



Some Tools For Developing a Compiler

MOHAMMAD REZA BAHRAMI

DEPARTMENT OF COMPUTER ENGINEERING

Outline

- ▶ Scanner Developing Using Flex
- ▶ Developing Parser with PGen.
- ▶ Scripting and Project Development

چرا از ابزار استفاده می کنیم؟

◀ علی رغم اینکه می توانیم، خودمان کامپایلرها را توسعه دهیم از ابزار استفاده می کنیم...

◀ توسعه ساده تر می شود.

◀ احتمال بروز مشکلات ناشی از بی دقتی در توسعه کاهش می یابد.

◀ استفاده مجدد ساده تر می شود.

◀ سرعت توسعه افزایش می یابد.

◀ کد تغییرپذیری بالاتری خواهد داشت.

◀ و در ادامه خودتان متوجه خواهید شد که چرا!..... 😊

“

Flex, Lexical Analyzer Generator

”

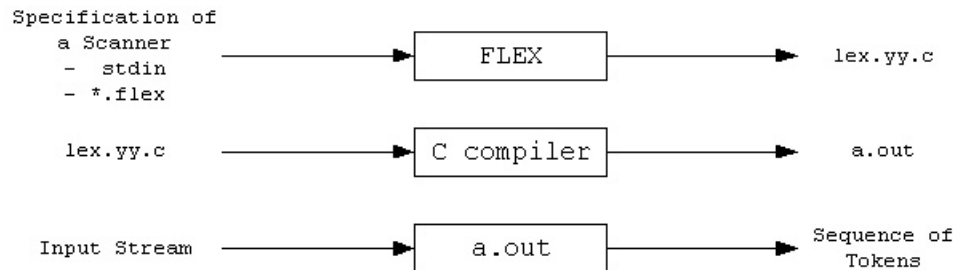
A TOOLS FOR DEVELOPING SCANNER, BASED ON LEX

Flex for Scanning!

- ◀ برای توسعه Scanner استفاده می شود.
- ◀ مبتنی بر Regular Expression ها است.
- ◀ دو پیاده سازی C و Java دارد.
- ◀ با ابزارهای دیگر می تواند تعامل کند.

How Does it Work?

- ◀ روند تبدیل یک برنامه flex به lexical Analyzer را در شکل زیر مشاهده می کنید.
- ◀ ابتدا کد توسط Flex به زبان C ترجمه شده، و سپس با استفاده از کامپایلر C کد اجرایی تولید می شود.



What is the Code Structure?

declarations

از نامش پیدا است!

%%

translation rules

%%

auxiliary functions

تعریف DFA یا همان
نوشتن عبارات منظم
دلخواه!

یک کد C که عیناً در خروجی کپی می‌شود

Let`s Work on an Example...

declarations

%%

translation rules

%%

auxiliary functions

کد C، مستقیماً داخل
برنامه نهایی کپی می‌شود.

```
1 %{\n2 #include <stdio.h>\n3 \n4 %}
```

مستقیم در ابتدای کد تولیدی تکرار می‌شود

```
6 %option yylineno\n7 %option noyywrap
```

تنظیماتی که امکاناتی را در اختیار ما می‌گذارند!

```
9 ID [a-zA-Z]+\n10 %%
```

تعریف RegEx ها، مثل متغیر

```
12 {ID} {printf("Become Happy!!! %s", yytext);}\n13 \n14 %%
```

قواعد گذار، بیانگر کارهایی است
که باید در حالت نهایی انجام
شوند. مثل گزارش کد به
Parser. با استفاده از این بخش
می‌توان کارهای زیادی را توسط
Scanner انجام داد

```
16 \n17 int main()\n18 {\n19     while(1)\n20     {\n21         yylex();\n22     }\n23     return 0;\n24 }\n25
```

