Project 3 :

Unix Shell

Summary

The program wish.c is a command line interpreter (CLI) or, as it is more commonly known, a shell. When you type in a command (in response to its prompt), the shell creates a child process that executes the command you entered and then prompts for more user input when it has finished.

Basic shell : wish

Your basic shell called wish is basically an interactive loop.

1. It repeatedly prints a prompt wish> , parses the input, executes the command specified on that line of input, and waits for the command to finish. This is repeated until the user types exit.
2. The shell can be invoked with either no arguments or a single argument; anything else is an error.
3. The shell also supports a batch mode, which instead reads input from a batch file and executes commands from therein. In batch mode, no prompt should be printed.

```
vagrant@precise32:/vagrant/Linux/project_3$ gcc wish.c -o wish -std=gnu99
vagrant@precise32:/vagrant/Linux/project_3$ ./wish
wish> ls
batch.txt  batch_output.txt  output.txt  r.txt  wish  wish.c
wish> ls -la
total 44
drwxr-xr-x 1 vagrant vagrant   288 Jan 21 09:08 .
drwxr-xr-x 1 vagrant vagrant   192 Dec 28 23:19 ..
-rw-r--r-- 1 vagrant vagrant  6148 Jan 21 09:07 .DS_Store
-rw-r--r-- 1 vagrant vagrant    99 Jan 21 09:05 batch.txt
-rw-r--r-- 1 vagrant vagrant    18 Jan 21 09:05 batch_output.txt
-rw-r--r-- 1 vagrant vagrant    50 Jan 21 09:06 output.txt
-rw-r--r-- 1 vagrant vagrant    24 Jan 21 09:07 r.txt
-rwxr-xr-x 1 vagrant vagrant 12127 Jan 21 09:08 wish
-rw-r--r-- 1 vagrant vagrant  6376 Jan 21 09:03 wish.c
wish> nonexistentcommand
An error has occurred while executing the command
wish> exit
vagrant@precise32:/vagrant/Linux/project_3$ cat batch.txt
ls
pwd
echo "Hello from batch mode"
echo "Batch mode test" > batch_output.txt
cat batch_output.txt
vagrant@precise32:/vagrant/Linux/project_3$ ./wish batch.txt
batch.txt  batch_output.txt  output.txt  r.txt  wish  wish.c
/vagrant/Linux/project_3
"Hello from batch mode"
"Batch mode test"
```

Paths

The path variable contains the list of all directories to search, in order, when the user types a command.

1. For example, when the user types ls, and path is set to include both /bin and /usr/bin, try access("/bin/ls", X_OK). If that fails, try "/usr/bin/ls". If that fails too, it is an error.

```
vagrant@precise32:/vagrant/Linux/project_3$ ./wish
wish> ls
batch.txt  batch_output.txt  output.txt  r.txt  wish  wish.c
wish> path
wish> ls
An error has occurred while executing the command
wish> path /bin /usr/bin
wish> ls
batch.txt  batch_output.txt  output.txt  r.txt  wish  wish.c
wish> exit
```

Built-in Commands

Whenever your shell accepts a command, it should check whether the command is a built-in command or not. If it is, it should not be executed like other programs. Instead, your shell will invoke your implementation of the built-in command.

1. When the user types exit, your shell should simply call the exit system call with 0 as a parameter. It is an error to pass any arguments to exit.
2. The cd command always takes one argument (0 or >1 args should be signaled as an error).
3. The path command takes 0 or more arguments, with each argument separated by whitespace from the others. A typical usage would be like this: wish> path /bin /usr/bin, which would add /bin and /usr/bin to the search path of the shell.

```
vagrant@precise32:/vagrant/Linux/project_3$ ./wish
wish> exit 1
An error has occurred: 'exit' doesn't take arguments
An error has occurred while executing the command
wish> cd
An error has occurred while changing directory
wish> cd /tmp /nonexistent
An error has occurred while changing directory
wish> cd /
wish> pwd
/
wish> cd vagrant/Linux/project_3
wish> pwd
/vagrant/Linux/project_3
wish> path /bin /usr/bin
wish> ls
batch.txt  batch_output.txt  output.txt  r.txt  wish  wish.c
wish> exit
```

Redirection

Many times, a shell user prefers to send the output of a program to a file rather than to the screen. Usually, a shell provides this nice feature with the > character.

