

Course Project

First Deliverable: The Lexer

Quin'darius Lyles-Woods

qlyleswo@students.kennesaw.edu

Concepts of Programming Languages

Professor Jose Garrido

Section W01

4308



Bachelors of Computer Science

Kennesaw State University

1100 South Marietta Pkwy SE

Marietta, GA 30060

September 27, 2021

Task

The development of an interpreter for a subset of **Basic Language**.

Project Goals

- Process a **Basic Language** source code file.
- Tokenize the source code file.
- Detect syntactical error.
- Display appropriate error messages during runtime.

Deliverable Goals

- Develop a grammar for the subset of **Basic Language** that I will be using.
- Arrays for the *Keywords* and *Identifiers*
- Ability to input source code to lexer.
- Output tokens scanned.

Subset of the Basic Language

The subset of basic that I want to define for the course project is going to be kept as minimal as possible to focus on the process of developing an interpreter. With that in mind the lowest we can go is turing complete of course. For this to be true the language doesn't need much.

	Tokens
• Recursive Operators	• LET
– FOR...TO...NEXT	• IF
• Conditional Jumps	• FOR
– IF...THEN...{ELSE}	• GOTO
– GOTO	• +
	• -
	• *
• Variables	• /
– LET	• =

Backus Normal Form of Basic Subset

BASIC PROGRAM ::=
 | *EXPRESSION*, *EXPRESSION*
 | *EXPRESSION*
EXPRESSION ::=
 | *LET*
 | *IF*
 | *FOR*
 | *GOTO*
 | *IDENTIFIER* + *EXPRESSION*
 | *IDENTIFIER* - *EXPRESSION*
 | *IDENTIFIER* * *EXPRESSION*
 | *IDENTIFIER* / *EXPRESSION*
LET ::=
 | *IDENTIFIER* = *EXPRESSION*
IDENTIFIER ::=
 | [*a* - *zA* - *Z*]
 | [*a* - *zA* - *Z*], [*a* - *zA* - *Z*]
IF ::=
 | *EXPRESSION* then *STATEMENT*
 | *EXPRESSION* then *STATEMENT* else *IF*
FOR ::=
 | *EXPRESSION* to *EXPRESSION* *STATEMENT* next
GOTO ::=
 | [0 - 9]
 | [0 - 9], [0 - 9]

Source Code

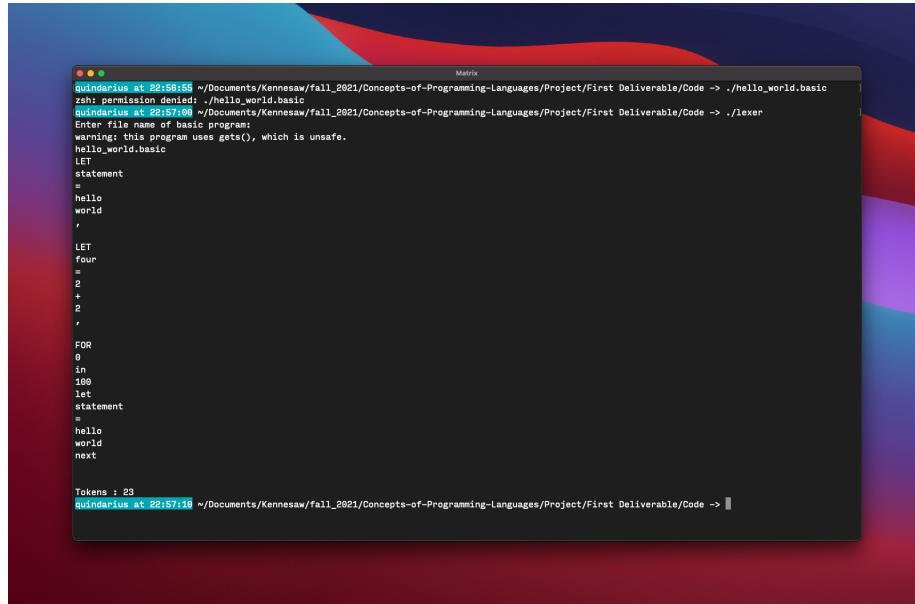
```
0 #include <stdio.h>
1
2 #define IN 1
3 #define OUT 0
4
5 int main()
6 {
7     char file_name[25];
8     FILE *fp;
9     printf("Enter file name of basic program: \n");
10    gets(file_name);
11    fp = fopen(file_name, "r");
12    if (fp == NULL)
13    {
14        perror("Error opening this file");
15    }
16    int character,           // Gets individual character input
17        tokens,             // Count of tokens
18        state;              // Determines if we are inside or outside a word
19    state = OUT;
20    tokens = 0;
21    while ((character = fgetc(fp)) != EOF) //Loops through the input
22    {
23        printf("%c",character);
24        // If statement to check if we are in a word
25        if(character == ' ' || character == '\n' || character == '\t')
26        {
27            printf("\n");
28            state = OUT;
29        }
30        // If statement to tell if we are starting or finishing a word
31        else if(state == OUT)
32        {
33            state = IN;
34            tokens++;
35        }
36    }
37    printf("\nTokens : %d\n",tokens);
38    fclose(fp);
39 }
```

