Họ và tên: Bùi Lê Nhật Tri

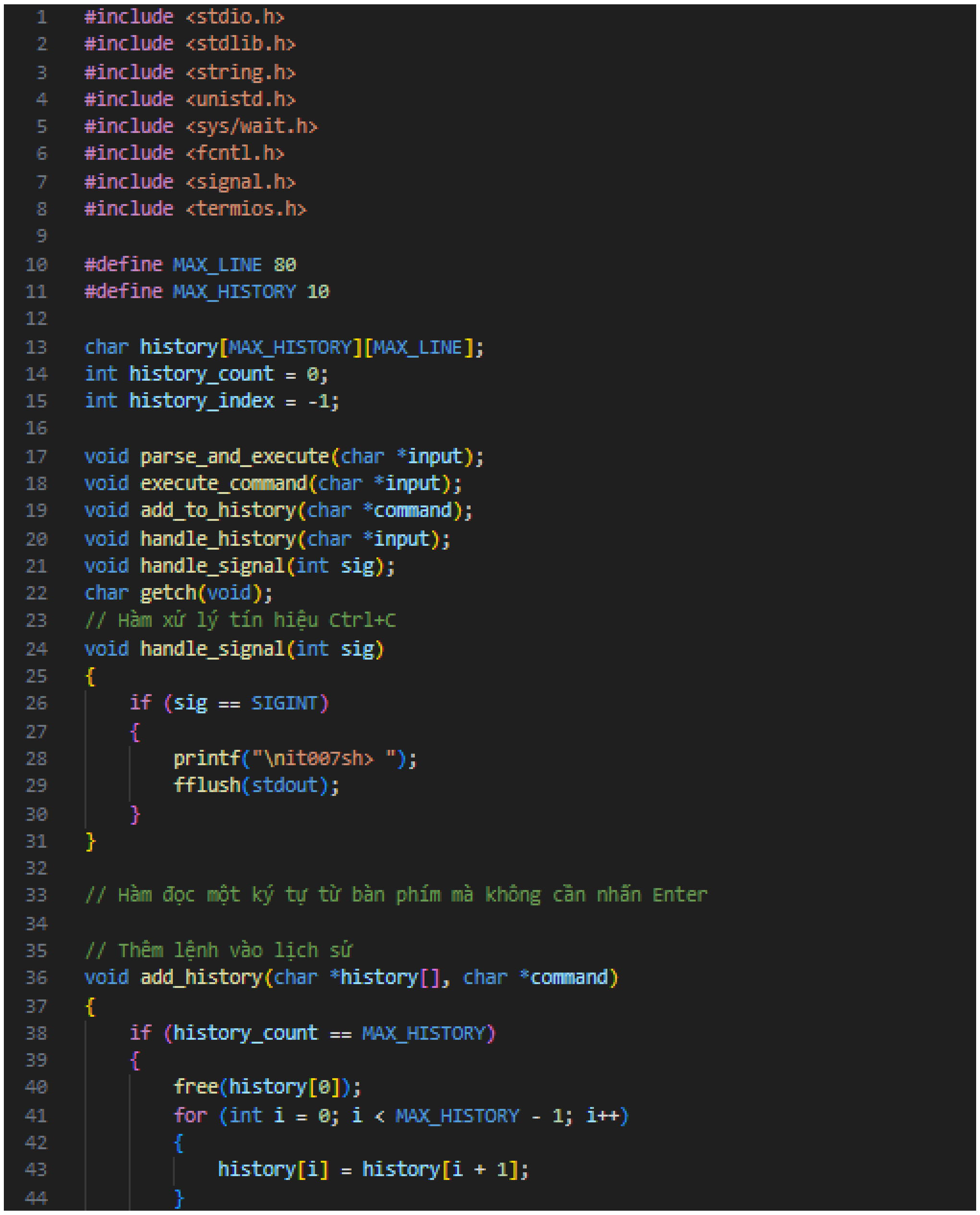
MSSV: 23521634

Lớp: IT00007.P11.1

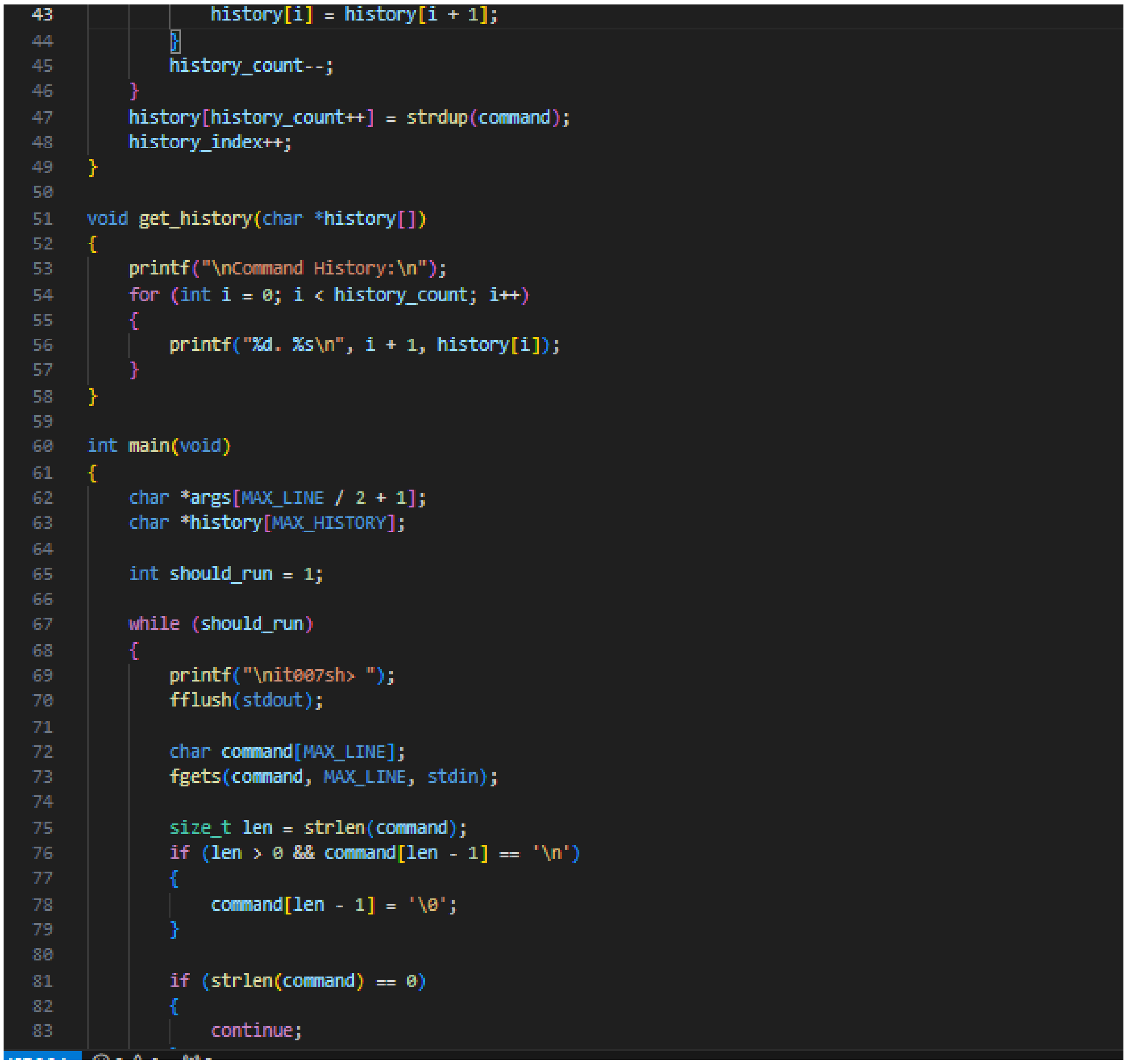
BÁO CÁO LAB 6

Câu 1,2:

