Quiz 1

Which of the following will trigger an interrupt?

|  |  |
| --- | --- |
|  | software execution |
|  | I/O completion |
|  | CPU execution |
|  | kernel function |
| **Question 2** | |  | 1 / 1 point |

Which of the following technology reduces the overhead for bulk data movement?

|  |  |
| --- | --- |
|  | Storage Area Network (SAN) |
|  | Direct Memory Access (DMA) |
|  | Non-uniform Memory Access (NUMA) |
|  | Symmetric Multiprocessing (SMP) |
| **Question 3** | |  | 1 / 1 point |

\_\_\_\_ operating systems are designed primarily to maximize resource utilization.

|  |  |
| --- | --- |
|  | Handheld computer |
|  | Mainframe |
|  | Network |
|  | PC |
| **Question 4** | |  | 1 / 1 point |

Which of the following programs runs all the time on the computer?

|  |  |
| --- | --- |
|  | compiler |
|  | assembler |
|  | text editor |
|  | kernel |
| **Question 5** | |  | 1 / 1 point |

Which of the following storage medium is the slowest regarding access speed?

|  |  |
| --- | --- |
|  | cache |
|  | register |
|  | main memory |
|  | solid state drive |
| **Question 6** | |  | 1 / 1 point |

A system call runs in kernel mode only.

|  |  |
| --- | --- |
|  | False |
|  | True |
| **Question 7** | |  | 1 / 1 point |

Where is the location of L1 cache?

|  |  |
| --- | --- |
|  | inside CPU core |
|  | inside registers |
|  | inside main memory |
|  | inside processor but outside CPU core |
| **Question 8** | |  | 1 / 1 point |

Which of the following stage triggers the CPU switch from user program to interrupt processing?

|  |  |
| --- | --- |
|  | I/O request |
|  | interrupt signaled |
|  | transfer done |
|  | interrupt handled |
| **Question 9** | |  | 1 / 1 point |

The device controller \_\_\_\_\_ an interrupt by asserting a signal on the interrupt request line.

|  |  |
| --- | --- |
|  | dispatches |
|  | raises |
|  | clears |
|  | catches |
| **Question 10** | |  | 1 / 1 point |

Which of the following contains the addresses of all the service routines?

|  |  |
| --- | --- |
|  | system call table |
|  | interrupt vector |
|  | page table |
|  | file descriptor table |

Quiz 2

|  |  |  |
| --- | --- | --- |
| **Question 1** |  | 1 / 1 point |

Applications compiled on one operating system can be directly executable on other operating systems due to common structure.

|  |  |  |
| --- | --- | --- |
|  | True | |
|  | False | |
| **Question 2** |  | 1 / 1 point | |

Which of the following operating system structure involves using loadable kernel module in design?

|  |  |  |
| --- | --- | --- |
|  | monolithic structure | |
|  | layered structure | |
|  | microkernel | |
|  | modular approach | |
| **Question 3** |  | 1 / 1 point | |

The \_\_\_\_\_\_\_\_ provides a portion of the system call interface for UNIX and Linux.

|  |  |  |
| --- | --- | --- |
|  | POSIX | |
|  | Java | |
|  | Standard C library | |
|  | Standard API | |
| **Question 4** |  | 1 / 1 point | |

Which of the following operating system structure is the one for MS-DOS?

|  |  |  |
| --- | --- | --- |
|  | monolithic structure | |
|  | layered structure | |
|  | microkernel | |
|  | modular approach | |
| **Question 5** |  | 1 / 1 point | |

What is the relationship between library call open() and open() system call?

|  |  |  |
| --- | --- | --- |
|  | open() system call is part of open() | |
|  | open() system call in an alternate version of open() | |
|  | open() is a predecessor version of open() system call | |
|  | open() invokes open() system call to get service from operating system | |
| **Question 6** |  | 1 / 1 point | |

Microkernels use \_\_\_\_\_ for communication.

|  |  |  |
| --- | --- | --- |
|  | message passing | |
|  | shared memory | |
|  | system calls | |
|  | virtualization | |
| **Question 7** |  | 1 / 1 point | |

Which of the following is the only gateway between user space and kernel space?

|  |  |  |
| --- | --- | --- |
|  | user interface | |
|  | system call | |
|  | operating system | |
|  | hardware drivers | |
| **Question 8** |  | 1 / 1 point | |

Which of the following is not a type of command interpreter?

|  |  |  |
| --- | --- | --- |
|  | Bourne shell | |
|  | C shell | |
|  | Korn shell | |
|  | KDE or GNOME | |
| **Question 9** |  | 1 / 1 point | |

Which of the following operating system structure is the one for Mach OS?

|  |  |  |
| --- | --- | --- |
|  | monolithic structure | |
|  | layered structure | |
|  | microkernel | |
|  | modular approach | |
| **Question 10** |  | 1 / 1 point | |

Which of the following defines the view of the operating system seen by most users?

|  |  |
| --- | --- |
|  | application and system programs |
|  | system calls |
|  | device drivers |
|  | library calls |

Quiz 3

|  |  |  |
| --- | --- | --- |
| **Question 1** |  | 1 / 1 point |

Applications compiled on one operating system can be directly executable on other operating systems due to common structure.

|  |  |  |
| --- | --- | --- |
|  | True | |
|  | False | |
| **Question 2** |  | 1 / 1 point | |

Which of the following operating system structure involves using loadable kernel module in design?

|  |  |  |
| --- | --- | --- |
|  | monolithic structure | |
|  | layered structure | |
|  | microkernel | |
|  | modular approach | |
| **Question 3** |  | 1 / 1 point | |

The \_\_\_\_\_\_\_\_ provides a portion of the system call interface for UNIX and Linux.

|  |  |  |
| --- | --- | --- |
|  | POSIX | |
|  | Java | |
|  | Standard C library | |
|  | Standard API | |
| **Question 4** |  | 1 / 1 point | |

Which of the following operating system structure is the one for MS-DOS?

|  |  |  |
| --- | --- | --- |
|  | monolithic structure | |
|  | layered structure | |
|  | microkernel | |
|  | modular approach | |
| **Question 5** |  | 1 / 1 point | |

What is the relationship between library call open() and open() system call?

|  |  |  |
| --- | --- | --- |
|  | open() system call is part of open() | |
|  | open() system call in an alternate version of open() | |
|  | open() is a predecessor version of open() system call | |
|  | open() invokes open() system call to get service from operating system | |
| **Question 6** |  | 1 / 1 point | |

Microkernels use \_\_\_\_\_ for communication.

