**3. Know about the following topics and explore them (Write a note on your understandings)**

**Areas for exploration,  
● Child process - fork()  
● Handling common signals  
● Exploring different Kernel crashes  
● Time complexity  
● Locking mechanism - mutex/spinlock**

**Child Process – fork():**

fork() is a system call in Linux/Unix that is used to create a new process. This new process is called the **child**, and it's basically a copy of the **parent** process.

* After fork() is called, both parent and child start running from the same line of code.
* It returns 0 in the child process and the child's PID in the parent process.
* Mostly used when we want to create parallel/concurrent tasks like running background processes or building server-client programs.

pid\_t pid = fork();

if (pid == 0) {

// Child process

} else {

// Parent process

}

**Handling Common Signals:**

Signals are used to notify a process when some event happens, like when we press Ctrl+C or there's an error. Some common ones are:

* SIGINT – Sent when we press Ctrl+C.
* SIGTERM – A polite way to ask a process to stop.
* SIGSEGV – Happens when there’s an invalid memory access (like segmentation fault).
* SIGKILL – Forcefully kills a process (this one can’t be caught or ignored).
* SIGCHLD – Sent to the parent when a child process ends or stops.

**Different Kernel Crashes**

Kernel crashes usually happen due to bugs in kernel-level code (like device drivers). This can lead to **kernel panic**, where the system stops working or restarts.  
Some common reasons:

* Null pointer access in kernel code
* Invalid memory operations
* Deadlocks between processes or threads
* Buffer overflows in the kernel

**Time Complexity**

Time complexity tells us how an algorithm's runtime grows as the input size increases. It's a way to measure how efficient an algorithm is.

Some common time complexities:

* O(1) – Constant time (doesn’t change with input size)
* O(log n) – Logarithmic (like binary search)
* O(n) – Linear (loops through input once)
* O(n log n) – Slightly more than linear (like merge sort)
* O(n²) – Quadratic (nested loops – like bubble sort)

**Locking Mechanism – Mutex / Spinlock**

When multiple threads try to access the same resource, we need to protect it using locks.

**Mutex (Mutual Exclusion)**

* If a thread can’t get the lock, it waits (sleeps).
* Good when the wait is long or unpredictable.
* Used in both user-space and kernel-space.

**Spinlock**

* If it can’t get the lock, it keeps checking (spinning) in a loop.
* Best when the wait is very short.
* Mostly used in the kernel or real-time systems.
* Not good for long waits since it wastes CPU cycles.