Chapter 4 Important Points

1. Differences between a **thread** and a **process**
2. Benefits of **multi-threading** over **single threading**
   (a) Responsiveness
   (b) Resource Sharing
   (c) Economy
   (d) Scalability
3. Components of a thread (resources it uses)
4. Examples in which multi-threading is beneficial and in which it is not
5. Challenges of programming multi-threaded applications
   - Dividing activities
   - Balance
   - Data splitting
   - Data dependency
   - Testing and debugging
6. Difference between **parallelism** and **concurrency**
7. **Amdahl's Law** and its applications
8. Difference between **data parallelism** and **task parallelism** and examples of each
9. Threading models
   - **User threads** versus **kernel threads** - what are they, how are they different
   - **Many to one** thread model
   - **One to one** thread model
   - **Many to many** thread model
   - **Two-level thread** model
10. Thread libraries - what are they and what are examples
    - **Asynchronous** versus **synchronous** threading
    - **POSIX threads** - what it is
11. Implicit threading -
    (a) definition, examples
    (b) thread pools
    (c) **OpenMP** - what is it and how is it used
12. Threading issues
    (a) Interaction between `fork()` and multi-threading
    (b) How `exec()` works with multiple threads in a process
    (c) What are **signals** and what is the problem with sending and handling signals with multi-threaded programs
    (d) **Thread cancellation** - types of cancellation and how it is handled
13. Linux threads - `clone()` and how it works.