**fork, exec, and clone System Calls in Linux are used for process creation and management in Linux.**

**1. fork()**

* **The new process is an exact copy of the parent. It has its own memory space, file descriptors, and process ID with COW.**
* **Returns 0 in the child and child PID in the parent. Both parent and child continue execution from the same point.**

**2. exec()**

* **Replaces the current process with a new program. It does not return on success (the old process is overwritten). After fork(), the child process usually calls exec() to load and run a different program. The old process image (code, data, stack) is completely replaced. Used after fork() to execute a different program in the child.**

**Common Variants of exec:**

* **execl(), execv(), execle(), execve(), execlp(), execvp()**

**3. clone()**

* **More flexible than fork(), used for creating threads or lightweight processes. Allows sharing specific resources between parent and child (e.g., memory, file descriptors).**
* **Used internally by pthread\_create() for thread creation. Requires a stack for the child process.**

**Example: Using fork() and exec() Together**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**#include <sys/types.h>**

**#include <sys/wait.h>**

**int main() {**

**pid\_t pid = fork();**

**if (pid < 0) {**

**perror("fork failed");**

**exit(1);**

**} else if (pid == 0) { // Child process**

**printf("Child process: Executing ls\n");**

**execlp("ls", "ls", "-l", NULL);**

**perror("exec failed"); // Only runs if exec fails**

**exit(1);**

**} else { // Parent process**

**printf("Parent process: Waiting for child\n");**

**wait(NULL); // Wait for child to complete**

**printf("Parent process: Child finished execution\n");**

**}**

**return 0;**

**}**

**Why Use fork() Before exec()?**

1. **Separation of Process Creation & Execution**
   * **fork() allows the parent to continue running while the child loads a new program.**
2. **Process Control**
   * **Parent can handle errors before executing a new program.**
   * **Can set up redirections, signal handling, or other configurations.**
3. **Parallel Execution**
   * **The parent and child can run concurrently.**