Laboratory Manual

For

Language Translator

(IT 608)

B.Tech (IT)

SEM VI

June 2010

Faculty of Technology
Dharmsinh Desai University
Nadiad.
www.ddu.ac.in

Table of Contents

Sample experiment 1

1 AIM: Scan a file and replace all the occurrences of the word "username" with users login name

2 TOOLS/APPARATUS: Flex tool in Linux

3 STANDARD PROCEDURES:

3.1 Analyzing the Problem:

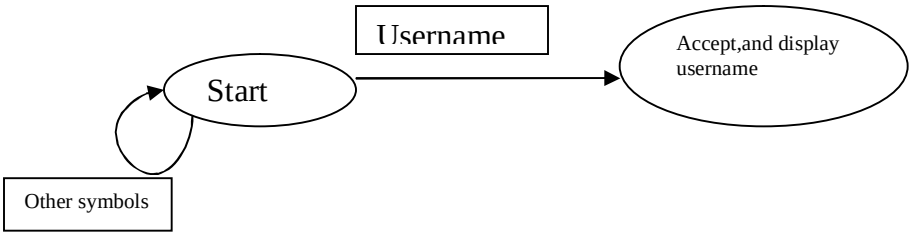
Analyze and find the token generation rules in language given.

Example : In given problem language, the valid token is the string“username”.

3.2 Designing the Solution:

- Draw the transition diagram for identification of lexemes making up tokens in given language.

  For example, for above problem transition diagram is as follows:-



- Identify the pattern matching rules (token generation rules) according to flex specifications.
- Find appropriate action to be taken on identifying a pattern in input data scanned. The action part should be written “C” langauge.

For example: In given example the pattern matching rule is “username” and corresponding action will have a printf statement to display name of current user logged in .

3.3 Implementing the Solution

3.3.1 Writing program for flex tool.

- Write program in any text editor like vi ,emacs etc ,available in linux.

Follow the indentation and spacing rules stated by tool while writing the input code. The major regions in input file to be given to “flex” are as follows:-

```
 ... definitions ...
%%
... rules ...
%%
... subroutines ...
```

For example : the code for above problem is

```
%{                                  //Definitions section
%}
%%
username     printf("%s",getlogin() );   //pattern and corresponding action rule
%%
main(int argc, char*argv[])
{
extern FILE *yyin;
yyin=fopen(argv[1],"r");
        //open the file provided on command line as input to  scanner s/w
generated
yylex();
      //execute the yylex routine where the required scanner logic is
implemented
printf("\n");
return 0;
}
```

- Now Save above program as myfile.l . This file is to be given to flex tool ,which would in turn interpret it and give as output a C program which has required logic to implement the required scanning task in stated language.

3.3.2 Compilation /Running and Debugging the Solution

On the linux command prompt give the following series of commands .

```
> flex myfile.l  // give the  specifications of target scanner needed to flex tool.

>  gcc –o scanner lex.yy.c –lfl   //compile the scanner software generated automatically
by flex

> ./scanner  <myfile.txt>  //execute the scanner software on the "myfile"
```

3.4 Testing the Solution

Suppose the "myfile.txt" is as follows:-

Current user is <username>

The output generated should be as follows:-

Current user is <user1> ***//assuming the current loged in user's name is user1***

4 Conclusions

Using flex tool , we can automatically generate a software to implement scanner phase of any given language.

Sample experiment 2

1 AIM: Generate a automated parser for simple calculator .Valid tokens are single digit operands ,'+', '*' '\n' . For valid expressions parsed in language, the corresponding evaluation result should be displayed as output .

2 TOOLS/APPARATUS: Yacc tool in linux

3 STANDARD PROCEDURES:

3.1 Analyzing the Problem:

Analyze and find CFG rules in language given.

Example : In given problem language as it is a calculator, the grammar rules are->

Lines-> Lines Line | ^

Line->Expr '\n'

Expr->Expr + Term | Term

Term->Term * Factor| Factor

Factor->number //here number is token indicating single digit operand

3.2 Designing the Solution:

- Find the syntax directed translation(SDT) scheme corresponding to the CFG rules found and the action required after successful parsing .

For example: In given example for the successful match of a valid term , action should be to pass the value computed for the valid factor to the term on LHS i.e

{ term.val= term.val*factor.val | factor.val}

- Find how to extract the attributes using yacc tool. For example here , if grammar specified to tool is "term->factor" and if we want to assign value of factor.val to term.val , we have to use "$$=$1" ,as per specifications of yacc.

3.3 Implementing the Solution

3.3.1 Writing program for yacc tool.

- Write program in any text editor like vi ,emacs etc ,available in linux.

Follow the indentation and spacing rules stated by tool while writing the input code.

The major regions in input file to be given to "yacc" are as follows:-

...definitions...

%%

...rules...

%%

...code...

For example : -the code for above problem is

```
%{                                  //definitions section
#include<stdio.h>
#include<stdlib.h>
%}
%token PLUS  MUL  NEWLINE           //tokens used
%token NUMBER
                                    /* grammar rules & actions section */
%%
```

```
lines : lines line
        |
        ;
line : expr NEWLINE { printf("%d\n> ", $1); }
        ;
expr : expr PLUS term { $$ = $1 + $3; }
        | term { $$ = $1; } /* default action */
        ;
term : term MUL factor { $$ = $1 * $3; }
         | factor { $$ = $1; }                    /* default action */
        ;
factor : NUMBER { $$ = $1; }                      /* default action */
        ;
%%
yylex() {                                  //lexer routine
int c;
do {
        c=getchar();
        switch (c)
                {
                case '0': case '1': case '2': case '3': case '4': case '5': case '6':
                case '7': case '8': case '9':
                        yylval= c –'0';
                        return NUMBER;
                case '+': return PLUS;
                case '\n': return NEWLINE;
                }
        } while (c!= EOF);
        return(EOF);
        }
main() {
yyparse();
}
```

- Now Save above program as calc.y . This file is to be given to yacc tool ,which would in turn interpret it and give as output a C program which has required logic to implement the required parsing and semantic tasks .

