

Họ và tên: Bùi Lê Nhật Tri

MSSV: 23521634

Lớp: IT00007.P11.1

BÁO CÁO LAB 6

Câu 1,2:

Code:

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <sys/wait.h>
6  #include <fcntl.h>
7  #include <signal.h>
8  #include <termios.h>
9
10 #define MAX_LINE 80
11 #define MAX_HISTORY 10
12
13 char history[MAX_HISTORY][MAX_LINE];
14 int history_count = 0;
15 int history_index = -1;
16
17 void parse_and_execute(char *input);
18 void execute_command(char *input);
19 void add_to_history(char *command);
20 void handle_history(char *input);
21 void handle_signal(int sig);
22 char getch(void);
23 // Hàm xử lý tín hiệu Ctrl+C
24 void handle_signal(int sig)
25 {
26     if (sig == SIGINT)
27     {
28         printf("\nit007sh> ");
29         fflush(stdout);
30     }
31 }
32
33 // Hàm đọc một ký tự từ bàn phím mà không cần nhấn Enter
34
35 // Thêm lệnh vào lịch sử
36 void add_history(char *history[], char *command)
37 {
38     if (history_count == MAX_HISTORY)
39     {
40         free(history[0]);
41         for (int i = 0; i < MAX_HISTORY - 1; i++)
42         {
43             history[i] = history[i + 1];
44         }

```

```

43         history[i] = history[i + 1];
44     }
45     history_count--;
46 }
47 history[history_count++] = strdup(command);
48 history_index++;
49 }
50
51 void get_history(char *history[])
52 {
53     printf("\nCommand History:\n");
54     for (int i = 0; i < history_count; i++)
55     {
56         printf("%d. %s\n", i + 1, history[i]);
57     }
58 }
59
60 int main(void)
61 {
62     char *args[MAX_LINE / 2 + 1];
63     char *history[MAX_HISTORY];
64
65     int should_run = 1;
66
67     while (should_run)
68     {
69         printf("\nit007sh> ");
70         fflush(stdout);
71
72         char command[MAX_LINE];
73         fgets(command, MAX_LINE, stdin);
74
75         size_t len = strlen(command);
76         if (len > 0 && command[len - 1] == '\n')
77         {
78             command[len - 1] = '\0';
79         }
80
81         if (strlen(command) == 0)
82         {
83             continue;

```

```

81     if (strcmp(command) == 0)
82     {
83         continue;
84     }
85
86     add_history(history, command);
87
88     int i = 0;
89     args[i] = strtok(strdup(command), " ");
90     while (args[i] != NULL)
91     {
92         i++;
93         args[i] = strtok(NULL, " ");
94     }
95     args[i] = NULL;
96     if (strcmp(args[0], "exit") == 0)
97     {
98         should_run = 0;
99     }
100    else if (strcmp(args[0], "history") == 0)
101    {
102        get_history(history);
103    }
104    else if (strcmp(args[0], "HF") == 0)
105    {
106        history_index--;
107        if (history_index >= 0)
108        {
109            strcpy(command, history[history_index]);
110            printf("%s\n", command);
111
112            i = 0;
113            args[i] = strtok(strdup(command), " ");
114            while (args[i] != NULL)
115            {
116                i++;
117                args[i] = strtok(NULL, " ");
118            }
119            args[i] = NULL;
120
121            pid_t pid = fork();
122            if (pid < 0)
123            {
124                fprintf(stderr, "Error: Failed\n");

```

```

123     {
124         fprintf(stderr, "Fork failed\n");
125         return 1;
126     }
127     else if (pid == 0)
128     {
129         if (execvp(args[0], args) == -1)
130         {
131             perror("Command not found");
132             exit(EXIT_FAILURE);
133         }
134     }
135     else
136         wait(NULL);
137 }
138 else
139 {
140     printf("Invalid history index\n");
141 }
142 }
143 else
144 {
145     pid_t pid = fork();
146     if (pid < 0)
147     {
148         fprintf(stderr, "Fork failed\n");
149         return 1;
150     }
151     else if (pid == 0)
152     {
153         if (execvp(args[0], args) == -1)
154         {
155             perror("Command not found");
156             exit(EXIT_FAILURE);
157         }
158     }
159     else
160     {
161         wait(NULL);

```

