Họ và tên: Nguyễn Minh Nguyễn

Mã số sinh viên: 23521046

Lớp: IT007.P14

# 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	$\boxtimes$	$\boxtimes$	$\boxtimes$	$\boxtimes$	$\boxtimes$
Chụp hình minh chứng (chạy ít nhất 3 lệnh)					
Giải thích code, kết quả					

Tự chấm điểm: 10

<Tên nhóm>\_LAB6.pdf

<sup>\*</sup>Lưu ý: Xuất báo cáo theo định dạng PDF, đặt tên theo cú pháp:

## 6.4. BÀI TẬP THỰC HÀNH

```
#include <stdlib.h>
#include <stdlib.h>
#include cunistd.h>
#include <signal.h>
#include <signal.h>
#include <sys/wait.h>
#include <fort.h>
#define MAX_CMDS_LEN 1024
#define MAX_CMDS_10
#define MAX_CMDS_10
#define MAX_CMDS_10 // Kich thước lịch sử lệnh
int history_Count = 0; // Đểm số lệnh trong lịch sử

void add_to_history(const char *cmd) {
    if (history_count < MISTORY_SIZE) {
        history[history_count+] = strdup(cmd); // Luu lệnh vào lịch sử
    } else {
        for (int i = 1; i < MISTORY_SIZE; i++) {
            history[i - 1] = history[i]; // Di chuyển lệnh lên
        }
    }

void print_history() {
    for (int i = 0; i < history_count; i++) {
        printf("%d: %s\n", i + 1, history[i]);
    }

void execute_command(char *cmd) {
        char *commands[MAX_CMUS];
        char *input_file = MUL;
        char *output_file = MUL;
        char *output_file = MUL;
        char *output_file = MUL;
    }
```

```
int background = 0;

// Phân tich cũ pháp lệnh để tách các lệnh theo dấu '|'
char *token = strtok(cmd, "|");
int cmd_count = 0;

while (token ≠ NULL) {
    commands[cmd_count+] = token;
    token = strtok(wull, "|");
}

commands[cmd_count] = NULL; // Kết thúc danh sách các lệnh

int pipefd[2 * (cmd_count - 1)]; // Màng để lưu file descriptors của các pipe

// Tao các pipe

for (int i = 0; i < cmd_count - 1; i++) {
    if (pipe(pipefd + i * 2) < 0) {
        perror("pipe failed");
        exit(EXII_FAILURE);
    }
}

for (int i = 0; i < cmd_count; i++) {
    char *args[MAX_MAN_LEN];
    token = strtok(commands[i], "\n");
    int j = 0;
    input_file = NULL; // Reset input and output files for each command output_file = NULL;

// Phân tich cũ pháp để tìm tệp đầu vào và đầu ra
    while (token ≠ NULL);

if (strcmp(token, "5") = 0) {
        background = 1; // Chay lệnh ở chế độ nến
    } else if (strcmp(token, "<") = 0) {
        input_file = strtok(NULL, "\n"); // Luu tên tệp đầu vào
    }
    etse if (strcmp(token, "<") = 0) {
        input_file = strtok(NULL, "\n"); // Luu tên tệp đầu vào
    }
}
etse if (strcmp(token, ">") = 0) {
        input_file = strtok(NULL, "\n"); // Luu tên tệp đầu vào
        output_file = strtok(NULL, "\n"); // Luu tên tệp đầu ra
    }
}
```

```
// Dong tat ca cae pipe descriptors
for (int k = 0; k < 2 * (cmd_count - 1); k++) {
    close(pipefd[k]);
}

execvp(args[0], args); // Thực thi lệnh
perror("execvp failed");
else if (pid < 0) {
    perror("fork failed");
}

// Dong tat ca cae pipe descriptors trong tiến trình cha
for (int i = 0; i < 2 * (cmd_count - 1); i++) {
    close(pipefd[i]);
}

// Dori tiến trình con kết thúc nếu không chạy ở chế độ nến
if (ibackground) {
    for (int i = 0; i < cmd_count; i++) {
        wait(NULL); // Bợi các tiến trình con
}
}

void sigint_handler(int sig) {
    printf("\nReceived SIGINT. Use 'exit' to quit the shell.\n");
}

int main() {
    char cmd[nAk_con_uth];
    // Đảng kỳ hàm xử lý tin hiệu SIGINT
    signal(SIGINT, sigint_handler);
while (1) {
    printf("iteO7sh> ");
    if (ifgets(cmd, sizeof(cmd), stdin)) {
```

```
break; // Thoát nếu có lỗi khi đọc

// Nếu lệnh là "exit", thoát khỏi chương trình
if (strncmp(cmd, "exit", 4) = 0) {
    break;
}

// Nếu lệnh là "HF", in ra lịch sử lệnh
if (strncmp(cmd, "HF", 2) = 0) {
    print_history();
    continue; // Trở lại vòng lặp
}

// Thêm lệnh vào lịch sử
add_to_history(cmd);
    execute_command(cmd);
}

// Giải phóng bộ nhớ cho lịch sử lệnh
for (int i = 0; i < history_count; i++) {
    free(history[i]);
}

return 0;
```

