Assignment 2 Report

COMP 3300 Operating Systems 104435995 Bilal Malik

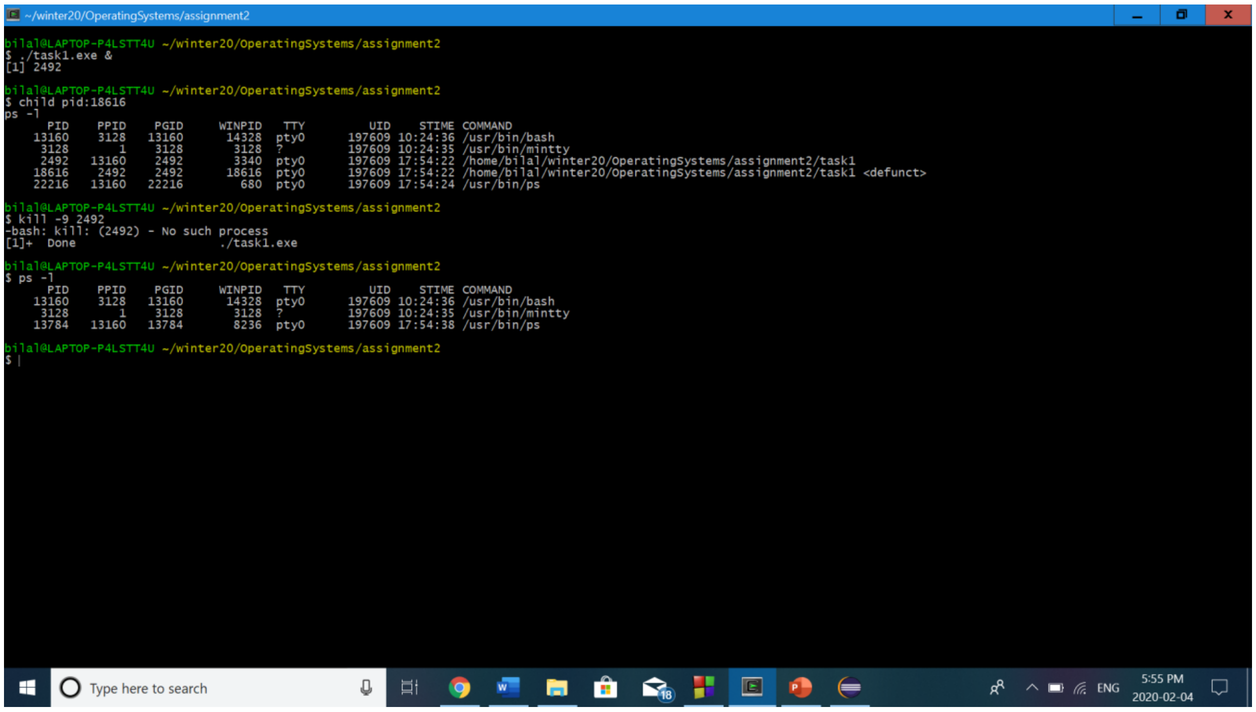
January 30, 2020

I confirm that I will keep the content of this assignment confidential. I confirm that I have not received any unauthorized assistance in preparing for or writing this assignment. I acknowledge that a mark of 0 may be assigned for copied work. Bilal Malik 104435995

Task 1

For task 1, I utilized the fork system call to break up the execution between two separate processes. The child prints its process ID then attempts to terminate itself. Unfortunately, for the child process, the parent is sleeping for ten seconds. It is during the situation, that the child process now falls under the zombie process status. This is because the child attempts to return certain types of data it holds to its parent before it can terminate, but the parent process is in a blocked status due to the nature of the sleep function. Until this task of sleeping is complete, the parent cannot tend to its child's needs. Hence the child process is in a zombie status, it is not dead yet, but cannot do anything else but await its termination. I then locate the zombie process in my process list by running the 'ps -l' command. When the command is run, all processes are listed, zombie processes are listed with a <defunct> next to there path name. Thus, I locate the zombie process, then find its parent by looking under the ppid column.