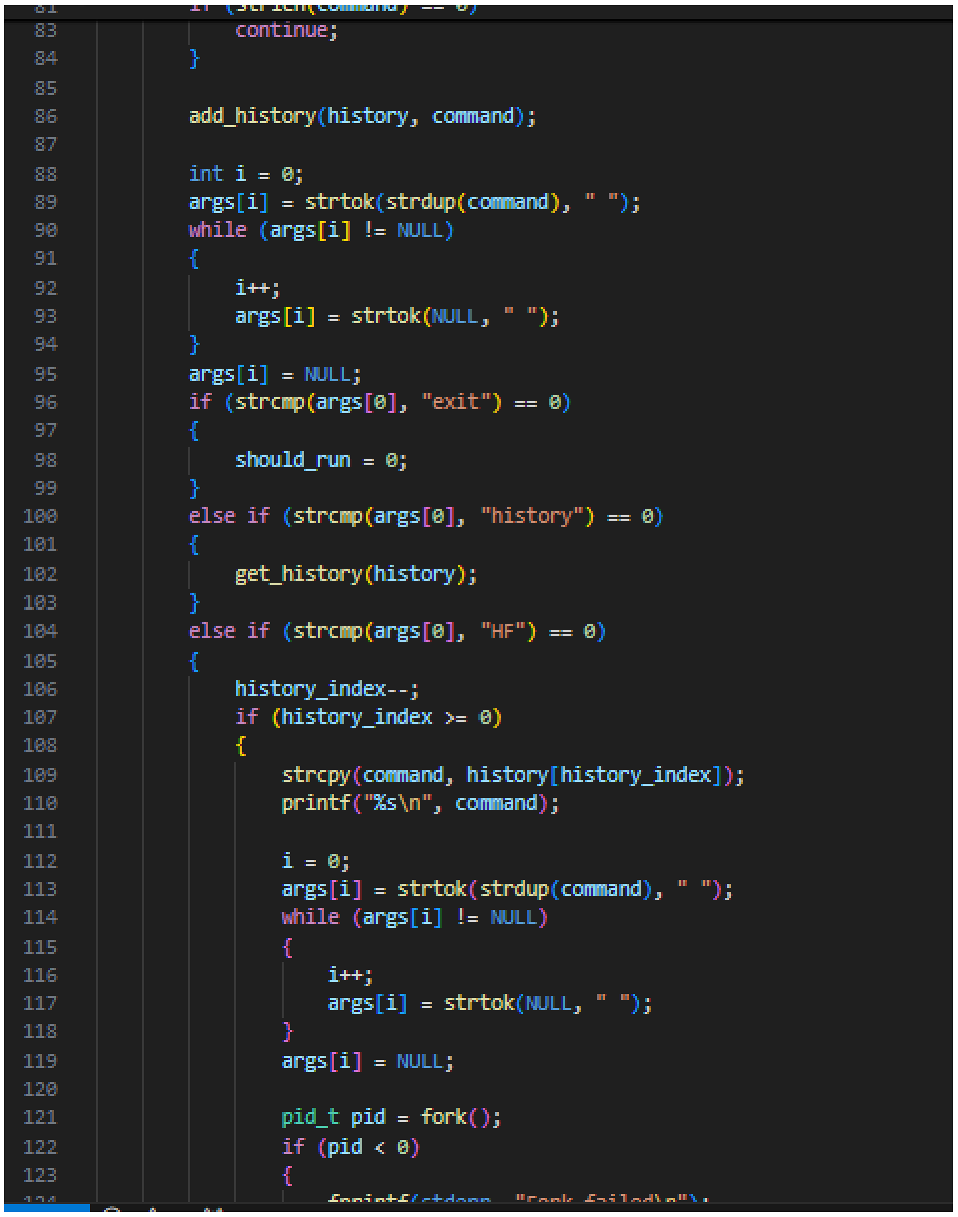
Code:



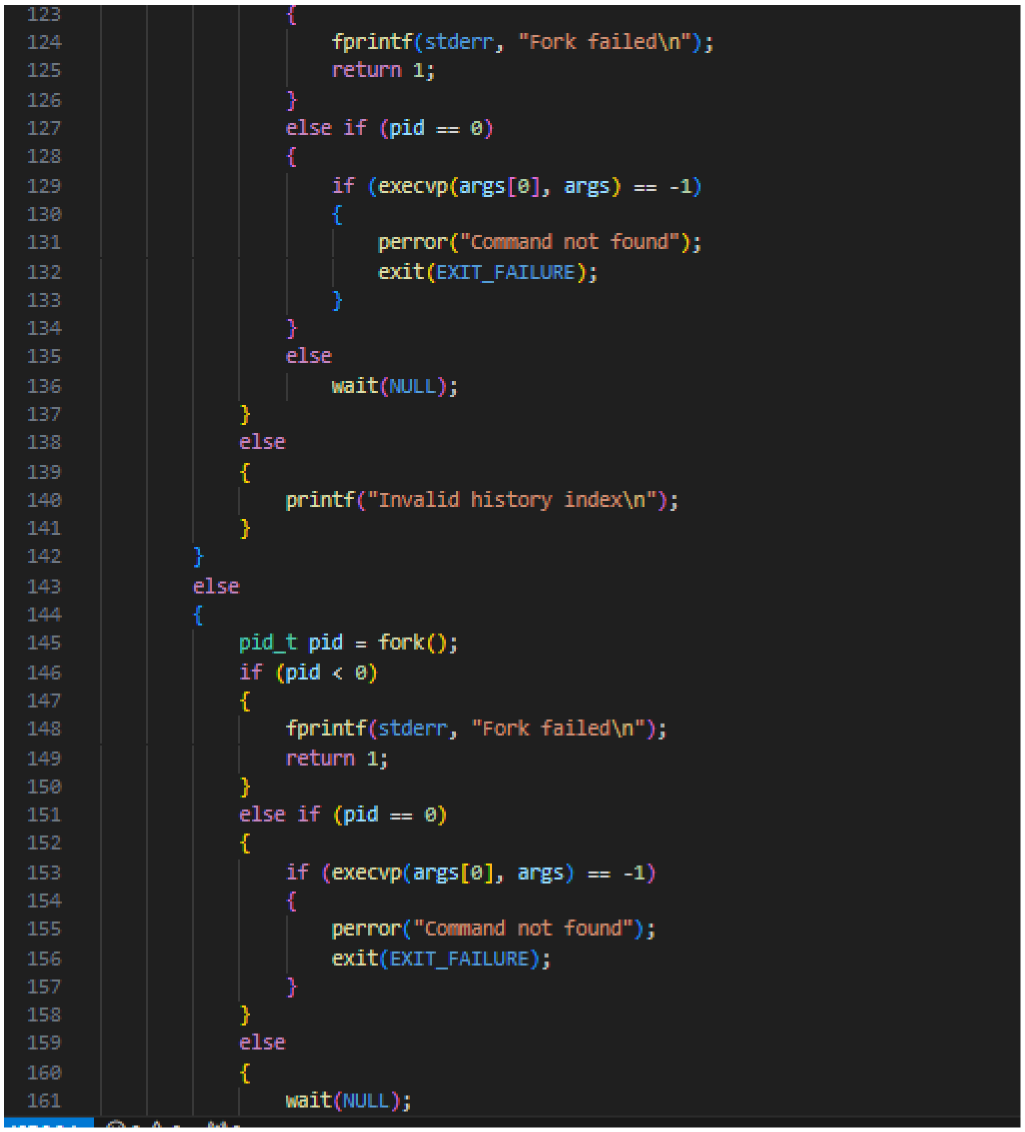
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <signal.h>
#include <termios.h>

#define MAX_LINE 80
#define MAX_HISTORY 10

char history[MAX_HISTORY][MAX_LINE];
int history_count = 0;
int history_index = -1;

void parse_and_execute(char *input);
void execute_command(char *input);
void add_to_history(char *command);
void handle_history(char *input);
void handle_signal(int sig);
char getch(void);
// Hàm xử lý tín hiệu Ctrl+C
void handle_signal(int sig)
{
    if (sig == SIGINT)
    {
        printf("\nit007sh> ");
        fflush(stdout);
    }
}