|  |  |  |
| --- | --- | --- |
|  | message passing | |
|  | shared memory | |
|  | system calls | |
|  | virtualization | |
| **Question 7** |  | 1 / 1 point | |

Which of the following is the only gateway between user space and kernel space?

|  |  |  |
| --- | --- | --- |
|  | user interface | |
|  | system call | |
|  | operating system | |
|  | hardware drivers | |
| **Question 8** |  | 1 / 1 point | |

Which of the following is not a type of command interpreter?

|  |  |  |
| --- | --- | --- |
|  | Bourne shell | |
|  | C shell | |
|  | Korn shell | |
|  | KDE or GNOME | |
| **Question 9** |  | 1 / 1 point | |

Which of the following operating system structure is the one for Mach OS?

|  |  |  |
| --- | --- | --- |
|  | monolithic structure | |
|  | layered structure | |
|  | microkernel | |
|  | modular approach | |
| **Question 10** |  | 1 / 1 point | |

Which of the following defines the view of the operating system seen by most users?

|  |  |
| --- | --- |
|  | application and system programs |
|  | system calls |
|  | device drivers |
|  | library calls |

Quiz 4

|  |  |  |
| --- | --- | --- |
| **Question 1** |  | 1 / 1 point |

The major factor impacting efficiency of multithreading on a multiprocessor system is the number of user threads currently running.

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |
| View question 1 feedback | | |

|  |  |  |
| --- | --- | --- |
| **Question 2** |  | 1 / 1 point |

To execute a thread T we need to invoke the run() method using T.run() call in the program.

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |
| View question 2 feedback | | |

|  |  |  |
| --- | --- | --- |
| **Question 3** |  | 1 / 1 point |

The most common technique for writing multithreaded Java programs is \_\_\_\_\_.

|  |  |  |
| --- | --- | --- |
|  | extending the Thread class and overriding the run() method | |
|  | implementing the Runnable interface and defining its run() method | |
|  | designing your own Thread class | |
|  | using the CreateThread() function | |
| **Question 4** |  | 1 / 1 point | |

Which of the following options to deliver signals in multithreaded program should be applied to a synchronous signal?

|  |  |  |
| --- | --- | --- |
|  | deliver the signal to the thread to which the signal applies | |
|  | deliver the signal to every thread in the process | |
|  | deliver the signal to certain threads in the process | |
|  | assign a specific thread to receive all signals for the process | |
|  | all of the above | |
| **Question 5** |  | 1 / 1 point | |

Which of the following benefits go to multithreaded programming?

|  |  |  |
| --- | --- | --- |
|  | responsiveness | |
|  | resource sharing | |
|  | economy | |
|  | scalability | |
|  | all of the above | |
| **Question 6** |  | 1 / 1 point | |

Which of the following items are shared across multiple threads belonging to the same process?

|  |  |  |
| --- | --- | --- |
|  | code, data, files | |
|  | registers | |
|  | stack | |
|  | all of the above | |
| **Question 7** |  | 1 / 1 point | |

One-to-one model provides more concurrency than the many-to-one model by allowing another thread to run when a thread makes a blocking system call.

|  |  |  |
| --- | --- | --- |
|  | True | |
|  | False | |
| **Question 8** |  | 1 / 1 point | |

\_\_\_\_\_\_\_\_\_\_\_\_ refers to the ability of an OS to support multiple, concurrent paths of execution within a single process.

|  |  |  |
| --- | --- | --- |
|  | Multiprocessing | |
|  | Multiprogramming | |
|  | Multithreading | |
|  | Multitasking | |
| **Question 9** |  | 1 / 1 point | |

It takes less time to terminate a process than a thread.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | True | |
|  |  | False | |
| **Question 10** | |  | 1 / 1 point | |

In a pure user thread facility, all the work of thread management is done by the application without the kernel being aware of these threads.

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |

EXAM 1

|  |  |  |
| --- | --- | --- |
| **Question 1** |  | 1 / 1 point |

Any alteration of a process' resource (data or file) by one thread affects the other threads in the same process.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | True | |
|  |  | False | |
| **Question 2** | |  | 1 / 1 point | |

Which of the following state transition cannot happen for a process?

|  |  |  |
| --- | --- | --- |
|  | Waiting to Running | |
|  | Waiting to Ready | |
|  | Running to Waiting | |
|  | Running to Ready | |
| **Question 3** |  | 1 / 1 point | |

Almost all versions of Unix, Linux, and Mac OS X use

\_\_\_POSIX\_\_\_

 API for system calls.

|  |  |  |
| --- | --- | --- |
| **Question 4** |  | 3 / 3 points |

Suppose *T* is an instance of a class that extends *Thread* class in Java. Explain the difference in execution of the statements *T.run()*and *T.start()* in main() code.

When T.run() is called it executes the code in the same thread and does not start a new thread. The main thread will be blocked until the execution of T.run() is completed.

When T.start() is called it creates a new thread for t and then calls the run() method in that new thread. This allows the main thread to continue executing without waiting for t to complete its execution.

**The correct answer is not displayed for Written Response type questions.**

|  |  |  |
| --- | --- | --- |
| **Question 5** |  | 1 / 1 point |

If thread T1 calls a join() on thread T2, then T2 can execute only after T1 has finished its execution.

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |
| View question 5 feedback | | |

|  |  |  |
| --- | --- | --- |
| **Question 6** |  | 0 / 1 point |

Interrupts are provided primarily as a way to improve processor utilization.

|  |  |  |  |
| --- | --- | --- | --- |
| Correct Answer |  | True | |
| Incorrect Response |  | False | |
| **Question 7** | |  | 1 / 1 point | |

The major factor impacting efficiency of multithreading on a multiprocessor system is the number of user threads currently running.

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |
| View question 7 feedback | | |

|  |  |  |
| --- | --- | --- |
| **Question 8** |  | 1 / 1 point |

Which of the following UT -> KT thread mapping model is suitable for a multithreaded Active Window Manager application on a desktop OS ?

|  |  |  |
| --- | --- | --- |
|  | many-to-many | |
|  | many-to-one | |
|  | one-to-one | |
|  | hybrid | |
| **Question 9** |  | 1 / 1 point | |

In a layered OS design, if memory management layer is above the scheduling layer, then the memory management layer can return services invoked by the scheduling layer.

