Báo cáo thực hành môn Hệ điều hành - Giảng viên: Trần Hoàng Lộc.

| |
|---|
| Họ và tên: Dương Thuận Trí<br>Mã số sinh viên: 22521517<br>Lớp: 1 |

# HỆ ĐIỀU HÀNH
# BÁO CÁO LAB 6

## CHECKLIST

## 6.4. BÀI TẬP THỰC HÀNH

| | Câu 1 | Câu 2 | Câu 3 | Câu 4 | Câu 5 |
|---|---|---|---|---|---|
| **Trình bày giải thuật** | ☐ | ☐ | ☐ | ☐ | ☐ |
| **Chụp hình minh chứng (chạy ít nhất 3 lệnh)** | ☐ | ☐ | ☐ | ☐ | ☐ |
| **Giải thích code, kết quả** | ☐ | ☐ | ☐ | ☐ | ☐ |

**Tự chấm điểm:** 10

**Lưu ý: Xuất báo cáo theo định dạng PDF, đặt tên theo cú pháp:*
***<Tên nhóm>_LAB6.pdf***

### 6.4. BÀI TẬP THỰC HÀNH

## 1. Câu 1

Trả lời:

### ➢ Giải thuật:

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

#define MAX_LINE 80

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

    while (shouldrun) {
        printf("it007sh>");
        fflush(stdout);
        char command[MAX_LINE]; // mang dung` de chua lenh
        fgets(command, MAX_LINE, stdin);
        char *arg = strtok(command, " \n");
        int i = 0;
        while (arg != NULL) {
            args[i++] = arg;
            arg = strtok(NULL, " \n");
        }
        args[i] = NULL;
        pid_t pid = fork();
        if (pid < 0) {
            fprintf(stderr, "Fork that bai!\n");
            return 1;
        } else if (pid == 0) {
            execvp(args[0], args);
            fprintf(stderr, "Khong tim thay lenh: %s\n", args[0]);
            exit(EXIT_FAILURE);
        } else {
            int status;
            waitpid(pid, &status, 0);
        }
```

```
    }

    return 0;
}
```

## ➢ Output:

```
triduong@duongthuantriubuntu:~$ ./bai1
it007sh>ls
bai1  bai1.c  Desktop  Documents  Downloads  Music  Pictures  Public  s  snap  Templates  test2.txt  test.txt  Videos
it007sh>cat test2.txt
Hello
I'm Duong Thuan Tri
This is test2.txt

it007sh>cp test2.txt test3.txt
it007sh>cat test3.txt
Hello
I'm Duong Thuan Tri
This is test2.txt
```