Đoan code của cả 5 câu

#### Câu 1:

Khi người dùng nhập lệnh như it007sh> cat abc.txt, chương trình sẽ tạo một tiến trình con để thực thi lệnh này. Giải thuật thực hiện bao gồm việc sử dụng hàm fork() để tạo tiến trình con, sau đó sử dụng execvp() để thay thế tiến trình con bằng chương trình thực thi lệnh cat. Trong khi tiến trình con đang hoạt động, shell sẽ không cho phép người dùng nhập lệnh mới cho đến khi lệnh cat hoàn tất. Sau khi lệnh thực thi xong, dấu nhắc it007sh> sẽ được hiển thị lại, cho phép người dùng nhập lệnh tiếp theo.

Trong đoạn mã trên, phần thực thi lệnh được thực hiện trong hàm execute\_command. Khi người dùng nhập lệnh như cat abc.txt, hàm này được gọi để xử lý lệnh.

```
// Phân tích cú pháp để tìm tệp đấu vào và đấu ra
while (token ≠ NULL) {
   if (strcmp(token, "&") = 0) {
      background = 1; // Chạy lệnh ở chế độ nến
   } else if (strcmp(token, "<") = 0) {
      input_file = strtok(NULL, "\n"); // Lưu tên tệp đấu vào
   else if (strcmp(token, ">") = 0) {
      output_file = strtok(NULL, "\n"); // Lưu tên tệp đấu ra
   } else {
      args[j++] = token; // Lưu các tham số lệnh
   }
   token = strtok(NULL, "\n");
}
args[j] = NULL; // Kết thúc danh sách tham số
```

**Tách lệnh**: Đầu tiên, lệnh được tách thành các phần tử nhỏ hơn bằng cách sử dụng strtok, nhằm phân chia các lệnh (nếu có nhiều lệnh được phân cách bằng ký tự |).

**Tạo tiến trình con**: Vòng lặp for sẽ lặp qua từng lệnh tách được. Trong mỗi vòng lặp, fork() được gọi để tạo một tiến trình con. Nếu fork() trả về 0, có nghĩa là chúng ta đang trong tiến trình con.

```
execvp(args[0], args); // Thực thi lệnh
    perror("execvp failed");
    exit(EXIT_FATLURE);
} else if (pid < 0) {
    perror("fork failed");
}</pre>
```

**Thực thi lệnh**: Trong tiến trình con, execvp(args[0], args) được gọi để thay thế tiến trình con bằng chương trình thực thi lệnh cat (hoặc bất kỳ lệnh nào khác). Nếu lệnh thực thi thành công, tiến trình con sẽ không trở lại shell.

Chờ lệnh hoàn tất: Trong tiến trình cha, sau khi gọi execute\_command, shell sẽ đợi cho tiến trình con hoàn tất bằng cách sử dụng wait(NULL) (nếu không chạy ở chế độ nền), đảm bảo rằng người dùng không thể nhập lệnh mới cho đến khi lệnh trước đó hoàn tất.

## Kết quả:

```
(kali® kali)-[~/OS_lab6]

$ ./lab
it007sh> ls
lab lab.c
it007sh> touch text.txt
it007sh> ls
lab lab.c text.txt
it007sh> echo "Hello" > text.txt
it007sh> cat text.txt
it007sh> cat text.txt
```

#### Câu 2

Giải thuật:

l**ưu lệnh vào lịch sử**: Mỗi khi người dùng nhập lệnh, lệnh đó sẽ được thêm vào mảng lịch sử bằng cách gọi hàm add to history.

**Hiến thị lịch sử**: Khi người dùng nhập HF, hàm print\_history sẽ được gọi để in ra tất cả các lệnh đã lưu trong mảng lịch sử.

Giải thích code:

```
char *history[NISTORY_SIZE]; // Mång để lưu trữ lịch sử lệnh
int history_count = 0; // Đếm số lệnh trong lịch sử

void add_to_history(const char *cmd) {
   if (history_count < NISTORY_SIZE) {
      history[history_count+] = strdup(cmd); // Lưu lệnh vào lịch sử
   } else {
      free(history[0]); // Giải phóng lệnh cũ nhất
      for (int i = 1; i < NISTORY_SIZE; i++) {
         history[i - 1] = history[i]; // Di chuyển lệnh lên
      }
      history[NISTORY_SIZE - 1] = strdup(cmd); // Thêm lệnh mới vào cuối
   }
}

void print_history() {
   for (int i = 0; i < history_count; i++) {
       printf("%d: %s\n", i + 1, history[i]);
   }
}</pre>
```

Mảng history được sử dụng để lưu trữ các lệnh đã thực thi, với kích thước được xác định bởi HISTORY SIZE.

Hàm add\_to\_history kiểm tra xem mảng lịch sử đã đầy chưa. Nếu chưa, lệnh mới sẽ được thêm vào. Nếu đã đầy, lệnh cũ nhất sẽ được giải phóng và các lệnh khác sẽ được di chuyển lên để tạo chỗ cho lệnh mới.