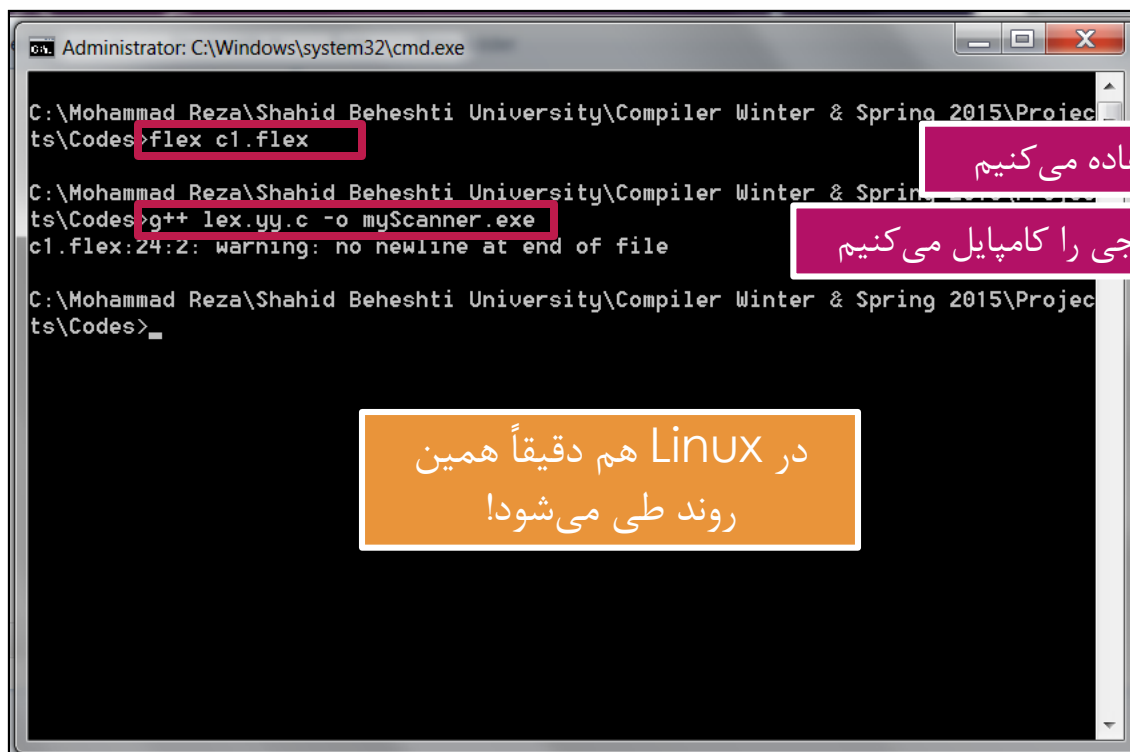

Example... Output

```
1498
1499 #ifdef YY_USE_PROTOS
1500 static void yy_flex_free( void *ptr )
1501 #else
1502 static void yy_flex_free( ptr )
1503 void *ptr;
1504 #endif
1505 {
1506     free( ptr );
1507 }
1508
1509 #if YY_MAIN
1510 int main()
1511 {
1512     yylex();
1513     return 0;
1514 }
1515 #endif
1516 #line 14 "c1.flex"
1517
1518 .....
1519 int main()
1520 {
1521     while (!feof(yyin))
1522     {
1523         yylex();
1524     }
1525     return 0;
1526 }
```