```
triduong@duongthuantriubuntu:~$ ./bai1
it007sh>cattt test2.txt
Khong tim thay lenh: cattt
```

## ➢ Giải thích:

- Mảng args[MAX_LINE / 2 + 1] để lưu trữ các đối số của lệnh. Số lượng đối số tối đa được giới hạn là (MAX_LINE / 2) + 1.
- Mảng command[MAX_LINE] để lưu trữ lệnh được nhập từ người dùng. Độ dài tối đa của lệnh được giới hạn bởi MAX_LINE.
- Dùng hàm fgets(command, MAX_LINE, stdin) để đọc lệnh từ người dùng và lưu vào mảng command
- Dùng strtok(command, " \n") để tách lệnh thành các đối số và lưu tạm vào chuỗi arg (strtok đầu tiên dùng để tách đối số đầu tiên- ví dụ như “ls”, “cat”,…), ký tự " " và "\n" là những ký tự phân tách (phân tách khi gặp khoảng trắng/ xuống dòng). Ta lưu từng đối số vào mảng args (args[0] chính là phần tử chứa đối số lệnh)
- “pid_t pid = fork()”: Tạo một tiến trình con.
  - pid<0: fork thất bại, in thông báo lỗi và thoát.

  - pid==0: đây là tiến trình con, sử dụng execvp(args[0], args) để thực thi lệnh. Nếu execvp thất bại, in thông báo lỗi và thoát với mã lỗi.
  - pid>0: đây là tiến trình cha, đợi tiến trình con kết thúc và lưu trữ trạng thái.
- Giải thích kết quả:
  - "ls": hiển thị danh sách các file/thư mục ở thư mục hiện tại
  - "cat test2.txt": hiển thị nội dung file test2.txt
  - "cp test2.txt test3.txt": sao chép test2.txt thành một file mới có tên test3.txt ngay tại thư mục hiện tại.
  - "cat test3.txt": hiển thị nội dung file test3.txt ( có cùng nội dung với file test2.txt vì đây là một file sao chép từ test2.txt)
  - "cattt test2.txt": không tồn tại lệnh này nên in thông báo lỗi

## 2. Câu 2

Trả lời:

### ➢ Giải thuật:

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

#define MAX_LINE 80
#define HF_size 20 // so cau lenh toi da duoc luu trong lich su

char history[HF_size][MAX_LINE];
// dung mang de luu lich su nhung cau lenh da nhap
int his_index = 0;

void add_to_history(const char *command) {
    strcpy(history[his_index], command);
    his_index = (his_index + 1) % HF_size;
    // thay cau lenh cu~ nhat bang cau lenh moi nhat neu da dat so cau lenh toi
//da duoc luu
}
```

```c
void print_history() {
    int i = 0;
    int print_index = his_index;
    do {
        if (strlen(history[print_index]) > 0) {
            printf("%s\n", history[print_index]);
        }
        print_index = (print_index + 1) % HF_size;
        i++;
    } while (i < HF_size);
}

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

    while (shouldrun) {
        printf("it007sh>");
        fflush(stdout);
        char command[MAX_LINE]; // mang dung` de chua lenh
        fgets(command, MAX_LINE, stdin);
        command[strcspn(command, "\n")] = '\0';
        if (strcmp(command, "HF") == 0)
        {
            print_history(); // in ra truoc khi luu "HF" vao lich su
            add_to_history(command);
            continue;  //sang vong lap tiep theo
        }
        add_to_history(command);

        char *arg = strtok(command, " \n");
        int i = 0;
        while (arg != NULL) {
            args[i++] = arg;
            arg = strtok(NULL, " \n");
        }
        args[i] = NULL;
        pid_t pid = fork();
        if (pid < 0) {
            fprintf(stderr, "Fork that bai!\n");
            return 1;
        } else if (pid == 0) {
            execvp(args[0], args);
            fprintf(stderr, "Khong tim thay lenh: %s\n", args[0]);
```

```
            exit(EXIT_FAILURE);
        } else {
            int status;
            waitpid(pid, &status, 0);
        }
    }
    return 0;
}
```

## ➢ Output:

```
triduong@duongthuantriubuntu:~$ ./bai2
it007sh>ls
bai1  bai1.c  bai2  bai2.c  Desktop  Documents  Downloads  Music  Pictures  Public  s  snap  Templates  test2.txt  test3.txt  Videos
it007sh>cat test2.txt
Hello
I'm Duong Thuan Tri
This is test2.txt

it007sh>HF
ls
cat test2.txt
it007sh>mv test3.txt test4.txt
it007sh>ls
bai1  bai1.c  bai2  bai2.c  Desktop  Documents  Downloads  Music  Pictures  Public  s  snap  Templates  test2.txt  test4.txt  Videos
it007sh>HF
ls
cat test2.txt
HF
mv test3.txt test4.txt
ls
```

```
it007sh>cat test4.txt
Hello
I'm Duong Thuan Tri
This is test3.txt

it007sh>rm test4.txt
it007sh>ls
bai1  bai1.c  bai2  bai2.c  Desktop  Documents  Downloads  Music  Pictures  Public  s  snap  Templates  test2.txt  Videos
it007sh>HF
ls
cat test2.txt
HF
mv test3.txt test4.txt
ls
HF
cat test4.txt
rm test4.txt
ls
it007sh>
```

## ➢ Giải thích:

- Dùng mảng history[HF_size][MAX_LINE] để lưu những câu lệnh đã nhập (HF_size chính là số lượng tối đa câu lệnh được lưu, nếu bị quá số lượng tối đa, câu lệnh mới nhất sẽ thay thế câu lệnh cũ nhất trong hàm add_to_history)
- Hàm add_to_history(const char* command) để thêm lệnh mới vào lịch sử, ghi đè lệnh cũ nhất nếu đã đạt số lệnh tối đa được lưu trong lịch sử.
- Hàm print_history() để in ra lịch sử câu lệnh (HF_size hoặc nhỏ hơn HF_size câu lệnh gần nhất)

- Điều kiện if (strcmp(command, "HF") == 0): xét nếu người dùng nhập "HF", chương trình sẽ gọi print_history() trước khi gọi add_to_history(command) để in ra lịch sử trước khi lưu "HF" vào lịch sử. “continue” sang vòng lập kế tiếp để người dùng nhập câu lệnh khác.
- Những giải thuật khác để thực thi lệnh khác lệnh “HF” (tương tự yêu cầu 1)
- Giải thích kết quả:
  - HF đầu tiên: in ra “ls” và “cat test2.txt” vì vừa nhập 2 câu lệnh đó.
  - Những HF sau tương tự: in ra những câu lệnh đã nhập vào trước đó kể cả lệnh nhập sai và lệnh HF.

