Space Complexity of C Compiler and Lex Tool

N.Abirami, S.Aishwarya, Z.Mahaboob Asfia, S.Suseela

***Abstract:** We can never stop stressing on the purpose and the importance of the compiler in the field of computer science. compiler does translates the high level programs like C,C++,java ,python and so on into low level language (machine language) which in turn computer processor use. Though the job of the compiler is to translate, depending on the properties of the programming languages the time complexity, space complexity and some of the other characteristics varies accordingly. Thus the purpose of the paper is concentrated on comparing such factors significantly the C compiler and the Lex tool. Our study reveals the best memory consumption among c compiler and lex tool.*

Keywords: Compiler, Computer Processor, Memory Consumption, Time Complexity.

I. INTRODUCTION

To create an executable program, it is must to get a target code from a source code using compiler [1]. In case of c language the code is written by a programmer as it is whereas in Lex, first the code is written in embedded c, then the lexical analyzer converts this into c code thereafter the compiler compiles it. Since the compiling process differs in terms of compiling it made sure that the comparison result will vary [2]. In this paper we will take the comparing parameters as Lines of code, Source file Memory, Line of code compiled, Backup file Memory, Executable file Memory. Lines of code refers to the no of coding lines present in the source program. Source file Memory is the file size of source code. Lines of code Compiled is actually the total no of lines the compiler has translated from the source code to get a target program.

Manuscript received on March 15, 2020.
Revised Manuscript received on March 24, 2020.
Manuscript published on March 30, 2020.

* Correspondence Author

N.Abirami*, UG Student, Department of Computer Science and Engineering in Periyar Maniammai Institute of Science & Technology (Deemed to be University), Thanjavur, Tamilnadu, India. Email: abinatsi@icloud.com

S.Aishwarya, UG Student, Department of Computer Science and Engineering in Periyar Maniammai Institute of Science & Technology (Deemed to be University), Thanjavur, Tamilnadu, India. Email: aishusankar66@gmail.com

Z.Mahaboob Asfia, UG Student, Department of Computer Science and Engineering in Periyar Maniammai Institute of Science & Technology (Deemed to be University), Thanjavur, Tamilnadu, India. Email: mahaboobasfia@gmail.com

S.Suseela, Assistant Professor, Department of Computer Science and Engineering in Periyar Maniammai Institute of Science & Technology (Deemed to be University), Thanjavur, Tamilnadu, India. Email: suseelas@pmu.edu

Backup file Memory is defined as when a program is executed then the compiler automatically generates a backup file which consumes some memory to be stored [8]. The file which is getting executed must be stored in the local system with its extension .exe this file memory is called executable file memory. Further the comparison between c compiler and lex tool is discussed in section II. The structure of lex tool and C compiler is briefly explains in section III and IV. Section V contains the Source code program involved for the analysis of this paper and thus the result is concluded in section VI.

II. COMPARISON BETWEEN C COMPILER AND LEX TOOL

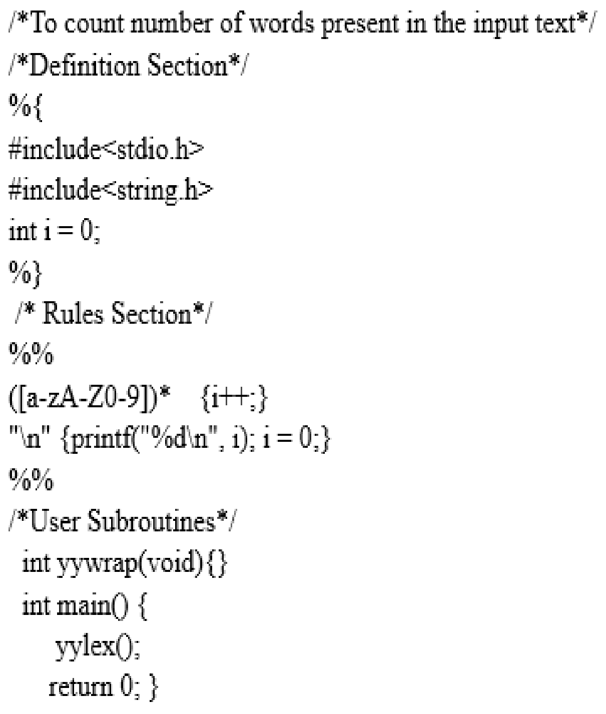
C COMPILER	LEX TOOL
C Compilers Compiles the C file as it is.	A lex first translates the lex file into a pure C file that only contains the C code [7].
It is Programming Language.	It is a Compiler.
It is an imperative procedural language [3].	It is a Computer program.
It is used for scripting system application.	It is used to generate scanners also known as tokenizer [5].
It helps to write coding for operating system and microcontrollers [4].	It helps to control flow of regular expression in the input stream.
It can recognize the Nested Structure.	It cannot recognize the Nested Structure.
C program is mostly used for developing applications.	Lex Program is mainly used to generate tokens.