```

159     }
160 }
161     wait(NULL);
162 }
163 }
164 }
165 return 0;
166 }

```

Kết quả:

```
abc
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:

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <sys/wait.h>
6  #include <fcntl.h>
7  #include <signal.h>
8
9  #define MAX_LINE 80          // The maximum length of command
10 #define MAX_HISTORY 10      // Maximum number of commands to store in history
11
12 char history[MAX_HISTORY][MAX_LINE];
13 int history_count = 0;
14
15 // Function to add a command to history
16 void add_to_history(char *command) {
17     if (history_count < MAX_HISTORY) {
18         strcpy(history[history_count], command);
19         history_count++;
20     } else {
21         for (int i = 1; i < MAX_HISTORY; i++) {
22             strcpy(history[i - 1], history[i]);
23         }
24         strcpy(history[MAX_HISTORY - 1], command);
25     }
26 }
27
28 // Function to execute a command
29 void execute_command(char **args, int background) {
30     pid_t pid = fork();
31     if (pid < 0) {
32         perror("Fork failed");
33         return;
34     }
35
36     if (pid == 0) { // Child process
37         if (execvp(args[0], args) == -1) {
38             perror("Execution failed");
39             exit(1);
40         }
41     } else { // Parent process
42         if (!background) {
43             wait(NULL); // Wait for the child process to finish
44         }
45     }
46 }

```

```

44     }
45 }
46 }
47
48 // Function to handle the "echo" command with file reading
49 void execute_echo_with_file(char *filename) {
50     FILE *file = fopen(filename, "r");
51     if (file == NULL) {
52         perror("Failed to open file");
53         return;
54     }
55
56     char ch;
57     while ((ch = fgetc(file)) != EOF) {
58         putchar(ch);
59     }
60     fclose(file);
61     putchar('\n');
62 }
63
64 // Function to handle command execution with input/output redirection
65 void execute_command_with_redirection(char **args, char *input_file, char *output_file, int background) {
66     pid_t pid = fork();
67     if (pid < 0) {
68         perror("Fork failed");
69         return;
70     }
71
72     if (pid == 0) { // Child process
73         if (input_file) {
74             int fd_in = open(input_file, O_RDONLY);
75             if (fd_in == -1) {
76                 perror("Open input file failed");
77                 exit(1);
78             }
79             dup2(fd_in, STDIN_FILENO);
80             close(fd_in);
81         }
82
83         if (output_file) {
84             int fd_out = open(output_file, O_WRONLY | O_CREAT | O_TRUNC, 0644);

```



```

83     if (output_file) {
84         int fd_out = open(output_file, O_WRONLY | O_CREAT | O_TRUNC, 0644);
85         if (fd_out == -1) {
86             perror("Open output file failed");
87             exit(1);
88         }
89         dup2(fd_out, STDOUT_FILENO);
90         close(fd_out);
91     }
92
93     if (execvp(args[0], args) == -1) {
94         perror("Execution failed");
95     }
96     exit(1);
97 } else { // Parent process
98     if (!background) {
99         wait(NULL); // Wait for the child process to finish
100     }
101 }
102 }
103
104 // Function to handle command piping (|) between two commands
105 void execute_pipe(char **args1, char **args2) {
106     int fd[2];
107     if (pipe(fd) == -1) {
108         perror("Pipe failed");
109         return;
110     }
111
112     pid_t pid1 = fork();
113     if (pid1 < 0) {
114         perror("Fork failed");
115         return;
116     }
117
118     if (pid1 == 0) { // First child process
119         dup2(fd[1], STDOUT_FILENO); // Write to pipe
120         close(fd[0]);
121         close(fd[1]);
122         if (execvp(args1[0], args1) == -1) {

```

