PES UNIVERSITY

(Established under Karnataka Act No.16 of 2013)

100-ft Ring Road, BSK III Stage, Bangalore – 560 085

Department of Computer Science & Engg

Session: Jan-May 2021

UE19CS254: Operating Systems

UNIT 2 Solutions

#	
	Chapter 4
1.	What resources are required to Creating threads? Ans: When a thread is creating the threads does not require any new resources to execute. The thread shares the resources of the process to which it belongs to and it requires a small data structure to hold a register set, stack, and priority.
2.	Under what circumstances user level threads are better than the kernel level threads? Ans: User-Level threads are managed entirely by the run-time system (user-level library).The kernel knows nothing about user-level threads and manages them as if they were single-threaded processes. User-Level threads are small and fast, each thread is represented by a PC, register, stack, and small thread control block. Creating a new thread, switching between threads, and synchronizing threads are done via procedure call. i.e. no kernel involvement. UserLevel threads are hundred times faster than KernelLevel threads. User level threads are simple to represent, simple to manage and fast and efficient.
3.	What is a thread? Ans: A thread otherwise called a lightweight process (LWP) is a basic unit of CPU utilization, it comprises of a thread id, a program counter, a register set and a stack. It shares with other threads belonging to the same process its code section, data section, and operating system resources such as open files and signals
4.	What are the benefits of multithreaded programming? Ans: The benefits of multithreaded programming can be broken down into four major categories: • Responsiveness • Resource sharing • Economy • Utilization of multiprocessor architectures
5.	Compare user threads and kernel threads. Ans: User threads: - User threads are supported above the kernel and are implemented by a thread library at the user level. Thread creation & scheduling are done in the user space, without kernel intervention. Therefore, they are fast to Creating and manage blocking system call will cause the entire process to block Kernel threads: - Kernel threads are supported directly by the operating system. Thread creation, scheduling and management are done by the operating system. Therefore, they are slower to Creating & manage compared to user threads. If the thread performs a blocking system call, the kernel can schedule another thread in the application for execution
6.	List two programming examples of multithreading giving improved performance over a single-threaded solution.

	Ans: • A Web server that services each request in a separate thread. • A parallelized application such as matrix multiplication where different parts of the matrix may be worked on in parallel. • An interactive GUI program such as a debugger where a thread is used to monitor user input, another thread represents the running application, and a third thread monitors performance.
Chapter 6 Process Synchronization	
1.	What are the requirements that a solution to the critical section problem must satisfy? Ans: The three requirements are • Mutual exclusion • Progress • Bounded waiting
2.	Define: Critical section problem. Ans: Consider a system consists of 'n' processes. Each process has segment of code called a critical section, in which the process may be changing common variables, updating a table, writing a file. When one process is executing in its critical section, no other process can allowed executing in its critical section

3.	Name two hardware instructions and their definitions which can be used for implementing mutual exclusion
	• TestAndSet boolean TestAndSet (boolean &target)
	{
	boolean rv = target;
	target = true;
	return rv;
	}
	• compare _and_swap
	int compare _and_swap(int *value, int expected, int new_value) {
	int temp = *value;
	if (*value == expected)
	*value = new_value;
	return temp;
	}

4.	Name some classic problem of synchronization? Ans: The Bounded – Buffer Problem. The Reader – Writer Problem, The Dining –Philosophers Problem
5.	Define entry section and exit section. Ans: The critical section problem is to design a protocol that the processes can use to cooperate. Each process must request permission to enter its critical section. The section of the code implementing this request is the entry section. The critical section is followed by an exit section. The remaining code is the remainder section.
Chapter 7 Deadlocks	
1.	What is resource-allocation graph? Ans: Deadlocks can be described more precisely in terms of a directed graph called a system resource allocation graph. This graph consists of a set of vertices V and a set of edges E. The set of vertices V is partitioned into two different types of nodes; P the set consisting of all active processes in the system and R the set consisting of all resource types in the system.
2.	Define Deadlock.

	Ans: A process requests resources; if the resources are not available at that time, the process enters a wait state. Waiting processes may never again change state, because the resources they have requested are held by other waiting processes. This situation is called a deadlock.
3.	Can a multithreaded solution using multiple user-level threads achieve better performance on a multiprocessor system than on a single processor system? Ans: A multithreaded system comprising of multiple user-level threads cannot make use of the different processors in a multiprocessor system simultaneously