1. Introduction
   1. Project overview

In this project a command line interpreter, also known as shell, was implemented. The shell called wish operates in a basic way: when you type in a command, the shell creates a child process that executes the command you entered and then prompts for more user input when it has finished.

* 1. Features

The shell can be invoked with either no arguments or a single argument; anything else is an error. The no-argument way is as follows:

prompt> ./wish

wish>

At this point, wish is running, and ready to accept commands. Type away!

The mode above is called *interactive* mode and allows the user to type commands directly. The shell also supports a *batch mode*, which instead reads input from a batch file and executes commands from therein. The way to run the shell with a batch file named batch.txt is as follows:

prompt> ./wish batch.txt

The shell has following features:

* The shell is an interactive loop: it repeatedly prints a prompt “wish> ”, parses the input, executes the command on that line of input, and waits for the command to finish. The loop runs until the user types “exit”.
* Interactive mode: allows the user to type commands directly.
* batch mode: reads input from a batch file and executes commands from therein.
* Built-in commands: The shell has three built-in commands which are “exit”, “cd” and “path”. With the “exit” command the shell simply calls the exit system call with 0 as a parameter. It is an error to pass any arguments to exit. The “cd” command always takes one argument, otherwise an error occurs. The “cd” command simply changes directories. The “path” command takes 0 or more arguments. A typical usage of “path” would be like this: wish> path /bin /usr/bin, which would add /bin and /usr/bin to the search path of the shell. If the user sets path to be empty, then the shell should not be able to run any programs (except built-in commands). The path command always overwrites the old path with the newly specified path.
* Other commands are run by creating a child process by using *fork()*, run by using *execv()* and waited to complete with *wait()*. Each command should only be able to run if it is defined in the path variable. The path variable contains the list of all directories to search, in order, when the user types a command.

The following features were NOT implemented:

* redirection.
* parallel commands.

The following errors are handled by the shell:

* Failing to use malloc
* Failing to use fork
* Failing to open a file
* Wrong number of arguments given to built-in commands.
* Wrong number of files given in batch mode.
* the command not found in path variable.

Whenever an error is encountered, the error message printed would be of type:

char error\_message[30] = "An error has occurred\n";

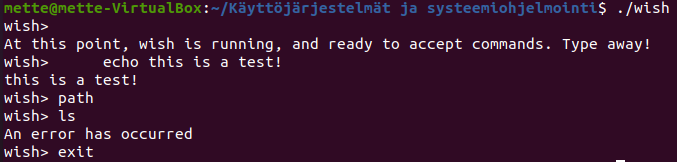
write(STDERR\_FILENO, error\_message, strlen(error\_message));

After most errors the shell continues processing after printing the error message. 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)*. The programs also exits by calling *exit(1)* when *malloc()* of *fork()* fails. If there are any program-related errors, the program will print its own error messages and exit.

1. Implementation

The shell works by running an infinite while loop in interactive mode. A prompt “wish> ” is printed repeatedly before a command. In batch mode no prompt is printed. The shell works by giving an input which is parsed in the *parseInput()* function. Parsing the input happens by creating a linked list and adding the arguments separated by space to the linked list. The program is robust to white space. After the the input is parsed, it is executed in the *executeCommand()* function. In the function it checks whether the command is a built-in command or not. If the command is not a built-in command, the command is searched from the path variable and if found from the path variable then executed by creating a child process. The parent process waits until the child process has completed.

An example of the shell in interactive mode:



An example of the shell in batch mode:

A screenshot of a computer screen

Description automatically generated

The input file test.txt:

A black text on a white background

Description automatically generated