3.3.2 Compilation /Running and Debugging the Solution

On the linux command prompt give the following series of commands.

```
>yacc  calc.y –lm
> gcc –o calc y.tab.c –ly
```

3.4 Testing the solution

```
    > ./calc
    2+3   //input entered
     5     //response generated.
    ^q    //exits
>
```

4 Conclusions

Using yacc tool, we can automatically generate software to implement language processor of given language which implements all the tasks of a language processor till code generation.

Tools / Apparatus: yacc and flex tools in Linux, turbo- c or gcc compiler in linux

EXPERIMENT-1

AIM:- Study the translation process

PROCEDURE:-

- Write a simple program in C/C ++ language and observe compiling ,linking and loading process.
  Note-Use turboc ,gcc compilers
- Explain gcc –c ,gcc –S, gcc –static,size <exefilename> , objdump –hrt <obj file>
- Find information about -
  (a) Compilers, (b) Interpreters, (c) linkers (d) loaders.
- Understand the characteristics of these types of files ->

a.out , .Class , COM files,ELF .exe,.JAR ,.XPI ,.DLL, .WAR ,XBE ,XCOFF,.ASM,.C ,.CPP,. CC,. CXX ,.CS ,.h,.HPP, .l,.y

EXPERIMENT-2

AIM:- Study flex.

PROCEDURE:- i) Study flex manual and try to find answer for following questions-

- What is flex?
- What are characteristics of flex?
- What is format of flex?
- Explain the steps to use flex .
- Is following LEX input program valid? If yes, how is the resultant scanner?

%%

ii)Generate scanner which identifies token ‘username’ and displays the current logged in username in response

iii) Generate scanner using lex, which simply counts number of lines and number of characters in input and displays it when EOF encountered in input.

EXPERIMENT-3

AIM:- Using flex tool generate scanner for a language
PROCEDURE:-

- Generate scanner using lex , which can identify following language –"D" ,and display appropriate messages on recognizing them

Rules of language "D"→
a) Keywords—if,for,while,do,exit,else,case,switch,until
b)Identifiers-starting with alphabet(big or small) and followed by one or more alphabets or numbers
c) Numbers- (1) float number →having one '.' and valid formats are → . Num or Num.Num or Num . 0 (2) integer number—>simple number, starting with digit followed by zero or more digits
d) Single line comments starting with ***
e) White space characthers are - blanks and tabs
f) punctuations are –[,],{,},(,) and comma
g) operators :=,<>,= = ,!= , <=, <=, (relational),+,-,*,% (mathametical)
h) string constant in ' ' Note-no enclosing quotes allowed in langauge

- Draw the transition diagram for the given "D" language
- Give name of at least 2 lexical phase generator software's,other than lex
- Give the list of meta character conventions used in Lex . I.e patterns allowed and their associated meanings.

EXPERIMENT-4

AIM:- Implementation of scanner phase (w/o using any automated tools) .
PROCEDURE:-

- Write scanner algorithm for implementing the “D” language
- Generate scanner in “C” code for language “D” given above.

EXPERIMENT-5

AIM:- Study yacc tool

PROCEDURE:-i)Study the online manual of yacc and find answers for following questions-

- What is yacc? Find its characteristics?
- How yacc works?
- What is format of typical file in yacc?

ii)Write small programs to understand the working of yacc

EXPERIMENT-6

AIM: - Implement language processors, using yacc
PROCEDURE:-

- Find SDD(syntax directed definition) for desktop calculator
- Create LP (language processor) for desk top calculator, using yacc

EXPERIMENT-7

AIM:- WAP for Left recursion problem Removal
PRODEDURE:-

- Write algorithm to remove left recursion problem
- Generate a program to remove left recursion problem from any given grammar rules

EXPERIMENT-8

AIM-Left factoring problem Removal
PROCEDURE:-

- Write algorithm to remove left factoring problem
- Generate a program to remove left factoring problem from given grammar rules

EXPERIMENT-9

AIM-first and follow set computation needed for LR parser generation
PROCEDURE:-

- Write algorithm to generate first and follow sets
- Generate a program to find first and follow set of a given grammar.

EXPERIMENT-10

AIM- Implement predictive parser (RDP) .
PROCEDURE:-

- Write algorithm for Recursive dependent parser for a desktop calculator's language.
- Write a program to implement predictive parser for desktop calculator.

EXPERIMENT-11

- Generate a LR Parser for any given language . (note- do not use any tools) .The user can specify the language dynamically. If the language has left recursion /left factoring problem, it should be removed, the first –follow sets should be computed , appropriate parse table should be generated . Then for any string, the parser should be able to deterministically give the result ,indicating the string is valid or invalid in the given grammar.

EXPERIMENT- 12

- Generate a tool like flex. The tools should read specifications of tokens in some specific format and generate equivalent C –code to implement the required logic to generate the scanner software.

References

- The theory and practice of Compiler Writing by Jean Paul, Tremblay and Paul G. Sorenson
- Compilers: Principles, Techniques and Tools, By A. V. Aho, R. Sethi, and J. D. Ullman. Publisher Addison-Wesley.
- Compiler design in C by Allen Holup, Publisher-PHI
- Compiler Construction by Dhamdhere, Publisher- McMillan India