Hàm print\_history duyệt qua mảng lịch sử và in ra từng lệnh cùng với chỉ số tương ứng, giúp người dùng dễ dàng nhận diện các lệnh đã thực thi.

Khi người dùng nhập HF, chương trình kiểm tra và gọi print\_history, sau đó quay lại vòng lặp để chờ lệnh tiếp theo mà không thực thi thêm lệnh nào.

## Kết quả:

```
-(kali® kali)-[~/0S_lab6]
it007sh> ls
lab lab.c
it007sh> touch text.txt
it007sh> ls
lab lab.c text.txt
it007sh> echo "Hello" > text.txt
it007sh> cat text.txt
"Hello"
it007sh> nano lab.c
it007sh> HF
1: ls
2: touch text.txt
3: ls
4: echo "Hello" > text.txt
5: cat text.txt
6: nano lab.c
it007sh>
```

Như trên hình đã sử dụng tổng cộng 6 lệnh. Khi nhập HF xuất ra các lệnh tương ứng theo thứ tư từ cũ nhất đến mới nhất.

#### Câu 3:

### Giải thuật:

**Phân tích cú pháp lệnh**: Khi người dùng nhập lệnh, chương trình sẽ phân tích xem có các ký tự > hoặc < trong lệnh hay không.

Xử lý chuyển hướng đầu ra: Nếu tìm thấy ký tự >, chương trình sẽ mở tệp đích và chuyển hướng đầu ra của lệnh đến tệp đó.

Xử lý chuyển hướng đầu vào: Nếu tìm thấy ký tự <, chương trình sẽ mở tệp nguồn và chuyển hướng đầu vào của lệnh từ tệp đó.

#### Giải thích code:

```
// Phân tích cú pháp để tìm tệp đấu vào và đấu ra
while (token ≠ NULL) {
   if (strcmp(token, "6") = 0) {
      background = 1; // Chạy lệnh ở chế độ nến
   } else if (strcmp(token, "<") = 0) {
      input_file = strtok(NULL, " \n"); // Lưu tên tệp đấu vào
   } else if (strcmp(token, ">") = 0) {
      output_file = strtok(NULL, " \n"); // Lưu tên tệp đấu ra
   } else {
      args[j++] = token; // Lưu các tham số lệnh
   }
   token = strtok(NULL, " \n");
}
args[j] = NULL; // Kết thúc danh sách tham số
```

Trong vòng lặp này, chương trình sẽ kiểm tra từng phần của lệnh. Nếu gặp ký tự <, nó sẽ lưu tên tệp vào biến input\_file. Nếu gặp ký tự >, nó sẽ lưu tên tệp vào biến output\_file.

```
// Xử lý tệp đấu vào
if (input_file) {
    int fd = open(input_file, O_ROONLY);
    if (fd < 0) {
        perror("open input file");
        exit(EXIT_FAILURE);
    }
    dup2(fd, STOIN_FILENO); // Chuyển hướng đấu vào close(fd);
}</pre>
```

Nếu có tệp đầu vào được chỉ định, chương trình sẽ mở tệp đó với quyền đọc. Nếu thành công, nó sẽ sử dung dup2() để chuyển hướng đầu vào của lênh từ têp.

```
// Xử lý tệp đấu ra
if (output_file) {
   int fd = open(output_file, O_WRONLY | O_CREAT | O_TRUNC, 0644);
   if (fd < 0) {
        perror("open output file");
        exit(EXIT_FAILURE);
   }
   dup2(fd, STDOUT_FILENO); // Chuyển hướng đấu ra
   close(fd);
}</pre>
```

Tương tự, nếu có tệp đầu ra được chỉ định, chương trình sẽ mở tệp đó với quyền ghi (tạo mới nếu chưa tồn tại, và cắt bỏ nội dung nếu đã tồn tại). Sau đó, nó sẽ chuyển hướng đầu ra của lệnh đến tệp.

Kết quả:

```
it007sh> ls
lab lab.c text.txt
it007sh> touch ls.txt
it007sh> ls > ls.txt
it007sh> cat < ls.txt
lab
lab.c
ls.txt
text.txt
it007sh>
```

Như trên hình output của lệnh ls đã được ghi vào file ls.txt.

#### Câu 4:

#### Giải thuật:

**Phân tích cú pháp lệnh**: Khi người dùng nhập lệnh có chứa ký tự |, chương trình sẽ tách lệnh thành các thành phần trước và sau ký tự này.

Tạo các tiến trình con: Mỗi lệnh sẽ được thực thi trong một tiến trình con riêng biệt.

**Thiết lập đường ống**: Sử dụng hàm pipe() để tạo một đường ống, cho phép kết nối đầu ra của lệnh đầu tiên với đầu vào của lệnh thứ hai.