III. STRUCTURE OF LEX PROGRAM

```
... definition section...

%%

...rules section...

%%

...user subroutines...
```

Fig.1

Example Program



```
/*To count number of words present in the input text*/
/*Definition Section*/
%{
#include<stdio.h>
#include<string.h>
int i = 0;
%}
 /* Rules Section*/
%%
([a-zA-Z0-9])*    {i++;}
"\n" {printf("%d\n", i); i = 0;}
%%
/*User Subroutines*/
  int yywrap(void){}
  int main() {
      yylex();
     return 0; }
```

Fig.2

The definition section can have *literal block*, *definitions*, *internal table declarations*, *start conditions*, and *translations*. The rules section can have c code and pattern lines. User subroutines are same as other programming languages, where we have main function [6].

IV. STRUCTURE OF C PROGRAM

A c program consists of following parts
Functions, Variables, Statements, Expressions, Pre-processor, commands, Comment lines [9].

A. SECTIONS IN C PROGRAM

There are six sections in c program

Documentation section consists of collection of comment lines. **Link section** contains instructions given to the compiler to connect with various functions.

Definition section defines constants. Every c program will have a **main function** and the main function will have two parts Declaration part and execution part. Main function starts with the curly brace and ends with a curly brace({}). The user defines functions comes inside main function.

Global Declaration Section contains global declaration of user variables. **Sub Program Section** is used to perform task, this section contains user defined sections to perform a task [9].

V. PROGRAM

Lex Program	C Program

%{
#include<stdio.h>
#include<string.h>
char line[100];
%}
%%
['\n']   { fprintf(yyout,"%s\n",line);}
(.*)   { strcpy(line,yytext); }
%%
int yywrap() {
   return 1;
}
int main() {
        extern FILE *yyin, *yyout;
        yyin=fopen("input.txt","r");
        yyout=fopen("output.txt","w");
        yylex();
}
	#include <stdio.h>
	#include <stdlib.h>
	int main()
	{
	   FILE *mp, *mp2;
	   char filename[90], c;
	   printf("Enter the filename to open \n");
	   scanf("%s", filename);
	   mp = fopen(filename, "r");
	   if (mp == NULL)
	   {
	      printf("Cannot open the file %s \n", filename);
	      exit(0);
	   }
	   printf("Enter the filename to Write \n");
	   scanf("%s", filename);
	   mp2 =

	fopen(filename, "w");
	   if (mp2 == NULL) {
	      printf("Cannot open the file %s \n", filename);
	      exit(0);
	   }
	   c = fgetc(mp);
	   while (c != EOF) {
	      fputc(c, fp2);
	      c = fgetc(mp);
	   }
	   printf("\nContents

	copied Successfully);
	   fclose(mp);
	   fclose(mp2);
	   return 0; }

VI. RESULT ANALYSIS

Features	Lex	C
LOC	18	31
Source File Memory	879 bytes	973 bytes
No of lines Compiled	18	31
Backup File Memory	30 bytes	38 bytes
Executable File Memory	17.790 bytes	14.312 bytes

When compared, the Lex and the C Program, we know that the Lex program has fewer Lines of Code and Consumes less memory than the C program. Thus the comparison shows that the Lex Program is adequate and satisfactory than the C program in terms of memory.

