LAB 06 – GROUP 5

CLASS : IT007.019.1 && IT007.019.2

Members:

1. Hà Huy Hoàng | 22520460
2. Nguyễn Duy Hoàng | 22520467
3. Nguyễn Hoàng Hiệp | 22520452
4. Nguyễn Hoàng Phúc | 22521129

SUMMARY

Task		Status	Members
Thực hành	1	Hoàn thành	4
	2	Hoàn thành	1
	3	Hoàn thành	2
	4	Hoàn thành	3
	5	Hoàn thành	1,2,3,4

Bài tập thực hành

Đoạn chương trình mẫu tạo ra dấu nhắc it007sh>:

```c
#include <stdio.h>
#include <unistd.h>
#define MAX LINE 80 /* The maximum length command */
int main(void) {
char *args[MAX LINE/2 + 1]; /* command line arguments */
int should run = 1; /* flag to determine when to exit program
*/
while (should run) { printf("it007sh>"); fflush(stdout);
/**
Do something
*/ }
return 0;
}
```

Dựa vào đoạn chương trình trên hãy thực hiện thêm các yêu cầu dưới đây

1. Thực thi command trong tiến trình con

```c
#include <stdio.h>
```

```c
#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;
    }
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

nguyenhoangphuc-22521129@LAPTOP-021S54DV:~$ gcc -o lab6 lab6.c
nguyenhoangphuc-22521129@LAPTOP-021S54DV:~$ ./lab6
it007sh> echo abc
abc
it007sh> echo

it007sh> echo abc
abc
it007sh>
```

2. Tạo tính năng sử dụng lại trên câu lệnh gần đây nhất

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

#define MAX_LINE 80 /* Độ dài tối đa của lệnh */

void process_command(char *command)
{
    // Kiểm tra nếu lệnh là "exit" thì thoát chương trình
    if (strcmp(command, "exit") == 0)
    {
        exit(0);
    }

    int status;
    pid_t pid = fork();

    if (pid == 0)
    {
```

```
            // Tách lệnh thành các đối số để thực thi bằng execvp
            char *argv[MAX_LINE / 2 + 1];
            int i = 0;

            while (command[i] != '\0')
            {
                // Gán con trỏ đến đầu từng đối số
                argv[i] = command + i;
                i++;

                // Di chuyển đến ký tự kết thúc của từng đối số
                while (command[i] != ' ' && command[i] != '\0')
                {
                    i++;
                }

                // Nếu là khoảng trắng thì đặt ký tự kết thúc và di chuyển đến
đối số tiếp theo
                if (command[i] == ' ')
                {
                    command[i] = '\0';
                    i++;
                }
            }

            // Ký tự kết thúc danh sách đối số
            argv[i] = NULL;

            // Thực thi lệnh bằng execvp và kiểm tra lỗi
            if (execvp(argv[0], argv) == -1)
            {
                perror("it007sh");
            }

            // Nếu execvp thành công, tiến trình con thoát
            exit(0);
        }
        else if (pid < 0)
        {
            // Xử lý lỗi khi tạo tiến trình con
            perror("it007sh");
        }
        else
        {
            // Đợi tiến trình con kết thúc
            while (waitpid(pid, &status, 0) != pid)
```

```
            ;
        }
    }

    int main(void)
    {
        char *args[MAX_LINE/2 + 1]; /* command line arguments */
        int should_run = 1;
        while (should_run)
        {
            // Đọc lệnh từ người dùng sử dụng readline
            char *command = readline("it007sh> ");

            // Kiểm tra nếu readline trả về NULL thoát chương trình
            if (command == NULL)
            {
                exit(0);
            }

            // Thêm lệnh vào lịch sử
            add_history(command);

            // Xử lý lệnh
            process_command(command);

            // Giải phóng bộ nhớ của lệnh
            free(command);
        }

        return 0;
    }
```

```
hoang-22520460@DESKTOP-EMKNK1L:~/Lab6$ ./Lab6_2
it007sh> ls -1
it007sh: Bad address
it007sh> echo abc
it007sh: Bad address
it007sh> echo abc
hoang-22520460@DESKTOP-EMKNK1L:~/Lab6$ ./Lab6_2
it007sh> ls -1
it007sh: Bad address
it007sh> echo abc
it007sh: Bad address
it007sh> ls -1
```

3. Chuyển hướng vào ra