Giải thích code:

```
// Phân tích cú pháp lệnh để tách các lệnh theo dấu '|'
char *token = strtok(cmd, "|");
int cmd_count = 0;

while (token ≠ NULL) {
    commands[cmd_count++] = token;
    token = strtok(NULL, "|");
}
commands[cmd_count] = NULL; // Kết thúc danh sách các lệnh
```

Đoạn mã này tách lệnh người dùng nhập vào thành các lệnh riêng biệt dựa trên ký tự |, lưu chúng vào mảng commands.

// Tao cac pipe
for (int i = 0; i < cmd\_count - 1; i++) {
 if (pipe(pipefd + i \* 2) < 0) {
basic=reversperror("pipe failed");
 exit(EXIT\_FAILURE);
 }
}</pre>

Ở đây, chương trình tạo ra các đường ống cần thiết cho số lượng lệnh. Mỗi đường ống

cần hai file descriptors (một cho đầu vào và một cho đầu ra).

```
if (pid = 0) { // Tien trinh con
    // Két nối đầu vào và đầu ra cho các lệnh trong pipeline
    if (i > 0) {
        dup2(pipefd[(i - 1) * 2], STDIN_FILENO); // Đầu vào từ pipe trước
    }
    if (i < cmd_count - 1) {
        dup2(pipefd[i * 2 + 1], STDOUT_FILENO); // Đầu ra vào pipe sau
    }
}</pre>
```

Trong mỗi tiến trình con, nếu không phải là lệnh đầu tiên, chương trình sẽ chuyển hướng đầu vào từ đường ống trước đó. Nếu không phải là lệnh cuối cùng, chương trình sẽ chuyển hướng đầu ra vào đường ống kế tiếp.

```
// Đóng tất cả các pipe descriptors trong tiến trình cha
for (int i = 0; i < 2 * (cmd_count - 1); i++) {
    close(pipefd[i]);
}</pre>
```

Sau khi thiết lập kết nối, tất cả các file descriptors của đường ống sẽ được đóng trong tiến trình con để không gây rò ri tài nguyên.

## Kết quả:

#### Câu 5:

### Giải thuật:

Đăng ký xử lý tín hiệu: Shell sẽ đăng ký một hàm xử lý tín hiệu cho tín hiệu SIGINT, cho phép chương trình xử lý khi người dùng nhấn Ctrl + C.

**Gửi tín hiệu đến tiến trình**: Khi tín hiệu SIGINT được nhận, hàm xử lý sẽ thực hiện việc dừng tiến trình hiện tại.

Giải thích code:

```
signal(SIGINT, sigint_handler);
```

Đoạn mã này đăng ký hàm sigint\_handler để xử lý tín hiệu SIGINT. Khi người dùng nhấn Ctrl + C, tín hiệu này sẽ được gửi đến shell.

```
void sigint_handler(int sig) {
    printf("\nReceived SIGINT. Use 'exit' to quit the shell.\n");
}
```

Hàm này sẽ được gọi khi tín hiệu SIGINT được nhận. Nó in ra thông báo và cho phép người dùng biết rằng lệnh hiện tại đã bị dừng.

```
F// Đợi tiến trình con kết thúc nếu không chạy ở chế độ nến
if (!background) {
    for (int i = 0; i < cmd_count; i++) {
        wait(NULL); // Đợi các tiến trình con
    }
}</pre>
Home
```

Nếu lệnh không chạy ở chế độ nền, shell sẽ chờ cho tiến trình con (lệnh đang chạy) kết thúc trước khi hiển thị lại dấu nhắc it007sh>. Kết quả:

