next reference is the longest.

**MULTIPLE CHOICE**

**COMPUTER SYSTEM OVERVIEW**

**1. The interrupt can occur at any time and therefore at any point in the execution of a user program. True/False.**

True.

**2. The operating system acts as an interface between the computer hardware and the human user. True/False.**

True.

**3. A system bus transfers data between the computer and its external environment.**

False. A system bus connects the major components of a computer system, combining the data bus, address bus, and control bus.

**4. In a uniprocessor system, multiprogramming increases processor efficiency by:**

Taking advantage of time wasted by long wait interrupt handling.

**5. The \_\_\_\_ routine determines the nature of the interrupt and performs whatever actions are needed.**

Interrupt handler.

**6. With interrupts, the processor can not be engaged in executing other instructions while an I/O operation is in progress. True/False.**

False, this is the purpose of interrupts.

**7. Instruction processing consists of the two steps:**

Fetch and execute.

**8. The fetched instruction is loaded into the Program Counter.**

False. The fetched instruction is loaded into Instruction Register (IR).

**9. The unit of data exchanged between cache and main memory is:**

Block size.

**10. The four main structural elements of a computer system are:**

Processor, Main Memory, I/O Modules, Secondary Memory.

**11. It is not possible for a communications interrupt to occur while a printer interrupt is being processed. True/False.**

False. Interrupts can occur during a printer interrupt.

**12. The \_\_\_ holds the address of the next instruction to be fetched.**

Program Counter.

**13. The \_\_\_ contains the data to be written into memory and receives the data read from memory.**

Memory buffer register.

**14. Small, fast memory located between the processor and main memory is called:**

Cache memory.

**15. Interrupts are provided primarily as a way to improve processor utilization. True/False.**

True.

**OPERATING SYSTEM OVERVIEW**

**1. A special type of programming language used to provide instructions to the monitor is \_\_\_\_.**

JCL – Job Control Language. This is a language which enables the user to define the tasks to be undertaken by the operating system.

**2. The OS frequently relinquishes control and must depend on the processor to allow it to regain control. True/False.**

True.

**3. One of the first time-sharing operating systems to be developed was the \_\_\_\_.**

The Compatible Time-Sharing System (CTSS).

**4. The processor itself is not a resource so the OS is not involved in determining how much of the processor is devoted to the execution of a user program. True/False.**

False. The processor is involved in how much of its time is devoted to the execution of a user program.

**5. A process consists of three components: an executable program, the associated data needed by the program, and the execution context of the program. True/False.**

True.

**6. A monolithic kernel is implemented as a single process with all elements sharing the same address space. True/False.**

True.

**7. A common strategy to give each process in the queue some time in turn is referred to as a \_\_\_ technique.**

Time Slicing.

**8. Both batch processing and time-sharing use multiprogramming.**

True.

**9. The \_\_\_\_ is the internal data by which the OS is able to supervise and control the process.**

Execution Context.

**10. The principle objective of Batch Multiprogramming is to minimize response time. True/False.**

False. Batch processing is the execution of a series of jobs in a program on a computer without manual intervention. That is, it is a processing mode in which the execution of a series of programs each on a set or batch of inputs, rather than a single input. The main objective is to increase processor efficiency.

**11. The technique where a system clock generates interrupts, and at each clock interrupt the OS regains control and assigns the processor to another user, is \_\_\_\_.**

Time Slicing.

**PROCESS DESCRIPTION AND CONTROL**

**1. A process is in the \_\_\_ state when it is in main memory and awaiting and event.**

Blocked.

**2. The processor itself only provides limited support for multiprogramming, and \_\_\_ is needed to manage the sharing of the processor and other resources by multiple applications at the same time.**

Software.

**3. A \_\_\_ is a unit of activity characterized by the execution of a sequence of instructions, a current state, and an associated set of system resources.**

Process.

**4. It is the principal responsibility of the \_\_\_ to control the execution of processes.**

OS.

**5. The principle function of the OS is to create, manage, and terminate processes. True/False.**

True.

**6. All processor designs include a register or set of register, often known as the program status word, which contains status information.**

True.

**7. When a process is in the \_\_\_ state it is in secondary memory but is available for execution as soon as it is loaded into main memory.**

Ready/Suspend.

**8. The OS must maintain \_\_\_ tables to manage processes.**

Process.

**9. The collection of program, data, stack, and attributes is referred to as the \_\_\_\_.**

Process image.

**THREADS**

**1. In a multithreaded environment, a \_\_\_\_ is define as the unit of resource allocation and a unit of protection.**

Process.

**2. The traditional approach of a single thread of execution per process in which the concept of a thread is not recognized, is referred to as a \_\_\_\_\_.**

Single-Threaded approach.

