Terminator 2

For this experiment my hypothesis is that yes a parent process cant terminate one of its child processes but a grandparent process for instance Project1B can not terminate a grandchild process in Project1A. In order for a parent to terminate one of its child processes you can use the kill() function. To test the first question in this experiment about a parent terminating one child I decided to set up an implementation of the kill() function in my A1 child process of Project1A program. To do this first the signal.h and sys/stat.h libraries were imported to use the kill function. The kill function takes the PID of the process you want to kill and then the signal term that you want to call. I used the PID of the first child process and the SIGTERM signal. SIGTERM signal is a generic signal used to cause program termination. The output to show this experiment is shown in figure 1, you can see child A1 is terminated by the parent process and doesn’t execute it’s I/O as the A1 output file is blank. Child process A2 still executes it’s i/o as seen in the A2 output file.

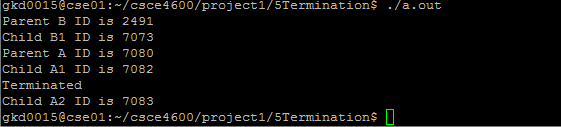


Figure 1

**Part 1 source code for Project1A** modified to show experiment

**Project1A.c**

//project 1 & Terminator 2 question 5

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <signal.h>

#include <sys/stat.h>

#include <fcntl.h>

pid\_t A1, A2;

//first child process

int fork1(){

int fd = open("A1\_file.txt", O\_WRONLY | O\_APPEND | O\_CREAT, 0644);

//child process created

A1 = fork ();

int A1pid = getpid();

if(A1!=0){

printf("Child A1 ID is %d\n", (int) A1);

//child process terminated by parent

kill (A1pid, SIGTERM);

dup2(fd, 1);

execl("/bin/ls", "ls", "-l", (char \*)0);

}

close(fd);

}

//second child process

int fork2(){

int fd = open("A2\_file.txt", O\_WRONLY | O\_APPEND | O\_CREAT, 0644);

//child 2 created

A2= fork ();

if(A2 != 0){

printf("Child A2 ID is %d\n", (int) A2);

dup2(fd, 1);

execl("/bin/ps", "ps", "-ael", (char \*)0);

}

close(fd);

}

int main(){

printf("Parent A ID is %d\n", (int) getppid());

fork1();

fork2();

}

**A1 Output**

(blank)