1013   yout	PID USER	PR	NI	VIRT	RES	SHR S	%CPU	%MEM	TIME	COMMAND
4374 kali 20 0 456664 101044 85368 S 1.3 5.0 0:13.18 qterminal 1303 kali 20 0 1264228 122312 76156 S 1.0 6.1 0:17.78 xfwm4 1304 kali 20 0 340272 29999 20688 S 0.7 1.5 0:08.77 panel-15-genmon 574 root 20 0 243596 11156 7808 S 0.3 0.6 0:05.68 wttoolsd 1289 kali 20 0 226436 7936 7168 S 0.3 0.6 0:05.68 at-spi2-registr 46017 root 20 0 22612 12892 9564 S 0.0 0.6 0:00.75 systemd 1 root 20 0 0 0 0 0 0 5 0.0 0.0 0:00.13 kworker/0:1-mm_percpu_wq 1 root 20 0 0 0 0 0 5 0.0 0.0 0:00.00 kthreadd 2 root 20 0 0 0 0 0 5 0.0 0.0 0:00.00 kthreadd 3 root 20 0 0 0 0 0 5 0.0 0.0 0:00.00 kworker/R-rcu_g 5 root 0 -20 0 0 0 0 1 0.0 0.0 0:00.00 kworker/R-rcu_g 5 root 0 -20 0 0 0 0 1 0.0 0.0 0:00.00 kworker/R-rcu_g 6 root 0 -20 0 0 0 1 0.0 0.0 0:00.00 kworker/R-rcu_g 7 root 0 -20 0 0 0 1 0.0 0.0 0:00.00 kworker/R-rcu_g 9 root 0 -20 0 0 0 1 0.0 0.0 0:00.00 kworker/R-rcu_g 11 root 20 0 0 0 0 1 0.0 0.0 0:00.00 kworker/R-rcu_g 9 root 0 -20 0 0 0 1 0.0 0.0 0:00.00 kworker/R-rcu_g 11 root 20 0 0 0 0 1 0.0 0.0 0:00.00 kworker/R-rcu_g 12 root 0 -20 0 0 0 0 1 0.0 0.0 0:00.00 kworker/R-rcu_g 13 root 20 0 0 0 0 1 0.0 0.0 0:00.00 kworker/R-do-floppy 12 root 0 -20 0 0 0 0 1 0.0 0.0 0:00.00 kworker/W-lo44:0-floppy 13 root 20 0 0 0 0 1 0.0 0.0 0:00.00 kworker/W-flo44:0-floppy 14 root 20 0 0 0 0 0 1 0.0 0.0 0:00.00 kworker/R-mm_pe 15 root 20 0 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_trace_kthread 16 root 20 0 0 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_trace_kthread 16 root 20 0 0 0 0 0 0 S 0.0 0.0 0:00.00 rcu_tasks_trace_kthread 16 root 20 0 0 0 0 0 S 0.0 0.0 0:00.00 fcu_tasks_trace_kthread 16 root 20 0 0 0 0 0 S 0.0 0.0 0:00.00 fcu_tasks_trace_kthread 17 root 20 0 0 0 0 0 S 0.0 0.0 0:00.00 fcu_tasks_trace_kthread 18 root rt 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0 20 root 20 0 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1 22 root -51 0 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1 24 root 20 0 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 25 root rt 0 0 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 27 root rt 0 0 0 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 28 root -51 0 0 0 0 0 0 S 0.0 0										
1303   Kali   20   0   1264228   122312   76156   5   1.0   6.1   0:17.78   x fwm4     1364   Kali   20   0   340272   29996   20688   S   0.7   1.5   0:08.77   panel-15-genmon     1700   20   0   243596   11156   7808   5   0.3   0.6   0:05.68   wtoolsd     1289   Kali   20   0   243596   11156   7808   5   0.3   0.4   0:01.95   at-spi2-registr     46017   root   20   0   0   0   0   0   0   0   0										
1364 kali										
1289   Kali   20   0   243596   11156   7808   S   0.3   0.6   0.95.68   vmtoolsd										
1289   Kali   20										
46017 root										
1 root										
2 root										
4 root	2 root	20	0	0	0	0 S	0.0	0.0		
4 root	3 root	20	0	0	0	0 S	0.0	0.0	0:00.00	pool workqueue release
5 root	4 root	0	-20	0	0	0 I	0.0	0.0		
7 root 0 -20 0 0 0 1 0.0 0.0 0:00.00 kworker/R-netns 9 root 0 -20 0 0 0 1 0.0 0.0 0:00.00 kworker/0:0H-events_highpri 11 root 20 0 0 0 0 1 0.0 0.0 0:00.00 kworker/0:0H-events_highpri 12 root 0 -20 0 0 0 0 1 0.0 0.0 0:00.00 kworker/0:0H-events_highpri 13 root 20 0 0 0 0 1 0.0 0.0 0:00.00 kworker/R-mm_pe 13 root 20 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_rude_kthread 14 root 20 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_rude_kthread 15 root 20 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_thread 16 root 20 0 0 0 0 0 0 0 0 0 0 0:00.00 ksoftirqd/0  17 root 20 0 0 0 0 0 0 0 0 0 0 0:00.00 ksoftirqd/0  17 root 20 0 0 0 0 0 0 0 0 0 0:00.00 ksoftirqd/0  18 root rt 0 0 0 0 0 0 0 0 0:00.00 ksoftirqd/0  19 root -51 0 0 0 0 0 0 0 0 0:00.00 idle_inject/0  20 root 20 0 0 0 0 0 0 0 0 0:00.00 idle_inject/0  21 root 20 0 0 0 0 0 0 0 0 0 0:00.00 idle_inject/1  22 root -51 0 0 0 0 0 0 0 0 0:00.00 idle_inject/1  23 root rt 0 0 0 0 0 0 0 0 0 0:00.00 ksoftirqd/1  24 root 20 0 0 0 0 0 0 0 0 0 0:00.00 kworker/1:0H-events_highpri  27 root 20 0 0 0 0 0 0 0 0 0 0:00.00 ksoftirqd/1  28 root -51 0 0 0 0 0 0 0 0 0 0:00.00 ksoftirqd/2  29 root rt 0 0 0 0 0 0 0 0 0 0:00.00 ksoftirqd/2  30 root 20 0 0 0 0 0 0 0 0 0 0:00.00 kworker/1:0H-events_highpri  31 root 20 0 0 0 0 0 0 0 0 0 0:00.00 ksoftirqd/2  32 root 0 -20 0 0 0 0 0 0 0 0 0:00.00 ksoftirqd/2  33 root 20 0 0 0 0 0 0 0 0 0:00.00 ksorker/2:0H-kblockd  33 root 20 0 0 0 0 0 0 0 0 0:00.00 cpuhp/3	5 root	0	-20	0	0	0 I	0.0	0.0	0:00.00	kworker/R-rcu_p
