Lex und Yacc

**„Lt. Worf, scannen Sie das Schiff!
300dpi, Sir?“**

Hagen Paul Pfeifer
hagen@jauu.net

29. April 2004

Überblick

- Einführung

Überblick

- Einführung
- Beispiele

Überblick

- Einführung
- Beispiele
- Êpilogue

Lex, Yacc, . . . ? Grundsätzliches!

Lex, Yacc, . . . ? Grundsätzliches!

„Lex und Yacc unterstützen Programme zu schreiben welche Eingaben (die einer gewissen regelmässigkeit unterliegen) transformieren, nutzen oder in einer anderen Weise als Eingabe nutzen.“

Lex

Allgemeines

Lex

Allgemeines

- Lex erzeugt C Code für lexikalischen Analyser (Scanner)

Lex

Allgemeines

- Lex erzeugt C Code für lexikalischen Analyser (Scanner)
- Benutzt Pattern um Token zu beschreiben

Lex

Allgemeines

- Lex erzeugt C Code für lexikalischen Analyser (Scanner)
- Benutzt Pattern um Token zu beschreiben
- Teillen des Input Streams in Einheiten

Lex

Allgemeines

- Lex erzeugt C Code für lexikalischen Analyser (Scanner)
- Benutzt Pattern um Token zu beschreiben
- Teillen des Input Streams in Einheiten
- Token? wc -> WORD; gcc -> keywords, comments; bc -> . . .

Lex

Allgemeines

- Lex erzeugt C Code für lexikalischen Analyser (Scanner)
- Benutzt Pattern um Token zu beschreiben
- Teillen des Input Streams in Einheiten
- Token? wc -> WORD; gcc -> keywords, comments; bc -> . . .
- Tokenbeschreibung ueber Reguläre Ausdrücke

Lex

Allgemeines

- Lex erzeugt C Code für lexikalischen Analyser (Scanner)
- Benutzt Pattern um Token zu beschreiben
- Teillen des Input Streams in Einheiten
- Token? wc -> WORD; gcc -> keywords, comments; bc -> . . .
- Tokenbeschreibung ueber Reguläre Ausdrücke
- GNU -> flex

Yacc

Allgemeines

Yacc

Allgemeines

- Yacc erzeugt C Code für Syntax Analyser (Parser)

Yacc

Allgemeines

- Yacc erzeugt C Code für Syntax Analyser (Parser)
- Benutzt grammatische Regeln um Token der Eingabe für die Erstellung eines Syntaxbaumes (Beispiel: Rechner)

Yacc

Allgemeines

- Yacc erzeugt C Code für Syntax Analyser (Parser)
- Benutzt grammatische Regeln um Token der Eingabe für die Erstellung eines Syntaxbaumes (Beispiel: Rechner)
- Token liefert Lex (gewöhnlich)

Yacc

Allgemeines

- Yacc erzeugt C Code für Syntax Analyser (Parser)
- Benutzt grammatische Regeln um Token der Eingabe für die Erstellung eines Syntaxbaumes (Beispiel: Rechner)
- Token liefert Lex (gewöhnlich)
- Beziehung zwischen Token beschreibt Yacc

Yacc

Allgemeines

- Yacc erzeugt C Code für Syntax Analyser (Parser)
- Benutzt grammatische Regeln um Token der Eingabe für die Erstellung eines Syntaxbaumes (Beispiel: Rechner)
- Token liefert Lex (gewöhnlich)
- Beziehung zwischen Token beschreibt Yacc
- Yacc erkennt ab Token gegen Grammatik matched oder !matched

Lex

- Syntax

```
%{
definition section
%}
%%
rules section ->
pattern action
%%
user subroutine section
```

Lex (reloaded)

- Minimales Beispiel:

```
%%
.|\n    ECHO;
%%
int main()   { yylex();  }
int yywrap() { return 1; }
```

Lex (reloaded)

- Minimales Beispiel II:

```
/* minimales beispiel II */
%%
[ \t]+                  ;
[0-9]+                  { printf("Digit: %s\n", yytext); }
[a-zA-Z][a-zA-Z0-9]*    { printf("Word : %s\n", yytext); }
.|\n                    { ECHO;                          }
```

Lex (reloaded)

- Minimales Beispiel III:

```
/* minimales beispiel II */
%{
unsigned char = 0, word = 0, line = 0;
}%
word [^ \t\n]+
eol  \n
%%
{word}  { ++word; char += yylen; }
{eol}   { ++char; ++line;         }
.       { ++char;                 }
```

Lex (reloaded)

- Tokennizer in Perl

```
#!/usr/bin/perl
$i = 1; $_ = <>;
while(1) {
  if    (/\G(\w+)/gc) { do_it($1) }
  elsif (/\G(\W+)/gc) {}
  else                { last };
}
sub do_it{ if   ($i^=1) {print(   (shift) . " " )}
           else         {print( uc(shift) . " " )}
}
```

Lex (reloaded)

Bauanleitung

Lex (reloaded)

Bauanleitung

- lex quelle.l -> lex.yy.c

Lex (reloaded)

Bauanleitung

- lex quelle.l -> lex.yy.c
- gcc -o ziel lex.yy.c ODER

Lex (reloaded)

Bauanleitung

- lex quelle.l -> lex.yy.c
- gcc -o ziel lex.yy.c ODER
- gcc -ll -o ziel lex.yy.c

Lex (reloaded)

States

- Pattern machted nur wenn im state
- Nützlich bei größeren Strings
- Beispiel: -file foo.bar

Yacc

Einführendes Beispiel

- Beispiel:

```
expression: NUMBER
    | expression '+' NUMBER
    | expression '-' NUMBER
    ;
```

Yacc

- Beispiel: (II/I)

```
%token NAME NUMBER
%%
statement:  expression  { printf("=%d\n", $1; }

expression: expression '+' NUMBER {$$=$1+$3;}
    |       expression '-' NUMBER {$$=$1-$3;}
    |       NUMBER                {$$=$1;}
    ;
```

Yacc (reloaded)

Union

```
%union {
    int   val;
    char *str;
}
%token <str> NAME
%token <val> NUMBER
%type <val> expression
%%
statement: NAME '=' expression {stbl[$1]=$3;}
    | expression
    ;

expression: expression '+' NUMBER {$$=$1+$3;}
```

```
    |       expression '-' NUMBER {$$=$1-$3;}
    |       NUMBER               {$$=$1;}
    ;
```

Yacc (reloaded)

Bauanleitung

Yacc (reloaded)

Bauanleitung

- yacc quelle.y -> y.tab.c und yacc.tab.h

Yacc (reloaded)

Bauanleitung

- yacc quelle.y -> y.tab.c und yacc.tab.h

oder

Yacc (reloaded)

Bauanleitung

- yacc quelle.y -> y.tab.c und yacc.tab.h

oder

- bison quelle.y -> yacc.tab.c und yacc.tab.h

Beispiele

Beispiele

- BRAINFUCK!

Beispiele

- BRAINFUCK!
- Konfigurationsdatei

Êpilogue

Êpilogue

- Flex: %array vs. %pointer

Êpilogue

- Flex: %array vs. %pointer
- Error Recovery

Êpilogue

- Flex: %array vs. %pointer
- Error Recovery
- Flex next generation