#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#include <string.h>

#define MAX\_LINE 80

int main(void) {

    char \*args[MAX\_LINE/2 + 1];

    int should\_run = 1;

    while (should\_run) {

        printf("it007sh>");

        fflush(stdout);

        char cmd[MAX\_LINE];

        fgets(cmd, MAX\_LINE, stdin);

        cmd[strlen(cmd)-1] = '\0';

        char \*pipe\_position = strchr(cmd, '|');

        if (pipe\_position != NULL) {

            \*pipe\_position = '\0';

            char \*cmd1 = cmd;

            char \*cmd2 = pipe\_position + 1;

            FILE \*pipe\_fp = popen(cmd1, "r");

            if (pipe\_fp == NULL) {

                perror("popen");

                exit(EXIT\_FAILURE);

            }

            pid\_t pid = fork();

            if (pid < 0) {

                perror("fork");

                exit(EXIT\_FAILURE);

            } else if (pid == 0) {

                dup2(fileno(pipe\_fp), STDIN\_FILENO);

                fclose(pipe\_fp);

                if (execlp("less", "less", NULL) == -1) {

                    perror("execlp");

                    exit(EXIT\_FAILURE);

                }

            } else {

                fclose(pipe\_fp);

                wait(NULL);

            }

        } else {

            int i = 0;

            args[i] = strtok(cmd, " ");

            while (args[i] != NULL) {

                i++;

                args[i] = strtok(NULL, " ");

            }

            pid\_t pid = fork();

            if (pid < 0) {

                perror("fork");

                exit(EXIT\_FAILURE);

            } else if (pid == 0) {

                execvp(args[0], args);

                perror("execvp");

                exit(EXIT\_FAILURE);

            } else {

                wait(NULL);

            }

        }

    }

    return 0;

}

Khởi tạo:

- Sử dụng thư viện và hằng số cần thiết.

- Định nghĩa một mảng để lưu trữ các đối số của lệnh và biến cờ để kiểm soát việc tiếp tục chương trình.

Vòng lặp chính:

- Hiển thị dấu nhắc của shell và đọc lệnh từ người dùng. (\*)

- Kiểm tra xem lệnh có chứa ký tự `|` không.

- Nếu có đường ống, sử dụng `popen` để tạo đường ống cho lệnh đầu tiên.

- Tách lệnh thành hai phần, tạo một tiến trình con để thực hiện lệnh thứ hai, và kết nối đầu vào của lệnh thứ hai với đầu ra của lệnh thứ nhất bằng lệnh pipe().

- Nếu không có dấu `|`, tách lệnh thành các đối số.

- Tạo một tiến trình con để thực hiện lệnh sử dụng `execvp`.

- Quay lại bước (\*) để đợi lệnh mới từ người dùng.

Chương trình này giúp hiểu cách một shell đơn giản có thể xử lý việc thực hiện lệnh và kết hợp chúng thông qua đường ống, giống như cách một shell thực tế hoạt động.