## 3. Câu 3

Trả lời:

### ➢ Giải thuật:

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

#define MAX_LINE 80
#define HF_size 20 // so cau lenh toi da duoc luu trong lich su

char history[HF_size][MAX_LINE];
int his_index = 0;
pid_t running_pid;

void add_to_history(const char *command) {
    strcpy(history[his_index], command);
    his_index = (his_index + 1) % HF_size;
    // thay cau lenh cu~ nhat bang cau lenh moi nhat neu da dat so cau lenh toi
da duoc luu
}

void print_history() {
    int i = 0;
    int print_index = his_index;
    do {
        if (strlen(history[print_index]) > 0) {
            printf("%s\n", history[print_index]);
        }
```

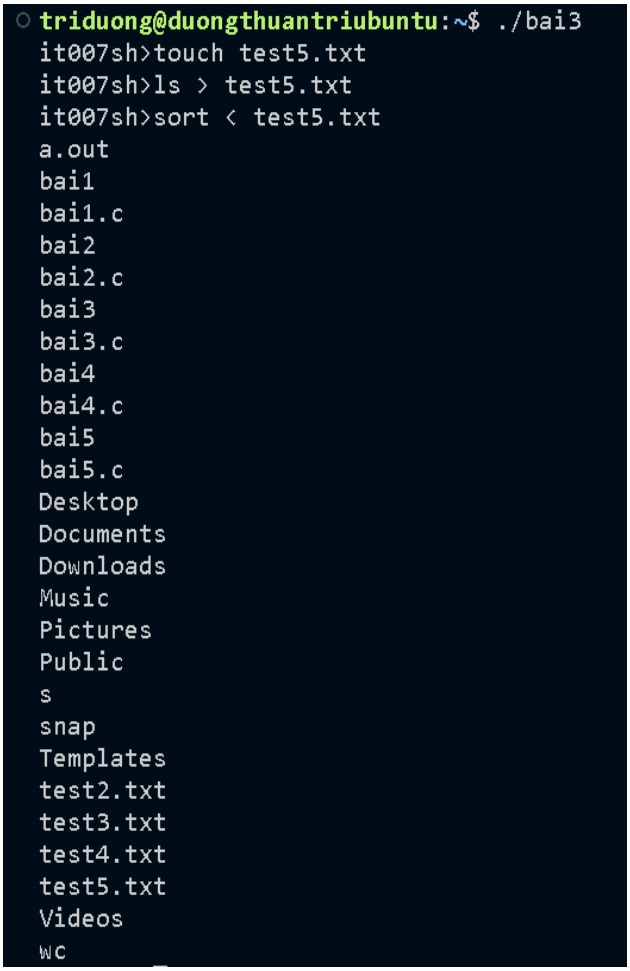
```
        print_index = (print_index + 1) % HF_size;
        i++;
    } while (i < HF_size);
}
// ham de xu li command binh thuong
void execute_command(char *args[]) {
    pid_t pid = fork();
    if (pid < 0) {
        fprintf(stderr, "Fork that bai!\n");
        exit(EXIT_FAILURE);
    } else if (pid == 0) {
        execvp(args[0], args);
        fprintf(stderr, "Khong tim thay lenh: %s\n", args[0]);
        exit(EXIT_FAILURE);
    } else {
        running_pid = pid;
        int status;
        waitpid(pid, &status, 0);
        running_pid = 0;
    }
}

// ham de xu li dau` ra/vao
void execute_redirection(char *args[], int input_fd, int output_fd) {
    pid_t pid = fork();
    if (pid < 0) {
        fprintf(stderr, "Fork that bai!\n");
        exit(EXIT_FAILURE);
    } else if (pid == 0) {
        if (input_fd != STDIN_FILENO) { //dau vao`
            dup2(input_fd, STDIN_FILENO);
            close(input_fd);
        }
        if (output_fd != STDOUT_FILENO) { //dau ra
            dup2(output_fd, STDOUT_FILENO);
            close(output_fd);
        }
        execvp(args[0], args);
        perror("execvp");
        exit(EXIT_FAILURE);
    } else {
        int status;
        waitpid(pid, &status, 0);
    }
```