کد ما

```
lex.yy.c
550
551 if ( ! yy_current_buffer )
552     yy_current_buffer =
553     yy_create_buffer( yyin, YY_BUF_SIZE );
554
555 yy_load_buffer_state();
556
557 while ( 1 )      /* loops until end-of-file is reached */
558 {
559     yy_cp = yy_c_buf_p;
560
561     /* Support of yytext. */
562     *yy_cp = yy_hold_char;
563
564     /* yy_bp points to the position in yy_ch_buf of the start of
565      * the current run.
566      */
567     yy_bp = yy_cp;
568
569     yy_current_state = yy_start;
570     yy_state_ptr = yy_state_buf;
571     *yy_state_ptr++ = yy_current_state;
572
573     yy_match:
574     do
575     {
576         register YY_CHAR yy_c = yy_ec[YY_SC_TO_UI(*yy_cp)];
577         while ( yy_chk[yy_base[yy_current_state] + yy_c] != yy_current_state )
578         {
579             yy_current_state = (int) yy_def[yy_current_state];
580             if ( yy_current_state >= 7 )
581                 yy_C = yy_meta[(unsigned int) yy_c];
582             yy_current_state = yy_nxt[yy_base[yy_current_state] + (unsigned int) yy_c];
583             *yy_state_ptr++ = yy_current_state;
584             ++yy_cp;
585         }
586         while ( yy_base[yy_current_state] != 4 );
587     }
588     yy_find_action:
589     yy_current_state = *--yy_state_ptr;
590     yy_lp = yy_accept[yy_current_state];
591     find_rule: /* we branch to this label when backing up */
```

How to?



```
Administrator: C:\Windows\system32\cmd.exe

C:\Mohammad Reza\Shahid Beheshti University\Compiler Winter & Spring 2015\Projects\Codes>flex c1.flex

C:\Mohammad Reza\Shahid Beheshti University\Compiler Winter & Spring 2015\Projects\Codes>g++ lex.yy.c -o myScanner.exe
c1.flex:24:2: warning: no newline at end of file

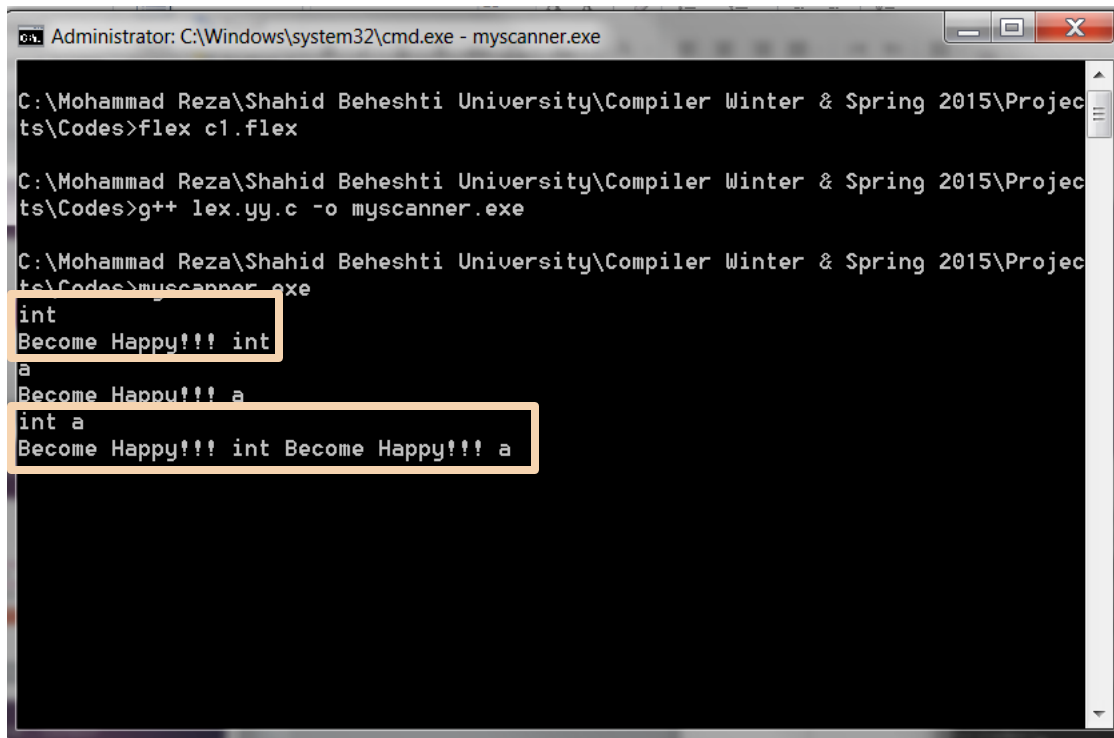
C:\Mohammad Reza\Shahid Beheshti University\Compiler Winter & Spring 2015\Projects\Codes>
```

۱. ابتدا از Flex استفاده می کنیم

۲. سپس خروجی را کامپایل می کنیم

در Linux هم دقیقاً همین
روند طی می شود!