9 root 0 -20 0 0 0 1 0.0 0.0 0:00.00 kworker/0:0H-events_highpri 11 root 20 0 0 0 0 1 0.0 0.0 0:00.00 kworker/u64:0-floppy 12 root 0 -20 0 0 0 1 0.0 0.0 0:00.00 kworker/u64:0-floppy 13 root 20 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_kthread 14 root 20 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_kthread 15 root 20 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_trace_kthread 15 root 20 0 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_trace_kthread 16 root 20 0 0 0 0 0 0 0 0 0 0 0:00.00 rcu_tasks_trace_kthread 16 root 20 0 0 0 0 0 0 0 0 0 0:00.00 rcu_tasks_trace_kthread 17 root 20 0 0 0 0 0 0 0 0 0 0:00.00 rcu_tasks_trace_kthread 18 root rt 0 0 0 0 0 0 0 0 0:00.00 rcu_tasks_trace_kthread 19 root -51 0 0 0 0 0 0 0 0 0:00.00 idle_inject/0 20 root 20 0 0 0 0 0 0 0 0 0:00.00 idle_inject/0 21 root 20 0 0 0 0 0 0 0 0 0:00.00 idle_inject/1 22 root -51 0 0 0 0 0 0 0 0 0:00.00 idle_inject/1 23 root rt 0 0 0 0 0 0 0 0 0:00.00 idle_inject/1 24 root 20 0 0 0 0 0 0 0 0 0:00.00 idle_inject/1 25 root 20 0 0 0 0 0 0 0 0 0:00.00 idle_inject/1 26 root 0 -20 0 0 0 0 0 0 0 0:00.00 idle_inject/2 27 root 20 0 0 0 0 0 0 0 0 0:00.00 idle_inject/2 28 root -51 0 0 0 0 0 0 0 0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 0 0 0 0 0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 0 0 0 0 0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 0 0 0 0 0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 0 0 0 0 0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 0 0 0 0 0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 0 0 0 0 0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 0 0 0 0 0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 0 0 0 0 0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 0 0 0 0 0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 0 0 0 0 0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 0 0 0 0 0 0:00.00 idle_inject/2	6 root	0	-20	0	0	0 I	0.0	0.0	0:00.00	kworker/R-slub_
11 root	7 root	0	-20	0	0	0 I	0.0	0.0	0:00.00	kworker/R-netns
12 root	9 root	0	-20	0	0		0.0	0.0	0:00.00	kworker/0:0H-events_highpri
13 root 20 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_kthread 14 root 20 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_rude_kthread 15 root 20 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_rude_kthread 16 root 20 0 0 0 0 0 5 0.0 0.0 0:00.00 ksoftirqd/0  17 root 20 0 0 0 0 0 R 0.0 0:00.00 ksoftirqd/0  17 root 20 0 0 0 0 0 0 0 0 0:00.00 ksoftirqd/0  18 root rt 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/0  20 root 20 0 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/0  21 root 20 0 0 0 0 0 5 0.0 0.0 0:00.00 cpuhp/0  21 root 20 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/1  22 root -51 0 0 0 0 5 0.0 0.0 0:00.00 cpuhp/1  22 root -51 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/1  23 root rt 0 0 0 0 5 0.0 0.0 0:00.01 riggation/1  24 root 20 0 0 0 0 5 0.0 0.0 0:00.17 riggation/1  25 root 0 -20 0 0 0 0 5 0.0 0.0 0:00.00 ksoftirqd/1  26 root 0 -20 0 0 0 0 5 0.0 0.0 0:00.00 ksoftirqd/1  27 root 20 0 0 0 0 5 0.0 0.0 0:00.00 ksoftirqd/1  28 root -51 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/2  29 root rt 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/2  29 root rt 0 0 0 0 0 5 0.0 0.0 0:00.00 ksoftirqd/2  30 root 20 0 0 0 0 5 0.0 0.0 0:00.00 ksortirqd/2  30 root 20 0 0 0 0 5 0.0 0.0 0:00.00 ksortirqd/2  31 root 20 0 0 0 0 0 5 0.0 0.0 0:00.00 ksortirqd/2  32 root 0 -20 0 0 0 0 0 0 0 0:00.00 ksortirqd/2  33 root 20 0 0 0 0 0 0 0 0 0:00.00 ksorker/2:0H-kblockd  33 root 20 0 0 0 0 0 0 0 0 0:00.00 cpuhp/3	11 root	20	0	0	0	0 I	0.0	0.0	0:00.00	kworker/u64:0-floppy
14 root 20 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_rude_kthread 15 root 20 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_trace_kthread 16 root 20 0 0 0 0 0 5 0.0 0.0 0:00.00 rcu_tasks_trace_kthread 16 root 20 0 0 0 0 0 8 0.0 0.0 0:00.00 rcu_tasks_trace_kthread 17 root 20 0 0 0 0 0 8 0.0 0.0 0:00.00 rcu_tasks_trace_kthread 18 root rt 0 0 0 0 8 0.0 0.0 0:00.06 ksoftirqd/0 19 root -51 0 0 0 0 8 0.0 0.0 0:00.12 migration/0 19 root -51 0 0 0 0 8 0.0 0.0 0:00.00 idle_inject/0 20 root 20 0 0 0 0 8 0.0 0.0 0:00.00 cpuhp/0 21 root 20 0 0 0 0 8 0.0 0.0 0:00.00 cpuhp/1 22 root -51 0 0 0 0 8 0.0 0.0 0:00.00 idle_inject/1 23 root rt 0 0 0 0 8 0.0 0.0 0:00.00 idle_inject/1 24 root 20 0 0 0 0 8 0.0 0.0 0:00.01 migration/1 24 root 20 0 0 0 0 8 0.0 0.0 0:00.06 ksoftirqd/1 26 root 0 -20 0 0 0 1 0.0 0:00.00 kworker/1:0H-events_highpri 27 root 20 0 0 0 0 8 0.0 0.0 0:00.00 cpuhp/2 28 root -51 0 0 0 0 8 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 8 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 8 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 8 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 8 0.0 0.0 0:00.00 idle_inject/2 30 root 20 0 0 0 0 8 0.0 0.0 0:00.00 ksoftirqd/2 32 root 0 -20 0 0 0 0 8 0.0 0.0 0:00.00 ksoftirqd/2 33 root 20 0 0 0 0 0 0 0 0 0.0 0:00.00 cpuhp/3	12 root	0	-20	0	0		0.0			
