Terminator 1

For this question the answer would be yes, the parent is allowed to terminate before one of its child processes. This is not encouraged as the child process would then become an orphan process. To show this in my experiment I will modify Program1A so that in the first and second child processes the parent would terminate before the child processes. In order to do this I will use the sleep() function so that the parent will terminate before the child. I will print out the child PID and PPID before the sleep (figure 1), and then after the sleep (figure 2). When the PPID after the sleep returns it will show the PPID to be 1, once the process becomes an orphan it is adopted by init process who has the PID of 1. As seen in the figures below first the proper PPID is returned but after the program is done running another instance of the process, the orphan child process, is still running on init and prints output when not intended as shown in figure 2. This is why terminating a parent before its child is discouraged because it crates orphan child processes.

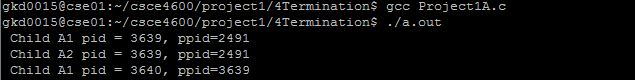


Figure 1

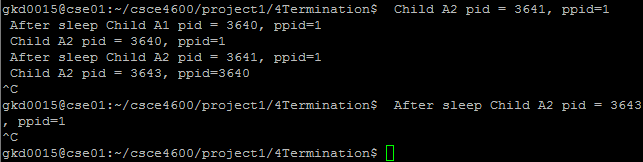


Figure 2

**Source Code** for code that was modified for experiment 4

**Project1A.c**

//project1 & question 4 Termination

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

pid\_t A1, A2;

//child process 1

int fork1(){

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

//create child

A1 = fork ();

//printing child and parent PID before sleep

printf(" Child A1 pid = %d, ppid=%d\n", (int)getpid(), (int)getppid());

if(A1==0){

//parent waits for 3 seconds and terminates creating orphan, child adopted by init with ppid = 1

sleep(3);

dup2(fd, 1);

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

printf(" After sleep Child A1 pid = %d, ppid=%d\n", (int)getpid(), (int)getppid());

}

close(fd);

}

//child process 2

int fork2(){

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

//create second child

A2= fork ();

//printing child and parent PID before sleep

printf(" Child A2 pid = %d, ppid=%d\n", (int)getpid(), (int)getppid());

if(A2==0){

//parent waits for 3 seconds and terminates creating orphan, child adopted by init with ppid = 1

sleep(3);

printf(" After sleep Child A2 pid = %d, ppid=%d\n", (int)getpid(), (int)getppid());

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();

return 0;

}