```
#include <stdio.h>
```

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

#define MAX_LINE 80

int main(void)
{
    char *args[MAX_LINE / 2 + 1];
    int should_run = 1;
    int command_executed = 0;

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

        char input[MAX_LINE];
        fgets(input, sizeof(input), stdin);

        int arg_count = 0;
        char *token = strtok(input, " \n");

        while (token != NULL)
        {
            if (strcmp(token, "<") == 0)
            {
                token = strtok(NULL, " \n");
                FILE *file = fopen(token, "r");
                if (file != NULL)
                {
                    dup2(fileno(file), STDIN_FILENO);
                    fclose(file);
                }
                else
                {
                    printf("Cannot open %s for input\n", token);
                    break;
                }
            }
            else if (strcmp(token, ">") == 0)
            {
                token = strtok(NULL, " \n");
                FILE *file = fopen(token, "w");
```

```
                if (file != NULL)
                {
                    dup2(fileno(file), STDOUT_FILENO);
                    fclose(file);
                }
                else
                {
                    printf("Cannot open %s for output\n", token);
                    break;
                }
            }
            else
            {
                args[arg_count] = token;
                arg_count++;
            }

            token = strtok(NULL, " \n");
        }

        args[arg_count] = NULL;

        if (arg_count > 0)
        {
            pid_t pid = fork();
            if (pid == 0)
            {
                execvp(args[0], args);
                perror("Error in execvp");
                exit(EXIT_FAILURE);
            }
            else
            {
                wait(NULL);
                command_executed = 1;
            }
        }

        if (command_executed)
        {
            break;
        }
    }

    return 0;
}
```

in.txt

```
a
d
f
g
j
c
q
p
```

out.txt

```
6.4_3_lab6.c
C:\Users\Legion 5 Pro\.ssh\id_rsa
C:\Users\Legion 5 Pro\.ssh\id_rsa.pub
C:\Users\Legion 5 pro\.ssh
C:\Users\Legion 5 pro\.ssh.pub
Myweb
count.sh
count.txt
dkhp.sh
elif_control2.sh
for_loop.sh
for_loop2.sh
hello
hello.c
in.txt
lab5_bai2a
lab5_bai2a.c
lab5_bai2b
lab5_bai2b.c
lab6_3.c
lab_2.sh
monhoc.txt
nguyen duy hoang
out.txt
password.sh
sjf.c
sjf_test.c
test
```

```
test.c
test1
test2
test_1
test_execl
test_execl.c
test_system
test_system.c
time
time.c
```

```
nguyenduyhoang-22520467@HOANGND04:~$ gcc lab6_3.c -o test2
nguyenduyhoang-22520467@HOANGND04:~$ ./test2
it007sh> sort < in.txt
a
c
d
f
g
j
p
q
nguyenduyhoang-22520467@HOANGND04:~$ ./test2
it007sh> ls > out.txt
nguyenduyhoang-22520467@HOANGND04:~$
```

4. Giao tiếp sử dụng cơ chế đường ống

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

#define MAX_LINE 80 /* Độ dài tối đa của lệnh */

int main(void) {
    char args[MAX_LINE/2 + 1]; / Các đối số của dòng lệnh */
    int should_run = 1; /* Cờ để xác định khi nào thoát khỏi chương trình */
    char input[MAX_LINE];
    int pipe_fd[2];
    pid_t pid, pid2;

    while (should_run) {
        printf("it007sh> ");
```

```
        fflush(stdout);

        // Nhận đối số từ dòng lệnh
        if (!fgets(input, MAX_LINE, stdin)) {
            perror("fgets");
            continue;
        }

        // Thay thế ký tự xuống dòng bằng null terminator
        size_t length = strlen(input);
        if (input[length - 1] == '\n') {
            input[length - 1] = '\0';
        }

        // Kiểm tra điều kiện thoát
        if (strcmp(input, "exit") == 0) {
            should_run = 0;
            continue;
        }

        // Kiểm tra xem có sử dụng pipe không
        char *second_command = strchr(input, '|');
        if (second_command != NULL) {
            *second_command = '\0';
            second_command++;
        }

        // Tách dòng lệnh thành các đối số
        char *token = strtok(input, " ");
        int i = 0;
        while (token != NULL) {
            args[i++] = token;
            token = strtok(NULL, " ");
        }
        args[i] = NULL;

        // Tạo ống nếu cần
        if (second_command) {
            if (pipe(pipe_fd) == -1) {
                perror("pipe");
                exit(EXIT_FAILURE);
            }
        }

        // Tạo tiến trình con
        pid = fork();
```

```
        if (pid == 0) { // Tiến trình con
            if (second_command) {
                // Chuyển đầu ra của tiến trình con thành đầu vào của ống
                dup2(pipe_fd[1], STDOUT_FILENO);
                close(pipe_fd[0]);
                close(pipe_fd[1]);
            }
            // Thực hiện lệnh đầu tiên
            execvp(args[0], args);
            perror("execvp");
            exit(EXIT_FAILURE);
        } else if (pid < 0) {
            perror("fork");
            exit(EXIT_FAILURE);
        }

        // Tiến trình cha thực hiện lệnh thứ hai nếu có pipe
        if (second_command && pid > 0) {
            pid2 = fork();
            if (pid2 == 0) {
                // Chuyển đầu vào của tiến trình con thành đầu ra của ống
                dup2(pipe_fd[0], STDIN_FILENO);
                close(pipe_fd[1]);
                close(pipe_fd[0]);

                // Tách lệnh thứ hai thành các đối số
                token = strtok(second_command, " ");
                i = 0;
                while (token != NULL) {
                    args[i++] = token;
                    token = strtok(NULL, " ");
                }
                args[i] = NULL;

