簡易 shell 作業

1094841 李昱佑

繳交文件說明：

my_shell.c 為我所撰寫的程式碼，my_shell 為 my_shell 的執行檔。
而 child.c、child、c3、c3.c、guess_ab、guess_ab.c 為老師所提供的子程式。

功能規劃說明:（紅字為新加入的功能）

1. my_shell 一開啟會先輸出自己的 pid，如” Parent(my_shell) pid is xxx”。
2. 無論是否為老師提供的程式，只要在 my_shell 輸入其他程式的路徑(如：linux 內建的/bin/ls)，皆能正常的運行使用者所指定的程式。
3. 只要在 my_shell 提示符號後的輸入中最後為&，my_shell 就不會等待 child；反之，輸入的最後無&，則 my_shell 就會等待 child 執行完畢。
4. 當 child 執行結束後，回到 my_shell 時，my_shell 會輸出剛結束的 child 的 pid 與 status()，如” I'm parent(my_shell). Ended child pid: xxx, status: xxx”。
5. 如果輸入的 child program 不在目錄中或輸入的絕對路徑有誤，my_shell 會 output “xxx is not found”。
6. 如果 execl()執行失敗，會將 fork 出的 child 結束掉。
7. 如果 fork()失敗，會輸出 fork()失敗的資訊。

my_shell 程式說明: (my_shell.c 裡已寫入完整註解)

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

void input_split(char *input_str, char *path)
{
    // 用迴圈方式將 input string 中的路徑存放到 path
    for (int i = 0; i < strlen(input_str); i++)
    {
        if (input_str[i] == ' ' || input_str[i] == '&' || input_str[i] == '\n')
        {
            // 假如讀到空格或 \n 或 & 表示 input string 的 path 已結束
            path[i] = '\0'; // 將 path 最後一格填上 \0 變為字串
            break;
        }
        else
        {
            // 將 path index i 的 element 填上 input 的 i 的 element
            path[i] = input_str[i];
        }
    }
}
```

```c
int main()
{
    char no_wait = 0;
    pid_t parent_pid = getpid();
    printf("Parent(my_shell) pid is %d\n", parent_pid);
    for (;;) // 跑無限迴圈
    {
        char path[1026];      // store path，linux 路徑最長限制為 1024byte + 字串最後一格填 \0
        char input_str[1028]; // store input string，比 path 多 3byte 來存 &\n
        int child_exit_status;
        printf("1094841 ms> "); // print 提示符號

        if (fgets(input_str, sizeof(input_str), stdin))
        {
            no_wait = 0; // 先設為 0
            if (input_str[strlen(input_str) - 2] == '&')
            {
                //假如 input string 的最後一個字為 &
                no_wait = 1; // 表示 my_shell 不需等待 chlild
            }

            // 將 input string 中的路徑 存放到 path
            input_split(input_str, path);
            if (strlen(path) == 0)
                continue;
```

```c

            if (no_wait == 1)
            {
                // my_shell 不會 wait child
                pid_t pid = fork(); // 產生child
                if (pid == 0)
                {
                    // child 會跑以下程式碼
                    execl(path, path, NULL); // 將 child 改成指定路徑的程式碼
                    printf("%s is not found.\n", path);
                    exit(1); // execl 失敗就會中止 child
                }
                else if (pid > 0)
                {
                    // parent 什麼都不做
                }
                else
                {
                    // fork() 失敗
                    printf("fork() failed.\n");
                }
            }
```

```
75                else
76                {
77                    // my_shell 會 wait child
78                    pid_t pid = fork(); // 產生child
79
80                    if (pid == 0)
81                    {
82                        // child 會跑以下程式碼
83                        execl(path, path, NULL);
84                        printf("\n%s is not found.\n", path);
85                        exit(7); // execl 失敗就會中止 child ， 並回傳 7
86                    }
87                    else if (pid > 0)
88                    {
89                        // parent 會跑以下程式碼
90                        pid_t ended_child_pid = wait(&child_exit_status);
91                        if (WEXITSTATUS(child_exit_status) != 7)
92                        {
```

```
93                          //假如不是因為找不到而結束的則會印出以下
94                          printf("\nI'm parent(my_shell). Ended child pid: %d, status: %d\n", ended_child_pid, child_exit_status);
95                      }
96                  }
97                  else
98                  {
99                      // fork() 失敗
100                     printf("fork() failed.\n");
101                 }
102             }
103         }
104     }
105     return 0;
106 }
```

操作說明:

1. 可觀看我錄製的 demo 影片 : https://youtu.be/xm8M3FG7DPQ

   一、進入 my_shell 後可輸入指定的程式(可為絕對路徑或相對路徑)

```
1094841 ms> child
I am the child. pid: 84294
1 Continue? Type s to quit.
I am the child. pid: 84294
2 Continue? Type s to quit.
I am the child. pid: 84294
3 Continue? Type s to quit.
```

或

```
1094841 ms> /bin/ls
c3                      guess_ab.c
c3.c                    my_shell
child                   my_shell.c
child.c                 ~$說 明 檔 .docx
guess_ab                說 明 檔 .docx
```

   二、如果輸入的程式不存在，則會顯示下圖：

```
1094841 ms> abcd1234
abcd1234 is not found.
```

   三、如輸入的最後加上&，則 my_shell 不會等待 child：

```
1094841 ms> guess_ab &
1094841 ms> my pid is 84420
(guess_ab) Guess a 4-digit number. Type 'q' to quit.
(guess_ab) Guess: 1111

I'm parent(my_shell). Ended child pid: 84353, status: 0
1094841 ms> 1111 is not found.
1111
 0 A, 0 B
(guess_ab) Guess:
```

2. 其餘的測試操作與老師 demo 影片中相同