1. If a user types ls -la /tmp > output, nothing should be printed on the screen. Instead, the standard output of the ls program should be rerouted to the file output. In addition, the standard error output of the file should be rerouted to the file output.
2. If the output file exists before you run your program, you should simply overwrite it.
3. The exact format of redirection is a command (and possibly some arguments) followed by the redirection symbol followed by a filename. Multiple redirection operators or multiple files to the right of the redirection sign are errors.

```
vagrant@precise32:/vagrant/Linux/project_3$ ./wish
wish> ls -la /tmp > output.txt
wish> cat output.txt
total 8
drwxr-xr-x  2 root root 4096 Jan 20 15:17 .
drwxr-xr-x 23 root root 4096 Dec 26 13:47 ..
wish> ls > output.txt
wish> cat output.txt
batch.txt
batch_output.txt
output.txt
r.txt
wish
wish.c
wish> ls > output.txt > output2.txt
An error has occurred while handling redirection
wish> ls >
An error has occurred while handling redirection
wish> exit
```

Parallel Commands

Your shell will also allow the user to launch parallel commands.

1. Instead of running cmd1 and then waiting for it to finish, your shell should run cmd1, cmd2, and cmd3 (each with whatever arguments the user has passed to it) in parallel, before waiting for any of them to complete.

```
vagrant@precise32:/vagrant/Linux/project_3$ ./wish
wish> ls & invalidcmd & pwd
/vagrant/Linux/project_3
An error has occurred while executing the command
batch.txt  batch_output.txt  output.txt  r.txt  wish  wish.c
wish> ls & pwd & echo "Hello"
"Hello"
/vagrant/Linux/project_3
batch.txt  batch_output.txt  output.txt  r.txt  wish  wish.c
wish> ls & pwd & echo "Hello foeufneo" > output.txt
/vagrant/Linux/project_3
batch.txt  batch_output.txt  output.txt  r.txt  wish  wish.c
wish> cat output.txt
"Hello foeufneo"
wish> ls > output.txt & pwd > batch_output.txt
wish> cat output.txt
batch.txt
batch_output.txt
output.txt
r.txt
wish
wish.c
wish> cat batch_output.txt
/vagrant/Linux/project_3
wish> exit
```

Program Errors

You should print the standard error message whenever you encounter an error of any type.

1. The error message should be printed to stderr (standard error).
2. After any more errors, your shell simply continues processing after printing the one and only error message. However, if the shell is invoked with more than one file, or if the shell is passed a bad batch file, it should exit by calling exit(1).

Sources

I used the following sources while doing this project :

- ➔ Codes done for weekly exercises
- ➔ I got a lot of help from the TA Emma Pakarinen to fix the way the paths list and path_count variable were passed from one function to another using pointers. Also in fixing the adding of new paths to that list using dynamic memory management with the strdup() function.
- ➔ https://medium.com/@WinnieNgina/guide-to-code-a-simple-shell-in-c-bd4a3a4c41cd
- ➔ https://medium.com/@santiagobedoa/coding-a-shell-using-c-1ea939f10e7e
- ➔ https://www.geeksforgeeks.org/making-linux-shell-c/
- ➔ https://stackoverflow.com/questions/28502305/writing-a-simple-shell-in-c-using-fork-execvp
- ➔ https://stackoverflow.com/questions/70369289/why-cant-i-open-my-file-using-open-method
- ➔ https://www.reddit.com/r/learnprogramming/comments/4it2s3/would_someone_please_explainhelp_me_understand/
- ➔ https://stackoverflow.com/questions/4629643/implementing-a-unix-shell-in-c-check-if-file-is-executable
- ➔ https://www.geeksforgeeks.org/snprintf-c-library/
- ➔ https://stackoverflow.com/questions/3027320/why-first-arg-to-execve-must-be-path-to-executable
- ➔ https://stackoverflow.com/questions/19419338/trouble-with-strcmp-in-c
- ➔ https://stackoverflow.com/questions/21356795/exit0-in-c-only-works-occasionally
- ➔ https://www.linuxquestions.org/questions/linux-newbie-8/writing-a-shell-program-in-c-how-to-exit-4175572135/
- ➔ https://stackoverflow.com/questions/33721774/strange-bug-with-chdir-in-c
- ➔ https://solarianprogrammer.com/2019/04/03/c-programming-read-file-lines-fgets-getline-implement-portable-getline/
- ➔ https://stackoverflow.com/questions/2693776/removing-trailing-newline-character-from-fgets-input