**INDIAN INSTITUTE OF TECHNOLOGY (ISM) , DHANBAD**

**Department of Computer Science and Engineering**

**Operating System Lab**

|  |
| --- |
| REPORT |
| SHELL PROGRAMMING APPLICATION |
| OPERATING SYSTEM LAB PROJECT |

|  |
| --- |
| Team Members:  Aayush Dutt (17JE002903)  Abhinav Srivastava (17JE002910)  Devansh Saxena (17JE002920)  Jyoti Kumari (17JE002907)  Mayank Jain (17JE002935)  Piyush Chavan (17JE002914)  Samyak Tanted (17JE002905)  Tarun Lalchandani (17JE002904)  Ujjwal Mani Tripathi (17JE002921)  Vinit Sharma (17JE002922) |

Project Title :

Shell Programming application

Table of Content:

The shell implementation is divided into three parts:

* Parser
* Executor
* Shell Subsystems

Introduction:

The shell is a program that interacts with the user through a terminal or takes the input from a file and executes a sequence of commands that are passed to the Operating System.

Implementation of grammar in shell.l and shell.y to make our parser interpret the command lines and provide our executor with the correct information and performs below task.

**Parser:**

The Parser is the software component that reads the command line such as “ls -al” and puts it into a data structure called Command Table that will store the commands that will be executed.

To parse the line into a list of arguments. We are going to make a glaring simplification here, and say that we won’t allow quoting or backslash escaping in our command line arguments. Instead, we will simply use whitespace to separate arguments from each other. So the command echo "this message" would not call echo with a single argument “this message”, but rather it would call echo with two arguments: "this and message".

With those simplifications, all we need to do is “tokenize” the string using whitespace as delimiters. That means we can break out the classic library function “strtok”

The call correction(w) tries to choose the most likely spelling correction for w. There is no way to know for sure (for example, should "lates" be corrected to "late" or "latest" or "lattes" or ...?), which suggests we use probabilities. We are trying to find the correction *c*, out of all possible candidate corrections, that maximizes the probability that *c* is the intended correction, given the original word *w*:

argmax*c ∈ candidates* P(*c*|*w*)

By [Bayes' Theorem](http://en.wikipedia.org/wiki/Bayes'_theorem) this is equivalent to:

argmax*c ∈ candidates* P(*c*) P(*w*|*c*) / P(*w*)

Since P(*w*) is the same for every possible candidate *c*, we can factor it out, giving:

argmax*c ∈ candidates* P(*c*) P(*w*|*c*)

The four parts of this expression are:

1. **Selection Mechanism**: argmax  
   We choose the candidate with the highest combined probability.
2. **Candidate Model**: *c ∈ candidates*  
   This tells us which candidate corrections, *c*, to consider.
3. **Language Model**: P(*c*)  
   The probability that *c* appears as a word of English text. For example, occurrences of "the" make up about 7% of English text, so we should have P(*the*) = 0.07.
4. **Error Model**: P(*w*|*c*)  
   The probability that *w* would be typed in a text when the author meant *c*. For example, P(*teh*|*the*) is relatively high, but P(*theeexyz*|*the*) would be very low.

**Executor:**

The executor takes the command table generated by the parser and for every SimpleCommand in the array it creates a new process. It also if necessary creates pipes to communicate the output of one process to the input of the next one. Additionally, it redirects the standard input, standard output, and standard error if there are any redirections.

In the child process, we want to run the command given by the user. So, we use one of the many variants of the exec system call, execvp. The different variants of exec do slightly different things. Some take a variable number of string arguments. Others take a list of strings.

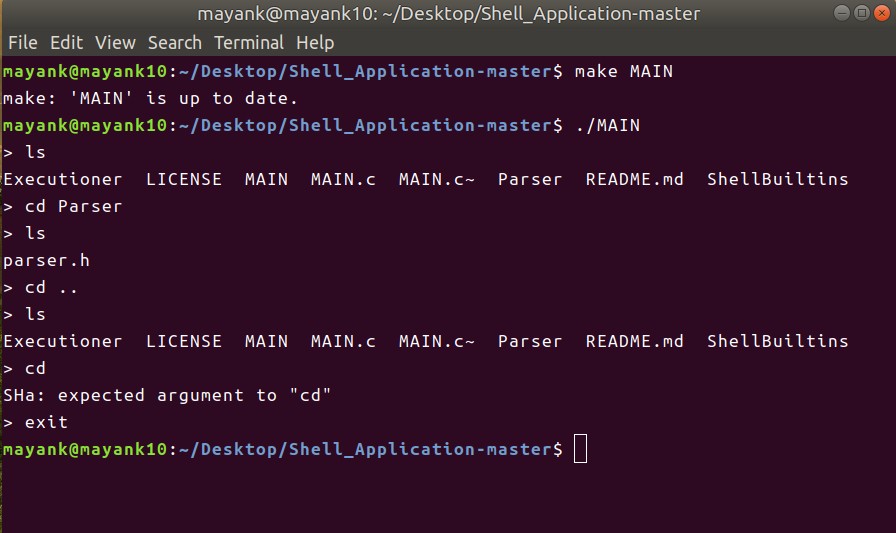
Still others let us specify the environment that the process runs with. Our executioner expects a program name and an array (also called a vector, hence the ‘v’) of string arguments (the first one has to be the program name). The ‘p’ means that instead of providing the full file path of the program to run, we are going to give its name, and let the operating system search for the program in the path.

**Implementing Wildcards in Shell**

No shell is complete without wildcards. Wildcards is a shell feature that allows one single command to be performed on multiple files that match the wildcard.

A wildcard describes filenames that match the wildcard. A wildcard works by iterating over all the files in the current directory or the directory described in the wildcard and then as arguments to the command those filenames that match the wildcard.

In general the “\*” character matches 0 or more characters of any type. The character ”?” matches one character of any type.



**Shell Program Application**

References:

<https://norvig.com/spell-correct.html>

<https://www.geeksforgeeks.org/>

<https://www.wikipedia.org/>