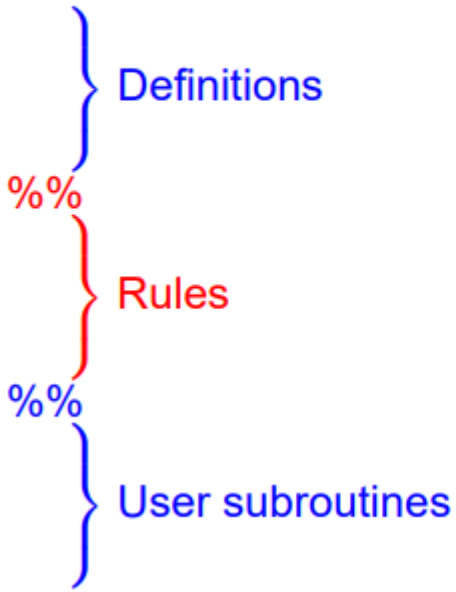
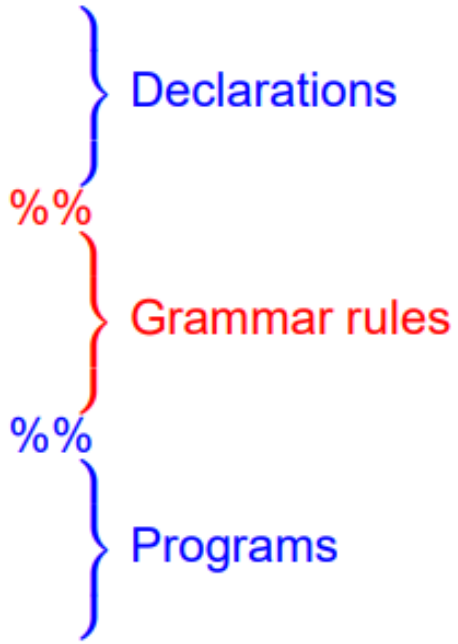
Structure of Lex Specification File