**3. The \_\_\_\_ are the fundamental entities that can be scheduled and dispatched to run on one of the system processors.**

Kernel threads.

**4. A principle disadvantage of the \_\_\_ approach is that the transfer of control from one thread to another within the same process requires a mode switch to the kernel.**

KLT – Kernel Level Threads.

**5. Any alteration of a resource by one thread affects the environment of the other threads in the same process. True/False.**

True.

**6. \_\_\_ are characterized by the presence of many single-threaded processes.**

Multiprocess applications.

**7. A thread enters the \_\_\_ state, after waiting, if it is ready to run but the resources are not available.**

Transition.

**8.** **In a pure ULT facility, all of the work of thread management is done by the application, and the kernel is not aware of the existence of threads. True/False.**

True.

**9. A \_\_\_ is a dispatchable unit of work that executes sequentially and is interruptible so that the processor can turn to another thread.**

Thread.

**10. The blocked state in which the process is waiting for an event, such as the end of an I/O operation, the availability of a resource, or a signal from another process is the \_\_\_ state.**

Interruptible.

**11. The \_\_\_ state is when the thread has terminated.**

ZOMBIE.

**12. Any alteration of a resource by one thread affects the environment of the other threads in the same process.**

True.

**CONCURRENCY (MUTUAL EXCLUSION AND SYNCHRONISATION)**

**1.** **It is possible in a single-processor system to not only interleave the execution of multiple processes but also to overlap them. True/False.**

False. Single-processor system do not execute multiple processes.

**2. A process that is waiting access to a critical section does not consume processor time. True/False.**

False.

**3. The case of cooperation by sharing covers processes that interact with other processes without being explicitly aware of them. True/False.**

True.

**4. The management of multiple processes within a uniprocessor system is \_\_\_\_.**

Multiprogramming.

**5. As an extension of the principles of modular design and structured programming, some applications can be effectively programmed as a set of concurrent processes. True/False.**

True.

**6. A \_\_\_ is a data type that is used to block a process or thread until a particular condition is true.**

Condition variable.

**7. It is possible for one process to lock the mutex and for another process to unlock it. True/False.**

False. Once a mutex has been locked by a process, it can only be unlocked by that same process.

**8. Concurrent processes do not come into conflict with each other when they are competing for use of the same resource.**

False. Processes can come into conflict, this is why there is a need for process scheduling.

**CONCURRENCY (DEADLOCK AND STARVATION)**

**1. The OS may pre-empt the second process and require it to release its resources if a process requests a resource that is currently held by another process. True/False.**

True.

**2. With \_\_\_\_ only one process may use a resource at a time and no process may access a resource unit that has been allocated to another process.**

Mutual exclusion.

**3. An indirect method of deadlock prevention is to prevent the occurrence of a circular wait. True/False.**

False. An direct method of deadlock prevention is to prevent the occurrence of a circular wait.

**4. The \_\_\_ condition can be prevented by requiring that a process request all of its required resources at one time and blocking the process until all requests can be granted simultaneously.**

Hold and wait.

**5. Deadlock is permanent because none of the events are ever triggered. True/False.**

True.

**6.** **A closed chain of processes exists, such that each process holds at least one resource needed by the next process in the chain is the condition of \_\_\_\_.**

Circular wait.

**7. Once the processes have progressed into the \_\_\_\_, those processes will deadlock.**

Fatal region.

**8. Requested resources are granted to processes whenever possible with \_\_\_\_.**

Mutual Exclusion.

**9. The dining philosophers’ problem can be representative of problems dealing with the coordination of shared resources which may occur when an application includes concurrent threads of execution.**

True. The dining philosopher’s problem is often used in concurrent algorithm design to illustrate synchronization issues and techniques for resolving them.

**UNIPROCESSOR SCHEDULING**

**1. This is a decision whether to add a new process to the set of processes that are currently active.**

Long-term scheduling.

**2. FCFS performs much better for short processes than long ones. True/False.**

False. FCFS performs much better for long processes due to non-pre-emptive.

**3. The strategy that schedules processes based on their group affiliation is generally referred to as \_\_\_\_\_\_\_\_\_ .**

Fair share scheduling.

**4. Typically, the swapping-in function for processes is based on the need to manage \_\_\_\_\_\_\_ .**

The degree of multiprogramming.

**5. The \_\_\_ specifies the instants in time at which the selection function is exercised.**

Decision mode.

**6. One problem with a pure priority scheduling scheme is that lower-priority processes may suffer starvation. True/False.**

True.

**7. The main objective of long-term scheduling is to allocate processor time in such a way as to optimize one or more aspects of system behaviour. True/False.**

False. The main objective of long-term scheduling is to decide whether there is enough memory to allow new programs or jobs into the system.

