**CS2205 (OS) Midterm Practice Questions**

1. What are the three main purposes of an operating system?
2. What is the purpose of the *command interpreter*? Why is it usually separate from the kernel?
3. What is a *kernel*? Briefly describe the difference(s)between *microkernel* and *monolithic* kernel design approaches.
4. What is a *system call*? **Figure 1** shows a view of various services of an operating system (OS). Based on the figure, describe how the system calls support user interfaces (your description should include the details of *user application*, *mode bit*, and *mode transition*).



**Figure 1.**

1. What is a *process?* Briefly describe its various states.Why is it important in a computer system?
2. Describe the differences among short-term, medium-term, and long-term scheduling.
3. Distinguish the terms: *interrupt*, *trap* (*system call*), and *mode bit*.
4. Describe the actions taken by a kernel to context-switch between processes.
5. Is it possible to have thread concurrency but not parallelism? Explain.
6. Can a multithreaded solution using multiple user-level threads achieve better performance on a multiprocessor system than on a single processor system? Explain.
7. Using Amdahl’s Law, calculate the speedup gain of an application that has a 70 percent parallel component for (a) two processing cores and (b) four processing cores.
8. What are the two models of interprocess communication (IPC)? What are the strengths and weaknesses of the two approaches?
9. Identify the nature of the process including its IPC model which is shown below:

 while (true)

 {

 if (counter == 0)

 ; /\* do nothing \*/

 next\_item = buffer[out];

 out = (out + 1) % BUFFER\_SIZE;

 counter--;

 }

1. What is a *semaphore*? Complete the semaphoresolution of the **bounded buffer producer-consumer** problem by filling the blanks of the given *producer* and *consumer* structures:

 A **bounded buffer** with ***n*** locations;

 Semaphore **mutex** initialized to the value 1;

 Semaphore **full** initialized to the value 0;

 Semaphore **empty** initialized to the value ***n***;

 **Producer process**

 do { produce an Item;

 14.1 (1 point) ------------------------------------

 14.2 (1 point) ------------------------------------

 add item to the buffer; *//Critical Section*

 14.3 (1 point) ------------------------------------

 signal (full); // *signal to consumer that the buffer is full*

 } while (TRUE);

**Consumer process**

 do {

 14.4 (1 point) ------------------------------------

 14.5 (1 point) ------------------------------------

 remove an item from the buffer; *//Critical Section*

 signal (mutex); // *signal to producer that the mutex is free*

 14.6 (1 point) ------------------------------------

 } while (TRUE);

1. What are two differences between **user-level threads** and **kernel-level threads**? Under what circumstances is one type better than the other?
2. How do the **user-level threads** recognized (map) by an OS for their execution? Describe the pro and cons of these three mapping methods.
3. Consider the following set of processes with their *arrival time*, *CPU burst time*, and *priority* details:

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Arrival Time (ms)** | **CPU Burst (ms)** | **Priority** |
| *P*1 | 0 | 8 | 3 |
| *P*2 | 1 | 4 | 2 |
| *P*3 | 2 | 7 | 4 |
| *P*4 | 3 | 10 | 1 |

Draw the *Gantt chart*, and show the *average waiting time* of the processes based on the following scheduling algorithms:

 17.1) First-Come First-Served (FCFS)

 17.2) Shortest-Job-First (SJF)

 17.3) Priority

 17.4) Round-Robin (time quantum = 4).

1. How could a system be designed to allow a choice of operating systems from which to boot? What would the bootstrap program need to do?
2. The services and functions provided by an operating system can be divided into two main categories. Briefly describe the two categories, and discuss how they differ.
3. What are the five major activities of an operating system with regard to file management?
4. What is the main advantage of the microkernel approach to system design? How do user programs and system services interact in a microkernel architecture? What are the disadvantages of using the microkernel approach?
5. What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other?
6. What resources are used when a thread is created? How do they differ from those used when a process is created?
7. A CPU-scheduling algorithm determines an order for the execution of its scheduled processes. Given n processes to be scheduled on one processor, how many different schedules are possible?
8. Assume that an operating system maps user-level threads to the kernel using the many-to-many model and that the mapping is done through the use of LWPs. Furthermore, the system allows program developers to create real-time threads. Is it necessary to bind a real-time thread to an LWP?