red : required
blue : optional

Yacc: A Parser Generator

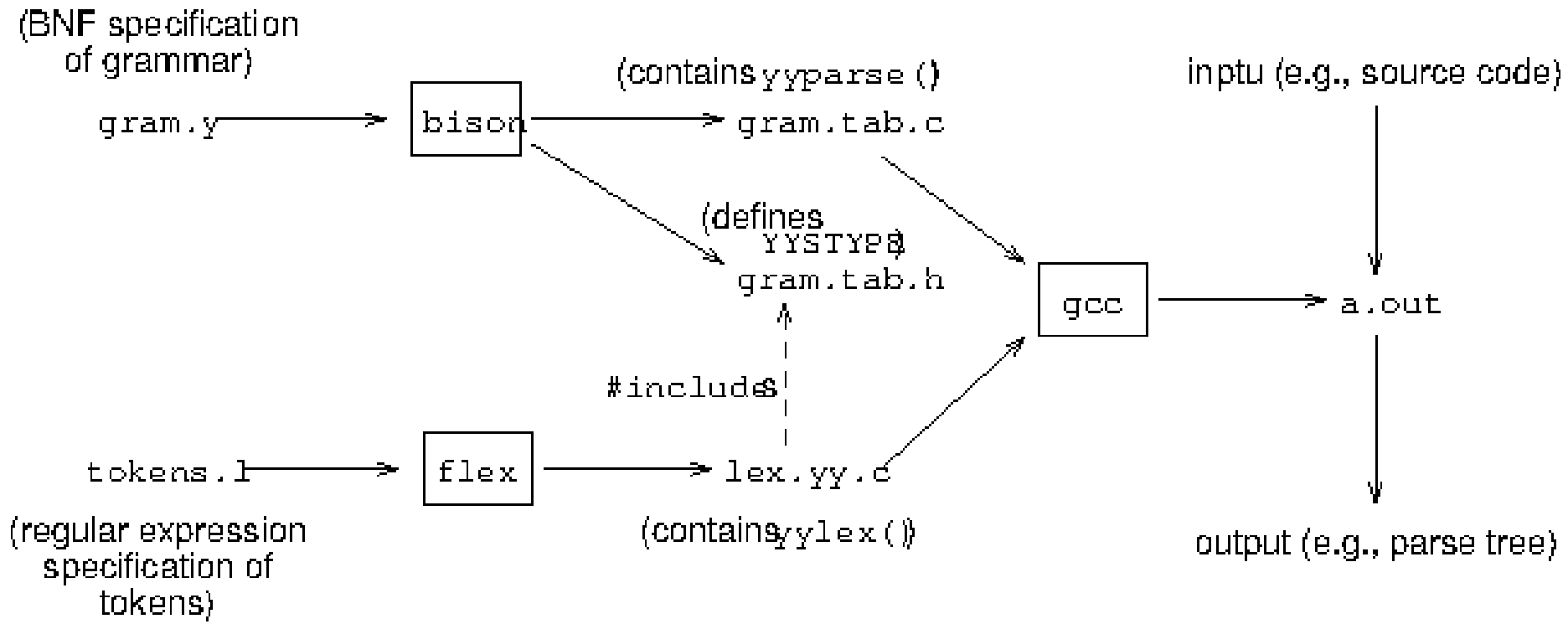


red : required
blue : optional

Yükleme Komutları	Derleme Komutları

```
sudo apt-get update
sudo apt-get install bison
sudo apt-get install flex
sudo apt-get install byacc
```

```
yacc -d gramer.y
flex kelime.l
gcc y.tab.c lex.yy.c -lfl
```


(BNF specification of grammar)
gram.y
bison
(contains yyparse())
gram.tab.c
inptu (e.g., source code)
(defines YYSTYPE)
gram.tab.h
gcc
a.out
#includes
tokens.l
flex
lex.yy.c
(contains yylex())
(regular expression specification of tokens)
output (e.g., parse tree)