Result!



```
Administrator: C:\Windows\system32\cmd.exe - myscanner.exe

C:\Mohammad Reza\Shahid Beheshti University\Compiler Winter & Spring 2015\Projects\Codes>flex c1.flex

C:\Mohammad Reza\Shahid Beheshti University\Compiler Winter & Spring 2015\Projects\Codes>g++ lex.yy.c -o myscanner.exe

C:\Mohammad Reza\Shahid Beheshti University\Compiler Winter & Spring 2015\Projects\Codes>myscanner.exe
int
Become Happy!!! int
a
Become Happy!!! a
int a
Become Happy!!! int Become Happy!!! a
```

A more Complex Code

```
1  %{
2  #include <stdio.h>
3  #include <iostream>
4  using namespace std;
5  #define IC_CODE 1
6
7  %}
8
9  %option yylineno
10 %option noyywrap
11
12 ID [a-zA-Z]+[_]
13 IC [0-9]+
14 %%
15 [ \t\n]
16 {ID} {printf("Become Happy!!! %s\n", yytext);}
17
18 {IC} {int tempIC = atoi(yytext);
19      printf("IC found: %d\n", tempIC);
20      return IC_CODE;
21      }
22
23 "=" {printf("Equal Found at %d\n", yylineno);}
24
25 in {cout << "In found!\n";}
26
27 <<EOF>> {return 0;}
28
29 %%
30
31 int main()
32 {
33     FILE * in = fopen("in.txt", "r+");
34     if(in == NULL)
35     {
36         printf("Cannot Open the File Specified!\n");
37         exit(EXIT_FAILURE);
38     }
39     yyin = in;
40     while(yylex());
41
42     return 0;
43 }
44
45
```



in.txt - Notepad

File Edit Format

```
id
id_
long_ 1234
=
in
```

```
C:\Mohammad Reza\Shahid Beheshti
ts\Codes>Scanner.exe
idBecome Happy!!! id_
Become Happy!!! long_
IC found: 1234
Equal Found at 4
In found!
```

Look Inside...(Declaration)

```
1  %{\n2  #include <stdio.h>\n3  #include <iostream>\n4  using namespace std;\n5  #define IC_CODE 1\n6\n7  %}\n8\n9  %option yylineno\n10 %option noyywrap\n11\n12 ID [a-zA-Z]+[_]\n13 IC [0-9]+\n14 %%\n15
```

کد قرار دادی بین Parser و Scanner

%option yylineno ، به صورت
اتوماتیک شماره خط را نیز محاسبه می کند.
%option noyywrap ، پیش تر برای
خواندن چندین Source Code
استفاده می شده، نیاز به لینک کتابخانه
دیگری دارد که چون نیاز نداریم، از خیرش
می گذریم!

Look Inside...(Transition)

Ignore Whitespaces

اگر الگوی IC را یافتی، کارهای زیر را انجام بده و در نهایت کد آن را گزارش کن!
{ } برای نشان دادن این است که Pattern تعریف شده است.

شماره خط در این متغیر و واژه یافت شده در yytext ذخیره

برای تشخیص انتهای فایل به صورت امن (راههای دیگری نیز هست!)

```
14  %%
15  [ \t\n]
16  {ID} {printf("Become Happy!!! %s\n", yytext);}
17
18  {IC} {int tempIC = atoi(yytext);
19        printf("IC found: %d\n",tempIC);
20        return IC_CODE;
21      }
22
23  "=" {printf("Equal Found at %d\n",yylineno)}
24
25  in {cout << "In found!\n";}
26
27  <<EOF>> {return 0;}
28
29  %%
30
```

Look Inside... (Auxiliary Functions)

```
30
31 int main()
32 {
33     FILE * in = fopen("in.txt","r+");
34     if(in == NULL)
35     {
36         printf("Cannot Open the File Specified!\n");
37         exit(EXIT_FAILURE);
38     }
39     yyin = in;
40     while(yylex());
41
42     return 0;
43 }
44
45
```

yyin ورودی Flex است که به صورت پیش فرض بر روی stdin تنظیم شده است.