// Hàm đọc một ký tự từ bàn phím mà không cần nhấn Enter

// Thêm lệnh vào lịch sử
void add_history(char *history[], char *command)
{
    if (history_count == MAX_HISTORY)
    {
        free(history[0]);
        for (int i = 0; i < MAX_HISTORY - 1; i++)
        {
            history[i] = history[i + 1];
        }
```



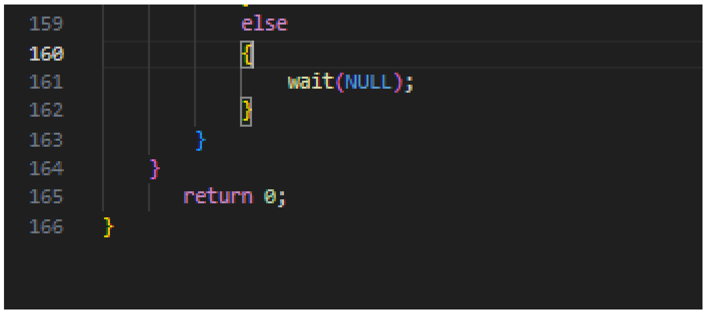
```
            history[i] = history[i + 1];
        }
        history_count--;
    }
    history[history_count++] = strdup(command);
    history_index++;
}

void get_history(char *history[])
{
    printf("\nCommand History:\n");
    for (int i = 0; i < history_count; i++)
    {
        printf("%d. %s\n", i + 1, history[i]);
    }
}

int main(void)
{
    char *args[MAX_LINE / 2 + 1];
    char *history[MAX_HISTORY];

    int should_run = 1;

    while (should_run)
    {
        printf("\nit007sh> ");
        fflush(stdout);

        char command[MAX_LINE];
        fgets(command, MAX_LINE, stdin);

        size_t len = strlen(command);
        if (len > 0 && command[len - 1] == '\n')
        {
            command[len - 1] = '\0';
        }

        if (strlen(command) == 0)
        {
            continue;
```



```c
        if (strlen(command) == 0)
            continue;
        }

        add_history(history, command);

        int i = 0;
        args[i] = strtok(strdup(command), " ");
        while (args[i] != NULL)
        {
            i++;
            args[i] = strtok(NULL, " ");
        }
        args[i] = NULL;
        if (strcmp(args[0], "exit") == 0)
        {
            should_run = 0;
        }
        else if (strcmp(args[0], "history") == 0)
        {
            get_history(history);
        }
        else if (strcmp(args[0], "HF") == 0)
        {
            history_index--;
            if (history_index >= 0)
            {
                strcpy(command, history[history_index]);
                printf("%s\n", command);

                i = 0;
                args[i] = strtok(strdup(command), " ");
                while (args[i] != NULL)
                {
                    i++;
                    args[i] = strtok(NULL, " ");
                }
                args[i] = NULL;

                pid_t pid = fork();
                if (pid < 0)
                {
                    fprintf(stderr, "Fork failed\n");
```

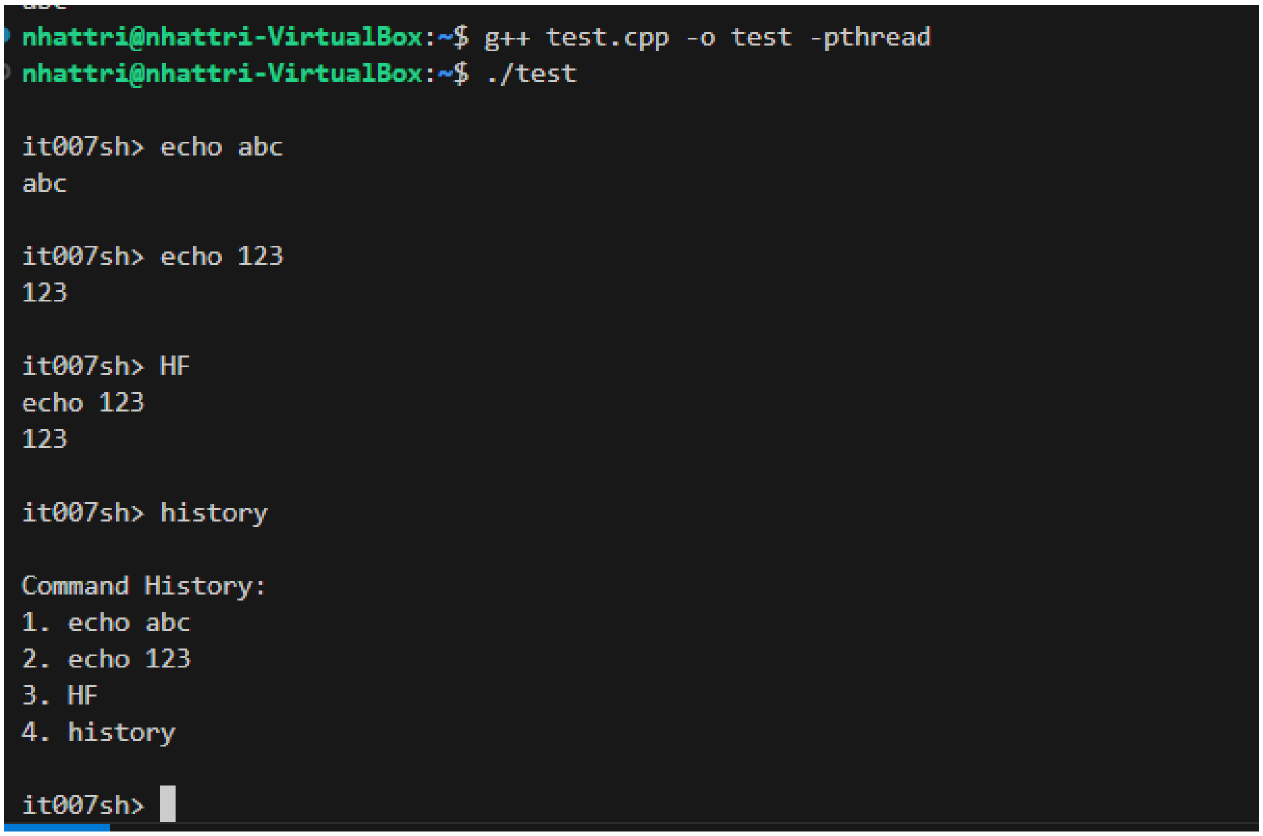


```c
                {
                    fprintf(stderr, "Fork failed\n");
                    return 1;
                }
                else if (pid == 0)
                {
                    if (execvp(args[0], args) == -1)
                    {
                        perror("Command not found");
                        exit(EXIT_FAILURE);
                    }
                }
                else
                    wait(NULL);
            }
            else
            {
                printf("Invalid history index\n");
            }
        }
        else
        {
            pid_t pid = fork();
            if (pid < 0)
            {
                fprintf(stderr, "Fork failed\n");
                return 1;
            }
            else if (pid == 0)
            {
                if (execvp(args[0], args) == -1)
                {
                    perror("Command not found");
                    exit(EXIT_FAILURE);
                }
            }
            else
            {
                wait(NULL);
```