```
}

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

    while (shouldrun) {
        printf("it007sh>");
        fflush(stdout);
        char command[MAX_LINE]; // mang dung` de chua lenh
        fgets(command, MAX_LINE, stdin);
        command[strcspn(command, "\n")] = '\0';
        if (strcmp(command, "HF") == 0)
        {
            print_history(); // in ra truoc khi luu "HF" vao lich su
            add_to_history(command);
            continue;  //sang vong lap tiep theo
        }
        add_to_history(command);

        char *arg = strtok(command, " \n");
        int i = 0;
        while (arg != NULL) {
            args[i++] = arg;
            arg = strtok(NULL, " \n");
        }
        args[i] = NULL;
        // 2 bien output va input de dat co` khi co chuyen dau ra/vao ('>' '<')
        int output = 0;
        int input = 0;

        for (int j = 0; j < i; j++) {
            if (strcmp(args[j], ">") == 0) {
                output = 1;
                args[j] = NULL;
                int fd = open(args[j + 1], O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR
| S_IWUSR | S_IRGRP | S_IROTH);
                if (fd == -1) {
                    perror("open");
                    exit(EXIT_FAILURE);
                }
                execute_redirection(args, STDIN_FILENO, fd); // tach thanh` ham`
//rieng de gon gang hon
                break;
```

```
            } else if (strcmp(args[j], "<") == 0) {
                input = 1;
                args[j] = NULL;
                int fd = open(args[j + 1], O_RDONLY);
                if (fd == -1) {
                    perror("open");
                    exit(EXIT_FAILURE);
                }
                execute_redirection(args, fd, STDOUT_FILENO);
                break;
            }
        }
        // khi khong co chuyen huong dau ra/vao thi xu li binh thuong`
        if (!output && !input) {
            execute_command(args); // tach thanh` ham` rieng de gon gang hon
        }
    }

    return 0;
}
```

## ➢ Output:

```
triduong@duongthuantriubuntu:~$ ./bai3
it007sh>ls
a.out  bai1.c  bai2.c  bai3.c  bai4.c  bai5.c   Documents  Music     Public  snap       test2.txt  Videos
bai1   bai2    bai3    bai4    bai5    Desktop  Downloads  Pictures  s       Templates  test3.txt  wc
it007sh>touch test4.txt
it007sh>ls > test4.txt
it007sh>cat test4.txt
a.out
bai1
bai1.c
bai2
bai2.c
bai3
bai3.c
bai4
bai4.c
bai5
bai5.c
Desktop
Documents
Downloads
Music
Pictures
Public
s
snap
Templates
test2.txt
test3.txt
test4.txt
Videos
wc
```





➢ **Giải thích:**

- Vòng lặp for (int j = 0; j < i; j++) { ... } để quét qua mảng các đối số của lệnh tìm xem có kí tự '>' hay '<' không.
- Chuyển hướng đầu ra (>):
  - Nếu tìm thấy đối số là >, đặt cờ output thành 1, loại bỏ dấu > khỏi danh sách đối số (args[j] = NULL).
  - Mở một file descriptor (fd) cho đầu ra (open(args[j + 1], O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)).
  - Gọi hàm execute_redirection để thực hiện lệnh với chuyển hướng đầu ra.
- Chuyển hướng đầu vào (<) tương tự:
  - Nếu tìm thấy đối số là <, đặt cờ input thành 1, loại bỏ dấu < khỏi danh sách đối số.
  - Mở một file descriptor (fd) cho đầu vào (open(args[j + 1], O_RDONLY);).

  - Gọi hàm execute_redirection để thực hiện lệnh với chuyển hướng đầu vào.
- Hàm execute_redirection: Đây là hàm thực hiện lệnh với chuyển hướng đầu vào và đầu ra. Nó nhận vào một mảng các đối số (args), một file descriptor cho đầu vào (input), và một file descriptor cho đầu ra (output). ( tùy vào tham số truyền vào để quyết định xử lí đầu vào/ra).
  - Dùng fork() để tạo một tiến trình con để thực hiện lệnh.
  - Tiến trình con (pid == 0):
    - Điều kiện if (input_fd != STDIN_FILENO): xét nếu có chuyển hướng đầu vào, sử dụng dup2(input_fd, STDIN_FILENO) để sao chép file descriptor của đầu vào vào STDIN_FILENO.
    - Gọi close(input) để đóng file descriptor của đầu vào, vì nó đã được sao chép vào STDIN_FILENO.
    - Điều kiện if (output_fd != STDOUT_FILENO): xét nếu có chuyển hướng đầu ra, sử dụng dup2(output_fd, STDOUT_FILENO) để sao chép file descriptor của đầu ra vào STDOUT_FILENO.
    - Gọi close(output) đóng file descriptor của đầu ra, vì nó đã được sao chép vào STDOUT_FILENO.
    - execvp(args[0], args): Thực hiện lệnh với các đối số đã được truyền vào. Nếu lệnh thất bại, in ra thông báo lỗi và thoát với mã lỗi.
  - Tiến trình cha: Đợi tiến trình con kết thúc. Trong khi đợi, chương trình cha bị đứng lại và không chạy lệnh tiếp theo cho đến khi tiến trình con hoàn thành.
- Giải thích kết quả:
  - “ls > test4.txt”: lưu output của lệnh ls vào file test4.txt vừa mới tạo ( chuyển hướng đầu ra)
  - “sort < test5.txt”: dùng nội dung của file test5.txt làm input của lệnh sort ( chuyển hướng đầu vào) -> in ra nội dung test5.txt đã được sắp xếp.
  - “cat test5.txt > test6.txt”: lưu nội dung của test5.txt vào test6.txt ( chuyển hướng đầu ra).

## 4. Câu 4

Trả lời:

### ➢ Giải thuật:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