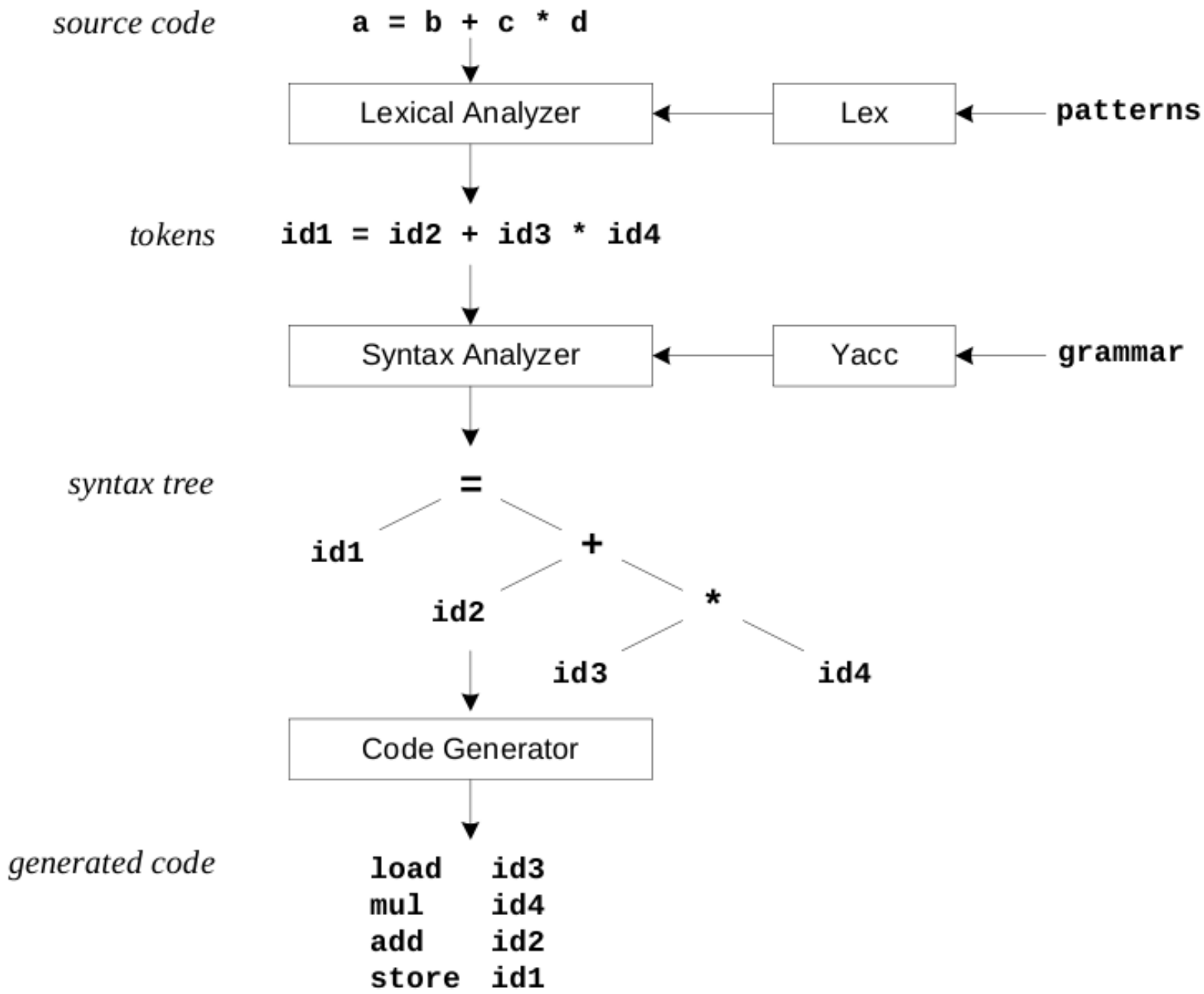


Figure 1: Compilation Sequence

metacharacter	description
+	previous expression can match one or more times
*	previous expression can match zero or more times
?	previous expression can match zero or one time
.	can match any character except the carriage return '\n'

Expression	*Matches*
`abc`	`abc`
`abc*`	`ab, abc, abcc, abccc, ...`
`abc+`	`abc, abcc, abccc, ...`
`a(bc)+`	`abc, abcbc, abcbcbc, ...`
`a(bc)?`	`a, abc`
`[abc]`	`a, b, c`
`[a-z]`	any letter, `a` through `z`
`[a\-z]`	`a, -, z`
`[-az]`	`-, a, z`
`[A-Za-z0-9]+`	one or more alphanumeric characters
`[ \t\n]+`	whitespace
`[^ab]`	anything except: `a, b`
`[a^b]`	`a, ^, b`
`[a|b]`	`a, |, b`
`a|b`	`a` or `b`

`a`	matches `a`
`abc`	matches `abc`
`[abc]`	matches `a`, `b` or `c`
`[a-f]`	matches `a`, `b`, `c`, `d`, `e`, or `f`
`[0-9]`	matches any digit
`X+`	mathces one or more of X
`X*`	mathces zero or more of X
`[0-9]+`	matches any integer
`(...)`	grouping an expression into a single unit
`|`	alternation (or)
`(a|b|c)*`	is equivalent to `[a-c]*`
`X?`	X is optional (0 or 1 occurrence)
`if(def)?`	matches `if` or `ifdef` (equivalent to `if|ifdef`)
`[A-Za-z]`	matches any alphabetical character
`.`	matches any character except newline character
`\.`	matches the . character
`\n`	matches the newline character
`\t`	matches the tab character
`\\`	matches the `\` character
`[ \t]`	matches either a space or tab character
`[^a-d]`	matches any character other than a,b,c and d

```
%{
    /* Definition section */
  #include "y.tab.h"
 %}

/* Rule Section */
 %%
[aA] {return A;}
[bB] {return B;}
\n {return NL;}
.    {return yytext[0];}
 %%

int ywwrap()
 {
   return 1;
 }
```

