Group_7 Assignment_6 report

(1) **(2 pt)** Write a program that creates a zombie process.
(2) **(1 pt)** Verifies that the process is a zombie in your code using **ps(1) command**.

- Below is the outcome of the process

```
mygodimato@RPi400Group7:~/Advanced-UNIX-Programming_Student/assignment6 $ ./assignment6
Parent process is sleeping...
Child process is terminating...
Zombie process found: 34229 Z+
Parent process is doing some work...
Parent process has cleaned up the zombie.
```

  - In the outcome, we can find that the zombie process has ID 34229

(3) **(1 pt)** Describe your implementation in your report.

- Below is the implementation

```c
pid_t child_pid;
char command[256];

// Fork a child process
child_pid = fork();

if (child_pid == 0) {
  // Child process : immediately exit to become a zombie
  printf("Child process is terminating...\n");
  _exit(0);
```

- First, we create a child process by `fork` and call the `_exit` to terminate it and make it a zombie process.

```
} else {
  // Parent process
  printf("Parent process is sleeping...\n");
  sleep(10); // Sleep to ensure child process terminates first

  // Use ps(1) command within the program to check for zombie processes
  sprintf(command, "ps -o pid,state -p %d", child_pid);
  FILE *ps_output = popen(command, "r");
  if (ps_output == NULL) {
    perror("popen failed");
    exit(EXIT_FAILURE);
  }

  // Read the output of the ps(1) command
  char buf[1024];
  while (fgets(buf, sizeof(buf), ps_output) != NULL) {      You, 12 hours a
    // Check if the process state is 'Z' for zombie
    if (strstr(buf, "Z")) {
      printf("Zombie process found: %s", buf);
      // This is where we verify that the child is a zombie process
    }
  }
```

- Second, for the parent process, it will wait for 10 sec to ensure child process terminates first, then it will call the `ps(1)` to get all the process.
  - for calling the `ps(1)` we choose to use `popen` , `perror` and `pclose` to deal with the interaction with terminal.
  - After getting the `ps(1)` outcome, we choose to use `fget` to get line by line of data and use `strstr()` to find the “Z” in the `ps(1)`, and print it to the terminal.

```c
    // Close the ps command pipe
    pclose(ps_output);

    // Parent does some work
    printf("Parent process is doing some work...\n");      You, 12 hours ago

    // Clean up the zombie by waiting for the child process to change state
    waitpid(child_pid, NULL, 0);

    printf("Parent process has cleaned up the zombie.\n");
}
```

- Last, we call `pclose` to close the ps command, use `waitpid` to clean up zombie, and call return to terminate the whole process.