

## 2008년 6월 13일 강의에서 다룬 예제에 대하여

컴파일러 수강생 여러분,

2008년 1학기 마지막 강의에서 다룬 Lex와 Yacc의 예제 중 하나가 제대로 동작하지 않아서 선생으로서 대단히 민망스러웠습니다. 제가 확인한 뒤 그 내용을 웹에 올리겠다고 약속했는데, 요 며칠 사이에 급한 일이 많이 생기는 바람에 늦어져서 또한 미안합니다.

결론부터 말하자면, 마지막 예제가 제대로 동작하지 않았던 이유는 실습 서버에 설치되어 있는 Yacc이 제대로 기능을 다하지 않아서입니다. (즉, 제가 뭘 몰라서 그랬던 것은 아니라는 거죠. 흠, 흠.)

그 예제의 Lex와 Yacc을 위한 소스(source)를 다시 보겠습니다. 먼저 Lex 소스는 다음과 같습니다.

```
%{
#include "y.tab.h"
%}

delim  [ Wt]
number [1-9][0-9]*

%%

{delim}+      { }
{number}     { sscanf(yytext, "%d", &yyval); return NUMBER; }
Wn | .      { return yytext[0]; }
```

그리고 Yacc 소스는 다음과 같았죠.

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

%token NUMBER

%%

line      : expr 'Wn'          { printf("%dWn", $1); }
          ;
expr      : expr '+' term     { $$ = $1 + $3; }
          | expr '-' term     { $$ = $1 - $3; }
          | term
          ;
```

```

term    : term '*' factor      { $$ = $1 * $3; }
        | term '/' factor     { $$ = $1 / $3; }
        | factor
        ;
factor  : '(' expr ')'         { $$ = $2; }
        | NUMBER
        ;

```

```
%%
```

```

int main()
{
    yyparse();
    return 0;
}

```

대단히 간단한 내용입니다. 상기와 같은 소스들을 가지고 다음과 같이 Lex와 Yacc을 돌린 뒤, 결과로 만들어진 C 파일들을 컴파일 시켜 실행 파일을 만들어 수행시켰고, 오류를 받았죠.

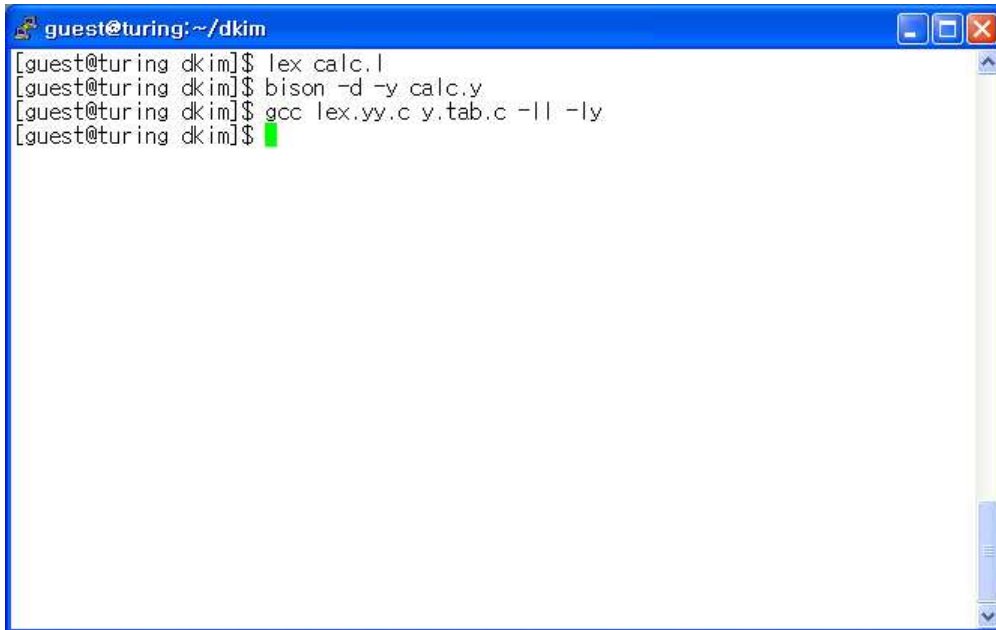
```

guest@turing:~/dkim
[guest@turing dkim]$ lex calc.l
[guest@turing dkim]$ yacc -d calc.y
[guest@turing dkim]$ gcc lex.yy.c y.tab.c -ll -ly
calc.l: In function 'yylex':
calc.l:11: error: 'yylval' undeclared (first use in this function)
calc.l:11: error: (Each undeclared identifier is reported only once
calc.l:11: error: for each function it appears in.)
[guest@turing dkim]$ █

```

오류 메시지는 보다시피 'yylval'이란 이름의 변수가 선언되지 않았다는 뜻입니다. 그런데 원래 이런 메시지가 나오면 안 되는 것이거든요. 엄연히 'yylval'은 Yacc의 출력 파일인 'y.tab.c'에 정의되어 있습니다. 그리고 Lex의 출력 파일인 'lex.yy.c'에서는 이러한 외부 변수(external variable)들의 선언부인 'y.tab.h' 파일을 읽어 들이고 있으므로 'yylex' 함수에서 'yylval'을 모른다는 일은 있어서는 안 됩니다. 도무지 이해할 수 없었으나, Yacc이 출력하는 선언부 파일 'y.tab.h'을 열어보니 정말 'yylval' 변수에 대한 선언이 포함되어 있지 않았습니니다. 그렇다면 상기와 같은 오류 메시지는 당연한 일이죠.

그래서 수업 시간에 얘기한 GNU의 Yacc 대체물인 Bison을 다음과 같이 대신 돌려봤습니다.



```
guest@turing:~/dkim
[guest@turing dkim]$ lex calc.l
[guest@turing dkim]$ bison -d -y calc.y
[guest@turing dkim]$ gcc lex.yy.c y.tab.c -ll -ly
[guest@turing dkim]$
```

여기서 Bison의 '-y' 옵션은 출력 파일을 Yacc의 관례를 따라 'y.tab.c'란 이름으로 만들라는 뜻입니다. 보다시피 제대로 컴파일되었습니다. 그리고 Bison에서 선언된 이름들을 외부에서 사용하기 위한 헤더 파일인 'y.tab.h'를 열어보면 'yyval'이 외부 변수임이 바르게 선언되어 있습니다. (이것으로 보아, Linux 상의 Yacc은 Bison에 밀려 제대로 관리가 되지 않고 있는 듯합니다.)

실행을 시켜 볼까요? 아래 캡처 화면과 같이 제대로 실행됨을 볼 수 있습니다.



```
guest@turing:~/dkim
[guest@turing dkim]$ lex calc.l
[guest@turing dkim]$ bison -d -y calc.y
[guest@turing dkim]$ gcc lex.yy.c y.tab.c -ll -ly
[guest@turing dkim]$ ./a.out
(342 + 726) / 3 - 2 * (846 - 561)
-214
[guest@turing dkim]$
```

사소한 오류를 미리 체크하지 못하고 수업 시간에 학생 여러분에게 불편을 주어 매우 미안합니다. 반성하여 앞으로는 이런 일이 없도록 신경을 쓰겠습니다. ;-)