**A2 Output**

F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD

4 S 0 1 0 0 80 0 - 4153 poll\_s ? 00:24:08 init

1 S 0 2 0 0 80 0 - 0 kthrea ? 00:00:02 kthreadd

1 S 0 3 2 0 80 0 - 0 run\_ks ? 00:50:46 ksoftirqd/0

1 S 0 6 2 0 -40 - - 0 cpu\_st ? 00:05:27 migration/0

5 S 0 7 2 0 -40 - - 0 watchd ? 00:00:29 watchdog/0

1 S 0 8 2 0 -40 - - 0 cpu\_st ? 00:05:04 migration/1

1 S 0 10 2 0 80 0 - 0 run\_ks ? 00:05:17 ksoftirqd/1

5 S 0 12 2 0 -40 - - 0 watchd ? 00:00:23 watchdog/1

1 S 0 13 2 0 -40 - - 0 cpu\_st ? 00:04:48 migration/2

1 S 0 15 2 0 80 0 - 0 run\_ks ? 00:04:52 ksoftirqd/2

5 S 0 16 2 0 -40 - - 0 watchd ? 00:00:23 watchdog/2

1 S 0 17 2 0 60 -20 - 0 rescue ? 00:00:00 cpuset

1 S 0 18 2 0 60 -20 - 0 rescue ? 00:00:00 khelper

5 S 0 19 2 0 80 0 - 0 devtmp ? 00:00:00 kdevtmpfs

1 S 0 20 2 0 60 -20 - 0 rescue ? 00:00:00 netns

1 S 0 22 2 0 80 0 - 0 bdi\_sy ? 00:00:16 sync\_supers

1 S 0 23 2 0 80 0 - 0 bdi\_fo ? 00:00:01 bdi-default

1 S 0 24 2 0 60 -20 - 0 rescue ? 00:00:00 kintegrityd

1 S 0 25 2 0 60 -20 - 0 rescue ? 00:00:00 kblockd

1 S 0 26 2 0 60 -20 - 0 rescue ? 00:00:00 ata\_sff

1 S 0 27 2 0 80 0 - 0 hub\_th ? 00:00:00 khubd

1 S 0 28 2 0 60 -20 - 0 rescue ? 00:00:00 md

1 S 0 30 2 0 80 0 - 0 watchd ? 00:00:05 khungtaskd

1 S 0 31 2 0 80 0 - 0 kswapd ? 00:03:56 kswapd0

1 S 0 32 2 0 60 -20 - 0 rescue ? 00:00:00 vmstat

1 S 0 33 2 0 85 5 - 0 ksm\_sc ? 00:00:00 ksmd

1 S 0 34 2 0 99 19 - 0 khugep ? 00:00:00 khugepaged

1 S 0 35 2 0 80 0 - 0 fsnoti ? 00:00:00 fsnotify\_mark

1 S 0 36 2 0 80 0 - 0 ecrypt ? 00:00:00 ecryptfs-kthrea

1 S 0 37 2 0 60 -20 - 0 rescue ? 00:00:00 crypto

1 S 0 45 2 0 60 -20 - 0 rescue ? 00:00:00 kthrotld

1 S 0 46 2 0 80 0 - 0 scsi\_e ? 00:00:00 scsi\_eh\_0

1 S 0 47 2 0 80 0 - 0 scsi\_e ? 00:00:00 scsi\_eh\_1

1 S 0 69 2 0 60 -20 - 0 rescue ? 00:00:00 devfreq\_wq

1 S 0 196 2 0 60 -20 - 0 rescue ? 00:00:00 ttm\_swap

1 S 0 271 2 0 60 -20 - 0 rescue ? 00:00:00 mpt\_poll\_0

1 S 0 272 2 0 60 -20 - 0 rescue ? 00:00:00 mpt/0

1 S 0 302 2 0 80 0 - 0 scsi\_e ? 00:00:00 scsi\_eh\_2

1 S 0 353 2 0 80 0 - 0 kjourn ? 00:01:12 jbd2/sda1-8

……………………………………………….

Question 2

It is not possible for a parent process in a program, Project1B, launching another program, Project1A, to terminate it’s grandchildren processes, child processes in Project1A. I expect this because the only process that can acquire exit statuses from its distant Nth generation grandchildren is the init process. Even that is a special case rule implemented by the kernel. A process can only wait for its direct children to die, it cannot wait for its children’s children to die. There really was no way to show this by modifying code as it is not possible in this setting. One way to work around this issue would to just have the child of the parent process terminate its own child, or grandchild of the original parent process. I tried to use the kill() function to terminate the child process of the original parent in ProjectB1 but the child process still executed A1 and A2 in ProjectA1. This can be seen by figure 1 showing the output that the kill() function did terminate something but as you can see by the output of A1 and A2 the processes were not terminated.

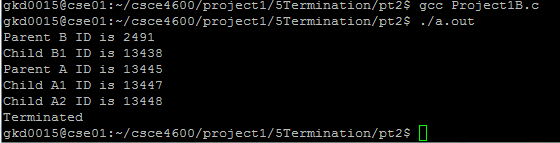


Figure 1

**Source code for ProjectB1.c** that was modified

//Project 1 & Termination 5 part 2

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <signal.h>

#include <sys/stat.h>

#include <fcntl.h>

pid\_t B1;

int fork1(){

int fd = open("B1\_file.txt", O\_WRONLY | O\_APPEND | O\_CREAT, 0644);

//child process to launch Project1A

B1 = fork ();

int B1pid = getpid();

if(B1!=0){

printf("Parent B ID is %d\n", (int) getppid());

printf("Child B1 ID is %d\n", (int) B1);

int status = system("gcc Project1A.c");

int status2 = system("./a.out");