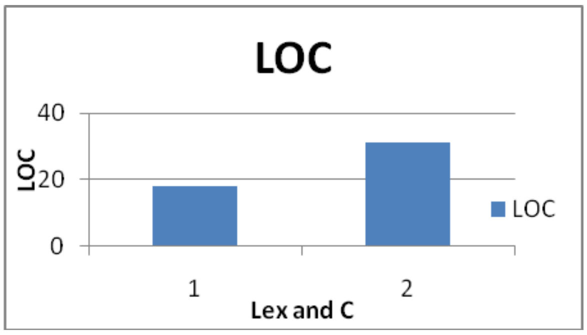


Fig. 3

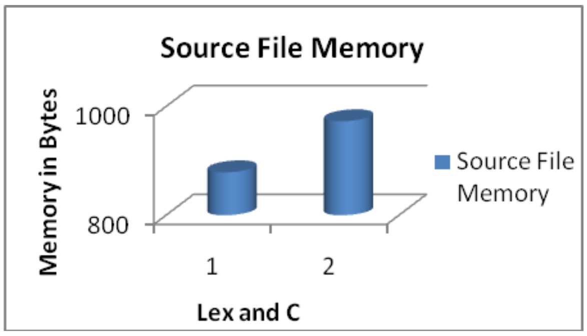


Fig. 4

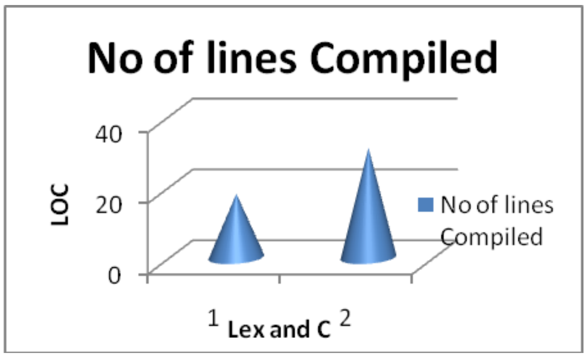


Fig. 5

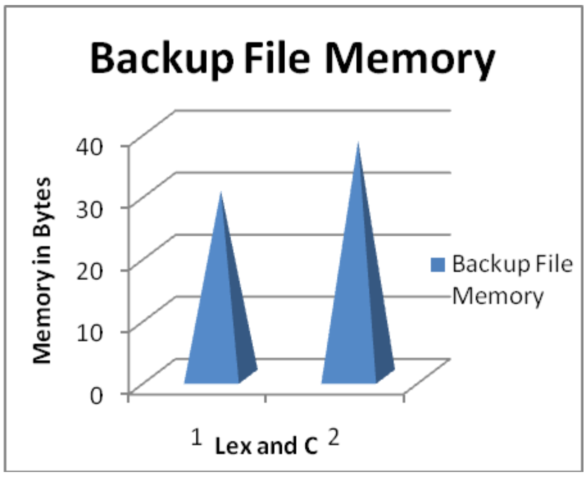


Fig. 6

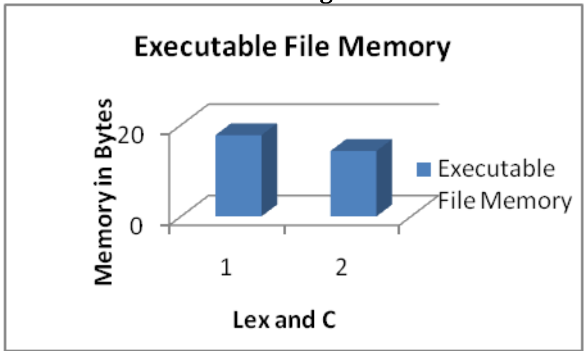


Fig. 7

VII. CONCLUSION

In conclusion, just as there are two sides to a coin there are two sides to every programming languages . It is too difficult to predict the best one than the other. We illustrate that the comparison is made, c compiler has 31 lines and lex tool has 18 Lines of code (Fig.3). The memory consumed by the source file (Fig.4) is 973 bytes by C compiler and 879 bytes by Lex tool. In Fig.5 the number of lines compiled by c is 31 lines and lex tool has compiled 18 lines. In Fig.6 the memory consumed by the backup file is 38 bytes by C compiler and 30 bytes by lex tool. In Fig.7 the memory consumed by the executable file is 14.312 bytes by C compiler and 17.790 bytes by lex tool. Our work substantiates the result obtained from this review will be helpful in the field of research and academic purposes.

ACKNOWLEDGMENT

It is our great pleasure to thank our Assistant Professor Ms S.Suseela to encouraging us to do this paper effectively. We would also like to thank our dear parents for their support and the encouragement.

REFERENCE

1. K. C. Louden, Compiler Construction: Principles and Practice, 1997.
2. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, "Compiler Principles Technique Tools" in , Reading, Massachusetts:Addison-Wesley, pp. 432-600.

3. F. Mulla, S. Nair and A. Chhabria, "Cross Platform C Compiler," *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, Pune, 2016, pp. 1-4.
4. M. Upadhyaya, "Simple calculator compiler using Lex and YACC," *2011 3rd International Conference on Electronics Computer Technology*, Kanyakumari, 2011, pp. 182-187.
5. Tom Neimann, "A Compact Guide To Lex & Yacc".
6. M.E. Lesk, E. Schmidt, "Lex_a lexical analyzer generator", *tech. rep. Bell Telephone Laboratories*, 1975.
7. S. Triantafyllis, M. Vachharajani, N. Vachharajani, D. I. August, "Compiler optimization-space exploration", *International Symposium on*, 2003.
8. J. Hartung, S. L. Gay and S. G. Haigh, "A practical C language compiler/optimizer for real-time implementations on a family of floating point DSPs," *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*, New York, NY, USA, 1988, pp. 1674-1677 vol.3.
9. N. Lossing, P. Guillou and F. Irigoin, "Effects Dependence Graph: A Key Data Concept for C Source-to-Source Compilers," *2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, Raleigh, NC, 2016, pp. 167-176.

AUTHORS PROFILE

N.Abirami, UG Student, Department of Computer Science and Engineering in Periyar Maniammai Institue of Science & Technology (Deemed to be University), Thanjavur, Tamilnadu, India.

S.Aishwarya, UG Student, Department of Computer Science and Engineering in Periyar Maniammai Institue of Science & Technology (Deemed to be University), Thanjavur, Tamilnadu, India.

Z.Mahaboob Asfia, UG Student, Department of Computer Science and Engineering in Periyar Maniammai Institue of Science & Technology (Deemed to be University), Thanjavur, Tamilnadu, India.

S.Suseela[CSI Life Membership No.00093481], working as a Assistant Professor in Department of Computer Science and Engineering in Periyar Maniammai Institute of Science & Technology, Thanjavur. More than 15 years of teaching experience and published4 articles in CSI Communication, more than 15 papers in national and international journals. Currently pursuing Ph.D. in National Institute of Technology Trichy in the Department of Computer Applications. Research interest is Multimedia Wireless Sensor Networks and Compiler Design, Theory of Computation. Email:suseelass@gmail.com