```c
#include <sys/types.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <signal.h>

#define MAX_LINE 80
#define HF_size 20 // so cau lenh toi da duoc luu trong lich su

char history[HF_size][MAX_LINE];
int his_index = 0;
pid_t running_pid;

void add_to_history(const char *command) {
    strcpy(history[his_index], command);
    his_index = (his_index + 1) % HF_size;
    // thay cau lenh cu~ nhat bang cau lenh moi nhat neu da dat so cau lenh toi
da duoc luu
}

void print_history() {
    int i = 0;
    int print_index = his_index;
    do {
        if (strlen(history[print_index]) > 0) {
            printf("%s\n", history[print_index]);
        }
        print_index = (print_index + 1) % HF_size;
        i++;
    } while (i < HF_size);
}
// ham de xu li command binh thuong
void execute_command(char *args[]) {
    pid_t pid = fork();
    if (pid < 0) {
        fprintf(stderr, "Fork that bai!\n");
        exit(EXIT_FAILURE);
    } else if (pid == 0) {
        execvp(args[0], args);
        fprintf(stderr, "Khong tim thay lenh: %s\n", args[0]);
        exit(EXIT_FAILURE);
    } else {
        running_pid = pid;
        int status;
        waitpid(pid, &status, 0);
```

```
        running_pid = 0;
    }
}

// ham de xu li dau` ra/vao
void execute_redirection(char *args[], int input, int output) {
    pid_t pid = fork();
    if (pid < 0) {
        fprintf(stderr, "Fork that bai!\n");
        exit(EXIT_FAILURE);
    } else if (pid == 0) {
        if (input != STDIN_FILENO) { //dau vao`
            dup2(input, STDIN_FILENO);
            close(input);
        }
        if (output != STDOUT_FILENO) { //dau ra
            dup2(output, STDOUT_FILENO);
            close(output);
        }
        execvp(args[0], args);
        perror("execvp");
        exit(EXIT_FAILURE);
    } else {
        int status;
        waitpid(pid, &status, 0);
    }
}

void execute_pipeline(char *args1[], char *args2[]) {
    int pipe_fd[2];
    if (pipe(pipe_fd) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }
    pid_t pid1 = fork();
    if (pid1 == 0) {
        close(pipe_fd[0]);
        dup2(pipe_fd[1], STDOUT_FILENO);
        close(pipe_fd[1]);
        execvp(args1[0], args1);
        perror("execvp");
        exit(EXIT_FAILURE);
    } else if (pid1 > 0) {
        waitpid(pid1, NULL, 0);
```

```
        pid_t pid2 = fork();
        if (pid2 == 0) {
            close(pipe_fd[1]);
            dup2(pipe_fd[0], STDIN_FILENO);
            close(pipe_fd[0]);
            execvp(args2[0], args2);
            perror("execvp");
            exit(EXIT_FAILURE);
        } else if (pid2 > 0) {
            close(pipe_fd[0]);
            close(pipe_fd[1]);
            waitpid(pid2, NULL, 0);
        } else {
            perror("fork");
            exit(EXIT_FAILURE);
        }
    } else {
        perror("fork");
        exit(EXIT_FAILURE);
    }
}