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |
| View question 9 feedback | | |

|  |  |  |
| --- | --- | --- |
| **Question 10** |  | 1 / 1 point |

Which of the following events causes a process transit from ready to running state?

|  |  |  |
| --- | --- | --- |
|  | An interrupt due to a fork() call | |
|  | Completion of an I/O | |
|  | OS issues a timer interrupt | |
|  | Scheduling dispatch | |
| **Question 11** |  | 1 / 1 point | |

How many processes exist at the end of the following code?

main(){

     fork();

     fork();

     fork();

}

|  |  |  |
| --- | --- | --- |
|  | 3 | |
|  | 4 | |
|  | 6 | |
|  | 8 | |
|  | cannot be determined | |
| **Question 12** |  | 1 / 1 point | |

A blocking system call by a thread will block the entire process when thread mapping is done using \_\_\_\_\_\_\_\_\_\_\_\_ model.

|  |  |  |
| --- | --- | --- |
|  | many-to-one | |
|  | one-to-one | |
|  | many-to-many | |
|  | hybrid | |
| **Question 13** |  | 1 / 1 point | |

Computing in multi-chip SMP is less costly in terms of communication overhead than multicore SMP.

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |
| View question 13 feedback | | |

|  |  |  |
| --- | --- | --- |
| **Question 14** |  | 0 / 1 point |

In all circumstances, an OS must handle all interrupts that have been generated.

|  |  |  |
| --- | --- | --- |
| Incorrect Response |  | True |
| Correct Answer |  | False |
| View question 14 feedback | | |

|  |  |  |
| --- | --- | --- |
| **Question 15** |  | 1 / 1 point |

A user program, if executing in \_\_\_\_\_\_\_\_\_\_ , then certain areas of memory are protected from the user's use, and certain instructions cannot be executed.

|  |  |  |
| --- | --- | --- |
|  | task mode | |
| Incorrect Response | kernel mode | |
|  | batch mode | |
| Correct Answer | user mode | |
| **Question 16** |  | 1 / 1 point | |

In a time-sharing system, a user's program is preempted at regular intervals, but due to relatively slow human reaction time this occurrence is usually transparent to the user.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | True | |
|  |  | False | |
| **Question 17** | |  | 1 / 1 point | |

The key states for a thread are:  running,

\_\_\_ready\_\_\_

and blocked.

|  |  |  |
| --- | --- | --- |
| **Question 18** |  | 1 / 1 point |

In designing a time-sharing system, the key objective is to achieve

|  |  |  |
| --- | --- | --- |
|  | low communication overhead | |
|  | high throughput | |
|  | high degree of resource sharing | |
|  | low response time | |
| **Question 19** |  | 1 / 1 point | |

In a uniprocessor system, multiprogramming increases processor efficiency by

|  |  |  |
| --- | --- | --- |
|  | Disabling all interrupts except those of highest priority | |
|  | Taking advantage of time wasted by long wait interrupt handling | |
|  | Eliminating all idle processor cycles | |
|  | Increasing processor speed | |
| **Question 20** |  | 2 / 2 points | |

Differentiate between blocking and non-blocking system call.

A blocking system call suspends the execution of a program until the requested operation is completed by the operating system. The program is not allowed to continue until this requested operation is completed.

A non-blocking system call allows the program to continue executing while the the requested operation is being executed. The program is allowed to continue while this requested operation is executed.

**The correct answer is not displayed for Written Response type questions.**

|  |  |  |
| --- | --- | --- |
| **Question 21** |  | 1 / 1 point |

If a process is swapped out of memory, all of its threads are necessarily swapped out too.

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |
| View question 21 feedback | | |

|  |  |  |
| --- | --- | --- |
| **Question 22** |  | 1 / 1 point |

A monolithic kernel is implemented as a single process with all elements sharing the same address space.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | True | |
|  |  | False | |
| **Question 23** | |  | 1 / 1 point | |

Which of the following is NOT an OS's responsibility?

|  |  |  |
| --- | --- | --- |
|  | Program optimization | |
|  | Interfacing with hardware | |
|  | Optimized resource utilization | |
|  | Separation of privileges | |
| **Question 24** |  | 1 / 3 points | |

Suppose OS-A supports one-to-one and OS-B supports many-to-one models for mapping user threads to kernel threads. Typically, one-to-one is preferred for most of the multi-threaded applications, and hence, OS-A would be the preferred OS. Can you suggest a multi-threaded application that would be advantageous to run on OS-B than OS-A? Explain.

A file server would be a great example of this. A file server allows users to upload documents on specific servers, normally used by a company. There would be thousands of employees uploading files onto this server everyday so a many-to-one model would be necessary. A many-to-one model allows multiple threads to be run at the same time, which would make uploading files onto a file server a lot faster.

A one-to-one

B many-to-one

Can you suggest a multi-threaded application that would be advantageous to run on many-to-one than one-to-one?

**The correct answer is not displayed for Written Response type questions.**

|  |  |  |
| --- | --- | --- |
| View question 24 feedback | | |
| **Question 25** |  | 1 / 1 point | |

In microkernel structure, individual modules can directly communicate with each other using predefined interfaces.

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |
| View question 25 feedback | | |

|  |  |  |
| --- | --- | --- |
| **Question 26** |  | 1 / 1 point |

In a multithreaded code on UNIX, an exec() invoked by a thread will always replace all threads of the entire process by the new program loaded through exec().

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | True | |
|  |  | False | |
| **Question 27** | |  | 1 / 1 point | |

Which of the following is not a typical benefit of multithreading?

|  |  |  |
| --- | --- | --- |
|  | Effective use of multiprocessor systems | |
|  | Enhanced responsiveness of the program | |
|  | Ease of development | |
|  | Increased sharing of resources | |
| **Question 28** |  | 1 / 1 point | |

In some systems, commands are implemented as codes that are interpreted and executed by the shell program.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | True | |
|  |  | False | |
| **Question 29** | |  | 1 / 1 point | |

Which of the following is not a benefit of multiprocessor system?

|  |  |  |
| --- | --- | --- |
|  | Increased convenience of operation | |
|  | More fault-tolerance | |
|  | Increased sharing of hardware resources | |
|  | Higher throughput | |
| **Question 30** |  | 2 / 2 points | |

Why an exec() is usually  invoked after a fork() in process creation?

Fork() is used to create a new process by duplicating the calling process. This creates a Parent and a Child process that continue executing the instructions after the fork call. The Child process is an exact copy of the Parent process.

Exec() is used to to replace the Child's process image with a new process image. This allows the child to execute a different program with different arguments.