lexer.c

Compiling

```
quindarius at 22:55:33 ~/Documents/Kennesaw/fall_2021/Concepts-of-Programming-Languages/Project/First Delivera
ble/Code -> cc lexer.c -o lexer
quindarius at 22:56:00 ~/Documents/Kennesaw/fall_2021/Concepts-of-Programming-Languages/Project/First Delivera
ble/Code -> 
```

Output

A terminal window titled 'Matrix' with a dark background and light-colored text. The prompt is 'quindarius at 22:56:55 ~/Documents/Kennesaw/fall_2021/Concepts-of-Programming-Languages/Project/First Deliverable/Code ->'. The user enters './hello_world.basic', which results in a 'zsh: permission denied: ./hello_world.basic' error. The user then enters './lexer', which prompts 'Enter file name of basic program:'. The user enters 'hello_world.basic', which results in a 'warning: this program uses gets(), which is unsafe.' message. The terminal then displays the output of the lexer, which is a list of tokens: 'LET', 'statement', '=', 'hello', 'world', ',', 'LET', 'four', '=', '2', '+', '2', '/', 'FOR', '0', 'in', '100', 'let', 'statement', '=', 'hello', 'world', 'next'. The prompt is now 'quindarius at 22:57:18 ~/Documents/Kennesaw/fall_2021/Concepts-of-Programming-Languages/Project/First Deliverable/Code ->'.

```
quindarius at 22:56:55 ~/Documents/Kennesaw/fall_2021/Concepts-of-Programming-Languages/Project/First Deliverable/Code -> ./hello_world.basic
zsh: permission denied: ./hello_world.basic
quindarius at 22:57:08 ~/Documents/Kennesaw/fall_2021/Concepts-of-Programming-Languages/Project/First Deliverable/Code -> ./lexer
Enter file name of basic program:
warning: this program uses gets(), which is unsafe.
hello_world.basic
LET
statement
=
hello
world
,
LET
four
=
2
+
2
/
FOR
0
in
100
let
statement
=
hello
world
next
Tokens : 23
quindarius at 22:57:18 ~/Documents/Kennesaw/fall_2021/Concepts-of-Programming-Languages/Project/First Deliverable/Code -> |
```

0.1 Summary

After finishing this project I have learned what it takes to build a simple lexer(Scanner) for a language. This feels very incomplete but I know that it is only on piece out of many when it comes to building an intepreter. I am glad I kept the subset super small since working by myself I could make the project a little more approachable and actually used the free mind space to make the lexer in C so I could use a very portable language that someday in the future might come in handy. I can see where this work will also help me in my Natural Language Processing class. stripping text of tokens is such a huge part of the field and I was glad to be able to do this in C because when working with large amounts of data you would probably want to use a fast language. C fills this requirement but now I need to do my part and learn how to optimize the program.