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

    while (shouldrun) {
        printf("it007sh>");
        fflush(stdout);
        char command[MAX_LINE]; // mang dung` de chua lenh
        fgets(command, MAX_LINE, stdin);
        command[strcspn(command, "\n")] = '\0';
        if (strcmp(command, "HF") == 0)
        {
            print_history(); // in ra truoc khi luu "HF" vao lich su
            add_to_history(command);
            continue;  //sang vong lap tiep theo
        }
        add_to_history(command);

        char *arg = strtok(command, " \n");
        int i = 0;
```

```
        while (arg != NULL) {
            args[i++] = arg;
            arg = strtok(NULL, " \n");
        }
        args[i] = NULL;
        // 2 bien output va input de dat co` khi co chuyen dau ra/vao ('>' '<')
        int output = 0;
        int input = 0;
        int pipeline = 0;
        for (int j = 0; j < i; j++) {
            if (strcmp(args[j], ">") == 0) {
                output = 1;
                args[j] = NULL;
                int fd = open(args[j + 1], O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR
| S_IWUSR | S_IRGRP | S_IROTH);
                if (fd == -1) {
                    perror("open");
                    exit(EXIT_FAILURE);
                }
                execute_redirection(args, STDIN_FILENO, fd); // tach thanh` ham`
rieng de gon gang hon
                break;
            } else if (strcmp(args[j], "<") == 0) {
                input = 1;
                args[j] = NULL;
                int fd = open(args[j + 1], O_RDONLY);
                if (fd == -1) {
                    perror("open");
                    exit(EXIT_FAILURE);
                }
                execute_redirection(args, fd, STDOUT_FILENO);
                break;
            }
            else if (strcmp(args[j], "|") == 0) { // xet pipeline
                pipeline = 1;
                args[j] = NULL;
                char *args2[MAX_LINE / 2 + 1];
                int k = 0;
                for (int l = j + 1; l < i; l++) {
                    args2[k++] = args[l];
                }
                args2[k] = NULL;
                execute_pipeline(args, args2);
                break;
```

```
            }
        }
        // xu li binh thuong khi khong co redirect, pipeline
        if (!output && !input && !pipeline) {
            execute_command(args);
        }
    }

    return 0;
}
```

## ➢ Output:

```
triduong@duongthuantriubuntu:~$ ./bai4
it007sh>ls
a.out  bai1.c  bai2.c  bai3.c  bai4.c  bai5.c   Documents  Music     Public  snap       test2.txt  test4.txt  test6.txt  Videos
bai1   bai2    bai3    bai4    bai5    Desktop  Downloads  Pictures  s       Templates  test3.txt  test5.txt  test7.txt  wc
it007sh>ls | wc -l
28
it007sh>ls | head -5
a.out
bai1
bai1.c
bai2
bai2.c
```