The following is an execution of the code, proceeded by the killing of the necessary process:



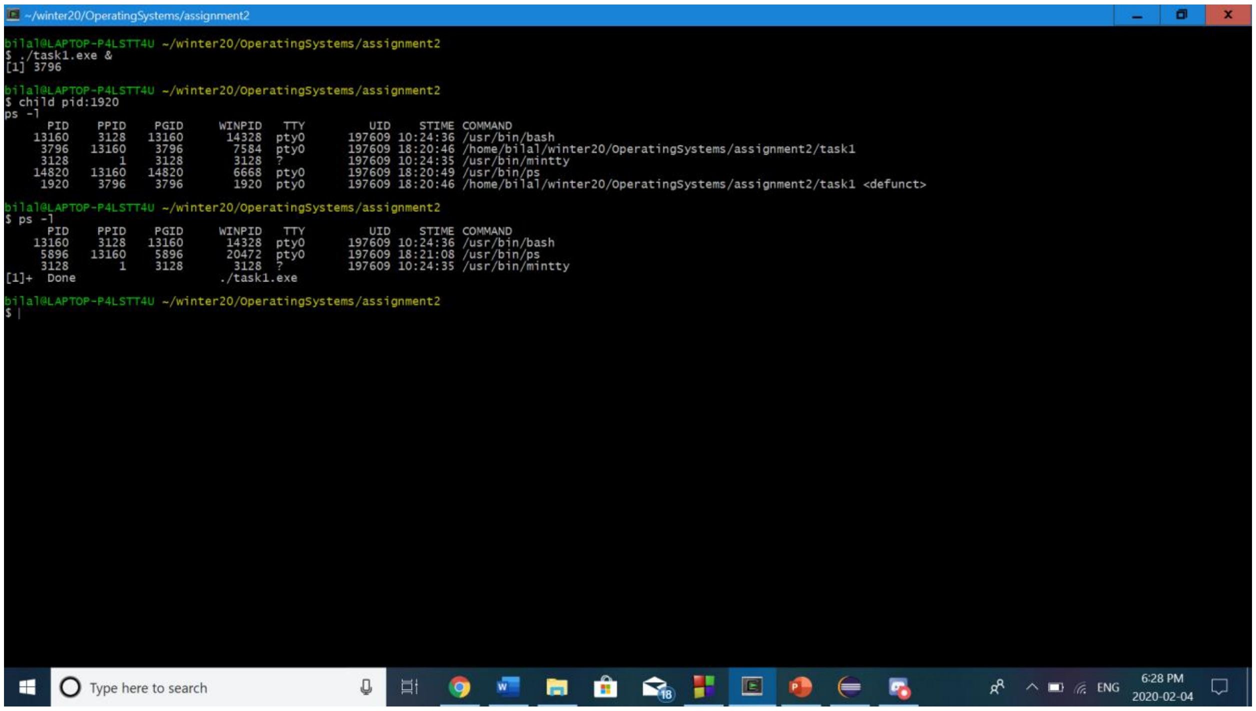
~/winter20/OperatingSystems/assignment2

```
bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$ ./task1.exe &
[1] 2492

bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$ child pid:18616
ps -l
      PID    PPID    PGID     WINPID   TTY         UID    STIME COMMAND
    13160    3128   13160      14328  pty0      197609 10:24:36 /usr/bin/bash
     3128       1    3128       3128  ?         197609 10:24:35 /usr/bin/mintty
     2492   13160    2492       3340  pty0      197609 17:54:22 /home/bilal/winter20/OperatingSystems/assignment2/task1
    18616    2492    2492      18616  pty0      197609 17:54:22 /home/bilal/winter20/OperatingSystems/assignment2/task1 <defunct>
    22216   13160   22216        680  pty0      197609 17:54:24 /usr/bin/ps

bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$ kill -9 2492
-bash: kill: (2492) - No such process
[1]+  Done                    ./task1.exe

bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$ ps -l
      PID    PPID    PGID     WINPID   TTY         UID    STIME COMMAND
    13160    3128   13160      14328  pty0      197609 10:24:36 /usr/bin/bash
     3128       1    3128       3128  ?         197609 10:24:35 /usr/bin/mintty
    13784   13160   13784       8236  pty0      197609 17:54:38 /usr/bin/ps

bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$ |
```

Here, the zombie process 18616, has a parent process 2492. A ‘kill -9 2492’ command is called, followed by another ‘ps -l’ command; here the zombie process and its parent process are now gone. Another way of confirming this, is by looking at the command path. The only two processes running from a path carrying the current directory and the name of the execution, are no where to be found on the second ‘ps -l’ call.