Some Regular Expression Rules in Flex

Character	Meaning
.	هر کاراکتری غیر از endlime
[]	هر کاراکتری که در آن واقع شود
- in [] e.g. [a-z]	بیانگر range است.
*	صفر یا بیشتر
+	یک یا بیشتر
	یا
\ e.g. \. Means dot	برای استفاده از metachar
^	اگر داخل [] بیاید به معنای عدم آمدن کاراکترهای داخل Brace است.
?	صفر یا بیشتر از عبارت قبلی

All Rules

EXPRESSION	MATCHES	EXAMPLE
<i>c</i>	the one non-operator character <i>c</i>	<i>a</i>
<i>\c</i>	character <i>c</i> literally	<i>*</i>
<i>"s"</i>	string <i>s</i> literally	<i>"**"</i>
<i>.</i>	any character but newline	<i>a.*b</i>
<i>^</i>	beginning of a line	<i>^abc</i>
<i>\$</i>	end of a line	<i>abc\$</i>
<i>[s]</i>	any one of the characters in string <i>s</i>	<i>[abc]</i>
<i>[^s]</i>	any one character not in string <i>s</i>	<i>[^abc]</i>
<i>r*</i>	zero or more strings matching <i>r</i>	<i>a*</i>
<i>r+</i>	one or more strings matching <i>r</i>	<i>a+</i>
<i>r?</i>	zero or one <i>r</i>	<i>a?</i>
<i>r{m,n}</i>	between <i>m</i> and <i>n</i> occurrences of <i>r</i>	<i>a[1,5]</i>
<i>r₁r₂</i>	an <i>r₁</i> followed by an <i>r₂</i>	<i>ab</i>
<i>r₁ r₂</i>	an <i>r₁</i> or an <i>r₂</i>	<i>a b</i>
<i>(r)</i>	same as <i>r</i>	<i>(a b)</i>
<i>r₁/r₂</i>	<i>r₁</i> when followed by <i>r₂</i>	<i>abc/123</i>

Figure 3.8: Lex regular expressions

Examples

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

Table 2: Pattern Matching Examples

Predefined Variables in Flex

Name	Function
<code>int yylex(void)</code>	call to invoke lexer, returns token
<code>char *yytext</code>	pointer to matched string
<code>yyldeng</code>	length of matched string
<code>yyldval</code>	value associated with token
<code>int yywrap(void)</code>	wrapup, return 1 if done, 0 if not done
<code>FILE *yyout</code>	output file
<code>FILE *yyin</code>	input file
<code>INITIAL</code>	initial start condition
<code>BEGIN</code>	condition switch start condition
<code>ECHO</code>	write matched string

Another Example

```
digit    [0-9]
letter   [A-Za-z]
%{
    int count;
}%
%%
    /* match identifier */
    {letter} ({letter}|{digit})*      count++;
%%
int main(void) {
    yylex();
    printf("number of identifiers = %d\n", count);
    return 0;
}
```

Check it Yourself!

Ambiguity in Flex`s Pattern

◀ در صورتی که ابهامی در عبارات وجود داشته باشد، Flex، به صورت زیر ابهام را برطرف می کند:

◀ همواره طولانی ترین پیشوند را در نظر می گیرد.

◀ اگر با شرط قبل ابهام حل نشد. عبارتی که زودتر در لیست Action ها آمده در نظر گرفته می شود.

