

# System Software and Operating Systems Laboratory

## Installation Procedure:

### 1. Sudo apt-get update

## To install Lex package:

Type the exact command (either of the two)

`sudo apt-get install flex`

OR

`sudo apt-get install flex-old`

Type 'y' when it asks for confirmation (see below image)



```
am@ubuntu:~$ sudo apt-get install flex
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libfl-dev libsigsegv2 n4
Suggested packages:
  bison build-essential
The following packages will be REMOVED:
  flex-old
The following NEW packages will be installed:
  flex libfl-dev libsigsegv2 n4
0 upgraded, 4 newly installed, 1 to remove and 541 not upgraded.
Need to get 422 kB of archives.
After this operation, 486 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

---

## To install Yacc package:

`sudo apt-get install bison`

Type 'y' when it asks for confirmation (see image below)

```

ann@ubuntu:~$ sudo apt-get install bison
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libbison-dev
Suggested packages:
  bison-doc
The following NEW packages will be installed:
  bison libbison-dev
0 upgraded, 2 newly installed, 0 to remove and 541 not upgraded.
Need to get 578 kB of archives.
After this operation, 1,798 kB of additional disk space will be used.
Do you want to continue? [Y/n] y

```

## SAMPLE PROGRAMS

### 1. lex code to count the number of a's in the input

```

%{
int c=0;          /* Global Variable */
%}
/* Rules Section */

%%
[a]      c++;
.        ;
%%

int main()
{
    yylex();
    printf("The no of a's in the given string %d\n",c);
}

```

#### Execution Steps:

Lex <lexfilename.l>

cc lex.yy.c -ll

./a.out

#### OUTPUT

aaabna (Press **Cntrl-D**)

The no of a's in the given string 4

## 2. lex code to count the number of lines, tabs and spaces used in the input

```
% {
#include<stdio.h>
int lc=0, sc=0, tc=0, ch=0; /*Global variables*/
% }

/*Rule Section*/
%%
\n lc++; //line counter
([ ])+ sc++; //space counter
\t tc++; //tab counter
. ch++; //characters counter
%%

int main()
{
    // The function that starts the analysis
    yylex();

    printf("\nNo. of lines=%d", lc);
    printf("\nNo. of spaces=%d", sc);
    printf("\nNo. of tabs=%d", tc);
    printf("\nNo. of other characters=%d\n", ch);

}
```

### **Execution Steps:**

Lex <lexfilename.l>

cc lex.yy.c -ll

./a.out

### **OUTPUT**

Good Morning  
No of lines = 1  
No of spaces = 1  
No of tabs = 0  
No of characters = 11

## 3. lex program to count number of words

```
% {
#include<stdio.h>
#include<string.h>
int i = 0;
```

```

% }

/* Rules Section*/
%%
([a-zA-Z0-9])* {i++;} /* Rule for counting number of words*/

"\n" {printf("%d\n", i); i = 0;}
%%

int yywrap(void){}

int main()
{
    // The function that starts the analysis
    yylex();

    return 0;
}

```

### **Execution Steps:**

Lex <lexfilename.l>

cc lex.yy.c -ll

./a.out

### **OUTPUT**

Hello Good Morning

## LAB PROGRAMS

**1 a. Write a LEX program to recognize valid arithmetic expression. Identifiers in the expression could be only integers and operators could be + and \*. Count the identifiers & operators present and print them separately.**

```
% {
#include<stdio.h>

int v=0,op=0,id=0,flag=0;

% }

%%

[a-zA-Z][0-9A-Za-z]* {id++;printf("\n Identifier:");ECHO;}

[+|\-|\*|\/=] {op++;printf("\n Operartor:");ECHO;}

"(" {v++;}

")" {v--;}

";" {flag=1;}

.\n {;}

%%

main()
{
printf("Enter the expression");
yylex();
if((op+1)==id && v==0 && flag==0)
{
printf("\n Expression is Valid\n");
printf("Number of identifiers:%d\n",id);
printf("Number of operators:%d\n",op);
}
else
printf("\n Expression is Invalid\n");
}
```

### **Execution Steps:**

Lex <lexfilename.l>

cc lex.yy.c -ll

./a.out

### **Output:**

Enter the expression

a+b Identifier: a

Operator: +

Identifier: b

Number of identifiers is: 2

Number of operators is: 1

Expression is valid

**b. Write YACC program to evaluate arithmetic expression involving operators: +, -, \*, and /**

### **LEX PART**

```
% {  
#include "y.tab.h"  
extern yylval;  
% }  
%%  
[0-9]+ { yylval=atoi(yytext); return num; }  
[+\-\\*\|] { return yytext[0]; }  
[(] { return yytext[0]; }  
[)] { return yytext[0]; }  
. { ; }  
\n { return 0; }  
%%
```

### **YACC PART**

```
% {  
#include <stdio.h>
```

```

#include<stdlib.h>

% }

%token num

%left '+' '-'
%left '*' '/'

%%

input:exp{printf("%d\n",$$);exit(0);}
exp:exp '+' exp{ $$=$1+$3;}
|exp '-' exp{ $$=$1-$3;}
|exp '*' exp{ $$=$1*$3;}
|exp '/' exp{if($3==0) {printf("divide by zero\n");
exit(0);}
else
$$=$1/$3;}
|('exp') { $$=$2;}
|num { $$=$1;};

%%

int yyerror()
{
printf("error\n");
exit(0);
}

int main()
{
printf("Enter a expression: ");
yyparse();
}

```

### **Execution Steps:**

Lex <lexfilename.l>

Yacc -d <yaccfilename.y>

```
cc lex.yy.c y.tab.c -ll  
./a.out
```

**Output:**

Enter an expression:

$(2+3)*5+9$

34