We can as well see the handling of zombie processes, and why they are called that by running the execution through without killing the parent. Eventually the parent will wake up and accept the child’s information, allowing the child to terminate, before terminating itself.

The following is the execution of the program followed by a ‘ps -l’ call where the zombie process and its parent are till alive, followed by a ‘ps -l’ call after there termination:

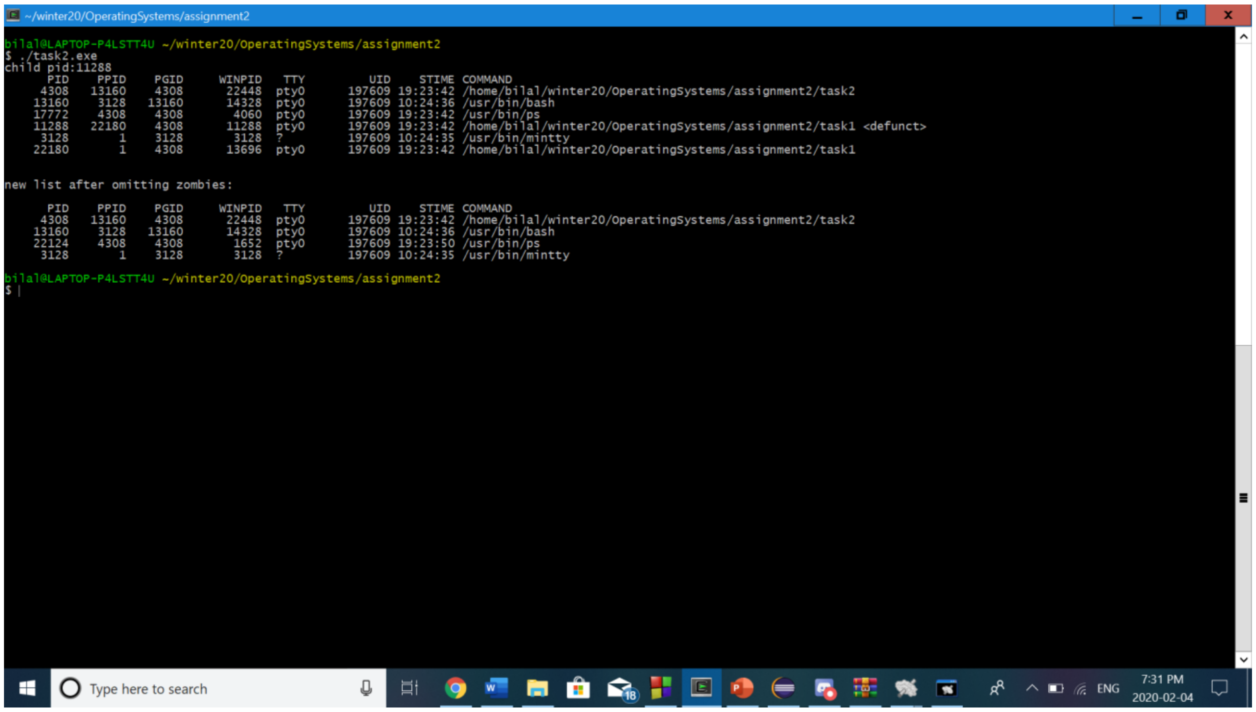


```
bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$ ./task1.exe &
[1] 3796

bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$ child pid:1920
ps -l
      PID    PPID    PGID     WINPID   TTY         UID    STIME COMMAND
    13160    3128   13160      14328  pty0      197609 10:24:36 /usr/bin/bash
     3796   13160    3796       7584  pty0      197609 18:20:46 /home/bilal/winter20/OperatingSystems/assignment2/task1
     3128       1    3128       3128  ?         197609 10:24:35 /usr/bin/mintty
    14820   13160   14820       6668  pty0      197609 18:20:49 /usr/bin/ps
     1920    3796    3796       1920  pty0      197609 18:20:46 /home/bilal/winter20/OperatingSystems/assignment2/task1 <defunct>

bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$ ps -l
      PID    PPID    PGID     WINPID   TTY         UID    STIME COMMAND
    13160    3128   13160      14328  pty0      197609 10:24:36 /usr/bin/bash
     5896   13160    5896      20472  pty0      197609 18:21:08 /usr/bin/ps
     3128       1    3128       3128  ?         197609 10:24:35 /usr/bin/mintty
[1]+  Done                    ./task1.exe

bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$ |
```

