1. Child Process – fork()

A child process is a copy of the current process created using the fork() system call in Unix/Linux. It enables a program to run two parallel processes: the parent and the child.

- When fork() is called:
    - The child process receives 0.
    - The parent process receives the PID of the child.
    - On failure, it returns -1.

2. Handling Common Signals

Signals are a way for the operating system or another process to notify a program of an event (such as termination, interruption, or errors).

Common signals:

- SIGINT – Sent when user types Ctrl+C
- SIGTERM – Requests process termination
- SIGKILL – Forcefully kills a process (cannot be caught or ignored)
- SIGSEGV – Segmentation fault (invalid memory access)

Why handle signals?

- To clean up resources before exit
- To avoid data loss
- To log or gracefully shut down applications

3. Exploring Different Kernel Crashes

A kernel crash occurs when the core part of the operating system encounters an unrecoverable error.

Causes of kernel crashes:

- NULL pointer dereference in kernel space
- Stack overflow
- Invalid memory access by kernel modules or drivers
- Race conditions in kernel code

4. Time Complexity

Time complexity describes how the time required for a program grows with the input size. It helps in evaluating the performance of algorithms.

Common time complexities:

- O(1) – Constant time: fast and efficient
- O(n) – Linear time: grows with input size
- O(log n) – Logarithmic time: efficient searching
- $O(n^2)$ – Quadratic time: slower for large inputs

Importance:
Time complexity is used to compare algorithms and determine which one is best suited for a task based on performance.

5. Locking Mechanism – Mutex / Spinlock

When multiple threads or processes access shared resources, a locking mechanism is required to avoid race conditions and maintain data integrity.

Mutex (Mutual Exclusion)

- A thread waits (blocks) if the lock is already held.
- Suitable when a thread might sleep while holding the lock.
- Used for longer critical sections.

Spinlock

- A thread keeps checking (spinning) until the lock is released.
- No blocking; uses CPU while waiting.
- Suitable for short critical sections.