1. *Thực thi command trong tiến trình con*

*Kết quả sẽ hiển thị nội dung của file abc.txt, kết thúc dòng lệnh sẽ hiển thị dấu nhắc it007sh> để người dùng nhập lệnh tiếp theo. Lưu ý rằng trong thời thực thi của lệnh echo abc, không cho người dùng nhập command mới*

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#include <string.h>

#define MAX\_LINE 80 /\* The maximum length command \*/

int main(void) {

    char inputBuffer[MAX\_LINE]; /\* Buffer to hold the command entered \*/

    int should\_run = 1; /\* Flag to determine when to exit the program \*/

    while (should\_run) {

        printf("it007sh> ");

        fflush(stdout);

        /\* Read the command from user input \*/

        fgets(inputBuffer, MAX\_LINE, stdin);

        /\* Remove newline character from the command \*/

        inputBuffer[strcspn(inputBuffer, "\n")] = '\0';

        /\* Check if the entered command is 'echo abc4' \*/

        if (strcmp(inputBuffer, "echo abc") == 0) {

            /\* Open the file 'abc.txt' for reading \*/

            FILE \*file = fopen("abc.txt", "r");

            if (file == NULL) {

                printf("Cannot open file abc.txt\n");

            } else {

                /\* Read and display the contents of 'abc.txt' \*/

                char c;

                while ((c = fgetc(file)) != EOF) {

                    putchar(c);

                }

                fclose(file);

            }

        } else {

            /\* Fork a child process \*/

            pid\_t pid = fork();

            if (pid < 0) {

                fprintf(stderr, "Fork failed\n");

                return 1;

            } else if (pid == 0) { /\* Child process \*/

                /\* Execute the command in the child process \*/

                execlp(inputBuffer, inputBuffer, (char \*)NULL);

                exit(0); /\* Terminate the child process after executing the command \*/

            } else { /\* Parent process \*/

                /\* Wait for the child process to complete \*/

                wait(NULL);

            }

        }

    }

    return 0;

}

*Kết quả:*

A screen shot of a computer

Description automatically generated

*5. Kết thúc lệnh đang thực thi*

*Kết quả sẽ liên tục hiển thị các process của hệ thống, khi sử dụng tổ hợp phím Ctrl + C, lệnh thực thi trên sẽ kết thúc và hiển thị dấu nhắc it007sh> mời người dùng nhập lệnh tiếp theo*

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#include <string.h>

#include <signal.h>

#define MAX\_LINE 80

int should\_run = 1;

void handleCtrlC(int signum) {

    printf("\nCtrl + C pressed\n");

    should\_run = 1; // Nếu có tín hiệu thì quay lại it007sh>

}

int main(void) {

    char inputBuffer[MAX\_LINE]; /\* Buffer to hold the command entered \*/

    signal(SIGINT, handleCtrlC); // Xử lý tín hiệu Ctrl + C

    while (should\_run) {

        printf("it007sh> ");

        fflush(stdout);

        fgets(inputBuffer, MAX\_LINE, stdin);

        inputBuffer[strcspn(inputBuffer, "\n")] = '\0';

        if (strcmp(inputBuffer, "echo abc") == 0) {

            /\* Open the file 'abc.txt' for reading \*/

            FILE \*file = fopen("abc.txt", "r");

            if (file == NULL) {

                printf("Cannot open file abc.txt\n");

            } else {

                /\* Read and display the contents of 'abc.txt' \*/

                char c;

                while ((c = fgetc(file)) != EOF) {

                    putchar(c);

                }

                fclose(file);

            }

        } else {

            /\* Fork a child process \*/

            pid\_t pid = fork();

            if (pid < 0) {

                fprintf(stderr, "Fork failed\n");

                return 1;

            } else if (pid == 0) { /\* Child process \*/

                /\* Execute the command in the child process \*/

                execlp(inputBuffer, inputBuffer, (char \*)NULL);

                exit(0); /\* Terminate the child process after executing the command \*/

            } else { /\* Parent process \*/

                /\* Wait for the child process to complete \*/

                wait(NULL);

            }

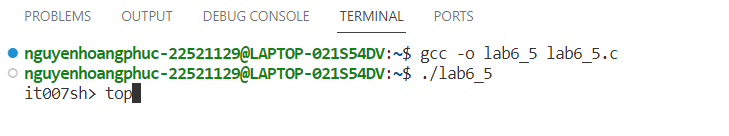
        }

    }

    return 0;

}

*Kết quả:*

**

*A screenshot of a computer

Description automatically generated*