**The correct answer is not displayed for Written Response type questions.**

|  |  |  |
| --- | --- | --- |
| **Question 31** |  | 0 / 2 points |

If the programs on a multiprogramming system waits, on average, 75% of the time for an I/O completion, then maximum number of programs (approximately) that can be loaded on the memory to achieve 90% CPU utilization is:

|  |  |  |
| --- | --- | --- |
| Correct Answer | 8 | |
|  | 7 | |
| Incorrect Response | 6 | |
|  | 9 | |
|  | 10 | |
| **Question 32** |  | 1 / 1 point | |

Which of the following is not a benefit of a system call API?

|  |  |  |
| --- | --- | --- |
|  | Enhanced abstraction | |
|  | Portability | |
|  | Convenience of use | |
|  | Increased speed of service | |
| **Question 33** |  | 0 / 1 point | |

Which of the following is shared by all threads of a multithreaded program?

|  |  |
| --- | --- |
| Correct Answer | code |
| Incorrect Response | heap |
|  | register values |
|  | stack |
|  | all of the above |

Quiz 5

|  |  |  |
| --- | --- | --- |
| **Question 1** |  | 1 / 1 point |

Which of the following parameters must be decided for a multilevel feedback queue scheduling algorithm?

1. number of feedback queues

2. entry criterion of a process to a specific queue

3. when to promote or demote processes from one queue to another

4. the scheduling algorithm for each queue

|  |  |  |
| --- | --- | --- |
|  | 1, 2, and 3 only | |
|  | All of 1, 2, 3, and 4 | |
|  | 3 and 4 only | |
|  | 1 and 3 only | |
| **Question 2** |  | 1 / 1 point | |

Measurement of a round-robin scheduling experiment has set the average process runtime for R ns before blocking for an I/O. The time quantum is set as Q ns which is equal to the context switch time. Which of the following expresses the CPU efficiency obtained in the experiment?

|  |  |  |
| --- | --- | --- |
|  | R/(R + R/Q) | |
|  | 1/2 | |
|  | R/(R + Q) | |
|  | (R\*Q)/(R + R/Q) | |
|  | R/(R\*Q + 1) | |
|  | R/2 | |
| **Question 3** |  | 1 / 1 point | |

Which of the following criteria is more important from the point of view of a particular process?

|  |  |  |
| --- | --- | --- |
|  | Turnaround time | |
|  | Throughput | |
|  | CPU utilization | |
|  | Response time | |
| **Question 4** |  | 1 / 1 point | |

Which of the following scheduling algorithms must be nonpreemptive?

|  |  |  |
| --- | --- | --- |
|  | Priority | |
|  | SJF | |
|  | FCFS | |
|  | RR | |
| **Question 5** |  | 1 / 1 point | |

Which of the following scheduling algorithms gives the minimum average response time?

|  |  |  |
| --- | --- | --- |
|  | RR | |
|  | SJF | |
|  | Multilevel queue | |
|  | FCFS | |
| **Question 6** |  | 1 / 1 point | |

A significant problem with priority scheduling algorithms is \_\_\_\_\_.

|  |  |  |
| --- | --- | --- |
|  | complexity | |
|  | starvation | |
|  | determining the length of the time quantum | |
|  | determining the length of the next CPU burst | |
| **Question 7** |  | 1 / 1 point | |

Which of the following can be a solution to the problem of indefinite blockage of low-priority processes?

|  |  |  |
| --- | --- | --- |
|  | Aging | |
|  | Starvation | |
|  | Multilevel queue | |
|  | All of the above | |
| **Question 8** |  | 1 / 1 point | |

Which of the following circumstances can preemptive scheduling take place?

|  |  |  |
| --- | --- | --- |
|  | when a process switches from the running state to the waiting state | |
|  | when a process switches from the waiting state to the ready state | |
|  | when a process terminates | |
|  | none of the above | |
| **Question 9** |  | 1 / 1 point | |

Which of the following CPU scheduling algorithms does not create starvation of a process?

|  |  |  |
| --- | --- | --- |
|  | FCFS | |
|  | RR | |
|  | Priority | |
|  | SJF | |
| **Question 10** |  | 1 / 1 point | |

It is not possible for a CPU scheduler to schedule processes based on their priority ranks and allowing them to run for a specific time quantum before giving chance to next process.

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |

Quiz 6

|  |  |  |
| --- | --- | --- |
| **Question 1** |  | 1 / 1 point |

Cooperating processes sharing resources in read-only mode do not encounter race condition.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | True | |
|  |  | False | |
| **Question 2** | |  | 1 / 1 point | |

A process may have multiple critical sections in its code.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | True | |
|  |  | False | |
| **Question 3** | |  | 1 / 1 point | |

Which of the following critical-section problem's requirements ensures programs will cooperatively determine what process will next enter its critical section?

|  |  |  |
| --- | --- | --- |
|  | mutual exclusion | |
|  | progress | |
|  | bounded waiting | |
|  | none of the above | |
| **Question 4** |  | 1 / 1 point | |

Which of the following is NOT true for Peterson's solution?

|  |  |  |
| --- | --- | --- |
|  | Mutual exclusion is preserved | |
|  | The progress requirement is satisfied | |
|  | The bounded-waiting requirement is met | |
|  | Peterson's solution works for synchronization among more than two processes | |
| **Question 5** |  | 1 / 1 point | |

Which of the following indicates that Pi can enter the critical section in Peterson's solution?

|  |  |  |
| --- | --- | --- |
|  | flag[j] == false or turn == i | |
|  | flag[j] == true or turn == i | |
|  | flag[j] == false or turn == j | |
|  | flag[j] == true and turn == j | |
| **Question 6** |  | 1 / 1 point | |

Which of the following is not true about test\_and\_set instruction?

|  |  |  |
| --- | --- | --- |
|  | It is a hardware instruction | |
|  | It is executed atomically | |
|  | Returns the original value of passed parameter | |
|  | Returns the new value of passed parameter | |
|  | Set the new value of passed parameter to "TRUE" | |
| **Question 7** |  | 1 / 1 point | |

Assume the binary variable lock is initialized to be 0, which of the following can be an implementation of the entry section to solve the critical-section problem?

|  |  |  |
| --- | --- | --- |
|  | while (compare and swap(&lock, 0, 1) != 0), do nothing; | |
|  | while (test and set(&lock)), do nothing; | |
|  | both A and B | |
|  | none of the above | |
| **Question 8** |  | 1 / 1 point | |

When mutex lock is implemented as a binary semaphore, what should its value be initialized to be?

|  |  |  |
| --- | --- | --- |
|  | 0 | |
|  | 1 | |
|  | -1 | |
|  | none of the above | |
| **Question 9** |  | 1 / 1 point | |

The counting semaphore is initialized to \_\_\_\_\_\_\_\_\_\_\_.

|  |  |  |
| --- | --- | --- |
|  | 0 | |
|  | 1 | |
|  | the number of resources available | |
|  | none of the above | |
| **Question 10** |  | 1 / 1 point | |

Which of the following is NOT true?

|  |  |
| --- | --- |
|  | Since semaphore and mutex lock are tools for synchronization, process that have used semaphores or mutex locks should not cause deadlocks |
|  | Semaphores and mutex locks may be shared resources that difference processes contend for, and hence deadlocks may occur |
|  | a set of processes is in a deadlocked state when every process in the set is waiting for an event that can be caused only by another process in the set |
|  | all of the above |

Quiz 7

|  |  |  |
| --- | --- | --- |
| **Question 1** |  | 1 / 1 point |

If three processes satisfy hold-and-wait condition when mutual exclusion is enforced and no preemption is allowed then they create a deadlock.

