

8. LEX 실습

A Lexical Analyzer Generator

충북대학교

이재성



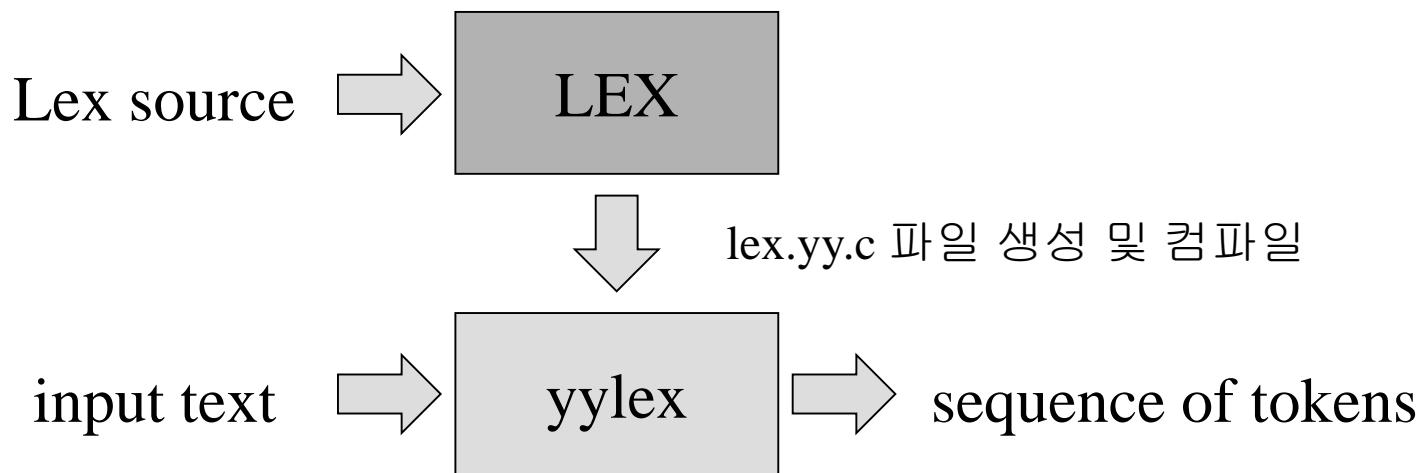


소개

■ Lex

- 입력된 정규표현을 프로그램으로 변환

■ Lex의 역할





Lex 소스 형식 (*.l)

■ 형식:

definitions	// 정의
%%	
rules	// 규칙
%%	// 이하 생략 가능
user routines	// 사용자 부프로그램



Lex 소스 예제(ex.l)

```
% {  
#define IF 100  
#define ID 101  
#define RELOP 102  
#define LE 201  
int yyval, tok;  
%}  
delim [ \t\n]  
ws {delim}+  
letter [A-Za-z]  
digit [0-9]  
id {letter}({letter}|{digit})*  
%%  
{ws} /* do nothing */  
if {yyval = 0; return(IF); }  
{id} {yyval = install_id(); return(ID); }  
"<=" {yyval= LE; return(RELOP); }  
%%  
int install_id() { return(1); }  
int yywrap() {return(0); }  
main() { while(1){ tok = yylex(); printf("%d %d\n", tok, yyval); }}
```



Flex bison download

■ Download 위치

- <https://sourceforge.net/projects/winflexbison/>
- ecampus 프로젝트 [과제]에서 사용될 도구 다운로드

