**Single and Multithreaded Processes**

* **Single-threaded Process**: One thread of execution.
* **Multithreaded Process**: Multiple threads share the same process resources (code, data, files)

A screenshot of a computer

AI-generated content may be incorrect.

**Why Need Multithreading**

* **Concurrency**: Allows multiple tasks to make progress simultaneously.
* **Efficiency**: Reduces the overhead of creating new processes for each task.
* **Example**: A web server using threads to handle multiple client requests efficiently.

**Multithreading**

* **Benefits**:
  + Simplifies code and increases efficiency.
  + Useful for applications performing multiple similar tasks (e.g., web servers).
* **Challenges**:
  + Complexity in managing shared resources and synchronization.

**Multithreaded Server Architecture**

1. **Request**: Client sends a request to the server.
2. **Thread Creation**: Server creates a new thread to service the request.
3. **Resume Listening**: Server resumes listening for additional client requests.

A diagram of a server

AI-generated content may be incorrect.

**Thread**

* **Definition**: A lightweight process (LWP) with its own thread ID, program counter, register set, and stack.
* **Sharing**: Shares code section, data section, and OS resources with other threads in the same process.

**Concurrency vs. Parallelism**

* **Concurrency**: Multiple tasks making progress over time (single-core systems).
* **Parallelism**: Multiple tasks executed simultaneously (multi-core systems).

***Multicore Programming***

* **Types of Parallelism**:
  + **Data Parallelism**: Same operation on different data subsets.
  + **Task Parallelism**: Different operations performed by different threads.
  + **Hybrid**: Combination of data and task parallelism.

**Benefits of Multithreading**

* **Responsiveness**: Continued execution even if part of the process is blocked.
* **Resource Sharing**: Easier sharing of resources within a process.
* **Economy**: Lower overhead compared to process creation.
* **Scalability**: Utilizes multicore architectures effectively.

**Programming Challenges**

* **Dividing Activities**: Efficiently dividing tasks among threads.
* **Balance**: Ensuring equal contribution from each thread.
* **Data Splitting**: Dividing data for separate processing.
* **Data Dependency**: Managing shared data access.
* **Testing and Debugging**: Dynamic behavior makes debugging complex

**Threads and Processes**

* **Process**: Defines address space and general attributes.
* **Thread**: Defines a sequential execution stream within a process.
* **Scheduling**: Threads are the unit of scheduling; processes are containers for threads.

**Amdahl’s Law**

* **Definition**: Describes the maximum speedup achievable through parallelization.
* **Formula**: Speedup = 1 / (S + (P/N)), where S is the serial portion, P is the parallel portion, and N is the number of processors.
* **Implications**: Diminishing returns as more processors are added