**Differences Between a Zombie Process and an Orphan Process**

| **Aspect** | **Zombie Process** | **Orphan Process** |
| --- | --- | --- |
| **Definition** | A zombie process is a process that has finished execution but still has an entry in the process table because its parent has not read its exit status (via wait() or waitpid()). | An orphan process is a process whose parent process has finished execution (or terminated) before it, making it "orphaned." The system assigns a new parent, typically the **init process** (PID 1). |
| **State** | It is a process that has completed execution but has not been fully removed from the process table. Its status is marked as Z (zombie). | It is a process that is still running after its parent has terminated. It is not a finished process but continues to run. |
| **Exit Status** | The process has already terminated, but its exit status is not yet collected by the parent process. | The process is still running and hasn't finished yet, so it doesn't have an exit status. |
| **Parent Process Status** | The parent process is still alive but has not collected the exit status of the terminated child process. | The parent process has terminated, leaving the orphan process without a parent. |
| **Process Table** | Zombie processes remain in the process table until the parent collects their exit status. | Orphan processes do not stay in the process table indefinitely. They continue to run, but their parent is replaced by the **init process**. |
| **Handling by System** | Zombies are eventually cleaned up by the **init process** (PID 1), or if the parent process reaps them. | Orphan processes are automatically adopted by the **init process** (PID 1) as their new parent. |
| **System Resource Impact** | Zombies consume a slot in the process table but do not consume CPU or memory. They just take up a PID entry. | Orphans do not consume a process table slot because they are re-parented. They continue running and consuming system resources like memory and CPU until they terminate. |
| **Common Causes** | Occurs when a parent process does not call wait() to collect the exit status of its child process. | Occurs when a parent process terminates while its child process is still running. |