■ 주요 스위치

- win_flex

- wincompat: 윈도우용 컴파일 호환 스위치

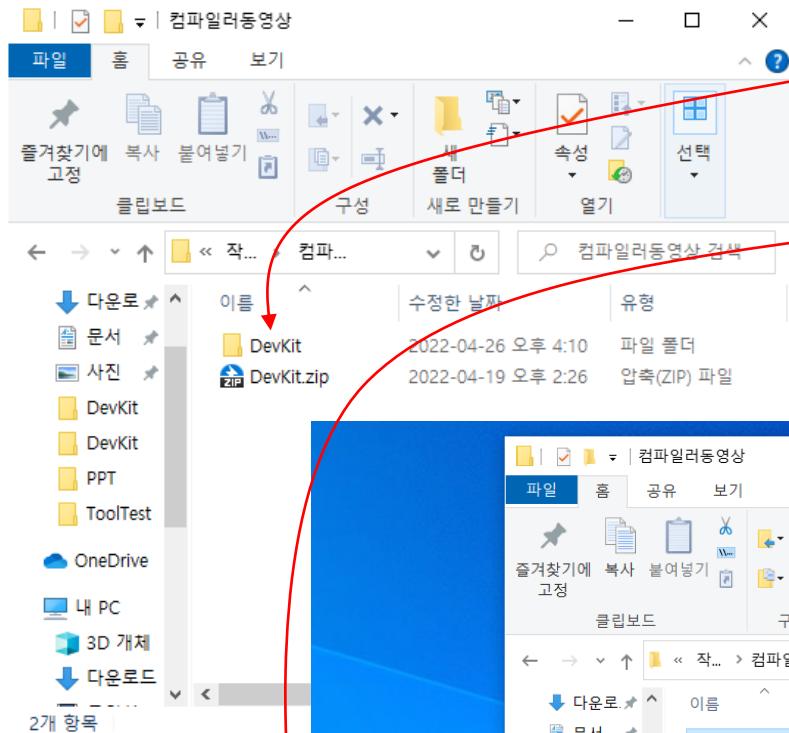
- o : 출력파일 이름 지정

- win_bison

- d : *.tab.h 파일 생성 -> lex 소스와 호환



명령어 창 열기



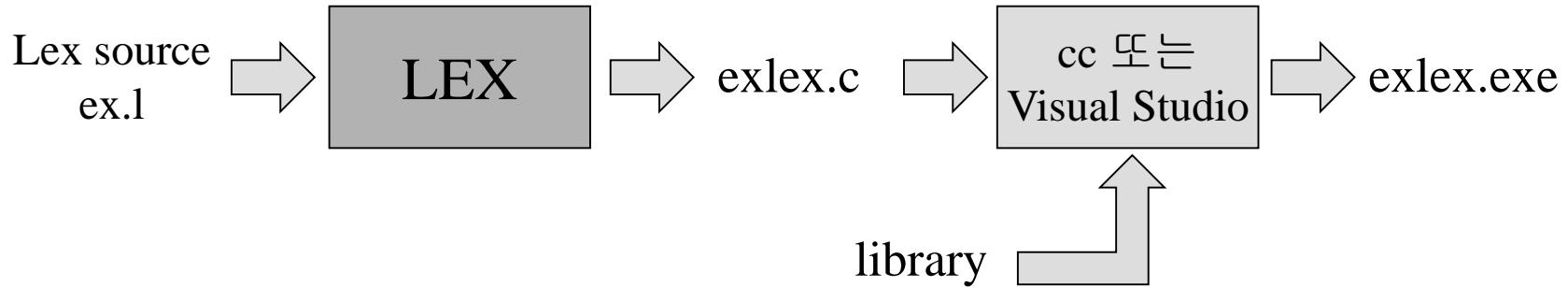
1. 원하는 폴더를 마우스로 선택
2. Shift키를 누른 채로 마우스 우측 클릭
3. “여기에 PowerShell 창 열기” 선택
4. 명령어 창이 열리면 “cmd” 명령 수행
5. 이후 필요한 명령 수행(flex 등)

```
PS D:\컴파일러동영상\DevKit> cmd
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

D:\컴파일러동영상\DevKit>
```



win_flex 수행 및 출력 C 파일 컴파일



■ PC용 Lex (win_flex):

```
C> win_flex --wincompat -o exlex.c ex.l
```

```
C> cc exlex.c
```

```
C> exlex
```

```
a
```

```
101 1
```

```
if
```

```
100 0
```

type flex.bat
flex ex

CBC vs

- Visual Studio 사용시에는
- 프로젝트를 새로 생성한 후,
- exlex.c를 소스에 추가하여 새로 build함
- 수행시에는
- Visual Studio에서 직접 실행하거나
- Debug 폴더에 만들어진 exlex.exe를
- 명령창에서 수행하는 방법이 있다.



ex.1의 확장 실습

■ 새로운 토큰 추가하기

- while 103
- for 104
- switch 105
- case 106

■ 주석 처리하기

- “//”로 시작하여 그 뒤 줄 끝까지의

■ 수행 예

- 출력: 빨간 색

```
while  
103 0  
for  
104 0  
//테스트  
// This is test  
sum  
101 1
```

```
%{  
#define IF 100  
#define ID 101  
#define RELOP 102  
#define WHILE 103  
#define FOR 104  
#define SWITCH 105  
#define CASE 106  
#define LE 201  
int yyval, tok;  
%}  
delim [ \t\n]  
ws {delim}+  
letter [A-Za-z]  
digit [0-9]  
id {letter}({letter}|{digit})*  
%%  
{ws} /* do nothing */  
if  
{yyval = 0; return(IF); }  
while {yyval = 0; return(WHILE); }  
for {yyval = 0; return(FOR); }  
switch {yyval = 0; return(SWITCH); }  
case {yyval = 0; return(CASE); }  
{id}  
{yyval = install_id(); return(ID); }  
"<="  
{yyval= LE; return(RELOP);}  
"/":*  
%%  
int install_id() { return(1); }  
int yywrap() {return(0); }  
101 1  
101 1  
101 1
```



Left context sensitivity 예제

<구현1>

```
% {  
int flag;  
% }  
%%  
^a {flag = 'a'; ECHO;}  
^b {flag = 'b'; ECHO;}  
^c {flag = 'c'; ECHO;}  
\n {flag = 0 ; ECHO;}  
magic { switch (flag){  
    case 'a': printf("first"); break;  
    case 'b': printf("second"); break;  
    case 'c': printf("third"); break;  
    default: ECHO; break;}  
}
```

<구현2>

```
%START AA BB CC  
%%  
^a {ECHO; BEGIN AA;}  
^b {ECHO; BEGIN BB;}  
^c {ECHO; BEGIN CC;}  
\n {ECHO; BEGIN 0;}  
<AA>magic printf("first");  
<BB>magic printf("second");  
<CC>magic printf("third");
```

수행 예: 출력(빨간색)
(구현1과 구현2가 동일)

a
a
magic
magic

a magic
a first
b magic
b second
a b magic
a b first



Lex 정규 표현 요약

x 문자 “x”.

“x” x가 연산자일지라도 “x”.

\x x가 연산자일지라도 “x”.

[xy] 문자 x 또는 y.

[x-z] 문자 x, y 또는 z.

[^x] x가 아닌 특정 문자.

. 개행이 아닌 특정 문자.

^x 라인의 시작일 때의 x.

x\$ 라인의 끝일 때의 x.

<y>x Lex가 시작 상태 y일 때 x.

x? 선택적인 x.

x* x의 인스턴스, 0개 부터 n개

x+ x의 인스턴스, 1개 부터 n개

x | y x 또는 y.

(x) x.

x/y y가 뒤따르는 x.

{xx} 정의 부문으로 부터의 xx의 변형.