Task 2

In task 2, I utilized the system function which outputs a string directly to the terminal to be run as a command. These commands are run based on the bash scripting language, thus I had to manipulate the bash language to output the correct process and kill its parent. The task 2 program begins with a system function call executing the task 1 programs executable in the background using the ampersand, then calling the ‘ps -l’ command to list all the processes. The process then sleeps for 3 seconds to give the user time to realize which process is going to be eliminated before finally eliminating the process. It then sleeps for 5 seconds; this is 5 plus the three from the previous sleep leaving us at 8 seconds. Where the process then lists the processes in the current directory and terminating. At this stage, had the ‘kill’ command failed, the two processes would still be seen in the ‘ps -l’ call. This is because the parent sleeps for 10 seconds in the original program, here the two separate sleep calls amount to eight seconds. Thus, if the program fails, we will know because of timing. This will not include the removed process and its child who was a zombie. All in all, this task automates the part of the firs task where you must manually find the process to kill. As well, this automated zombie process remover works with multiple zombie running in the directory. Zombie processes that weren't necessarily made by its own system call to task1.

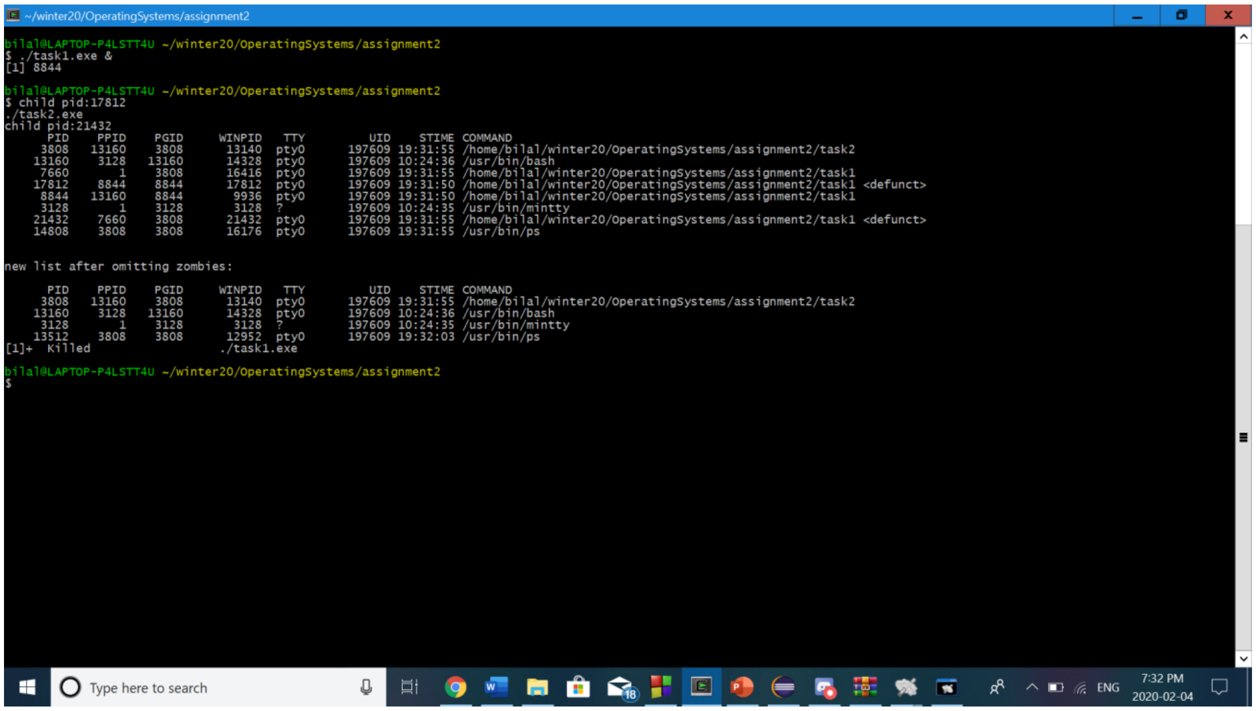
The following is the execution of the task2 program, no input is given to the command line on its execution by me, task2 can still be seen because at his stag it is outputting 'ps -l' to the screen:



```
~/winter20/OperatingSystems/assignment2

bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$ ./task2.exe
child pid:11288
      PID    PPID    PGID     WINPID   TTY         UID    STIME COMMAND
     4308   13160    4308      22448  pty0      197609 19:23:42 /home/bilal/winter20/OperatingSystems/assignment2/task2
    13160    3128   13160      14328  pty0      197609 10:24:36 /usr/bin/bash
    17772    4308    4308       4060  pty0      197609 19:23:42 /usr/bin/ps
    11288   22180    4308      11288  pty0      197609 19:23:42 /home/bilal/winter20/OperatingSystems/assignment2/task1 <defunct>
     3128       1    3128       3128  ?         197609 10:24:35 /usr/bin/mintty
    22180       1    4308      13696  pty0      197609 19:23:42 /home/bilal/winter20/OperatingSystems/assignment2/task1

new list after omitting zombies:

      PID    PPID    PGID     WINPID   TTY         UID    STIME COMMAND
     4308   13160    4308      22448  pty0      197609 19:23:42 /home/bilal/winter20/OperatingSystems/assignment2/task2
    13160    3128   13160      14328  pty0      197609 10:24:36 /usr/bin/bash
    22124    4308    4308       1652  pty0      197609 19:23:50 /usr/bin/ps
     3128       1    3128       3128  ?         197609 10:24:35 /usr/bin/mintty

bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$ |
```