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |
| View question 1 feedback | | |

|  |  |  |
| --- | --- | --- |
| **Question 2** |  | 1 / 1 point |

A "wait-for" graph is used in deadlock \_\_\_\_\_\_\_.

|  |  |  |
| --- | --- | --- |
|  | prevention | |
|  | avoidance | |
|  | ignorance | |
|  | detection | |
| **Question 3** |  | 1 / 1 point | |

In a resource allocation graph, when a process releases a resource,

|  |  |  |
| --- | --- | --- |
|  | a request edge is inserted. | |
|  | an assignment edge is inserted. | |
|  | a request edge is removed. | |
|  | an assignment edge is removed. | |
| **Question 4** |  | 1 / 1 point | |

The circular-wait condition for a deadlock implies the hold-and-wait condition.

|  |  |  |
| --- | --- | --- |
|  | True | |
|  | False | |
| **Question 5** |  | 1 / 1 point | |

A system will never enter a deadlocked state if

|  |  |  |
| --- | --- | --- |
|  | the system chooses to ignore the problem altogether. | |
|  | the system uses the detection and recovery technique. | |
|  | the system uses the deadlock avoidance technique. | |
|  | None of the above. | |
| **Question 6** |  | 1 / 1 point | |

Deadlocks can be prevented only if

|  |  |  |
| --- | --- | --- |
|  | all four necessary conditions cannot hold. | |
|  | at least one of the four necessary conditions cannot hold. | |
|  | mutual exclusion condition cannot hold. | |
|  | circular wait condition cannot hold. | |
| **Question 7** |  | 1 / 1 point | |

In a system that uses deadlock detection algorithm,

|  |  |  |
| --- | --- | --- |
|  | a deadlock is detected as soon as it occurs. | |
|  | a deadlock is detected just before it occurs. | |
|  | a deadlock is detected sometime after it has occurred but not necessarily immediately. | |
|  | a deadlock is detected sometime before it occurs, but not necessarily just before. | |
| **Question 8** |  | 1 / 1 point | |

A claim edge in a resource-allocation graph indicates that

|  |  |  |
| --- | --- | --- |
|  | a process has been allocated a resource. | |
|  | a process may request a resource. | |
|  | a process may release a resource that it has already been assigned. | |
|  | a process has requested a resource. | |
| **Question 9** |  | 2 / 2 points | |

Can a deadlock be created by a single process? Why or why not? (*Just 'yes' or 'no' answer is not sufficient. You need to provide explanation/justification to support your answer.*)

No it is not possible for a deadlock to be created by a single process. A deadlock happens when two or more processes are blocked indefinitely, waiting for each other to release resources that they hold. Deadlocks require at least two processes. In the question, we would only have one process, which means there wouldn't be another process for it to be waiting on.

**The correct answer is not displayed for Written Response type questions.**

Quiz 8

If the base register is loaded with value 12345 and limit register is loaded with value 1000, which of the following memory address access will not result in a trap to the operating system?

|  |  |  |
| --- | --- | --- |
|  | 12200 | |
|  | 12344 | |
|  | 12500 | |
|  | 13346 | |
| **Question 2** |  | 1 / 1 point | |

\_\_\_\_\_ is the method of binding instructions and data to memory performed by most general-purpose operating systems.

|  |  |  |
| --- | --- | --- |
|  | Load-time binding | |
|  | Compile time binding | |
|  | Interrupt binding | |
|  | Execution time binding | |
| **Question 3** |  | 1 / 1 point | |

Suppose the size of a process is 10,000 bytes and the relocation register is loaded with value 5000, which of the following memory address this process can access?

|  |  |  |
| --- | --- | --- |
|  | logical address 10,350 | |
|  | physical address 4,500 | |
|  | physical address 10,350 | |
|  | None of the above | |
| **Question 4** |  | 1 / 1 point | |

Fragmentation does not occur in a paging system.

|  |  |  |
| --- | --- | --- |
|  | True | |
|  | False | |
| **Question 5** |  | 1 / 1 point | |

Given the logical address 0xAEF9 (in hexadecimal) with a page size of 256 bytes, what is the page number?

|  |  |  |
| --- | --- | --- |
|  | 0xAE | |
|  | 0xF9 | |
|  | 0xA | |
|  | 0x00F9 | |
| **Question 6** |  | 1 / 1 point | |

Which of the following is true about dynamic storage allocation?

|  |  |  |
| --- | --- | --- |
|  | First fit is clearly better than best fit in terms of time and storage utilization. | |
|  | First fit requires less time for allocation than worst fit on average. | |
|  | Best fit is clearly better than first fit in terms of time and storage utilization. | |
|  | Worst fit provides the best storage utilization. | |
| **Question 7** |  | 1 / 1 point | |

Suppose a memory access with a TLB hit takes 5 ns. Assuming TLB overhead and other referencing overhead is zero, what is the slowdown percentage in access time if TLB hit rate is 90%?

|  |  |
| --- | --- |
|  | 5% |
|  | 10% |
|  | 20% |
|  | 1% |
| View question 7 feedback | |

|  |  |  |
| --- | --- | --- |
| **Question 8** |  | 1 / 1 point |

A process has 64 pages in its logical address space with a page size 1024 bytes. Assuming a byte is one memory location, how many bits are needed to express a logical address?

|  |  |
| --- | --- |
|  | 15 |
|  | 10 |
|  | 6 |
|  | 16 |
| View question 8 feedback | |

|  |  |  |
| --- | --- | --- |
| **Question 9** |  | 1 / 1 point |

External fragmentation is

|  |  |  |
| --- | --- | --- |
|  | when there is some unused memory that cannot be allocated to a process. | |
|  | when the amount of available memory is less than the size of a process. | |
|  | when a process is broken up into smaller parts for memory allocation. | |
|  | when there is enough total memory space to satisfy a request but the available spaces are not contiguous. | |
| **Question 10** |  | 1 / 1 point | |

A translation look-aside buffer is used to

|  |  |
| --- | --- |
|  | cache page table entries. |
|  | store the address of the page table in memory. |
|  | size of the logical address space of the currently running process. |
|  | store page size. |

Exam 2

|  |  |  |
| --- | --- | --- |
| **Question 1** |  | 1 / 1 point |

Which of the following indicates that Pi can enter the critical section in Peterson's solution?

|  |  |  |
| --- | --- | --- |
|  | flag[j] == false or turn == i | |
|  | flag[j] == true or turn == i | |
|  | flag[j] == false or turn == j | |
|  | flag[j] == true and turn == j | |
| **Question 2** |  | 1 / 1 point | |

If three processes satisfy hold-and-wait condition when mutual exclusion is enforced and no preemption is allowed then they create a deadlock.