```c
            else
            {
                wait(NULL);
            }
        }
    }
       return 0;
}
```

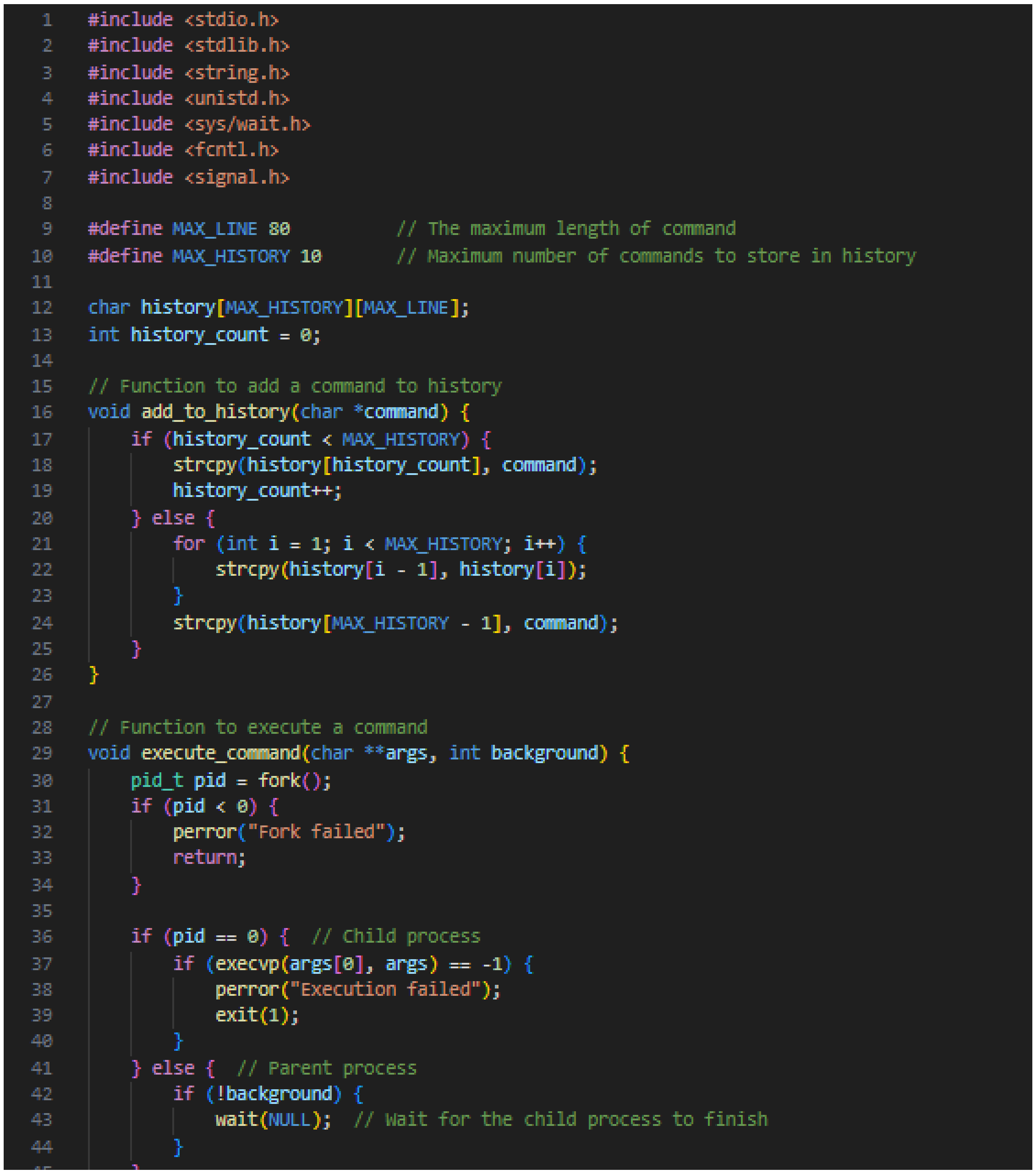
Kết quả:



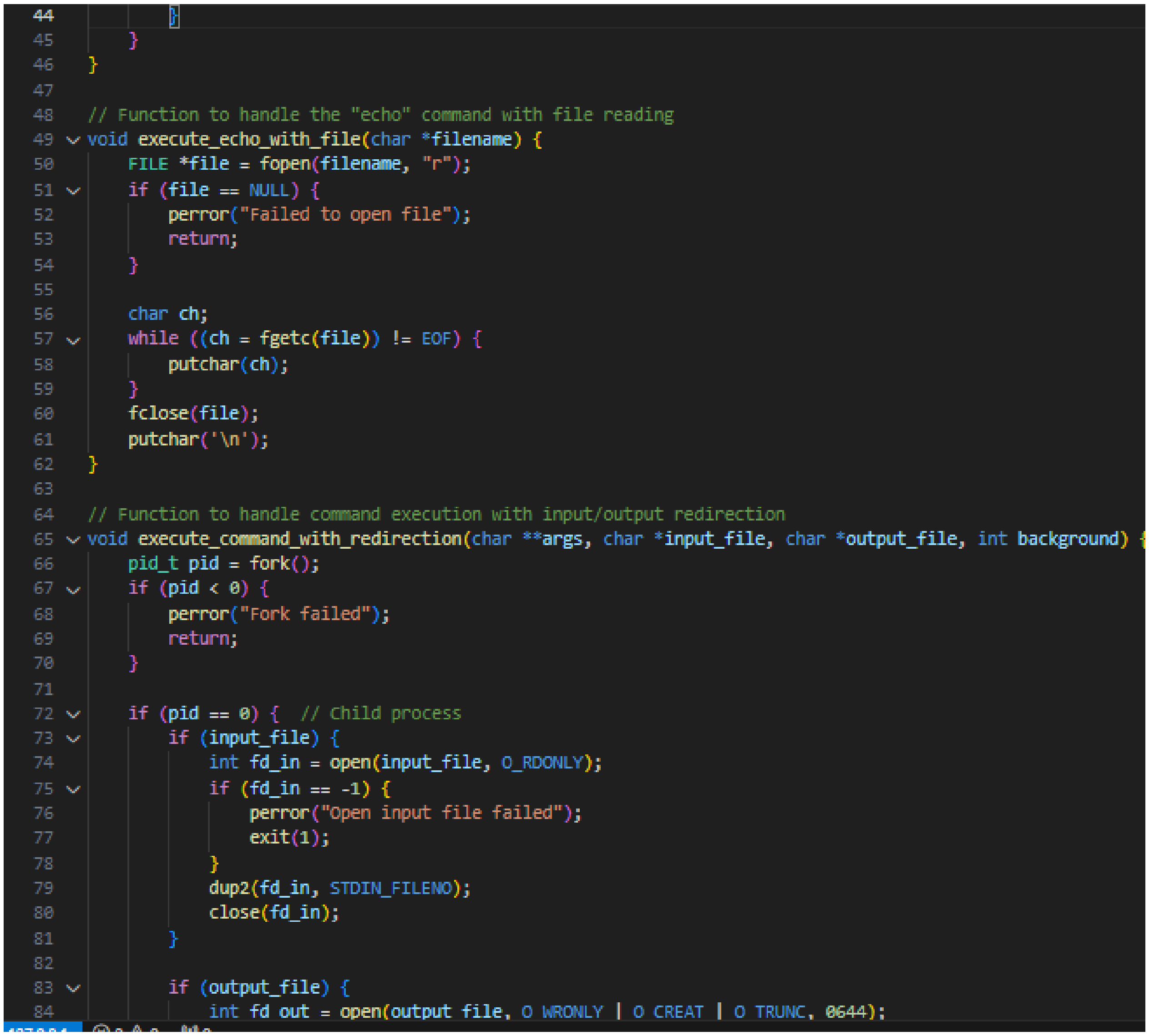
```
nhattri@nhattri-VirtualBox:~$ g++ test.cpp -o test -pthread
nhattri@nhattri-VirtualBox:~$ ./test