Read More On Flex (Bibliography)

- ▶ Levine John R., Flex & Bison, O`Reilly 2009
- ▶ Compilers, Principles, Techniques & Tools; A.V.Aho, M.S.Lam, R.Sethi, J.D.Ullman, 2nd Edition
- ▶ Lex and YACC Tutorial by Tom Niemann
- ▶ <http://dinosaur.compilertools.net/flex/manpage.html>
- ▶ <http://dinosaur.compilertools.net/lex/index.html>
- ▶ http://aquamentus.com/tut_lexyacc.html

+Start States

در برخی از این منابع، در مورد ساخت Parser نیز صحبت شده، توصیه می‌شود مطالعه آن
مباحث را تا تدریس این مطالب سر کلاس به تعویق بیندازید.



PGen: Parser Table Generator



BASED ON SYNTAX
GRAPH

DESIGNED BY
DR.JABERIPOUR

Why PGen?

- ◀ توصیف دستور زبان ساده و قابل فهم است (حتی برای کسانی که درس کامپایلر را نگذرانده‌اند).
- ◀ می‌توان بر بخش‌های دیگر کامپایلر تمرکز کرد.
- ◀ جدول پارس را خروجی می‌دهد و می‌توان به صورت دلخواه، عملیات Parse را انجام داد و بهینه‌سازی‌های دلخواه را انجام داد، به این ترتیب عملیات ساختاریابی Customizable خواهد بود.
- ◀ بیان گرامر با آن به شدت ساده است.

How to Use it?

گام اول: کشیدن نمودار

گام دوم: تولید جدول Parse

PGen - syngraph-new.pg

File View Run Help

Graph Name: MAI

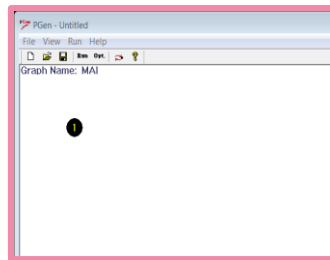
Parse Table Show

Legend:
S: Shift G: Goto PG: PushGoto R: Return A: Accept E: Error

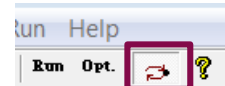
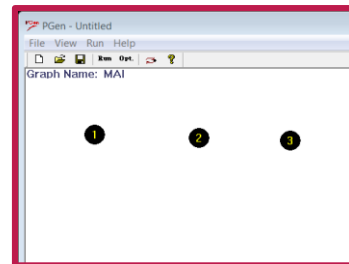
	!=	\$	()	*	+	-	.	/	:
0	E	E	E	E	E	E	E	E	E	E
1	E	E	E	E	E	E	E	E	E	E
2	E	E	E	E	E	E	E	E	E	E
3	E	E	E	E	E	E	E	E	E	E
4	E	E	E	E	E	E	E	E	E	S
5	E	E	E	E	E	E	E	E	E	E
6	E	E	E	E	E	E	E	E	E	E
7	E	E	E	E	E	E	E	E	E	S
8	E	E	E	E	E	E	E	E	E	E
9	E	E	E	E	E	E	E	E	E	E
10	E	E	E	E	E	E	E	S11	E	E

Save Ok

Draw a Graph



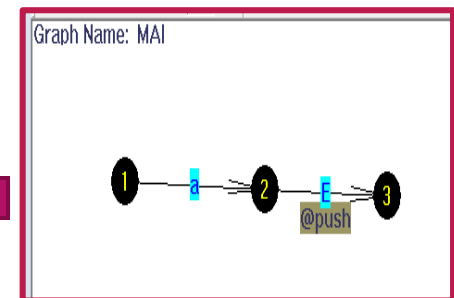
Left-Click



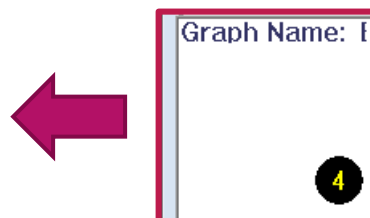
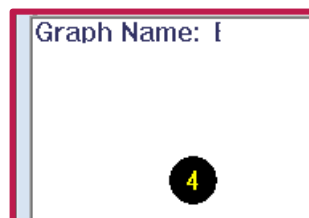
دو Node را به
هم متصل می کنیم



A screenshot of the 'Edge Properties ...' dialog box. It has a title bar with a close button. Inside, there are two input fields labeled 'Input' and 'Action'. To the right of each field are 'OK' and 'Cancel' buttons. At the bottom, there is a section labeled 'Edge Type:' with two radio buttons: 'Graph' and 'Token'. The 'Token' radio button is selected.