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |
| View question 2 feedback | | |

|  |  |  |
| --- | --- | --- |
| **Question 3** |  | 1 / 2 points |

Consider the following resource-allocation policy:

Requests for and releases of resources are allowed at any time. If a request for resources cannot be satisfied because the resources are not available, then we check any processes that are blocked waiting for resources. If a blocked process has the desired resources, then these resources are taken away from it and are given to the requesting process. The need for resources by the blocked process is updated accordingly to include the resources that were taken away.

Can deadlock occur under the above resource-allocation policy? If yes, explain how that can happen. If no, then explain why not.

Yes, deadlock can occur. In the above scenario there would be a circular wait condition. This would cause processes to request a resource and then get their resource taken away and then are placed in a blocked waiting state waiting for another resource.

**The correct answer is not displayed for Written Response type questions.**

|  |  |  |
| --- | --- | --- |
| View question 3 feedback | | |
| **Question 4** |  | 1 / 1 point | |

FCFS scheduling alogrithm tends to favor I/O bound processes over CPU-bound processes.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | True | |
|  |  | False | |
| **Question 5** | |  | 1 / 1 point | |

A significant problem with priority scheduling algorithms is \_\_\_\_\_.

|  |  |  |
| --- | --- | --- |
|  | complexity | |
|  | starvation | |
|  | determining the length of the next CPU burst | |
|  | determining the length of the time quantum | |
| **Question 6** |  | 2 / 3 points | |

Suppose a system with 512 bytes of physical memory uses a paging scheme where each page is of size 16 bytes. Here each byte indicates a memory address. Suppose a process has 108 bytes of logical address space. The page table for the process is as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Page# | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Frame# | 5 | 6 | 1 | 12 | 8 | 7 | 18 |

Find out the following for the CPU generated address 89:

i) page number, ii) offset, iii) physical address corresponding to this logical address.

i) page number: 89 to binary is 01011001 and with 16 bits you have 01011 or  in decimal

ii) offset: 1001 is 9 in decimal

iii) physical address: Frame *Page Size + offset = 8 \**16 + 9 = 137

**The correct answer is not displayed for Written Response type questions.**

|  |  |  |
| --- | --- | --- |
| View question 6 feedback | | |
| **Question 7** |  | 1 / 1 point | |

Which of the following CPU scheduling algorithms is unlikely to create starvation of a process?

|  |  |  |
| --- | --- | --- |
|  | Priority | |
|  | Round robin | |
|  | First-come-first-serve | |
|  | Shortest job first | |
| **Question 8** |  | 1 / 1 point | |

Which of the following circumstances can preemptive scheduling take place?

|  |  |  |
| --- | --- | --- |
|  | when a process switches from the running state to the waiting state | |
|  | when a process switches from the waiting state to the ready state | |
|  | when a process terminates | |
|  | none of the above | |
| **Question 9** |  | 0.5 / 1 point | |

Measurement of a round-robin scheduling experiment has set the average process runtime for R ns before blocking for an I/O. The time quantum is set as Q ns which is twice the context switch time S. Which of the following expresses the CPU efficiency obtained in the experiment?

|  |  |  |
| --- | --- | --- |
|  | S/(R + R/Q) | |
|  | R/(S + Q) | |
| Incorrect Response | 1/2 | |
| Correct Answer | 2/3 | |
|  | (R\*Q)/(S + R/Q) | |
|  | S/2 | |
|  | R/(R + S)\*Q | |
| **Question 10** |  | 1 / 1 point | |

Assume the binary variable lock is initialized to be 0, which of the following can be an implementation of the entry section to solve the critical-section problem?

|  |  |  |
| --- | --- | --- |
|  | while (compare and swap(&lock, 0, 1) != 0), do nothing; | |
|  | while (test and set(&lock)), do nothing; | |
|  | both A and B | |
|  | none of the above | |
| **Question 11** |  | 1 / 1 point | |

The "convoy effect" for processes is typically observed in

\_\_\_First Come First Serve\_\_\_Incorrect Response**(FCFS)**

scheduling algorithm.

|  |  |  |
| --- | --- | --- |
| **Question 12** |  | 1 / 1 point |

"Aging" in CPU scheduling context attempts to solve the issue of

|  |  |  |
| --- | --- | --- |
|  | mutual exclusion | |
|  | deadlock | |
|  | starvation | |
|  | all of these | |
|  | none of these | |
| **Question 13** |  | 0 / 1 point | |

A signal() (or, release()) cannot be invoked on a semaphore unless a wait() (or, acquire()) has already been invoked on the semaphore.

|  |  |  |  |
| --- | --- | --- | --- |
| Incorrect Response |  | True | |
| Correct Answer |  | False | |
| **Question 14** | |  | 1 / 2 points | |

Suppose a system has non-preemptive priority scheduling algorithm that allows user-defined methods of assigning priority values to processes. Explain how an FCFS scheduling can be implemented on the system using that non-preemptive priority algorithm.

For non-preemptive priority to occur with first come first serve, you would need to give all the processes the same priority for first come first serve to be implemented.

**The correct answer is not displayed for Written Response type questions.**

|  |  |  |
| --- | --- | --- |
| View question 14 feedback | | |
| **Question 15** |  | 0 / 1 point | |

Semaphore acquire() and release() methods are atomic operations.

|  |  |  |
| --- | --- | --- |
| Correct Answer |  | True |
| Incorrect Response |  | False |
| View question 15 feedback | | |

|  |  |  |
| --- | --- | --- |
| **Question 16** |  | 3 / 3 points |

An OS provides each process with 65536 bytes of address space divided into pages of size 4096 bytes each. A particular program has a text size of 32768 bytes, a data size of 16386 bytes, and a stack size of 15870 bytes. A page must contain either text, data, or stack, but not a mix of two or three of these three things. Explain whether the program would fit into the allocated memory space. (Just a 'yes' or 'no' answer is not sufficient. You must provide explanation/justification to support your answer.)

