**Process**

Zombie Process:

A process which has finished the execution but still has entry in the process table to report to its parent process is known as a zombie process. A child process always first becomes a zombie before being removed from the process table. The parent process reads the exit status of the child process which reaps off the child process entry from the process table.

These resources remain so that parent processes that want to check up on

the status of their children can obtain information relating to the life and termination

of those processes. Once the parent does so, the kernel cleans up the process for good

and the zombie ceases to exist.

However, anyone who has used Unix for a good while is sure to have seen zombie

processes sitting around. These processes, often called ghosts, have irresponsible pa‐

rents. If your application forks a child process, it is your application’s responsibility

(unless it is short-lived, as you will see shortly) to wait on the child, even if it will merely

discard the information gleaned. Otherwise, all of your process’s children will become

ghosts and live on, crowding the system’s process listing and generating disgust at your

application’s sloppy implementation.

What happens, however, if the parent process dies before the child, or if it dies before

it has a chance to wait on its zombie children? Whenever a process terminates, the Linux

kernel walks a list of its children and reparents all of them to the init process (the process

with a pid value of 1). This guarantees that no process is ever without an immediate

parent. The init process, in turn, periodically waits on all of its children, ensuring that

none remain zombies for too long—no ghosts! Thus, if a parent dies before its children

or does not wait on its children before exiting, the child processes are eventually reparented to init and waited upon, allowing them to fully exit. Although doing so is still considered good practice, this safeguard means that short-lived processes need not worry excessively about waiting on all of their children.

Orphan Process:

A process whose parent process no more exists i.e. either finished or terminated without waiting for its child process to terminate is called an orphan process.

In the following code, parent finishes execution and exits while the child process is still executing and is called an orphan process now.

However, the orphan process is soon adopted by init process, once its parent process dies.