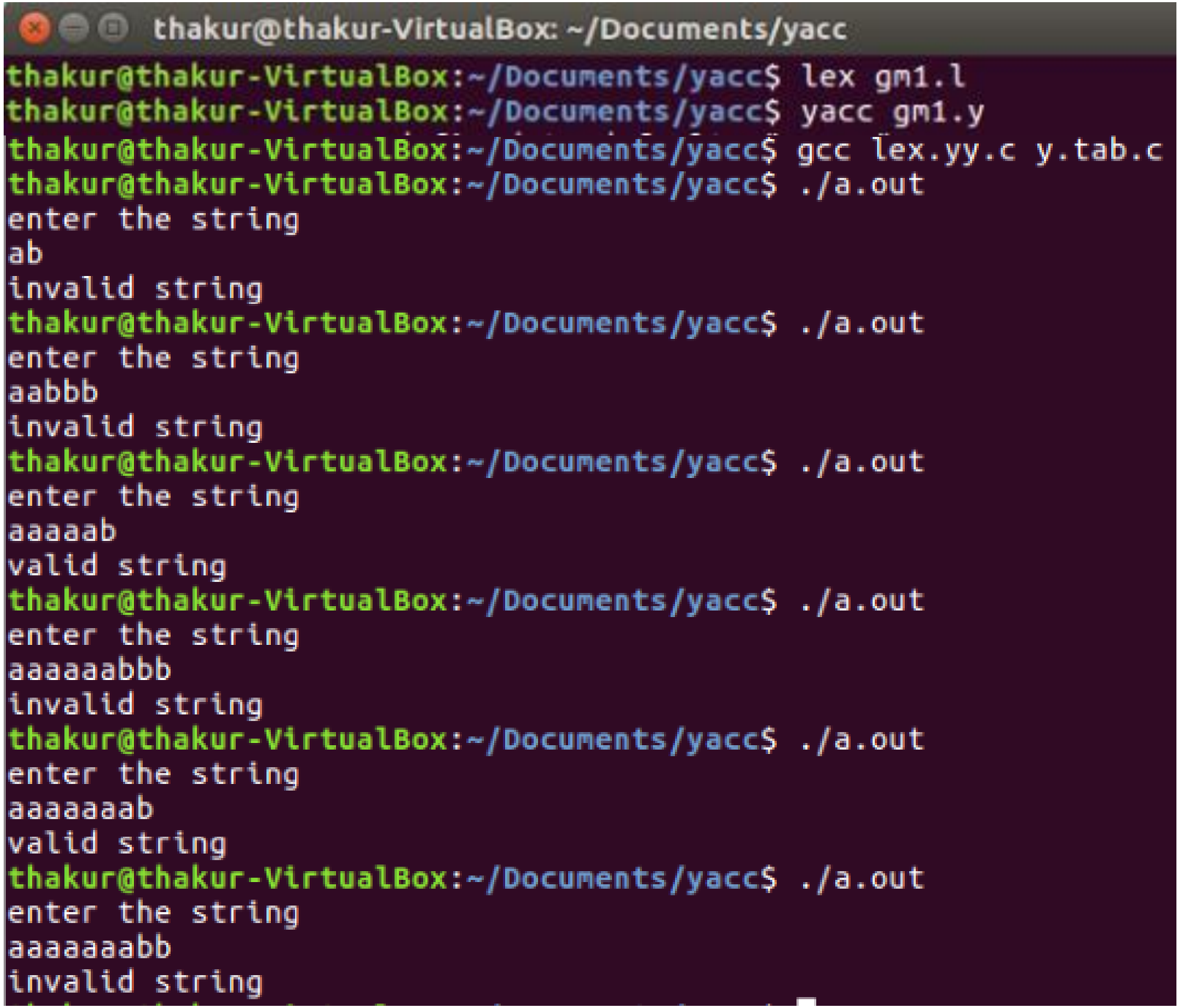
```
%{
    /* Definition section */
  #include<stdio.h>
  #include<stdlib.h>
 %}

%token A B NL

/* Rule Section */
%%
stmt: A A A A A S B NL {printf("valid string\n");
                exit(0);}
;
S: S A
|
;
%%

int yyerror(char *msg)
 {
  printf("invalid string\n");
  exit(0);
 }


//driver code
main()
 {
  printf("enter the string\n");
  yyparse();
 }
```


thakur@thakur-VirtualBox: ~/Documents/yacc
thakur@thakur-VirtualBox:~/Documents/yacc$ lex gm1.l
thakur@thakur-VirtualBox:~/Documents/yacc$ yacc gm1.y
thakur@thakur-VirtualBox:~/Documents/yacc$ gcc lex.yy.c y.tab.c
thakur@thakur-VirtualBox:~/Documents/yacc$ ./a.out
enter the string
ab
invalid string
thakur@thakur-VirtualBox:~/Documents/yacc$ ./a.out
enter the string
aabbb
invalid string
thakur@thakur-VirtualBox:~/Documents/yacc$ ./a.out
enter the string
aaaaab
valid string
thakur@thakur-VirtualBox:~/Documents/yacc$ ./a.out
enter the string
aaaaaabbb
invalid string
thakur@thakur-VirtualBox:~/Documents/yacc$ ./a.out
enter the string
aaaaaaab
valid string
thakur@thakur-VirtualBox:~/Documents/yacc$ ./a.out
enter the string
aaaaaaabb
invalid string