No the program could not fit into the allocated memory space. This is because pages can only contain one type of memory. the amount of pages we have avialable is 16: 65536/4096. The amount of pages we need is 17: text size we need 8, data size we need 4, stack size we need 4 with a remainder of 256 bites because you cannot have multiple data types in pages. This creates a 17 page.

**The correct answer is not displayed for Written Response type questions.**

|  |  |  |
| --- | --- | --- |
| **Question 17** |  | 1.5 / 3 points |

In a multiprocessor system, busy waiting is sometimes tolerated rather than blocking the process. In such an environment if context switch time is S, what is the maximum amount of busy waiting time that may be tolerated as as opposed to blocking the process? Justify/explain your answer.

The maximum amount of busy waiting time is dependent on the context switch time s. If s is small, then it is more efficient to have a process busy waiting on the resource. However if S is large, then it may be more efficient to block the process and perform a context switch and waste cpu cycles.

**The correct answer is not displayed for Written Response type questions.**

|  |  |  |
| --- | --- | --- |
| View question 17 feedback | | |
| **Question 18** |  | 1 / 2 points | |

Consider a preemptive priority scheduling algorithm based on dynamically changing priorities. Greater priority value implies higher priority. All processes are given a priority of 0 when they enter the ready queue. When a process is waiting for the CPU (in the ready queue, but not running), W is added to its priority value after each time unit. When it is running, R is added to its priority value after each time unit.  Here, W < R < 0. Indicate how the processes are scheduled in this scheme.

|  |  |  |
| --- | --- | --- |
|  | Alternating between first waiting process and the new coming process | |
|  | Round-robin with quantum (R-W) | |
| Incorrect Response | First-come-first-served | |
| Correct Answer | Last-in-first-served | |
| **Question 19** |  | 0 / 1 point | |

Suppose a logical address space of a process starts with address 0 and the page index starts from 0 as well (that is, the first page is page 0). If the page size is 32 bytes, in which page the logical address 769 be found?

|  |  |
| --- | --- |
| Incorrect Response | page number 24 |
|  | page 2 |
|  | Could be on any page |
| Correct Answer | page number 25 |
|  | page number 23 |
|  | page 26 |
| View question 19 feedback | |

|  |  |  |
| --- | --- | --- |
| **Question 20** |  | 1 / 1 point |

Match the following two conditions of critical section problem to the two aspects of concurrent execution of processes.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | |  | \_\_2\_\_ |  | Bounded waiting | |  | \_\_1\_\_ |  | Progress | |  | |  |  | | --- | --- | | **1**. | Deadlock | | **2**. | Starvation | | | |
| **Question 21** | | |  | 2 / 3 points | |

Suppose a contiguous allocation of memory left the following holes at some point:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hole # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Size (in KB) | 10 | 4 | 20 | 18 | 7 | 9 | 12 | 15 |

Suppose the following processes now appear (in order of their appearances): Pi of size 13KB, Pj of size 11KB, and Pk of size 9KB.

Calculate the difference between the sizes of the smallest hole left when the processes are fit by "best fit" and "worst fit" memory allocation strategies.

best fit:

pi will fit into hole 3 with 7kb of fragmentation

pj will fit into hole 8 with 4kb of fragmentation

pk will fit into hole 2 with 3kb of fragmentation

leaving the smallest hole of 3kb

worst fit:

pi will fit into hole 3 with 7kb of fragmentation

pj will fit into hole 3 with 4kb of fragmentation

pk will fit into hole 3 with 5kb of fragmentation

leaving the smallest hole of 4kb

difference: 1kb

**The correct answer is not displayed for Written Response type questions.**

|  |  |  |
| --- | --- | --- |
| View question 21 feedback | | |
| **Question 22** |  | 1 / 2 points | |

Examine the following RA-graph and determine whether a deadlock exists. If found, indicate the processes that are involved in the deadlock. Assume that all recourses are single instance.

![Diagram

Description automatically generated]()Diagram

Description automatically generated

Yes a deadlock exists. process p1 is holding resource r1 and waiting for resource r2. Resource 2 is being held by process p3. p3 is waiting for resource r4, which is being held by process p2. P2 is waiting for resource r1, which is being held by process p1. This has created a cyclic dependency, which causes a deadlock. The processes involved are p1, p2, p3.

**The correct answer is not displayed for Written Response type questions.**

|  |  |  |
| --- | --- | --- |
| View question 22 feedback | | |
| **Question 23** |  | 0 / 1 point | |

In any resource allocation graph, existence of a cycle indicates a deadlock is found.

|  |  |  |  |
| --- | --- | --- | --- |
| Incorrect Response |  | True | |
| Correct Answer |  | False | |
| **Question 24** | |  | 1 / 1 point | |

Which of the following parameters must be decided for a multilevel feedback queue scheduling algorithm?

1. number of feedback queues

2. entry criterion of a process to a specific queue

3. when to promote or demote processes from one queue to another

4. the scheduling algorithm for each queue

|  |  |  |
| --- | --- | --- |
|  | 3 and 4 only | |
|  | 1 and 3 only | |
|  | All of 1, 2, 3, and 4 | |
|  | 1, 2, and 3 only | |
| **Question 25** |  | 3 / 3 points | |

Consider the following proposed solution to the critical-section problem for two processes P0 and P1. The processes share the following data items:  
  
boolean ready[2]; //initialized to FALSE

int select; // initialized to 0 or 1  
  
The structure of process P0 is as follows. The other process P1 has similar structure (1 in place of 0).  
  
while ( TRUE ){  
        ready[0] = TRUE;  
        select == 1;  
        while ( ready[1] == TRUE && select == 1 );  
        { ... } //CRITICAL SECTION  
        ready[0] = TRUE;  
        { ... } //REMAINDER SECTION  
}   
  
Does this solve the critical section problem for P0 and P1? If so, explain how. If not, then explain why not. (without proper justification correct answer will not get full credit) .

No this does not solve the critical section.

This line creates an error: ready[0] = TRUE;

this would allow p[0] to enter the critical section even if p[1] is within the critical section. This would create a deadlock, if p[1] is within the critical section and hasn't set ready[1] to false.