**8. The medium-term scheduler is invoked whenever an event occurs that may lead to the blocking of the current process or that may provide an opportunity to pre-empt a currently running process in favour of another. True/False.**

False. The medium-term scheduler makes the decision to swap a job out or to suspend it until a more important process is finished. When the computer is less busy, the medium-term scheduler allows the suspended jobs to pass.

**9. Response time in an interactive system is an example of:**

User-oriented criteria for short-term scheduling policies.

**10. The operating system must make \_\_\_\_\_\_\_\_\_ types of scheduling decisions with respect to the execution of processes.**

Six.

**MEMORY MANAGEMENT**

**1. Overlay programming wastes the programmers time. True/False.**

True. Overlaying means the process of transferring a block of program code or other data into internal memory, replacing what is already stored. This method allows programs to be larger than a computer’s main memory.

**2. One technique for overcoming external fragmentation is \_\_\_\_.**

Compaction – shuffling the fragmented memory into one contiguous location.

**3. The page table for each process maintains \_\_\_\_.**

The logical memory location of the process.

**4. Main memory divided into a number of equal size frames is the \_\_\_\_\_\_\_\_\_\_ technique.**

Simple Paging.

**5. A \_\_\_\_\_\_\_\_\_\_\_ is a particular example of logical address in which the address is expressed as a location relative to some known point, usually a value in a processor register.**

Relative address.

**6.  A physical address is the location of a word relative to the beginning of the program and the processor translates that into a logical address. True/False.**

False. The physical address corresponds to the location of some data in main memory.

**7. Main memory divided into a number of static partitions at system generation time is \_\_\_\_\_\_\_ .**

Fixed partitioning.

**8. With \_\_\_\_\_\_\_\_\_\_ a process is loaded by loading all of its segments into dynamic partitions that need not be contiguous.**

Simple segmentation.

**9. The chunks of a process are known as \_\_\_\_\_\_\_\_\_\_ .**

Pages.

A page is what you want to store (such as a logical address), and a page frame is where you want to store it (physical address).

**10.** **The memory protection requirement must be satisfied by the operating system rather than the processor. True/False.**

False.

**11. Available chunks of memory are known as \_\_\_\_\_\_\_\_\_\_\_ .**

Frames.

**VIRTUAL MEMORY**

**1. The address of a storage location in main memory is the \_\_\_\_\_\_\_\_\_\_\_\_\_.**

Real address.

**2. With \_\_\_\_\_\_\_\_\_\_\_ pages other than the one demanded by a page fault are brought in.**

Pre-paging.

**3.  \_\_\_\_\_\_\_\_\_ is the concept associated with determining the number of processes that will be resident in main memory.**

Load Control.

**4.  \_\_\_\_\_\_\_\_\_ allows the programmer to view memory as consisting of multiple address spaces.**

Segmentation.

**5.  \_\_\_\_\_\_\_\_\_\_ is the range of memory addresses available to a process.**

Address space.

**6.  A \_\_\_\_\_\_\_\_\_ chooses only among the resident pages of the process that generated the page fault in selecting a page to replace.**

Local replacement policy.

**7. The \_\_\_\_\_\_\_\_\_ determines when a page should be brought into main memory.**

Fetch policy.

**8.  The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites. True/False.**

True.

**9. The design issue of page size is related to the size of physical main memory and program size.**

True.

**10.  \_\_\_\_\_\_\_\_\_\_ is transparent to the programmer and eliminates external fragmentation providing efficient use of main memory.**

Paging.

**11. The smaller the page size, the greater the amount of internal fragmentation. True/False.**

False. The larger the page size, the greater amount of internal fragmentation.

**12. The size of virtual storage is limited by the actual number of main storage locations. True/False.**

False.

**13. The \_\_\_\_\_\_\_\_\_ policy results in the fewest number of page faults.**

Optimal.

**I/O MANAGEMENT AND DISK SCHEDULING**

**1.  The SCAN policy favors jobs whose requests are for tracks nearest to both innermost and outermost tracks, and favors the latest arriving jobs.**

True.

**2. The hierarchical philosophy is that the functions of the operating system should be separated according to their complexity, their characteristic time scale, and their level of abstraction. True/False.**

True.

**3.  \_\_\_\_\_\_\_\_\_\_ is a technique that smoothes out peaks in I/O demand.**

Buffering: Buffering is the process of temporarily storing data that is passing between a processor and a peripheral. The usual purpose is to smooth out the difference in rates in which the two devices can handle data.

**4. Double buffering is when a process transfers data to (or from) one buffer while the operating system empties (or fills) the other. True/False.**

True. Double buffering: A programming technique that uses two buffers to speed up a computer that can overlap I/O with processing. Data in one buffer are being processed while the next set of data is read into the other one.

