Aayush Shah
D1 - 19BCE245
15 April 2021

Practical 9

Write a C program to implement a system call using the fork() and Exec() function.

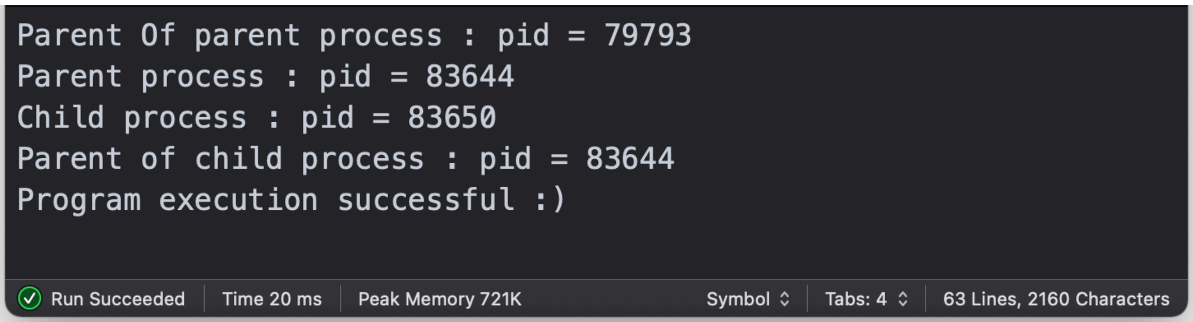
CODE :

```c
// Write a C program to implement a system call using the fork() and Exec() function.

#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/wait.h>

int main(){
  pid_t pid;
  int ret = 1;
  int status;
  pid = fork();

  if (pid == -1){          // Here, pid == -1 means error occured
      printf("Can't fork : error occured :(\n");
      exit(EXIT_FAILURE);
  }
  else if (pid == 0){      // Here, pid == 0 means child process created
```

```
        // getpid() returns process id of calling process which is of child process
        printf("Child process : pid = %u\n",getpid());
        // Here It will return Parent of child Process means Parent process it self
        printf("Parent of child process : pid = %u\n",getppid());

        // The argv list first argument should point to filename associated with file being executed and the array pointer must be terminated by NULL pointer.

        char * argv_list[] = {"ls","-lart","/home",NULL};

        // The execv() only return if error occured. The return value is -1
        execv("ls",argv_list);
        exit(0);
    }
    else{          // A positive number is returned for the pid of parent process
        // getppid() returns process id of parent of calling process which will return the parent of parent process's ID
        printf("Parent Of parent process : pid = %u\n",getppid());
        printf("Parent process : pid = %u\n",getpid());

        // The parent process calls waitpid() on the child waitpid() system call suspends execution of calling process until a child specified by pid argument has changed state
        // See wait() man page for all the flags or options used here
        if (waitpid(pid, &status, 0) > 0) {

            if (WIFEXITED(status) && !WEXITSTATUS(status))
                printf("Program execution successful :)\n");

            else if (WIFEXITED(status) && WEXITSTATUS(status)) {
                if (WEXITSTATUS(status) == 127) {        // execv failed
                    printf("execv failed :(\n");
```

```c
                }
                else
                    printf("Program terminated normally,"
                        " but returned a non-zero status :|\n");
            }
            else
                printf("Program didn't terminate normally :|\n");
        }
        else {          // waitpid() failed
            printf("waitpid() failed :(\n");
        }
        exit(0);
    }
    return 0;
}
```

OUTPUT :



```
Parent Of parent process : pid = 79793
Parent process : pid = 83644
Child process : pid = 83650
Parent of child process : pid = 83644
Program execution successful :)
```