Indian Institute of Technology (Indian School of Mines) Dhanbad

Dept. Of Computer science and Engineering

Operating System Lab (CSC15202) Project
2019-20

Shell Programming Application

OBJECTIVE: 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.

PROJECT DESCRIPTION: 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.

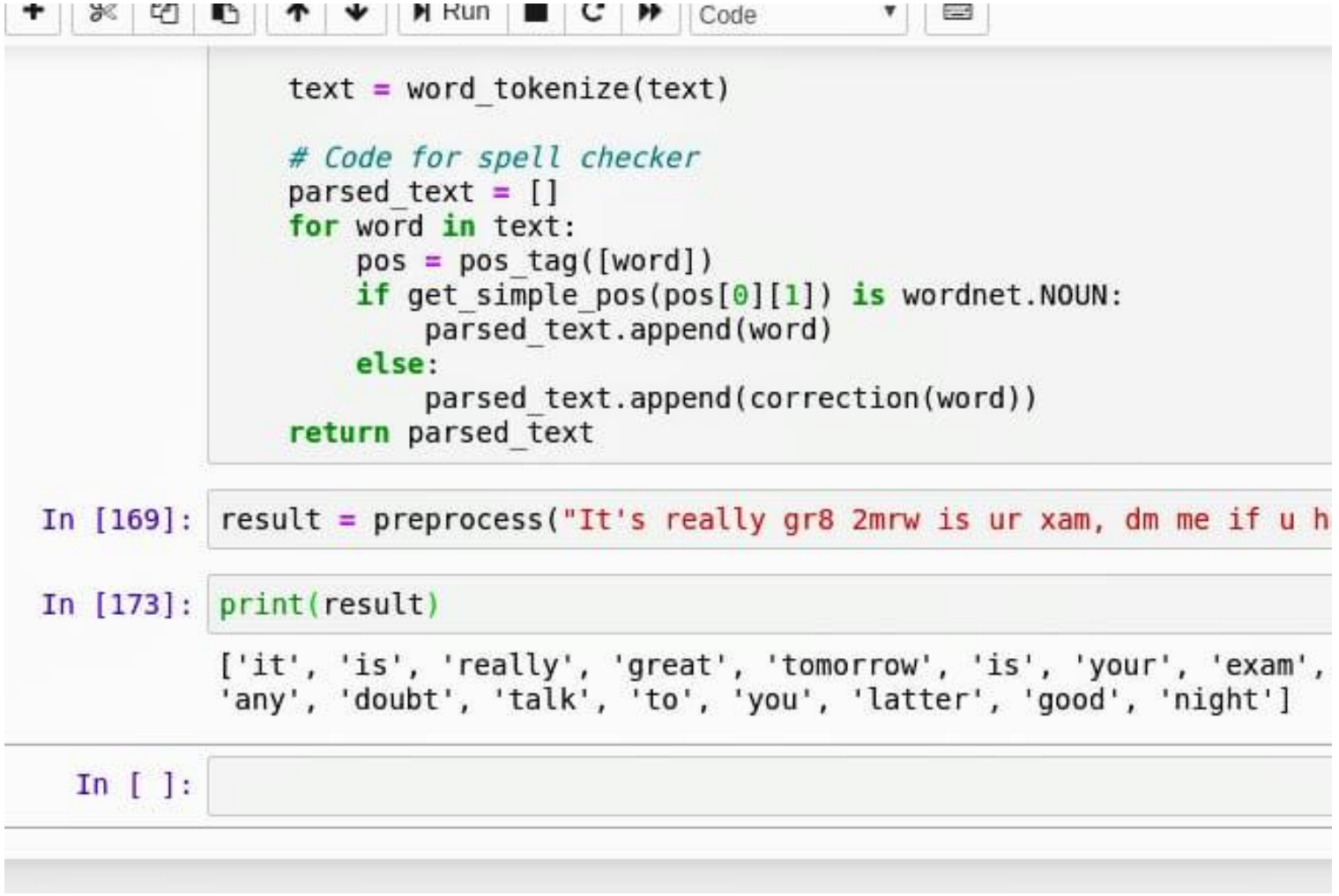
SOLUTION: The shell implementation is divided into three parts:

- Parser
- Executor
- Shell Subsystems

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".

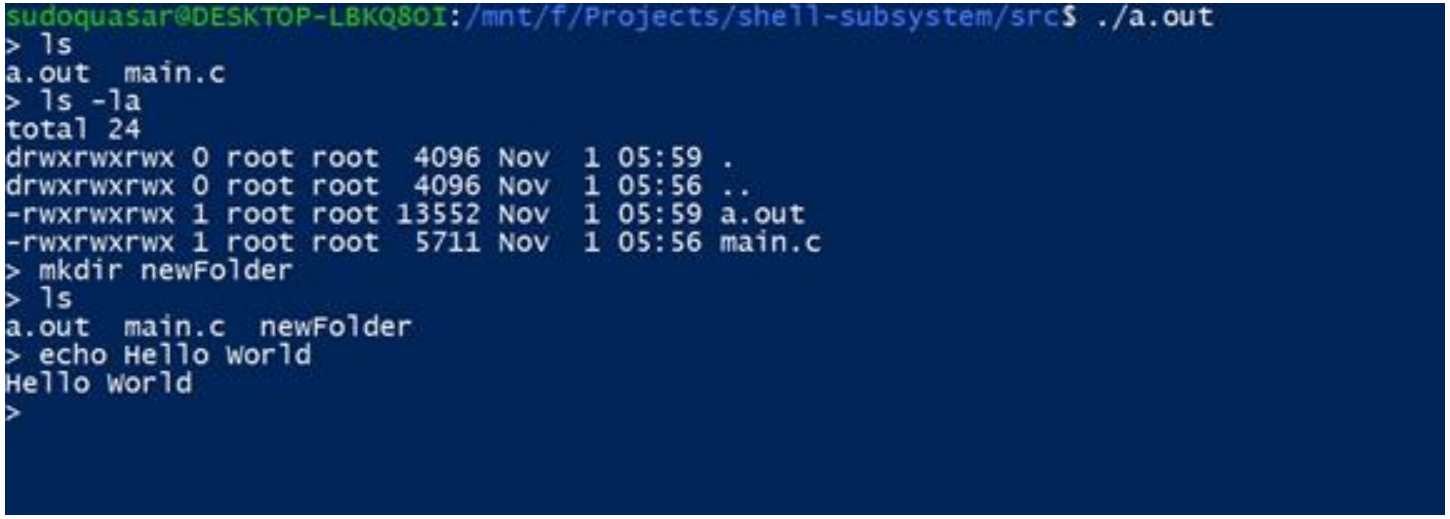
> With those simplifications, all we need to do is “tokenize” the string using whitespace as delimiters. That means we can break out the classic library function “strtok”