**5. The simplest type of support that the operating system can provide is \_\_\_\_\_\_\_\_\_ .**

Single buffering.

Circular buffer: A memory allocation scheme where memory is reused when an index writes over a previously used location.

**6. The \_\_\_\_\_\_\_\_\_\_ unit is capable of mimicking the processor and of taking over control of the system bus just like a processor.**

Direct Memory Access.

Interrupt-driven I/O: The processor issues an I/O command on behalf of a process. Non-blocking: Processor continues to execute instructions from the process that issued the I/O command. Blocking: The next instruction the processor executes is from the OS, which will put the current process in a blocked state and schedule another process.

Direct Memory Access (DMA): DMA module controls the exchange of data between main memory and I/O module through the system bus.

Programmed I/O: The processor issues an I/O command on behalf of a process to an I/O module – that process then busy waits for the I/O operation to be completed before proceeding.

**7.  In the C-SCAN disk scheduling algorithm, the disk arm is required to move in one direction only until it reaches the last track or there are no more requests to service in that direction, then it reverses direction and the scan proceeds in the opposite direction in the same fashion. True/False.**

False. The C-SCAN (Circular Scan) begins its scan toward the nearest end and works its way to the end of the system. Once it hits the bottom/top, it jumps to the other end and moves in the same direction.

**8. The requested operations and data are converted into appropriate sequences of I/O instructions, channel commands, and controller orders at the \_\_\_\_\_\_\_\_\_\_ layer.**

Device I/O.

**9. The actual queuing and scheduling of I/O operations occurs at the logical I/O module.**

False. This occurs at the scheduling and control layer.

**10.  \_\_\_\_\_\_\_\_\_\_ external devices are suitable for communicating with modems.**

Communication.

Communication: Suitable for communicating with remote devices, modems and digital line drivers.

Human Readable: Suitable for communicating with computer users, printers, terminals, video display, keyboards and mouse.

Machine readable: Suitable for communicating with electronic equipment, disk drives, USB keys, sensors, controllers and actuators.

**11. Access rights are managed at the physical organization layer.**

False.

**12. On a moveable-head system, the time it takes to position the head at the track is known as \_\_\_\_\_\_\_\_ .**

Seek time.

Rotational delay: Amount of time between information requests and how long it takes the hard drive to move to the sector where the requested data is located. I.e. A time measurement of how long before a rotating drive can transfer data.

Access time: The time taken to retrieve data from storage.

Transfer time: The time it takes to transmit or move data from one place to another.

Seek time: The time taken for a disk drive to locate the area on the disk where the data to be read is stored.

**FILE MANAGEMENT**

**1.** **An objective of the \_\_\_\_\_\_\_\_\_\_ is to provide a standardized set of I/O interface routines to user processes.**

File management system.

File Directory: A directory of files. Contains information about the files, including attributes, location and ownership.

Working directory: A directory of a hierarchical file system, if any, dynamically associated with each process.

File management system: A type of software that manages data files in a computer system.

**2.** **A \_\_\_\_\_\_\_\_\_\_ is responsible for starting I/O operations on a device and processing the completion of an I/O request.**

Device driver: A program that controls a particular type of device that is attached to your computer.

**3. Indexed allocation supports both sequential and direct access to the file and thus is the most popular form of file allocation. True/False.**

True.

Indexed allocation: Provides solutions to problems of contiguous and linked allocation. An index block is created having all pointers to files. Each file has its own index block which stores the addresses of disk space occupied by the file.

**4. A file allocation table (FAT) is used to keep track of the portions assigned to a file.**

True.

File Allocation Table (FAT): A file system developed for hard drives that used 12 or 16 bits for each cluster entry into the file allocation table. It provides a map of the clusters (the basic units of logical storage on a hard disk) that a file has been stored in.

**5. A \_\_\_ is a collection of related fields that can be treated as a unit by some application program.**

Record.

**6. \_\_\_\_\_\_\_\_\_ is the basic element of data.**

Field.

**7.  The \_\_\_\_\_\_\_\_\_ file exploits the capability found on disks to access directly any block of a known address.**

Indexed

Indexed-Sequential Access Method: A file management system that allows records to be accessed either sequentially (in the order they were entered) or randomly (with an index).

Sequential file: A file that contains records or other elements that are stored in a chronological order based on account number or some other identifying data. Must be read starting at the beginning of the file.

**8. The term \_\_\_\_\_\_\_\_\_ refers to the logical structuring of the records as determined by the way in which they are accessed.**

File organization.

**9. Records may only be of fixed length. True/False.**

False. Records may be of variable length.

**10.**  **File systems do not maintain the attributes associated with the files. True/False.**

False. File systems DO maintain the attributes associated with the files.