**The correct answer is not displayed for Written Response type questions.**

|  |  |  |
| --- | --- | --- |
| **Question 26** |  | 1 / 1 point |

The "compaction" strategy can resolve the issue of

|  |  |
| --- | --- |
|  | external fragmentation only |
|  | internal fragmentation only |
|  | both external and internal fragmentation |
|  | none of the above |

Quiz 9

|  |  |  |
| --- | --- | --- |
| **Question 1** |  | 1 / 1 point |

Assume there are enough free frames available in physical memory. With demand paging, a complete execution of an instruction of a process can cause multiple page faults (that is, faulting for multiple pages).

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |
| View question 1 feedback | | |

|  |  |  |
| --- | --- | --- |
| **Question 2** |  | 1 / 1 point |

A free-frame list

|  |  |  |
| --- | --- | --- |
|  | is a set of all frames that are filled with all zeros. | |
|  | is a set of all frames that are currently unallocated to any process. | |
|  | is a set of all frames that are currently being shared by at least two processes. | |
|  | is a set of all frames that are used for stack and heap memory. | |
| **Question 3** |  | 1 / 1 point | |

In demand paging,

|  |  |  |
| --- | --- | --- |
|  | a page loaded in memory may never be accessed. | |
|  | all pages that a program will access during execution are loaded in memory in the beginning. | |
|  | a page is loaded in memory only when it is needed during execution. | |
|  | a page is loaded in memory just before it is needed. | |
| **Question 4** |  | 1 / 1 point | |

The most preferred method of swapping a process is

|  |  |  |
| --- | --- | --- |
|  | to swap using the file system. | |
|  | to copy an entire file to swap space at process startup and then perform demand paging from the swap space. | |
|  | to demand-page from the file system initially but to write the pages to swap space as they are replaced. | |
|  | None of the above. | |
| **Question 5** |  | 1 / 1 point | |

A page fault must be preceded by a TLB miss.

|  |  |  |
| --- | --- | --- |
|  | True | |
|  | False | |
| **Question 6** |  | 1 / 1 point | |

The instruction that causes a page fault needs to be re-executed after the fault has been handled.

|  |  |  |
| --- | --- | --- |
|  | True | |
|  | False | |
| **Question 7** |  | 1 / 1 point | |

The dirty (modify) bit identifies

|  |  |  |
| --- | --- | --- |
|  | a page that has been corrupted. | |
|  | a page that needs to be reloaded when accessed. | |
|  | a page that is shared by multiple processes. | |
|  | a page that has been modified since it was loaded. | |
| **Question 8** |  | 1 / 1 point | |

A drawback of equal or proportional allocation is that

|  |  |  |
| --- | --- | --- |
|  | they are very expensive to compute. | |
|  | the processes that arrive earlier get more pages than the processes arriving later. | |
|  | the allocation varies according to the degree of multiprogramming. | |
|  | a high-priority process is treated the same as a low-priority process. | |
| **Question 9** |  | 1 / 1 point | |

A sign of thrashing is

|  |  |  |
| --- | --- | --- |
|  | the CPU utilization increases as the degree of multiprogramming is increased. | |
|  | the CPU utilization decreases as the degree of multiprogramming is increased. | |
|  | the CPU utilization increases as the number of pages allocated to each process is increased. | |
|  | the CPU utilization decreases as the number of pages allocated to each process is increased. | |
| **Question 10** |  | 1 / 1 point | |

The current best practice to avoid thrashing is to include enough physical memory.

|  |  |
| --- | --- |
|  | True |
|  | False |

Quiz 10

|  |  |  |
| --- | --- | --- |
| **Question 1** |  | 1 / 1 point |

Implementing files internally using contiguous disk block suffers from external fragmentation.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | True | |
|  |  | False | |
| **Question 2** | |  | 1 / 1 point | |

In the two-level directory, if a user refers to a particular file then\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | first MFD (master file directory) is searched, then UFD (user file directory) | |
|  | first UFD (user file directory) is searched, then MFD (master file directory) | |
|  | only MFD (master file directory) is searched | |
|  | only his/her own UFD (user file directory) is searched | |
| **Question 3** |  | 1 / 1 point | |

A two-level directory can be thought as a tree of height \_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | 2 and its root is UFD | |
|  | 1 and its root is MFD | |
|  | 2 and its root is MFD | |
|  | 1 and its root is UFD | |
| **Question 4** |  | 1 / 1 point | |

A relative path name defines a path from\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | the current directory | |
|  | from the UFD (user file directory) | |
|  | the root directory | |
|  | from the MFD (master file directory) | |
| **Question 5** |  | 1 / 1 point | |

Acyclic-graph directory structure\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | prohibits the sharing of files, but allows to share directories | |
|  | allows the sharing of files, but prohibits to share directories | |
|  | prohibits the sharing of files and directories | |
|  | allows to share files and directories | |
| **Question 6** |  | 1 / 1 point | |

The following information is presented for the prog.c

**-rwxr-xr-- 1 Jim staff 130 May 25 22:13 prog.c**

Users Jim, Sara and Mike are the members of the group staff

|  |  |  |
| --- | --- | --- |
|  | Jim can invoke read and execute operation on prog.c | |
|  | Alan can read prog.c | |
|  | Sara can modify prog.c | |
|  | Mike can read and write to prog.c | |
| **Question 7** |  | 1 / 1 point | |

Indexed allocation

|  |  |  |
| --- | --- | --- |
|  | supports direct access | |
|  | does not suffer from external fragmentation. | |
|  | all of the above | |
|  | none of the above | |
| **Question 8** |  | 1 / 1 point | |

The file allocation table (FAT) has one entry for each block and is indexed by block number.

|  |  |  |
| --- | --- | --- |
|  | True | |
|  | False | |
| **Question 9** |  | 1 / 1 point | |

What is the size of the bit vector of a 1TB disk with 512-byte blocks?

|  |  |  |
| --- | --- | --- |
|  | 28 MB | |
|  | 2 MB | |
|  | 2 KB | |
|  | 8 MB | |
|  | 32 KB | |
| **Question 10** |  | 1 / 1 point | |

A disk drive may have many partitions each of which contains a different file system.

|  |  |  |
| --- | --- | --- |
|  | True | |
|  | False | |
| **Question 11** |  | 1 / 1 point | |

The root-partition:

|  |  |  |
| --- | --- | --- |
|  | contains a list of the operating systems which may be booted. | |
|  | does not contain the operating system kernel, but it is the first mounted file system during the boot time. | |
|  | contains an operating system kernel and it is mounted during boot time. | |
|  | contains an operating system kernel and it is not necessary to be mounted. | |
| **Question 12** |  | 1 / 1 point | |

Size of physical disk does not have any effect on size of file allocation table.

|  |  |  |
| --- | --- | --- |
|  |  | True |
|  |  | False |