Right-Click



Some Points

- ◀ برای نهایی کردن یک Node روی آن کلیک راست کرده و گزینه Final را انتخاب کنید.
- ◀ برای اصلاح یک Node در حالت Transition روی آن کلیک راست کنید.
- ◀ به همراه نرم افزار یک شبه کامپایلر داده شده که معماری برنامه را پیشنهاد می دهد.
- ◀ یک برنامه نیز برای خواندن جدول پارس داده شده است.
- ◀ چند مثال و یک فایل Help نیز با نصب نرم افزار بر روی سیستم شما نصب می شوند آن ها را مطالعه کنید.
- ◀ برای استفاده جدول پارس، پس از تولید آن را ذخیره کنید.

Final Parse Table

Parse Table Show

Legend: S: Shift G: Goto PG: PushGoto R: Return A: Accept E: Error

	[]	and	begin	bid	bvalue	case	const	declaratio	dc	▲
60	E	E	E	E	PG13	E	E	E	E	E	
61	E	E	E	E	E	E	E	E	E	E	
62	R52	R52	R52	R52	R52	R52	R52	R52	R52	R!	
63	S64	E	E	E	E	E	E	E	E	E	
64	E	E	E	E	E	E	E	E	E	E	
65	E	E	E	E	E	E	E	E	E	E	
66	E	S67 @Fix	E	E	E	E	E	E	E	E	
67	E	E	E	E	E	E	E	E	E	E	
68	E	E	E	E	E	E	E	E	E	E	
69	R53	R53	R53	R53	R53	R53	R53				
70	E	E	E	E	E	E	E				

61	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem	0
62	4	52	NoSem	4	52	NoSem	4	52	NoSem	4	52	NoSem	4
63	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem	3	13	NoSem	C
64	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem	C
65	4	52	NoSem	4	52	NoSem	4	52	NoSem	4	52	NoSem	4
66	n	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem
67	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem	C
68	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem	C
69	1	67	@FixdigitArray1ASM	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem	C
70	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem	0	-1	NoSem	C

Action

Destination
Node

Semantic
Rule

Attention!



- ◀ مرحله به مرحله از کار خود Backup بگیرید! به یک نسخه هم اکتفا نکنید!
- ◀ این نرم افزار باگ های ناشناخته ای دارد؛ پس از اینکه بخشی را اضافه کردید، از حضور آن بخش مطمئن شده و بعد به مرحله بعد بروید!
- ◀ PGen امکان Undo (Ctrl+Z) ندارد! مراقب باشید!
- ◀ هرگز Node ابتدایی و انتهایی را یکی در نظر نگیرید (دلیل این موضوع را به خاطر ندارم!)
- ◀ یک باگ متداول (!): دیده شده، اولین فایل npt تولید شده (خروجی می باشد) که باید عددی برابر $L-2$ که L تعداد خطوط جدول (با احتساب خط اول) می باشد، برابر این عدد نمی باشد (!)، ابتدا آن را اصلاح کرده و سپس ذخیره کنید.

از توسعه نسخه سوم این نرم افزار، شدیداً استقبال می شود!



Easier
Development



BATCH FILE

How to Make Development Easier?

◀ همان طور که واضح است(!) نوشتن این همه دستور برای کامپایل کردن Scanner و یا Parser سخت است!

◀ همچنین باید سازوکاری برای تحویل یکسان پروژه‌ها در نظر گرفت! به طوری که بتوان اجازه استفاده از ابزار متفاوت را نیز داد!



How to Make a Batch File in Windows?