```

122     if (execvp(args1[0], args1) == -1) {
123         perror("Execution failed");
124         exit(1);
125     }
126 } else { // Parent process
127     pid_t pid2 = fork();
128     if (pid2 < 0) {
129         perror("Fork failed");
130         return;
131     }
132
133     if (pid2 == 0) { // Second child process
134         dup2(fd[0], STDIN_FILENO); // Read from pipe
135         close(fd[0]);
136         close(fd[1]);
137         if (execvp(args2[0], args2) == -1) {
138             perror("Execution failed");
139             exit(1);
140         }
141     } else { // Parent process
142         close(fd[0]);
143         close(fd[1]);
144         wait(NULL); // Wait for both child processes to finish
145         wait(NULL);
146     }
147 }
148 }
149
150 // Function to parse and execute a command
151 void parse_and_execute(char *input) {
152     char *args[MAX_LINE / 2 + 1];
153     int background = 0;
154     int redirect_output = 0;
155     int redirect_input = 0;
156     char *output_file = NULL;
157     char *input_file = NULL;
158
159     char *token = strtok(input, " ");
160     int i = 0;
161     while (token != NULL) {
162         if (strcmp(token, "&") == 0) {

```

```

163         background = 1;
164     } else if (strcmp(token, ">") == 0) {
165         redirect_output = 1;
166         token = strtok(NULL, " ");
167         output_file = token;
168     } else if (strcmp(token, "<") == 0) {
169         redirect_input = 1;
170         token = strtok(NULL, " ");
171         input_file = token;
172     } else {
173         args[i] = token;
174         i++;
175     }
176     token = strtok(NULL, " ");
177 }
178 args[i] = NULL;
179
180 if (args[0] == NULL) {
181     return; // Empty command, nothing to do
182 }
183
184 add_to_history(input);
185
186 if (strcmp(args[0], "history") == 0) {
187     for (int j = history_count - 1; j >= 0; j--) {
188         printf("%s\n", history[j]);
189     }
190     return;
191 }
192
193 if (strcmp(args[0], "echo") == 0 && i == 2) {
194     // If args[1] is a filename, read it and display contents
195     char file_path[256];
196     snprintf(file_path, sizeof(file_path), "%s.txt", args[1]);
197     execute_echo_with_file(file_path);
198     return;
199 }
200
201 if (redirect_output || redirect_input) {
202     execute_command_with_redirection(args, input_file, output_file, background);
203 } else {

```

```

203     } else {
204         for (int j = 0; j < i; j++) {
205             if (strcmp(args[j], "|") == 0) {
206                 args[j] = NULL;
207                 execute_pipe(args, &args[j + 1]);
208                 return;
209             }
210         }
211         execute_command(args, background);
212     }
213 }
214
215 // Signal handler for SIGINT (Ctrl+C)
216 void handle_signal(int sig) {
217     if (sig == SIGINT) {
218         printf("\nMời người dùng nhập lệnh tiếp theo.");
219         fflush(stdout);
220     }
221 }
222
223 int main(void) {
224     signal(SIGINT, handle_signal); // Set up signal handler for Ctrl+C
225     char input[MAX_LINE];
226
227     while (1) {
228         printf("it007sh> ");
229         fflush(stdout);
230
231         if (fgets(input, MAX_LINE, stdin) == NULL) {
232             perror("fgets failed");
233             continue;
234         }
235
236         input[strlen(input) - 1] = '\0'; // Remove the newline character
237
238         if (strcmp(input, "exit") == 0) {
239             break;
240         }
241
242         parse_and_execute(input);
243     }
244
245     return 0;

```

Kết quả:

```

nhattri@nhattri-VirtualBox:~$ g++ test.cpp -o test && ./test < in.txt
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> █
```