```
it007sh>ls | sort
a.out
bai1
bai1.c
bai2
bai2.c
bai3
bai3.c
bai4
bai4.c
bai5
bai5.c
Desktop
Documents
Downloads
Music
Pictures
Public
s
snap
Templates
test2.txt
test3.txt
test4.txt
test5.txt
test6.txt
test7.txt
Videos
wc
```

## ➢ Giải thích:

- Vòng lặp for (int j = 0; j < i; j++) { ... } để quét qua mảng các đối số của lệnh tìm xem có kí tự '|' (pipeline) hay không. ('<' và '>' xử lí tương tự yêu cầu 4)
- Nếu tìm thấy đối số là '|', đặt cờ pipeline thành 1, loại bỏ dấu '|' khỏi danh sách đối số (args[j]=NULL), sau đó tách lệnh thành hai phần, tạo một mảng mới để lưu lệnh thứ hai (args2[MAX_LINE / 2 +1]) và gọi hàm execute_pipeline(args, args2) để xử lý 2 phần lệnh.
- Hàm execute_pipeline:
  - Tạo ống (pipe_fd): Sử dụng pipe để tạo một ống giữa hai tiến trình con.
  - Tạo tiến trình con đầu tiên (pid1): Đóng đầu đọc của ống, thiết lập đầu ra của tiến trình con thành đầu vào của ống, và thực hiện lệnh đầu tiên.
  - Chờ tiến trình con đầu tiên kết thúc: Tiến trình cha đợi cho tiến trình con đầu tiên kết thúc.
  - Tạo tiến trình con thứ hai (pid2): Đóng đầu ghi của ống và thiết lập đầu vào của tiến trình con thành đầu ra của ống, sau đó thực hiện lệnh thứ hai.
  - Chờ tiến trình con thứ hai kết thúc: Tiến trình cha đợi cho tiến trình con thứ hai kết thúc.
- Giải thích kết quả:
  - "ls | wc -l": liệt kê file, thư mục ở thư mục hiện tại và chuyển output đến lệnh wc -l để đếm số dòng.
  - "ls | head -5": liệt kê file, thư mục của thư mục hiện tại và chuyển output đến lệnh head -5 để chỉ hiển thị 5 dòng đầu tiên.
  - "ls | sort": tương tự và xuất ra output đã sắp xếp theo thứ tự từ điển.

## 5. Câu 5

Trả lời:

### ➢ Giải thuật:

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

#define MAX_LINE 80
#define HF_size 20 // so cau lenh toi da duoc luu trong lich su

char history[HF_size][MAX_LINE];
int his_index = 0;
pid_t running_pid;
```

```c
void add_to_history(const char *command) {
    strcpy(history[his_index], command);
    his_index = (his_index + 1) % HF_size;
    // thay cau lenh cu~ nhat bang cau lenh moi nhat neu da dat so cau lenh toi
da duoc luu
}

void print_history() {
    int i = 0;
    int print_index = his_index;
    do {
        if (strlen(history[print_index]) > 0) {
            printf("%s\n", history[print_index]);
        }
        print_index = (print_index + 1) % HF_size;
        i++;
    } while (i < HF_size);
}
// ham de xu li command binh thuong
void execute_command(char *args[]) {
    pid_t pid = fork();
    if (pid < 0) {
        fprintf(stderr, "Fork that bai!\n");
        exit(EXIT_FAILURE);
    } else if (pid == 0) {
        execvp(args[0], args);
        fprintf(stderr, "Khong tim thay lenh: %s\n", args[0]);
        exit(EXIT_FAILURE);
    } else {
        running_pid = pid;
        int status;
        waitpid(pid, &status, 0);
        running_pid = 0;
    }
}

// ham de xu li dau` ra/vao
void execute_redirection(char *args[], int input, int output) {
    pid_t pid = fork();
    if (pid < 0) {
        fprintf(stderr, "Fork that bai!\n");
        exit(EXIT_FAILURE);
    } else if (pid == 0) {
        if (input != STDIN_FILENO) { //dau vao`
```

```
            dup2(input, STDIN_FILENO);
            close(input);
        }
        if (output != STDOUT_FILENO) { //dau ra
            dup2(output, STDOUT_FILENO);
            close(output);
        }
        execvp(args[0], args);
        perror("execvp");
        exit(EXIT_FAILURE);
    } else {
        int status;
        waitpid(pid, &status, 0);
    }
}

void execute_pipeline(char *args1[], char *args2[]) {
    int pipe_fd[2];

    if (pipe(pipe_fd) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }

    pid_t pid1 = fork();

    if (pid1 == 0) {
        close(pipe_fd[0]);

        dup2(pipe_fd[1], STDOUT_FILENO);
        close(pipe_fd[1]);

        execvp(args1[0], args1);
        perror("execvp");
        exit(EXIT_FAILURE);
    } else if (pid1 > 0) {
        waitpid(pid1, NULL, 0);

        pid_t pid2 = fork();

        if (pid2 == 0) {
            close(pipe_fd[1]);

            dup2(pipe_fd[0], STDIN_FILENO);
```

```
            close(pipe_fd[0]);

            execvp(args2[0], args2);
            perror("execvp");
            exit(EXIT_FAILURE);
        } else if (pid2 > 0) {
            close(pipe_fd[0]);
            close(pipe_fd[1]);

            waitpid(pid2, NULL, 0);
        } else {
            perror("fork");
            exit(EXIT_FAILURE);
        }
    } else {
        perror("fork");
        exit(EXIT_FAILURE);
    }
}

void signal_handler(int sig) {
    if (sig == SIGINT && running_pid > 0) {
        kill(running_pid, SIGINT);
    }
    else{
        printf("\n");
        exit(EXIT_SUCCESS);
    }
}

int main(void) {
    if (signal(SIGINT, signal_handler) == SIG_ERR) {
        perror("signal");
        exit(EXIT_FAILURE);
    }
    char *args[MAX_LINE / 2 + 1];
   int shouldrun = 1;

    while (shouldrun) {
        printf("it007sh>");
        fflush(stdout);
        char command[MAX_LINE]; // mang dung` de chua lenh
        fgets(command, MAX_LINE, stdin);
        command[strcspn(command, "\n")] = '\0';
```

```
        if (strcmp(command, "HF") == 0)
        {
            print_history(); // in ra truoc khi luu "HF" vao lich su
            add_to_history(command);
            continue;  //sang vong lap tiep theo
        }
        add_to_history(command);

        char *arg = strtok(command, " \n");
        int i = 0;
        while (arg != NULL) {
            args[i++] = arg;
            arg = strtok(NULL, " \n");
        }
        args[i] = NULL;
        // 2 bien output va input de dat co` khi co chuyen dau ra/vao ('>' '<')
        int output = 0;
        int input = 0;
        int pipeline = 0;
        for (int j = 0; j < i; j++) {
            if (strcmp(args[j], ">") == 0) {
                output = 1;
                args[j] = NULL;
                int fd = open(args[j + 1], O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR
| S_IWUSR | S_IRGRP | S_IROTH);
                if (fd == -1) {
                    perror("open");
                    exit(EXIT_FAILURE);
                }
                execute_redirection(args, STDIN_FILENO, fd); // tach thanh` ham`
rieng de gon gang hon
                break;
            } else if (strcmp(args[j], "<") == 0) {
                input = 1;
                args[j] = NULL;
                int fd = open(args[j + 1], O_RDONLY);
                if (fd == -1) {
                    perror("open");
                    exit(EXIT_FAILURE);
                }
                execute_redirection(args, fd, STDOUT_FILENO);
                break;
            }
            else if (strcmp(args[j], "|") == 0) { // xet pipeline
```

```
                pipeline = 1;
                args[j] = NULL;
                char *args2[MAX_LINE / 2 + 1];
                int k = 0;
                for (int l = j + 1; l < i; l++) {
                    args2[k++] = args[l];
                }
                args2[k] = NULL;
                execute_pipeline(args, args2);
                break;
            }
        }
        // xu li binh thuong khi khong co redirect, pipeline
        if (!output && !input && !pipeline) {
            execute_command(args);
        }
    }

    return 0;
}
```