it007sh> echo abc
abc

it007sh> echo 123
123

it007sh> HF
echo 123
123

it007sh> history

Command History:
1. echo abc
2. echo 123
3. HF
4. history

it007sh> 
```

Câu 3,4,5:

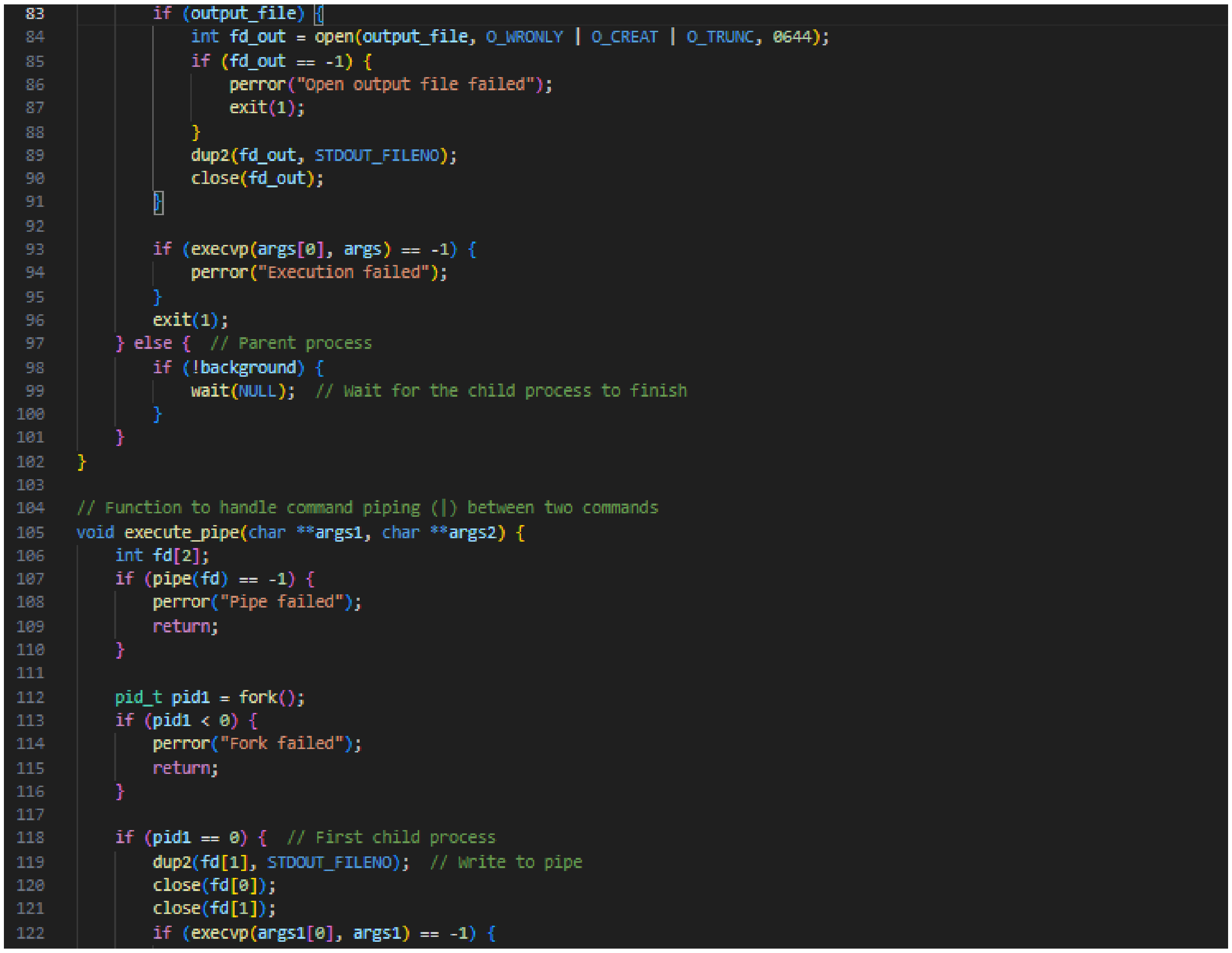
Code:



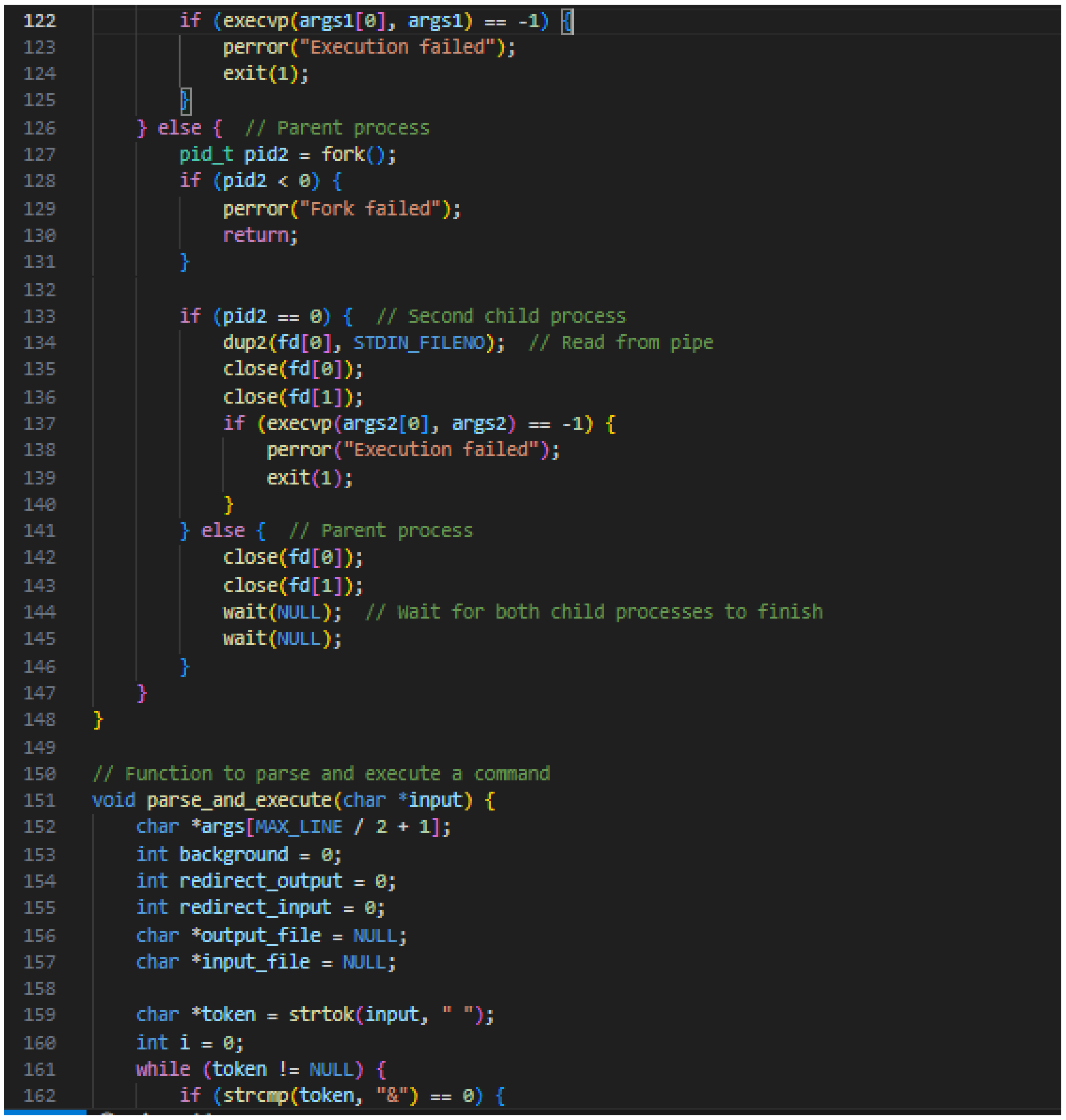
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <signal.h>

#define MAX_LINE 80          // The maximum length of command
#define MAX_HISTORY 10       // Maximum number of commands to store in history

char history[MAX_HISTORY][MAX_LINE];
int history_count = 0;

// Function to add a command to history
void add_to_history(char *command) {
    if (history_count < MAX_HISTORY) {
        strcpy(history[history_count], command);
        history_count++;
    } else {
        for (int i = 1; i < MAX_HISTORY; i++) {
            strcpy(history[i - 1], history[i]);
        }
        strcpy(history[MAX_HISTORY - 1], command);
    }
}

// Function to execute a command
void execute_command(char **args, int background) {
    pid_t pid = fork();
    if (pid < 0) {
        perror("Fork failed");
        return;
    }

    if (pid == 0) {  // Child process
        if (execvp(args[0], args) == -1) {
            perror("Execution failed");
            exit(1);
        }
    } else {  // Parent process
        if (!background) {
            wait(NULL);  // Wait for the child process to finish
        }
    }
```