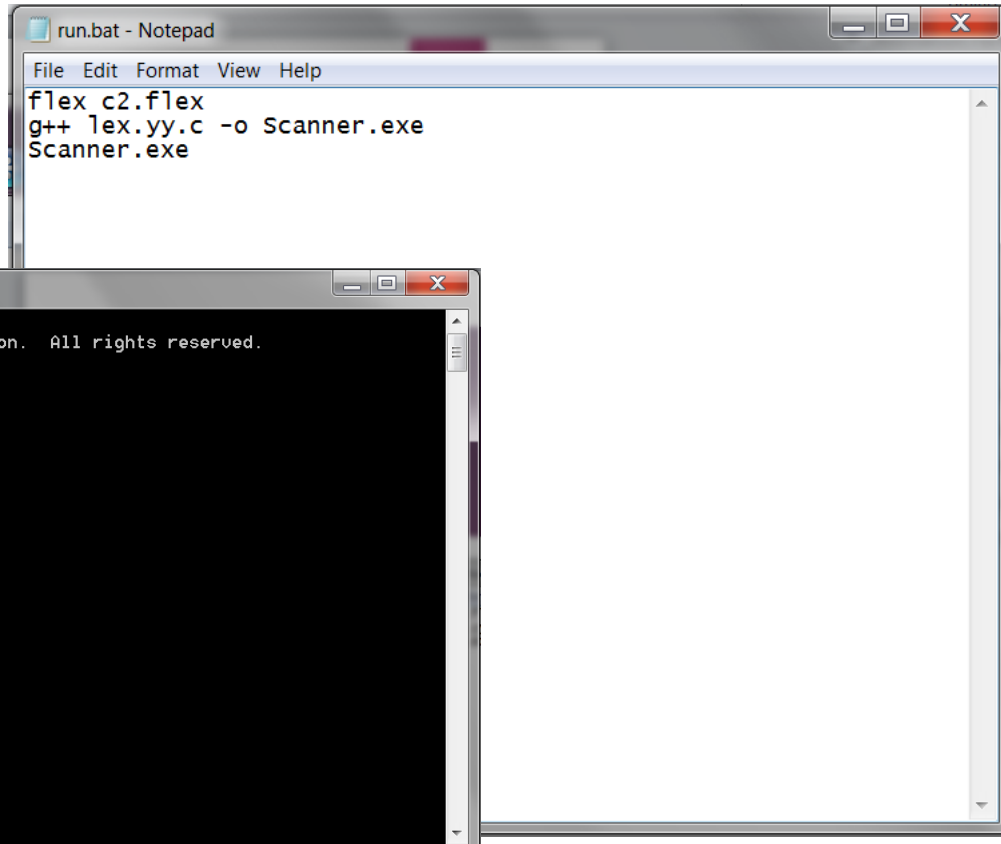
- ◀ یک فایل متنی باز کنید.
- ◀ هر دستوری که قصد داشتید در Command-line بزنید داخل این فایل بنویسید.
- ◀ فایل خود را ذخیره کنید.
- ◀ پسوند آن را به bat. تغییر دهید.
- ◀ فایل اجرایی شما آمده می باشد!

البته این فایل ها موارد استفاده و به طبع امکانات بیشتری دارند؛ اما تا همین جا، نیاز ما را رفع می کند.

Example



run.bat



The image shows a Windows desktop environment. In the background, a Notepad window titled "run.bat - Notepad" is open, displaying the following text:

```
File Edit Format View Help
flex c2.flex
g++ lex.yy.c -o Scanner.exe
Scanner.exe
```

In the foreground, a Command Prompt window titled "Administrator: C:\Windows\system32\cmd.exe" is open. It displays the following text:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Mohammad Reza>run.bat
```

Linux Solution: Shell Script

```
touch run.sh  
vim run.sh
```

```
#!/bin/sh  
flex c1.flex  
g++ lex.yy.c -o myScanner  
./myScanner
```

```
chmod 755 run.sh
```

Make it
Executable

```
./run
```

