OS Assignment 1
Description

I have designed a simple shell that can handle three internal commands – ‘cd’, ‘echo’ and ‘pwd’ and five external commands – ‘ls’, ‘cat’, ‘date’, ‘rm’ and ‘mkdir’. I have implemented a shell using C library functions and Linux system calls.

My code has a while loop in the main function, that runs until “exit” is typed on the command line.

I have made a readline() function to take user input by using fgets() function. Then I made a parse_line() function to split the user input. This process is done using the strtok() function, that tokenises a string based on a delimiter.

Then interpret() function is called, that interprets the command entered by the user, and calls the internal_commands() function or external_commands() function depending on the user input.

This function also checks for “&t” at the end of the command and subsequently creates a thread using pthread_create() function.

Internal Commands

Internal commands are those which are interpreted by the shell program itself without requiring a different program to handle the expected operations. Internal commands implemented are as follows-

cd

It is used to change the current working directory to a specified directory.

cd has been implemented using getcwd() and chdir().

Flags implemented :

1. cd ..
   This command is used to move to the parent directory of current directory

2. cd ~
   This command is used to change directory to the home directory.

Handling Errors:

1. It handles the error when too many arguments are passed.

2. It handles the error when no arguments are passed.

System calls used :

1. getcwd()

2. chdir()

echo

It is used to display line of text/string that are passed as an argument.

Flags implemented :

1. -n
   When this flag is used, it doesn't append a new line.

2. -e
   This flag enables interpretation of backslash escapes.

Handling Errors and corner cases:

1. It handles the error when no arguments are passed.

2. It handles the case when user enters multiple words (It is treated as a string and printed on the terminal).

pwd

This command prints the current working directory of the shell.

Flags implemented :

1. -L

This flag uses PWD from the environment. It is used for showing the symbolic positioning of current directory.

2. -P

This flag is used for showing the physical positioning of current directory.

Handling Errors :

1. It handles the error if the user enters more number of arguments than required.
2. It handles the error when the user enters an invalid command or uses an invalid flag.

System calls used :

1. getcwd()

2. getenv()

exit

This command exits the user from the shell.

System calls used :

1. exit()

External Commands

External commands are commands that aren't implemented by the shell itself.
The shell creates a new child process which executes these commands.
The following system calls have been used to run all external commands :

- **fork()**
- **execvp()**
- **waitpid()**

In order to handle the external commands, a child process is created every time any of the external commands are used on the shell. It creates a child process using fork() system call and using execvp() system call which is used to replace the current process with a new process, we pass the arguments to the external commands program and access them. We use waitpid() system call to allow the parent process to wait for the termination of the child process.
Error handling is done for system calls using perror().

date

This command prints the system date and time.

Flags implemented :

1. -u
   This flag prints the time in Universal Time Coordinated (UTC) / Greenwich Mean Time (GMT).

2. -R
   This flag prints date and time in RFC 5322 format.

Handling Errors and a corner case :

1. It handles the error when the user enters an invalid command.

2. It handles the error when too many arguments are passed.

3. The exact syntax of the Linux system has been followed by this command.

System calls used :

1. time()
2. localtime()

cat

This command is used to concatenate files and print them on the standard output.

Flags implemented :

1. -E
   This flag adds $ at the end of each line of the file.

2. -n
   This flag numbers all the lines in the file.

Handling Errors:

1. It handles the error if no file is given to be opened (No arguments passed).
2. It handles the file not found error.
3. It handles the error when the user enters an invalid command.
4. It handles the error when too many arguments are passed.

System calls used :

1. fopen()
2. fclose()

ls

This command is used to list all files in a directory. The ls command implementation makes use of the dirent structure in C, which stores directory entries. Firstly, a directory is opened using opendir(), and then individual entries are read using the readdir() function. At last the directory is closed using closedir().

Flags implemented :

1. -a
   Using this flag, the command does not ignore entries starting with .

2. -m
   This flag displays all files and directories inside current directory separated by commas.

Handling Errors:

1. It handles the error when the user enters an invalid command.

2. It handles the error when too many arguments are passed.

3. The return value of opendir() system call is used for a NULL check.
   If the return value is NULL, we handle the error by printing directory does not exist.

System calls used :

1. opendir()

2. readdir()

3. closedir()

rm
This command is be used to remove files or directories.

Flags implemented :

1. -i
   This flag takes confirmation from the user before deleting a file.

2. -v
   This flag displays a message after removing a file.

Handling Errors:

1. It handles the error when the user enters an invalid command.
2. It handles the error when too many arguments are passed.
3. It handles the error if no file is given to be opened (No arguments passed).
4. It handles the file not found error.

System calls used :

1. remove()

mkdir
This command is used to create new directories.

Flags implemented

1. -p
   Takes a path and creates all of the necessary directories on the path that don’t exist.

2. -v
   This flag prints a message after creating a directory.

Handling Errors:

1. It handles the error when the user enters an invalid command.
2. It handles the error when too many arguments are passed.
3. It handles the error where no directory is given to be created (no arguments are passed).
4. It handles the directory already exists error.

System calls used :

1. mkdir()

Name : Sanmay Sood
Roll Number : 2021095
Section : A
Branch : CSE