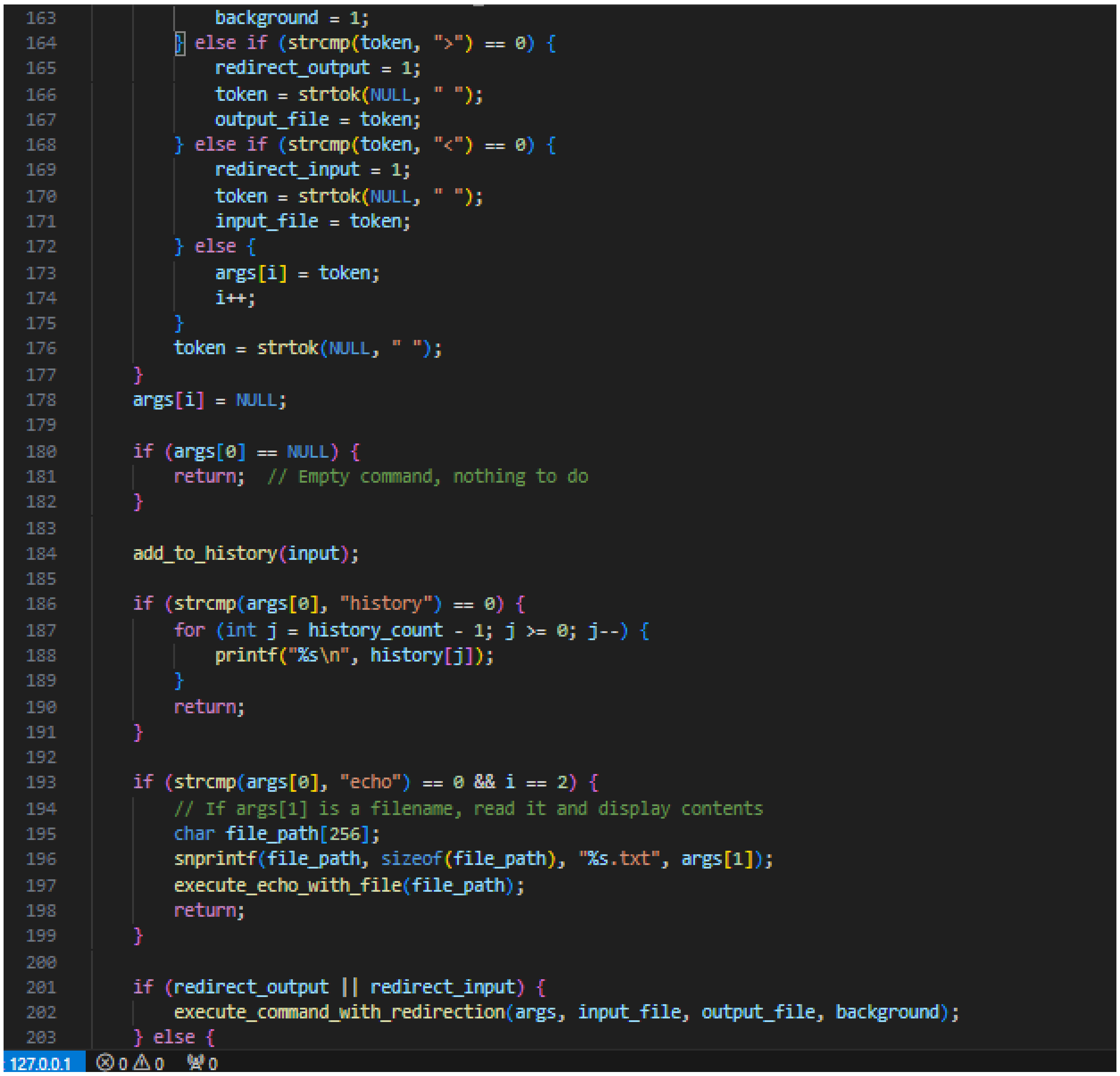
```c
        }
    }
}

// Function to handle the "echo" command with file reading
void execute_echo_with_file(char *filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        perror("Failed to open file");
        return;
    }

    char ch;
    while ((ch = fgetc(file)) != EOF) {
        putchar(ch);
    }
    fclose(file);
    putchar('\n');
}

// Function to handle command execution with input/output redirection
void execute_command_with_redirection(char **args, char *input_file, char *output_file, int background) {
    pid_t pid = fork();
    if (pid < 0) {
        perror("Fork failed");
        return;
    }

    if (pid == 0) {  // Child process
        if (input_file) {
            int fd_in = open(input_file, O_RDONLY);
            if (fd_in == -1) {
                perror("Open input file failed");
                exit(1);
            }
            dup2(fd_in, STDIN_FILENO);
            close(fd_in);
        }

        if (output_file) {
            int fd_out = open(output_file, O_WRONLY | O_CREAT | O_TRUNC, 0644);
```



```c
        if (output_file) {
            int fd_out = open(output_file, O_WRONLY | O_CREAT | O_TRUNC, 0644);
            if (fd_out == -1) {
                perror("Open output file failed");
                exit(1);
            }
            dup2(fd_out, STDOUT_FILENO);
            close(fd_out);
        }

        if (execvp(args[0], args) == -1) {
            perror("Execution failed");
        }
        exit(1);
    } else {  // Parent process
        if (!background) {
            wait(NULL);  // Wait for the child process to finish
        }
    }
}

// Function to handle command piping (|) between two commands
void execute_pipe(char **args1, char **args2) {
    int fd[2];
    if (pipe(fd) == -1) {
        perror("Pipe failed");
        return;
    }

    pid_t pid1 = fork();
    if (pid1 < 0) {
        perror("Fork failed");
        return;
    }

    if (pid1 == 0) {  // First child process
        dup2(fd[1], STDOUT_FILENO);  // Write to pipe
        close(fd[0]);
        close(fd[1]);
        if (execvp(args1[0], args1) == -1) {
```

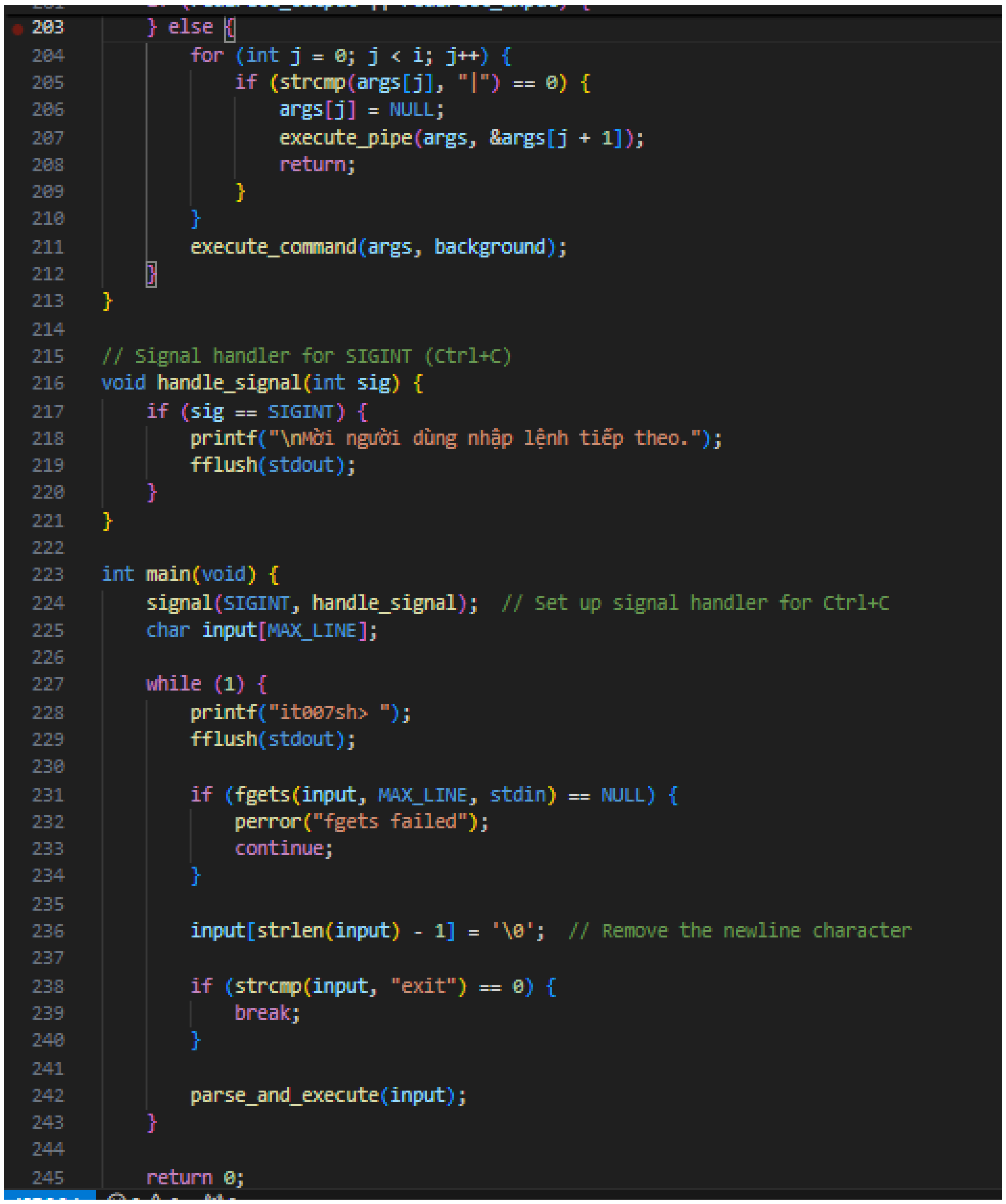


```
            if (execvp(args1[0], args1) == -1) {
                perror("Execution failed");
                exit(1);
            }
        } else {  // Parent process
            pid_t pid2 = fork();
            if (pid2 < 0) {
                perror("Fork failed");
                return;
            }

            if (pid2 == 0) {  // Second child process
                dup2(fd[0], STDIN_FILENO);  // Read from pipe
                close(fd[0]);
                close(fd[1]);
                if (execvp(args2[0], args2) == -1) {
                    perror("Execution failed");
                    exit(1);
                }
            } else {  // Parent process
                close(fd[0]);
                close(fd[1]);
                wait(NULL);  // Wait for both child processes to finish
                wait(NULL);
            }
        }
}

// Function to parse and execute a command
void parse_and_execute(char *input) {
    char *args[MAX_LINE / 2 + 1];
    int background = 0;
    int redirect_output = 0;
    int redirect_input = 0;
    char *output_file = NULL;
    char *input_file = NULL;

    char *token = strtok(input, " ");
    int i = 0;
    while (token != NULL) {
        if (strcmp(token, "&") == 0) {
```