The Executor

The executor takes the command table generated by the parser and for every Simple Command 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.

The figure below shows a command line "A | B | C | D". If there is a redirection such as "< infile" detected by the parser, the input of the first Simple Command A is redirected from infile. If there is an output redirection such as "> outfile", it redirects the output of the last Simple Command (D) to outfile.

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.



Shell Subsystems

Other subsystems that complete your shell are:

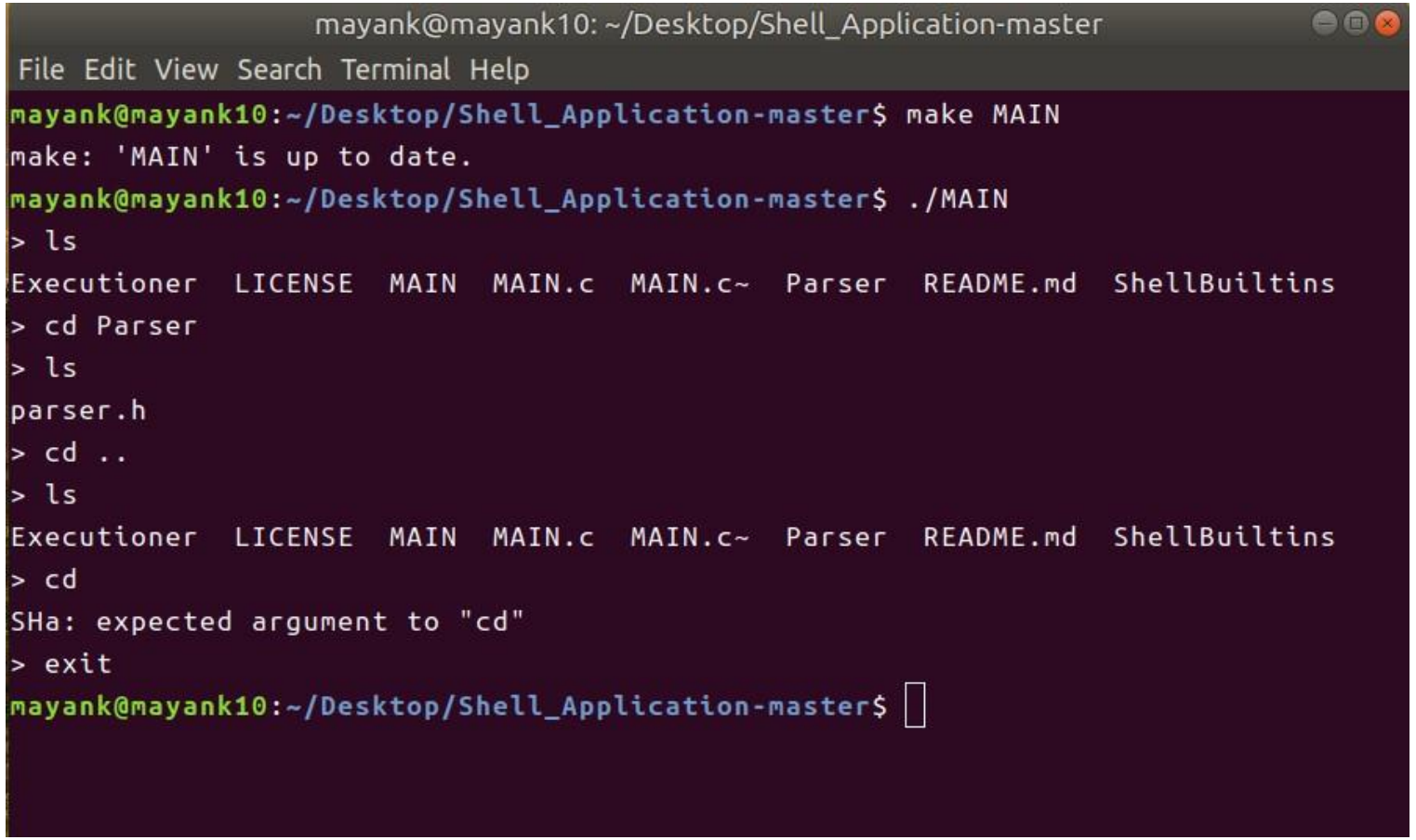
- Environment Variables: Expressions of the form ${VAR} are expanded with the corresponding environment variable. Also the shell should be able to set, expand and print environment vars.
- Wildcards: Arguments of the form a*a are expanded to all the files that match them in the local directory and in multiple directories .
- Subshells: Arguments between `` (backticks) are executed and the output is sent as input to the shell.

In this project we have implemented wildcards functionality

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.

Final Execution of Program



Project done by:.

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)