                // Thực hiện lệnh thứ hai
                execvp(args[0], args);
                perror("execvp");
                exit(EXIT_FAILURE);
            } else if (pid2 < 0) {
                perror("fork");
                exit(EXIT_FAILURE);
            }
        }

        // Đóng ống và chờ tiến trình con kết thúc
        if (second_command) {
```

```
            close(pipe_fd[0]);
            close(pipe_fd[1]);
            waitpid(pid, NULL, 0);
            waitpid(pid2, NULL, 0);
        } else {
            waitpid(pid, NULL, 0);
        }
    }
    return 0;
}
```

```
nguyenhoanghiep-22520452@HiepNH:~$ gcc lab6.c -o run
nguyenhoanghiep-22520452@HiepNH:~$ ./run
it007sh> echo hello
hello
it007sh> ls
'C:\Users\nghhi\.ssh'       case1.sh            pA.c                run
'C:\Users\nghhi\.ssh.pub'   case2.sh            pB.c                sjf.c
 FCFS.c                     exit                processA            snap
 FCFS.cpp                  'for file in *.sh'   processA.c          test
 Myweb                      lab2.sh             processB            testA
 bai252.sh                  lab5.c              processB.c          testB
 bai253.sh                  lab6.c              producer_consumer.c until_user.sh
 bai4                       new                 rr.c                while_for.sh
it007sh> ls -l | less
it007sh>
```

```
-rw-rw-r-- 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452  1508 Oct 23 14:22 pA.c
-rw-rw-r-- 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452  1377 Oct 23 14:23 pB.c
-rwxrwxr-x 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452 16312 Oct 23 14:15 processA
-rw-rw-r-- 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452  1507 Oct 23 15:12 processA.c
-rwxrwxr-x 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452 16264 Oct 23 14:15 processB
-rw-rw-r-- 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452  1376 Oct 23 15:11 processB.c
-rw-rw-r-- 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452  2641 Oct 23 15:37 producer_co
nsumer.c
-rw-rw-r-- 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452  1515 Nov 13 16:21 rr.c
-rwxrwxr-x 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452 16688 Dec 11 14:20 run
-rw-rw-r-- 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452  1808 Nov 13 16:32 sjf.c
drwx------ 3 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452  4096 Sep 25 13:32 snap
-rwxrwxr-x 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452 16688 Dec 11 13:59 test
-rwxrwxr-x 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452 16312 Oct 23 14:27 testA
-rwxrwxr-x 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452 16264 Oct 23 14:28 testB
-rwxrwxr-x 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452   157 Oct  9 14:01 until_user.
sh
-rwxrwxr-x 1 nguyenhoanghiep-22520452 nguyenhoanghiep-22520452    94 Oct  9 13:59 while_for.s
:
```

5. Kết thúc lệnh đang thực thi

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

#define MAX_LINE 80
```

```c
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 */
```

```c
                /* Wait for the child process to complete */
                wait(NULL);
            }
        }
    }

    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
nguyenhoangphuc-22521129@LAPTOP-021S54DV:~$ gcc -o lab6_5 lab6_5.c
nguyenhoangphuc-22521129@LAPTOP-021S54DV:~$ ./lab6_5
it007sh> top
```

```
top - 14:17:54 up  1:07,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  45 total,   1 running,  44 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.5 us,  0.5 sy,  0.0 ni, 98.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3799.1 total,   2507.4 free,    709.3 used,    582.5 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   2872.1 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    348 root      20   0  820140  80576  23252 S   6.7   2.1   0:11.19 python3.10
    678 nguyenh+  20   0  961920 102480  42056 S   6.7   2.6   0:26.20 node
      1 root      20   0  165984  11368   8356 S   0.0   0.3   0:01.20 systemd
      2 root      20   0    2456   1332   1220 S   0.0   0.0   0:00.00 init-systemd(Ub
      5 root      20   0    2500    196    196 S   0.0   0.0   0:00.18 init
     40 root      19  -1   47804  15144  14112 S   0.0   0.4   0:00.36 systemd-journal
     64 root      20   0   21960   5860   4368 S   0.0   0.2   0:00.54 systemd-udevd
     77 root      20   0    4764   1752   1212 S   0.0   0.0   0:01.81 snapfuse
     78 root      20   0    4496    156      8 S   0.0   0.0   0:00.00 snapfuse
     80 root      20   0    4496    156      8 S   0.0   0.0   0:00.00 snapfuse
     83 root      20   0    4628    160     12 S   0.0   0.0   0:00.00 snapfuse
     84 root      20   0    4496    152      0 S   0.0   0.0   0:00.00 snapfuse
     89 root      20   0    4764   1732   1196 S   0.0   0.0   0:03.66 snapfuse
     94 root      20   0    4496    188     44 S   0.0   0.0   0:00.00 snapfuse
     96 root      20   0    4800   1700   1272 S   0.0   0.0   0:02.57 snapfuse
    114 systemd+  20   0   25532  12616   8424 S   0.0   0.3   0:00.31 systemd-resolve
Ctrl + C pressed

it007sh> echo abc
abc
it007sh>
```