not be allocated the same PID. On modern UNIX-like systems (that comply with SUSv3 specification in this respect), the following special case applies: if the parent *explicitly* ignores SIGCHLD by setting its handler to `SIG_IGN` (rather than simply ignoring the signal by default) or has the `SA_NOCLDWAIT` flag set, all child exit status information will be discarded and no zombie processes will be left.[1]

Zombies can be identified in the output from the Unix `ps` command by the presence of a "`Z`" in the "STAT" column.[2] Zombies that exist for more than a short period of time typically indicate a bug in the parent program, or just an uncommon decision to not reap children (see example). If the parent program is no longer running, zombie processes typically indicate a bug in the operating system. As with other resource leaks, the presence of a few zombies is not worrisome in itself, but may indicate a problem that would grow serious under heavier loads. Since there is no memory allocated to zombie processes – the only system memory usage is for the process table entry itself – the primary concern with many zombies is not running out of memory, but rather running out of process table entries, concretely process ID numbers.

To remove zombies from a system, the SIGCHLD signal can be sent to the parent manually, using the `kill` command. If the parent process still refuses to reap the zombie, and if it would be fine to terminate the parent process, the next step can be to remove the parent process. When a process loses its parent, `init` becomes its new parent. `init` periodically executes the `wait` system call to reap any zombies with `init` as parent.

When a process ends via `exit`, all of the memory and resources associated with it are deallocated so they can be used by other processes. However, the process's entry in the process table remains. The parent can read the child's exit status by executing the `wait` system call, whereupon the zombie is removed. The `wait` call may be executed in sequential code, but it is commonly executed in a handler for the SIGCHLD signal, which the parent receives whenever a child has died.

After the zombie is removed, its process identifier (PID) and entry in the process table can then be reused. However, if a parent fails to call `wait`, the zombie will be left in the process table, causing a resource leak. In some situations this may be desirable – the parent process wishes to continue holding this resource – for example if the parent creates another child process it ensures that it will not be allocated the same PID. On modern UNIX-like systems (that comply with SUSv3 specification in this respect), the following special case applies: if the parent *explicitly* ignores SIGCHLD by setting its handler to `SIG_IGN` (rather than simply ignoring the signal by default) or has the `SA_NOCLDWAIT` flag set, all child exit status information will be discarded and no zombie processes will be left.[1]

Zombies can be identified in the output from the Unix `ps` command by the presence of a "`Z`" in the "STAT" column.[2] Zombies that exist for more than a short period of time typically indicate a bug in the parent program, or just an uncommon decision to not reap children (see example). If the parent program is no longer running, zombie processes typically indicate a bug in the operating system. As with other resource leaks, the presence of a few zombies is not worrisome in itself, but may indicate a problem that would grow serious under heavier loads. Since there is no memory allocated to zombie processes – the only system memory usage is for the process table entry itself –