15 root 20 0 0 0 0 1 0.0 0.0 0:00.00 rcu_tasks_trace_kthread 16 root 20 0 0 0 0 0 5 0.0 0.0 0:00.06 ksoftirqd/0  17 root 20 0 0 0 0 R 0.0 0.0 0:00.06 ksoftirqd/0  18 root rt 0 0 0 0 5 0.0 0.0 0:00.12 migration/0  19 root -51 0 0 0 0 5 0.0 0.0 0:00.01 migration/0  20 root 20 0 0 0 0 5 0.0 0.0 0:00.00 cpuhp/0  21 root 20 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/0  22 root 20 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/1  23 root rt 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/1  24 root 20 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/1  25 root 75 0 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/1  26 root 0 -20 0 0 0 0 5 0.0 0.0 0:00.00 ksoftirqd/1  26 root 0 -20 0 0 0 0 5 0.0 0.0 0:00.00 ksoftirqd/1  27 root 20 0 0 0 0 5 0.0 0.0 0:00.00 cpuhp/2  28 root -51 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/2  29 root rt 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/2  29 root rt 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/2  29 root rt 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/2  29 root rt 0 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/2  29 root rt 0 0 0 0 0 5 0.0 0.0 0:00.00 ksoftirqd/2  30 root 20 0 0 0 0 0 5 0.0 0.0 0:00.06 ksoftirqd/2  32 root 0 -20 0 0 0 0 0 5 0.0 0.0 0:00.06 ksoftirqd/2  33 root 20 0 0 0 0 0 0 0 0 0 0:00.00 cpuhp/3		20	0	0	0		0.0	0.0	0:00.00	rcu_tasks_kthread
16 root 20 0 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/0  17 root 20 0 0 0 0 R 0.0 0.0 0:00.06 ksoftirqd/0  18 root rt 0 0 0 0 S 0.0 0.0 0:00.12 migration/0  19 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0  20 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0  21 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/1  22 root -51 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/1  23 root rt 0 0 0 0 S 0.0 0.0 0:00.17 migration/1  24 root 20 0 0 0 0 S 0.0 0.0 0:00.17 migration/1  24 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/1  26 root 0 -20 0 0 0 1 0.0 0:00.06 ksoftirqd/1  27 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kworker/1:0H-events_highpri  27 root 20 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2  28 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2  29 root rt 0 0 0 0 S 0.0 0.0 0:00.00 kworker/1:0H-events_highpri  30 root 20 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2  30 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kworker/2:0H-kblockd  33 root 20 0 0 0 0 S 0.0 0.0 0:00.04 kworker/2:0H-kblockd	14 root	20		0			0.0	0.0		
17 root										
18 root rt 0 0 0 0 S 0.0 0.0 0:00.12 migration/0 19 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0 20 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0 21 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/1 22 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1 23 root rt 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1 24 root 20 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1 25 root 20 0 0 0 0 S 0.0 0.0 0:00.017 migration/1 26 root 0 -20 0 0 0 I 0.0 0:00.06 ksoftirqd/1 27 root 20 0 0 0 I 0.0 0:00.00 wworker/1:0H-events_highpri 27 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/2 28 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 S 0.0 0.0 0:00.014 migration/2 30 root 20 0 0 0 S 0.0 0.0 0:00.014 migration/2 32 root 0 -20 0 0 0 I 0.0 0:00.06 ksoftirqd/2 32 root 0 -20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/2 33 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/2 33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/3										
19 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0 20 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0 21 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/1 22 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1 23 root rt 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1 24 root 20 0 0 0 0 S 0.0 0.0 0:00.17 migration/1 25 root 0 -20 0 0 0 I 0 S 0.0 0.0 0:00.06 ksoftirqd/1 26 root 0 -20 0 0 0 I 0.0 0:00.00 kworker/1:0H-events_highpri 27 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/2 28 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/2 30 root 20 0 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/2 32 root 0 -20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/2 33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/3										
