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

Họ và tên: Lý Tuấn Khoa
Mã số sinh viên: 21522225
Lớp: CS4323.O11.CTTT

HỆ ĐIỀU HÀNH
BÁO CÁO LAB 6

1. Executing Command in a Child Process

My solution:

I use `fork()` to create a new process for each command and `execvp()` to execute the command in the child process. The parent process waits for the child to finish unless the command is specified to run in the background with "&". The shell exits when the user enters "exit". And here is an explanation of the code:

1.Header Files:

- `<sys/types.h>`: Provides data types used in system calls.
- `<stdio.h>`: Standard Input/Output functions.
- `<unistd.h>`: Provides access to the POSIX operating system API.
- `<string.h>`: String manipulation functions.
- `<stdlib.h>`: General utilities library.
- `<sys/wait.h>`: Declares the `wait()` function for process synchronization.

2. Constants:

- `MAX_LINE`: Maximum length of the command line.

3. Main Function:

- `char *args[MAX_LINE / 2 + 1]`: Array to store command-line arguments.
- `char line[MAX_LINE]`: Array to store the input command line.
- `int argc = 0`: Counter for the number of arguments.
- `int background = 0`: Flag to indicate if the command should run in the background.
- `pid_t pid`: Process ID variable.

4. Shell Loop:

- The program enters an infinite loop to repeatedly accept and process user commands until the user enters "exit."

5. User Input:

- `printf("21522225>");`: Display a simple shell prompt.
- `fgets(line, MAX_LINE, stdin);`: Read a line of input from the user.
- `line[strlen(line) - 1] = '\0';`: Remove the newline character from the input.

6. Parsing Input:

- The code then parses the input line into individual arguments and stores them in the `args` array.

7. Check for Exit:

- If the user enters "exit," the program breaks out of the loop and terminates.

8. Background Execution:

- The code checks if the last argument is "&" to determine if the command should run in the background.

9. Fork and Execute:

- `pid = fork();`: Create a new process.
- In the child process (`pid == 0`), it replaces its image with the specified command using `execvp(args[0], args);`.
- If `execvp` fails, an error message is printed using `perror("execvp");` this code to display when user enter a random char such as “dsjfhjfd” it will show an error in screen.

10. Parent Process:

- In the parent process (`pid > 0`), if the command is not running in the background, it waits for the child process to finish using `wait(NULL);`.

11. Repeat:

- The loop repeats for the next user input.

12. Exit:

- When the user enters "exit," the program terminates.

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

Picture of my code below:

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

#define MAX_LINE 80

int main(void) {
    char *args[MAX_LINE / 2 + 1];
    char line[MAX_LINE];
    int argc = 0;
    int background = 0;
    pid_t pid;

    while (1) {
        printf("21522225>");
        fflush(stdout);
        fgets(line, MAX_LINE, stdin);
        line[strlen(line) - 1] = '\0';
        argc = 0;
        background = 0;

        // Check for the exit command
        if (strcmp(line, "exit") == 0) {
            break;
        }

        for (int i = 0; i < strlen(line); i++) {
            if (line[i] != ' ') {
                args[argc] = &line[i];
                argc++;
                // Argument separator
                while (line[i] != ' ' && line[i] != '\0') {
                    i++;
                }
```

```
            }
        }
        args[argc] = NULL;

        // Check if the command should run in the background
        if (argc > 0 && strcmp(args[argc - 1], "&") == 0) {
            background = 1;
            args[argc - 1] = NULL;
        }

        pid = fork();

        if (pid == 0) {
            // Child process
            execvp(args[0], args);
            printf("excution fail \n");
            return 0;
        } else if (pid > 0) {
            // Parent process
            if (!background) {
                wait(NULL);
            }
        }
    }

    return 0;
}
```

After I executing the code it show the result below:

```
21522225>ls
bai1  bai3    bai3.sh  bai4.c   catprj    elif_control2.sh  for_loop.sh    password.sh  sjf    SRT    test.sh   try_variables.sh  while_for.sh
bai2  bai3.c  bai4     bai4.sh  catprj.c  for_loop2.sh      if_control.sh  RR.c         sjf.c  SRT.c  test.txt  until_user.sh
21522225>dskjlfdk
excution fail
21522225>exit
lytuankhoa_21522225/ $
```

In the provided shell program:

- First, I press `ls` command to successfully listed files and directories in the current directory.

- The non-existent command `dskjlfdk` failed, and the shell displayed "execution fail."

- The shell accepted user input, processed commands, and displayed a prompt (`21522225>`). Then I press “exit” to end the programs.

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

My reference:

https://drive.google.com/drive/folders/1twtkvypjUZjZ01Dx7u5n1IHChpvDYrih?usp=drive_link