As can be seen, the two processes running in task1, the zombie and parent are gone in the secondary 'ps -l' call. The task2 process still exists, as the ps -l call happen within the task2 program before its exit.

Here, I chose to run task1 in the background, before executing the task2. Now there are two zombie processes, each with corresponding parent processes. The task2 program is till successful in removing all these zombie processes from the working directory, as can be seen by the final 'ps -l' call.



```
~/winter20/OperatingSystems/assignment2

bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$ ./task1.exe &
[1] 8844

bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$ child pid:17812
./task2.exe
child pid:21432
      PID    PPID    PGID     WINPID   TTY         UID    STIME COMMAND
     3808   13160    3808      13140  pty0      197609 19:31:55 /home/bilal/winter20/OperatingSystems/assignment2/task2
    13160    3128   13160      14328  pty0      197609 10:24:36 /usr/bin/bash
     7660       1    3808      16416  pty0      197609 19:31:55 /home/bilal/winter20/OperatingSystems/assignment2/task1
    17812    8844    8844      17812  pty0      197609 19:31:50 /home/bilal/winter20/OperatingSystems/assignment2/task1 <defunct>
     8844   13160    8844       9936  pty0      197609 19:31:50 /home/bilal/winter20/OperatingSystems/assignment2/task1
     3128       1    3128       3128  ?         197609 10:24:35 /usr/bin/mintty
    21432    7660    3808      21432  pty0      197609 19:31:55 /home/bilal/winter20/OperatingSystems/assignment2/task1 <defunct>
    14808    3808    3808      16176  pty0      197609 19:31:55 /usr/bin/ps


new list after omitting zombies:

      PID    PPID    PGID     WINPID   TTY         UID    STIME COMMAND
     3808   13160    3808      13140  pty0      197609 19:31:55 /home/bilal/winter20/OperatingSystems/assignment2/task2
    13160    3128   13160      14328  pty0      197609 10:24:36 /usr/bin/bash
     3128       1    3128       3128  ?         197609 10:24:35 /usr/bin/mintty
    13512    3808    3808      12952  pty0      197609 19:32:03 /usr/bin/ps
[1]+  Killed                  ./task1.exe

bilal@LAPTOP-P4LSTT4U ~/winter20/OperatingSystems/assignment2
$
```

Thus, the program is truly a zombie remover and removes all zombie processes and parent from the current directory.