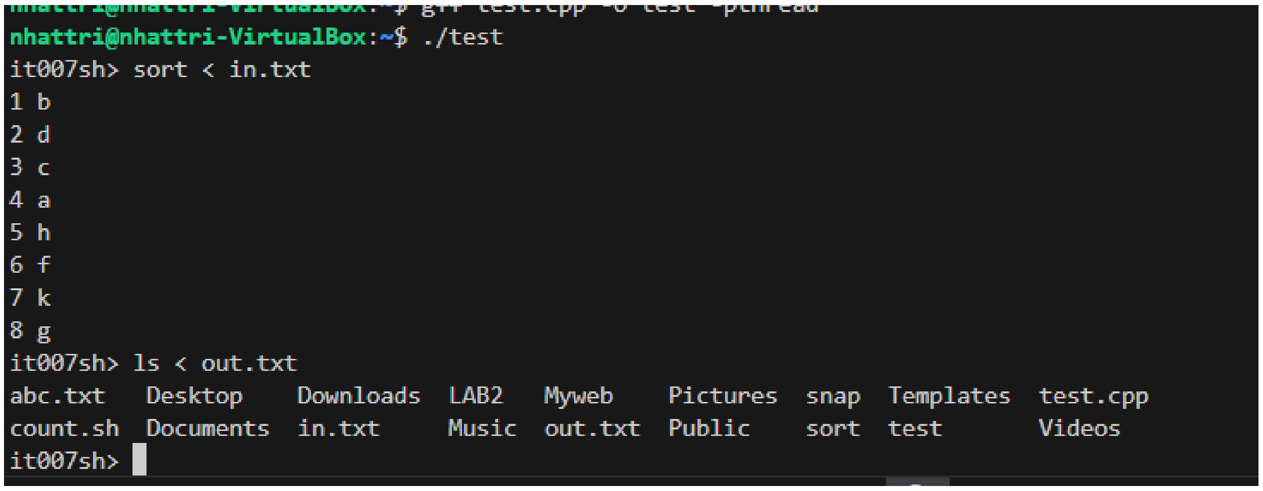
```
            background = 1;
        } else if (strcmp(token, ">") == 0) {
            redirect_output = 1;
            token = strtok(NULL, " ");
            output_file = token;
        } else if (strcmp(token, "<") == 0) {
            redirect_input = 1;
            token = strtok(NULL, " ");
            input_file = token;
        } else {
            args[i] = token;
            i++;
        }
        token = strtok(NULL, " ");
    }
    args[i] = NULL;

    if (args[0] == NULL) {
        return;  // Empty command, nothing to do
    }

    add_to_history(input);

    if (strcmp(args[0], "history") == 0) {
        for (int j = history_count - 1; j >= 0; j--) {
            printf("%s\n", history[j]);
        }
        return;
    }

    if (strcmp(args[0], "echo") == 0 && i == 2) {
        // If args[1] is a filename, read it and display contents
        char file_path[256];
        snprintf(file_path, sizeof(file_path), "%s.txt", args[1]);
        execute_echo_with_file(file_path);
        return;
    }

    if (redirect_output || redirect_input) {
        execute_command_with_redirection(args, input_file, output_file, background);
    } else {
```

127.0.0.1   ⓧ 0 ⚠ 0   0

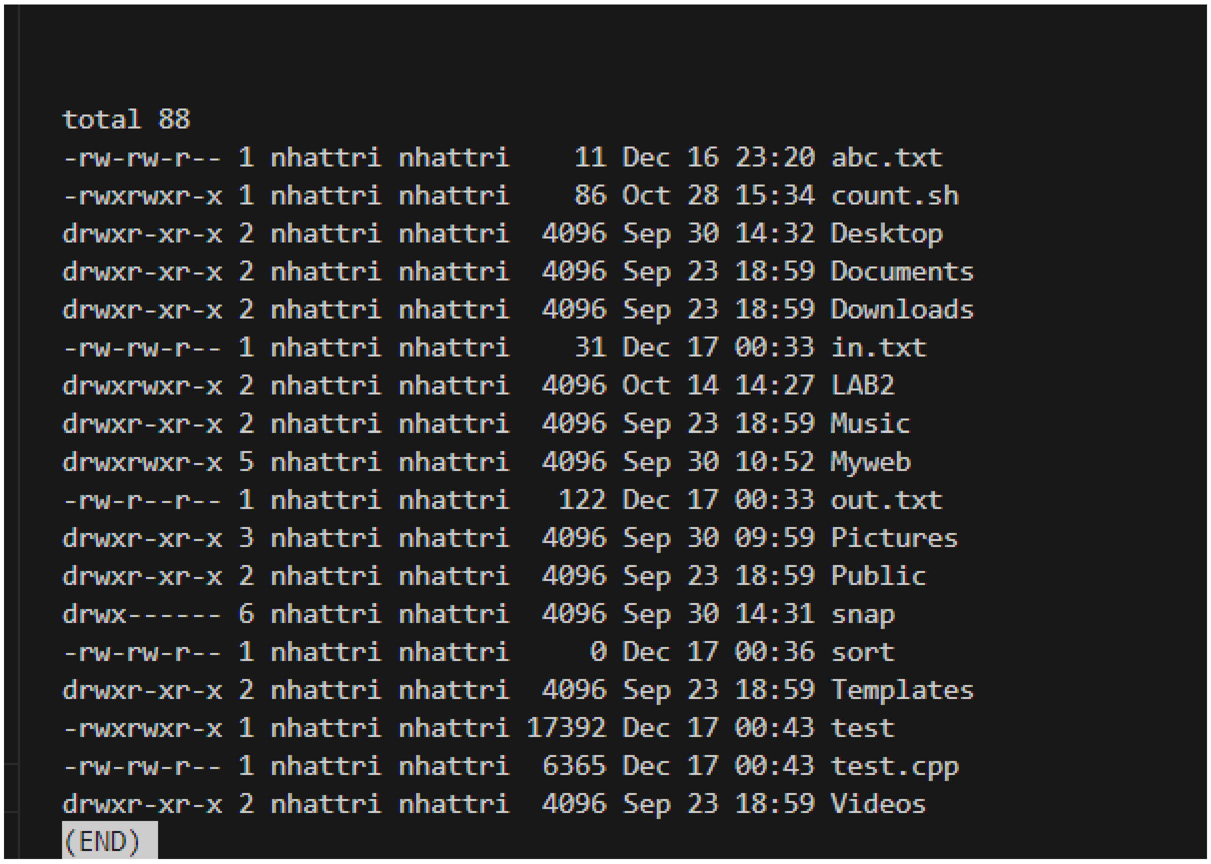


```c
    } else {
        for (int j = 0; j < i; j++) {
            if (strcmp(args[j], "|") == 0) {
                args[j] = NULL;
                execute_pipe(args, &args[j + 1]);
                return;
            }
        }
        execute_command(args, background);
    }
}

// Signal handler for SIGINT (Ctrl+C)
void handle_signal(int sig) {
    if (sig == SIGINT) {
        printf("\nMời người dùng nhập lệnh tiếp theo.");
        fflush(stdout);
    }
}

int main(void) {
    signal(SIGINT, handle_signal);  // Set up signal handler for Ctrl+C
    char input[MAX_LINE];

    while (1) {
        printf("it007sh> ");
        fflush(stdout);

        if (fgets(input, MAX_LINE, stdin) == NULL) {
            perror("fgets failed");
            continue;
        }

        input[strlen(input) - 1] = '\0';  // Remove the newline character

        if (strcmp(input, "exit") == 0) {
            break;
        }

        parse_and_execute(input);
    }

    return 0;
```