//trying to kill child processes A1 and A2

kill (B1pid, SIGTERM);

dup2(fd, 1);

execl("/bin/date", "date", 0, (char\*)0);

}

close(fd);

}

int main(){

fork1();

}

**A1 Output**

total 28

-rw------- 1 gkd0015 gkd0015 0 Mar 23 11:34 A1\_file.txt

-rw------- 1 gkd0015 gkd0015 23277 Mar 23 11:34 A2\_file.txt

-rwx------ 1 gkd0015 gkd0015 7660 Mar 23 11:34 a.out

-rw------- 1 gkd0015 gkd0015 87 Mar 22 15:06 B1\_file.txt

-rw-rw---- 1 gkd0015 gkd0015 761 Mar 22 15:11 Project1A.c

-rw-rw---- 1 gkd0015 gkd0015 562 Mar 22 15:14 Project1B.c

**A2 Output**

F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD

4 S 0 1 0 0 80 0 - 4153 poll\_s ? 00:24:09 init

1 S 0 2 0 0 80 0 - 0 kthrea ? 00:00:02 kthreadd

1 S 0 3 2 0 80 0 - 0 run\_ks ? 00:51:09 ksoftirqd/0

1 S 0 6 2 0 -40 - - 0 cpu\_st ? 00:05:28 migration/0

5 S 0 7 2 0 -40 - - 0 watchd ? 00:00:29 watchdog/0

1 S 0 8 2 0 -40 - - 0 cpu\_st ? 00:05:04 migration/1

1 S 0 10 2 0 80 0 - 0 run\_ks ? 00:05:18 ksoftirqd/1

5 S 0 12 2 0 -40 - - 0 watchd ? 00:00:23 watchdog/1

1 S 0 13 2 0 -40 - - 0 cpu\_st ? 00:04:49 migration/2

1 S 0 15 2 0 80 0 - 0 run\_ks ? 00:04:53 ksoftirqd/2

5 S 0 16 2 0 -40 - - 0 watchd ? 00:00:23 watchdog/2

1 S 0 17 2 0 60 -20 - 0 rescue ? 00:00:00 cpuset

1 S 0 18 2 0 60 -20 - 0 rescue ? 00:00:00 khelper

5 S 0 19 2 0 80 0 - 0 devtmp ? 00:00:00 kdevtmpfs

1 S 0 20 2 0 60 -20 - 0 rescue ? 00:00:00 netns

1 S 0 22 2 0 80 0 - 0 bdi\_sy ? 00:00:16 sync\_supers

1 S 0 23 2 0 80 0 - 0 bdi\_fo ? 00:00:01 bdi-default

1 S 0 24 2 0 60 -20 - 0 rescue ? 00:00:00 kintegrityd

1 S 0 25 2 0 60 -20 - 0 rescue ? 00:00:00 kblockd

1 S 0 26 2 0 60 -20 - 0 rescue ? 00:00:00 ata\_sff

1 S 0 27 2 0 80 0 - 0 hub\_th ? 00:00:00 khubd

1 S 0 28 2 0 60 -20 - 0 rescue ? 00:00:00 md

1 S 0 30 2 0 80 0 - 0 watchd ? 00:00:05 khungtaskd

1 S 0 31 2 0 80 0 - 0 kswapd ? 00:03:57 kswapd0

1 S 0 32 2 0 60 -20 - 0 rescue ? 00:00:00 vmstat

1 S 0 33 2 0 85 5 - 0 ksm\_sc ? 00:00:00 ksmd

1 S 0 34 2 0 99 19 - 0 khugep ? 00:00:00 khugepaged

1 S 0 35 2 0 80 0 - 0 fsnoti ? 00:00:00 fsnotify\_mark

1 S 0 36 2 0 80 0 - 0 ecrypt ? 00:00:00 ecryptfs-kthrea

1 S 0 37 2 0 60 -20 - 0 rescue ? 00:00:00 crypto

1 S 0 45 2 0 60 -20 - 0 rescue ? 00:00:00 kthrotld

**……………………………………….**