## ➢ Output:

```
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3623 triduong  20   0  800848  72232  39168 S   2,3   0,6   0:38.04 node
   3562 triduong  20   0  987416 127240  42496 S   0,7   1,1   0:45.36 node
  54387 root      20   0       0      0      0 I   0,3   0,0   0:01.13 kworker/1:1-mm_percpu_wq
  56215 triduong  20   0   15964   4352   3584 R   0,3   0,0   0:00.02 top
      1 root      20   0  168048  13056   8320 S   0,0   0,1   0:07.39 systemd
      2 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kthreadd
      3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 slub_flushwq
      6 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 netns
      8 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 kworker/0:0H-events_highpri
     10 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 mm_percpu_wq
     11 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_tasks_kthread
     12 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_tasks_rude_kthread
     13 root      20   0       0      0      0 I   0,0   0,0   0:00.00 rcu_tasks_trace_kthread
     14 root      20   0       0      0      0 S   0,0   0,0   0:00.29 ksoftirqd/0
     15 root      20   0       0      0      0 I   0,0   0,0   0:10.18 rcu_preempt
     16 root      rt   0       0      0      0 S   0,0   0,0   0:00.28 migration/0
     17 root     -51   0       0      0      0 S   0,0   0,0   0:00.00 idle_inject/0
     19 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/0
     20 root      20   0       0      0      0 S   0,0   0,0   0:00.00 cpuhp/1
     21 root     -51   0       0      0      0 S   0,0   0,0   0:00.00 idle_inject/1
it007sh>
```

```
Every 2,0s: date

Thứ hai, 01 Tháng 1 năm 2024 12:58:09 +07
```

```
it007sh>watch date
it007sh>
```


- **Giải thích:**
  - Gọi signal(SIGINT, signal_handler) để thiết lập sự kiện xử lý tín hiệu SIGINT (Ctrl+C) bằng cách gọi hàm signal_handler khi tín hiệu này được nhận.
  - Hàm signal_handler(int sig):
    - Điều kiện if (signo == SIGINT) kiểm tra xem tín hiệu nhận được có phải là SIGINT không (Ctrl+C).
    - Nếu có tiến trình đang chạy (running_pid != 0), thì gửi tín hiệu SIGINT đến tiến trình đó bằng hàm kill(running_pid, SIGINT). Điều này giúp dừng tiến trình đang chạy khi người dùng nhấn Ctrl+C.
    - Nếu không có tiến trình nào đang chạy (running_pid == 0), thoát chương trình shell bằng hàm exit(EXIT_SUCCESS).
  - Giải thích kết quả (gọi những tiến trình chạy liên tục: ‘top’, ‘watch date’, ‘yes’:
    - ‘top’: hiển thị thông tin động về các tiến trình đang chạy và sử dụng tài nguyên hệ thống.
    - ‘watch’: thực hiện lệnh date mỗi 2 giây và hiển thị kết quả.
    - ‘yes’: in ra màn hình một dòng chữ "y" liên tục.