20 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0 21 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/1 22 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1 23 root rt 0 0 0 0 S 0.0 0.0 0:00.17 migration/1 24 root 20 0 0 0 0 S 0.0 0.0 0:00.17 migration/1 26 root 0 -20 0 0 0 1 0.0 0:00.06 ksoftirqd/1 26 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/1 27 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kworker/1:0H-events_highpri 27 root 20 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 28 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 S 0.0 0.0 0:00.01 migration/2 30 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/2 32 root 0 -20 0 0 0 0 I 0.0 0:00.06 ksoftirqd/2 33 root 20 0 0 0 0 S 0.0 0.0 0:00.04 kworker/2:0H-kblockd 33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/3										
21 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/1 22 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1 23 root rt 0 0 0 0 S 0.0 0.0 0:00.17 migration/1 24 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/1 26 root 0 -20 0 0 0 0 I 0.0 0.0 0:00.00 kworker/1:0H-events_highpri 27 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/2 28 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 S 0.0 0.0 0:00.01 idle_inject/2 30 root 20 0 0 0 0 S 0.0 0.0 0:00.01 kworker/2:0H-kblockd 33 root 0 -20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/2										
22 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1 23 root rt 0 0 0 0 S 0.0 0.0 0:00.17 migration/1 24 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/1 26 root 0 -20 0 0 0 1 0.0 0:00.00 kworker/1:0H-events_highpri 27 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/2 28 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 S 0.0 0.0 0:00.14 migration/2 30 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/2 32 root 0 -20 0 0 0 0 I 0.0 0:00.06 ksoftirqd/2 33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/2										
23 root rt 0 0 0 0 S 0.0 0.0 0:00.17 migration/1 24 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/1 26 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 ksoftirqd/1 27 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/2 28 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 S 0.0 0.0 0:00.14 migration/2 30 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/2 32 root 0 -20 0 0 0 I 0.0 0:00.06 ksoftirqd/2 33 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/2 33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/3										
24 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/1 26 root 0 -20 0 0 0 I 0.0 0.0 0:00.06 ksoftirqd/1 27 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/2 28 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 S 0.0 0.0 0:00.14 migration/2 30 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/2 32 root 0 -20 0 0 0 I 0.0 0:00.04 kworker/2:0H-kblockd 33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/3										
26 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/1:0H-events_highpri 27 root 20 0 0 0 0 5 0.0 0.0 0:00.00 cpuhp/2 28 root -51 0 0 0 0 5 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 5 0.0 0.0 0:00.14 migration/2 30 root 20 0 0 0 0 5 0.0 0.0 0:00.06 ksoftirqd/2 32 root 0 -20 0 0 0 0 I 0.0 0:00.04 kworker/2:0H-kblockd 33 root 20 0 0 0 0 5 0.0 0.0 0:00.00 cpuhp/3										
27 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/2 28 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 S 0.0 0.0 0:00.14 migration/2 30 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/2 32 root 0 -20 0 0 0 0 I 0.0 0.0 0:00.04 kworker/2:0H-kblockd 33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/3										
28 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2 29 root rt 0 0 0 0 S 0.0 0.0 0:00.14 migration/2 30 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/2 32 root 0 -20 0 0 0 I 0.0 0:00.04 kworker/2:0H-kblockd 33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/3										
29 root rt 0 0 0 0 S 0.0 0.0 0:00.14 migration/2 30 root 20 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/2 32 root 0 -20 0 0 0 I 0.0 0.0 0:00.04 kworker/2:0H-kblockd 33 root 20 0 0 0 S 0.0 0.0 0:00.00 cpuhp/3										
30 root 20 0 0 0 0 S 0.0 0.0 0:00.06 ksoftirqd/2 32 root 0 -20 0 0 0 I 0.0 0.0 0:00.04 kworker/2:0H-kblockd 33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/3										
32 root 0 -20 0 0 0 I 0.0 0:00.04 kworker/2:0H-kblockd 33 root 20 0 0 0 S 0.0 0.0 0:00.00 cpuhp/3										
33 root 20 0 0 0 5 0.0 0.0 0:00.00 cpuhp/3										
eceived SIGINT. Use 'exit' to quit the shell.							0.0	0.0	0.00.00	Cpunp/ 3

Lệnh top

Như trên hình khi lệnh top khi đang thực thi bị dừng bởi tín hiệu CTRL+C và quay trở lại giao diện người dùng