**NAME : OBAFEMI DAVID ADENIYI**

**MATRIC NO : 2022001213**

**DEPARTMENT : COMPUTER SCIENCE**

**LEVEL : 300LEVEL**

**1. Compare and Contrast Processes and Threads**

**Similarities:**

* Both processes and threads are independent units of execution.
* Both can run concurrently, improving system responsiveness and throughput.
* Both can be prioritized, allowing the OS to allocate resources accordingly.

**Differences:**

* **Resource allocation**: Processes have their own memory space, file descriptors, and other resources. Threads share the same memory space and resources as the parent process.
* **Creation overhead**: Creating a new process is more resource-intensive than creating a new thread.
* **Communication**: Processes communicate using inter-process communication (IPC) mechanisms, while threads communicate using shared memory and synchronization primitives.
* **Scheduling**: Processes are scheduled by the OS, while threads are scheduled by the process itself (or the OS, depending on the implementation).
* **Termination**: Terminating a process terminates all its threads, while terminating a thread does not affect the process.

**2. Enumerate the two types of threads to be managed in a modern system 4. A process is an executing instance of an application. What does that mean?**

1. **User Threads**: These threads are created and managed by the application or programmer. They are also known as "application threads" or "user-level threads".
2. **Kernel Threads**: These threads are created and managed by the operating system kernel. They are also known as "system threads" or "kernel-level threads".

**4. Process definition:**

A process is an executing instance of an application, which means:

* A process is a separate entity from the application itself (the program code).
* A process is created when the application is executed or run.
* A process has its own memory space, resources, and execution context.
* A process can be thought of as a container for the application's code and data.

**3. Is Multitasking and Multiprogramming the same? Justify your answer with reason(s)**

No, Multitasking and Multiprogramming are not the same. Here's why:

**Multiprogramming**:

* Refers to the ability of an operating system to execute multiple programs or processes simultaneously.
* The OS allocates resources (CPU, memory, I/O devices) to each program, but only one program executes at a time.
* The OS switches between programs, giving each program a slice of CPU time (time slicing).
* The goal is to maximize CPU utilization and throughput.

**Multitasking**:

* Refers to the ability of an operating system to execute multiple tasks or threads within a single program simultaneously.
* Multiple tasks share the same memory space and resources, but each task has its own execution path.
* The OS switches between tasks, giving each task a slice of CPU time (time slicing).
* The goal is to improve responsiveness, user experience, and system utilization.

Key differences:

* **Number of programs**: Multiprogramming involves multiple programs, while Multitasking involves a single program with multiple tasks.
* **Resource allocation**: Multiprogramming allocates resources to each program, while Multitasking shares resources among tasks within a program.
* **Execution**: Multiprogramming switches between programs, while Multitasking switches between tasks within a program.

In summary, Multiprogramming is about executing multiple programs simultaneously, while Multitasking is about executing multiple tasks within a single program simultaneously.

**4. Highlight the 5 states of processes.**

**1. Newborn (or Created) State**:  
- A new process is created, and the operating system allocates resources to it.  
- The process is not yet ready to run.

**2. Ready State**:  
- The process is waiting for the CPU to become available to execute it.  
- The process is ready to run, but the CPU is busy with other processes.

**3. Running State**:  
- The process is currently being executed by the CPU.  
- The process is using the CPU and other system resources.

**4. Waiting (or Blocked) State**:  
- The process is waiting for an event to occur, such as I/O completion or a signal.  
- The process is not using the CPU, but is waiting for something to happen.

**5. Terminated (or Zombie) State**:  
- The process has finished executing and has exited.  
- The process is no longer using system resources, but its process table entry remains until it is removed.