Assignment 4

Adv C Programming Comprehensive Assignment

Title: Mastering Advanced C Programing Concepts - Exception, System Call and Process Control

Objective: The objective of this assignment is to reinforce your understanding of advanced C programming and assembly language concepts. You will work on a series of tasks that cover topics such as exception, system call, and process control.

Assignment Task:

Implementing a shell program supporting pipe and redirection

This is a simple shell program that supports pipes and redirection for basic commands. It splits user input into individual commands separated by a pipe. It also handles input and output redirection, and executes the commands using child processes. Use execvp to execute the command line. execvp and

execve are both functions in C used for executing other programs from within a C program, but execvp is a simpler version of the exec family of functions.

Download assignment4.zip that includes assignment4.c and assignment4.h. Compile the code with your implementation and execute it. You should capture a screenshot of the execution result from the C code.

Test these commands and capture the result.

- ls > test.txt
- cat < test.txt
- ps -a > test.txt
- tail -n 2 test.txt
- ls -al | pwd
- pwd | ps

```
/*
* File: assignment4.c
* Owner:
* Date: 4.2.2024
* Description: Implementing a shell program in C
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <fcntl.h>
#include "assignment4.h"

// Function to parse user input into command and arguments
void command_line_parse(char *input, char *command, char **arguments) {
    char *token = strtok(input, " ");
    strcpy(command, token);

    int arg_count = 0;
    while (token != NULL) {
        arguments[arg_count] = token;
        arg_count++;

        token = strtok(NULL, " ");
    }
    arguments[arg_count] = NULL;
}
```

```
void command_line_execute(char *command, char **arguments) {
    if (execvp(command, arguments) == -1) {
        perror("Command execution failed");
        exit(EXIT_FAILURE);
    }
}

int main() {
    char input[MAX_COMMAND_LENGTH];
    char command[MAX_COMMAND_LENGTH];
    char *arguments[MAX_ARGUMENTS];

    while (1) {
        printf("CSE320_SHELL> ");
        fflush(stdout);

        // Read user input
        if (fgets(input, sizeof(input), stdin) == NULL) {
            break;
        }

        // Remove newline character
        input[strcspn(input, "\n")] = '\0';

        // TODO: Exit the shell if the user enters "exit"

        // Parse the input into command and arguments
        command_line_parse(input, command, arguments);

        int pipes[2]; // Used for pipe if needed

        // Check for input/output redirection
        int input_fd = 0; // Default to stdin
        int output_fd = 1; // Default to stdout

        for (int i = 0; arguments[i] != NULL; i++) {
            if (strcmp(arguments[i], "<") == 0) {
                printf("Input Redirection Detected!\n");
                // Input redirection
                arguments[i] = NULL; // Remove "<" from the argument list
                input_fd = open(arguments[i + 1], O_RDONLY);
                if (input_fd == -1) {
                    perror("Input redirection failed");
                    exit(EXIT_FAILURE);
```

```
            }
            i++;
        } else if (strcmp(arguments[i], ">") == 0) {
            printf("Output Redirection Detected!\n");
            // Output redirection
            arguments[i] = NULL; // Remove ">" from the argument list
            output_fd = open(arguments[i + 1], O_WRONLY | O_CREAT |
O_TRUNC, 0666);
            if (output_fd == -1) {
                perror("Output redirection failed");
                exit(EXIT_FAILURE);
            }
            i++;
        } else if (strcmp(arguments[i], "|") == 0) {
            printf("Pipe Detected!\n");
            // Pipe
            arguments[i] = NULL; // Split the command into two parts
            if (pipe(pipes) == -1) {
                perror("Pipe creation failed");
                exit(EXIT_FAILURE);
            }

            // TODO: Fork a child process for the left side of the pipe

            // TODO: Fork another child process for the right side of the
pipe

        }
    }

    // Fork a child process to execute the command
    pid_t pid = fork();

    if (pid < 0) {
        perror("Fork failed");
        exit(EXIT_FAILURE);
    } else if (pid == 0) {
        // Child process
        // Redirect input if needed
        if (input_fd != 0) {
            dup2(input_fd, 0);
            close(input_fd);
        }

        // Redirect output if needed
```

```
            if (output_fd != 1) {
                dup2(output_fd, 1);
                close(output_fd);
            }
            command_line_execute(command, arguments);
        } else {
            // Parent process
            // TODO: Wait child process and check if it is exited or terminated

        }
    }

    return 0;
}
```

Documentation

1. Document your code thoroughly with comments explaining each section.
2. Prepare a report PDF document summarizing everything such as each step or functions that you implemented, challenges faced, and what you've learned during this assignment.

Submission Guidelines:

- Submit the consolidated source code (a zip file) for the entire assignment.
- Include a README file with instructions for compiling and running your program.
- Ensure your code is well-documented with comments.

Assessment Criteria:

Your assignment will be assessed based on the following criteria:

- Submit your own work (80% in points) even though it doesn’t execute as expected.

- Proper use of C data types, pointers, arrays, structures, and dynamic memory allocation.
- Efficiency and readability of the code.
- Effective use of memory, functions, and function pointers.
- Proper error handling and validation.
- Utilization of macros for code optimization.
- Utilization of C standard library functions.
- Quality and completeness of documentation.

Important Note: **Plagiarism will not be tolerated, and students are expected to produce their work independently.** Please perform this assignment by yourself and don't copy any solutions from any Generative AI applications like ChatGPT, your friends or online. If I find any things that imply plagiarism, you will lose the whole points and be reported.