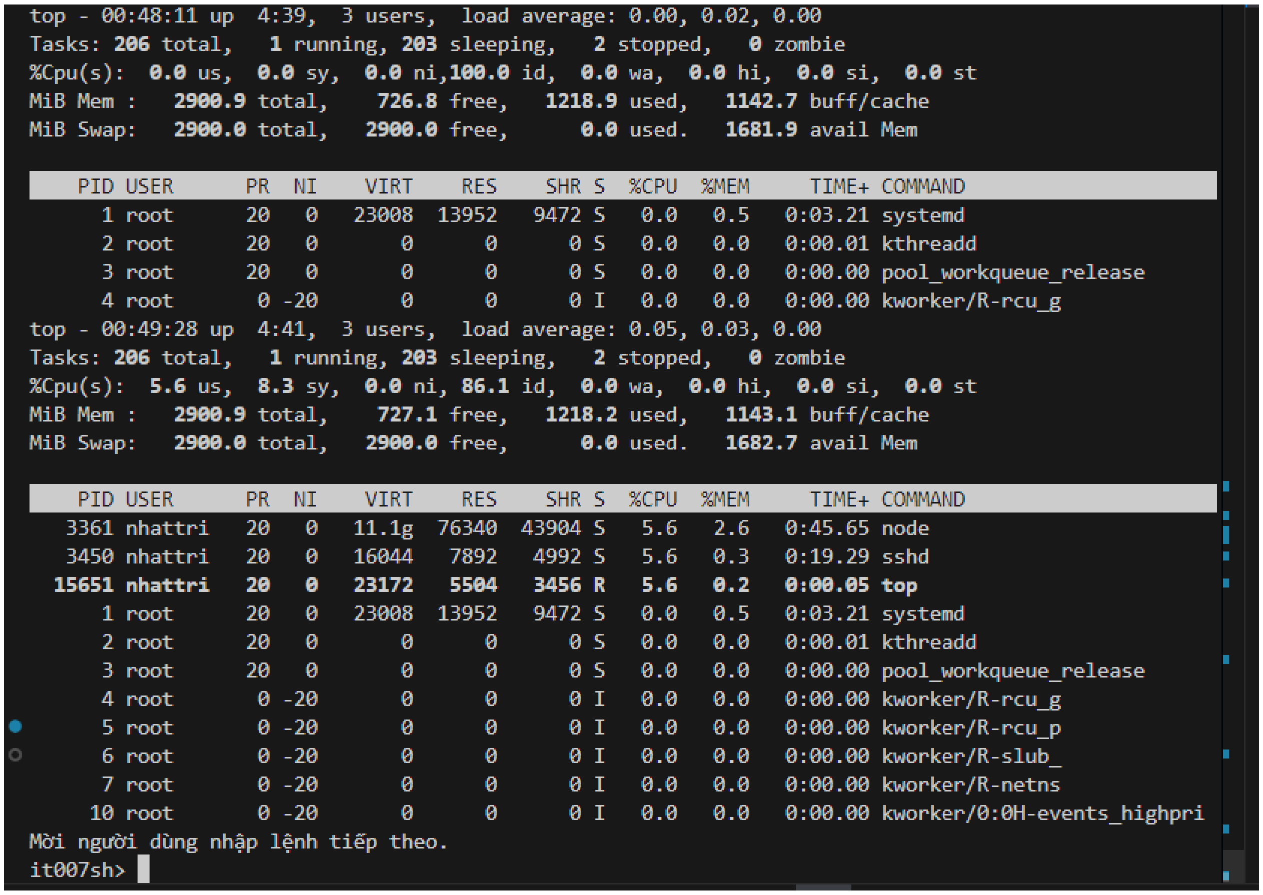
Kết quả:



```
nhattri@nhattri-VirtualBox:~$ g++ test.cpp -o test -pthread
nhattri@nhattri-VirtualBox:~$ ./test
it007sh> sort < in.txt
1 b
2 d
3 c
4 a
5 h
6 f
7 k
8 g
it007sh> ls < out.txt
abc.txt   Desktop    Downloads  LAB2   Myweb    Pictures  snap  Templates  test.cpp
count.sh  Documents  in.txt     Music  out.txt  Public    sort  test       Videos
it007sh> 
```



```
total 88
-rw-rw-r-- 1 nhattri nhattri    11 Dec 16 23:20 abc.txt
-rwxrwxr-x 1 nhattri nhattri    86 Oct 28 15:34 count.sh
drwxr-xr-x 2 nhattri nhattri  4096 Sep 30 14:32 Desktop
drwxr-xr-x 2 nhattri nhattri  4096 Sep 23 18:59 Documents
drwxr-xr-x 2 nhattri nhattri  4096 Sep 23 18:59 Downloads
-rw-rw-r-- 1 nhattri nhattri    31 Dec 17 00:33 in.txt
drwxrwxr-x 2 nhattri nhattri  4096 Oct 14 14:27 LAB2
drwxr-xr-x 2 nhattri nhattri  4096 Sep 23 18:59 Music
drwxrwxr-x 5 nhattri nhattri  4096 Sep 30 10:52 Myweb
-rw-r--r-- 1 nhattri nhattri   122 Dec 17 00:33 out.txt
drwxr-xr-x 3 nhattri nhattri  4096 Sep 30 09:59 Pictures
drwxr-xr-x 2 nhattri nhattri  4096 Sep 23 18:59 Public
drwx------ 6 nhattri nhattri  4096 Sep 30 14:31 snap
-rw-rw-r-- 1 nhattri nhattri     0 Dec 17 00:36 sort
drwxr-xr-x 2 nhattri nhattri  4096 Sep 23 18:59 Templates
-rwxrwxr-x 1 nhattri nhattri 17392 Dec 17 00:43 test
-rw-rw-r-- 1 nhattri nhattri  6365 Dec 17 00:43 test.cpp
drwxr-xr-x 2 nhattri nhattri  4096 Sep 23 18:59 Videos
(END)
```



```
top - 00:48:11 up  4:39,  3 users,  load average: 0.00, 0.02, 0.00
Tasks: 206 total,   1 running, 203 sleeping,   2 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   2900.9 total,    726.8 free,   1218.9 used,   1142.7 buff/cache
MiB Swap:   2900.0 total,   2900.0 free,      0.0 used.   1681.9 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
      1 root      20   0   23008  13952   9472 S   0.0   0.5   0:03.21 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthreadd
      3 root      20   0       0      0      0 S   0.0   0.0   0:00.00 pool_workqueue_release
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_g
top - 00:49:28 up  4:41,  3 users,  load average: 0.05, 0.03, 0.00
Tasks: 206 total,   1 running, 203 sleeping,   2 stopped,   0 zombie
%Cpu(s):  5.6 us,  8.3 sy,  0.0 ni, 86.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   2900.9 total,    727.1 free,   1218.2 used,   1143.1 buff/cache
MiB Swap:   2900.0 total,   2900.0 free,      0.0 used.   1682.7 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3361 nhattri   20   0   11.1g  76340  43904 S   5.6   2.6   0:45.65 node
   3450 nhattri   20   0   16044   7892   4992 S   5.6   0.3   0:19.29 sshd
  15651 nhattri   20   0   23172   5504   3456 R   5.6   0.2   0:00.05 top
      1 root      20   0   23008  13952   9472 S   0.0   0.5   0:03.21 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthreadd
      3 root      20   0       0      0      0 S   0.0   0.0   0:00.00 pool_workqueue_release
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_g
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_p
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-slub_
      7 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-netns
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
Mời người dùng nhập lệnh tiếp theo.
it007sh>
```