Lexikalische Analyse mit LEX

Jan Hendrik Dithmar
15.12.2005

Motivation

- Klassische Beispiele:
  - das zu entwickelnde Programm ist ein Compiler
  - die Eingabe ist eine Konfigurationsdatei
- Eingabe hat eine bestimmte Struktur
- Struktur muss erkannt werden
- Dazu muss die Struktur (Syntax der Eingabe) formal festgelegt sein

Beispiel: Konfigurationsdatei

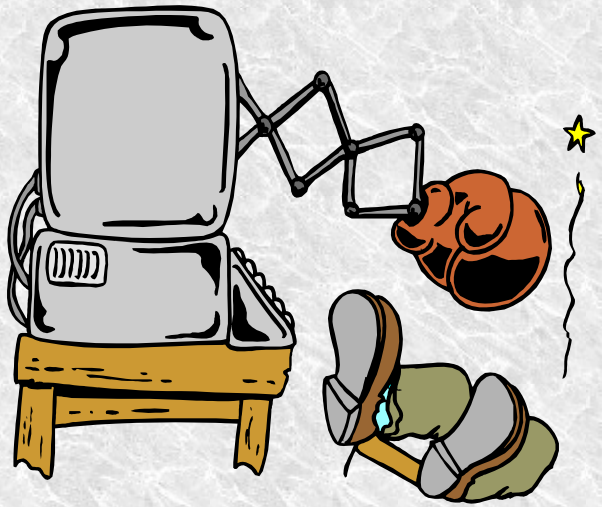
```
[General]
WindowState = Maximized
ShowSplashScreen = True

[Help]
ShowTooltips = True
ShowTipOfTheDay = False

[Config]
SaveTemporaryFiles = True
BrowserExecutable = firefox
StandardSearchEngine = http://www.google.de/
```

Eingaben parsen

- Man muss Eingaben (z.B. Konfigurationsdateien) parsen können.
- Solche Parser selbst zu programmieren ist zeitaufwändig und teilweise auch kompliziert.

Eingaben parsen (2)

- Es kann leicht passieren, dass man vom Arbeitsaufwand für die Entwicklung des Parsers „erschlagen“ wird.

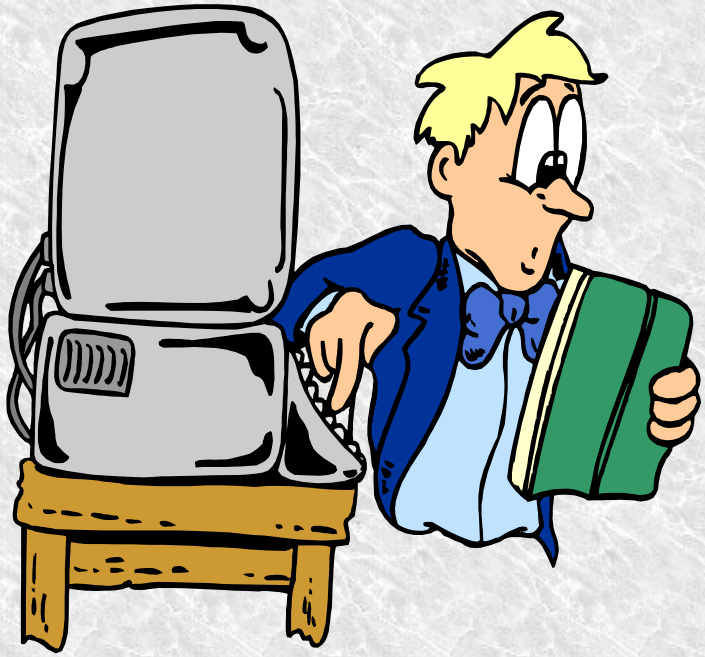


- Aus diesem Grund ist es einfacher, sich einen Scanner generieren zu lassen.

Lösung: LEX

- Mit LEX kann man z.B. folgende Aufgabe lösen:
  - Lexikalische Analyse

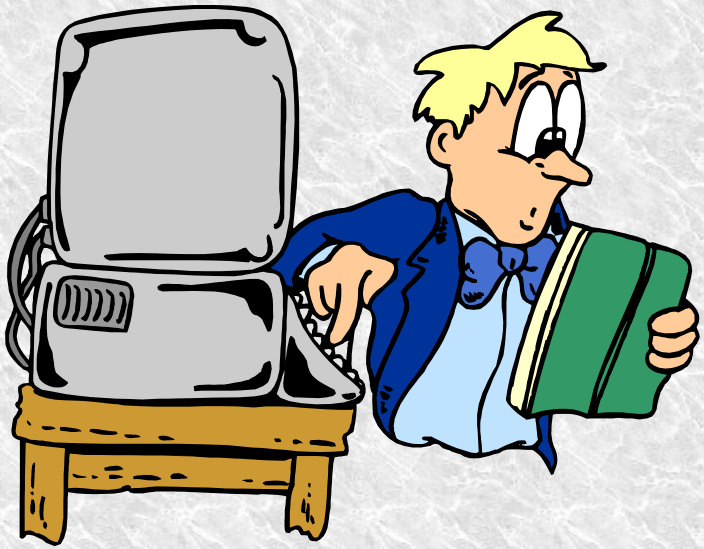
- Wie verwendet man LEX?



LEX-Spezifikation

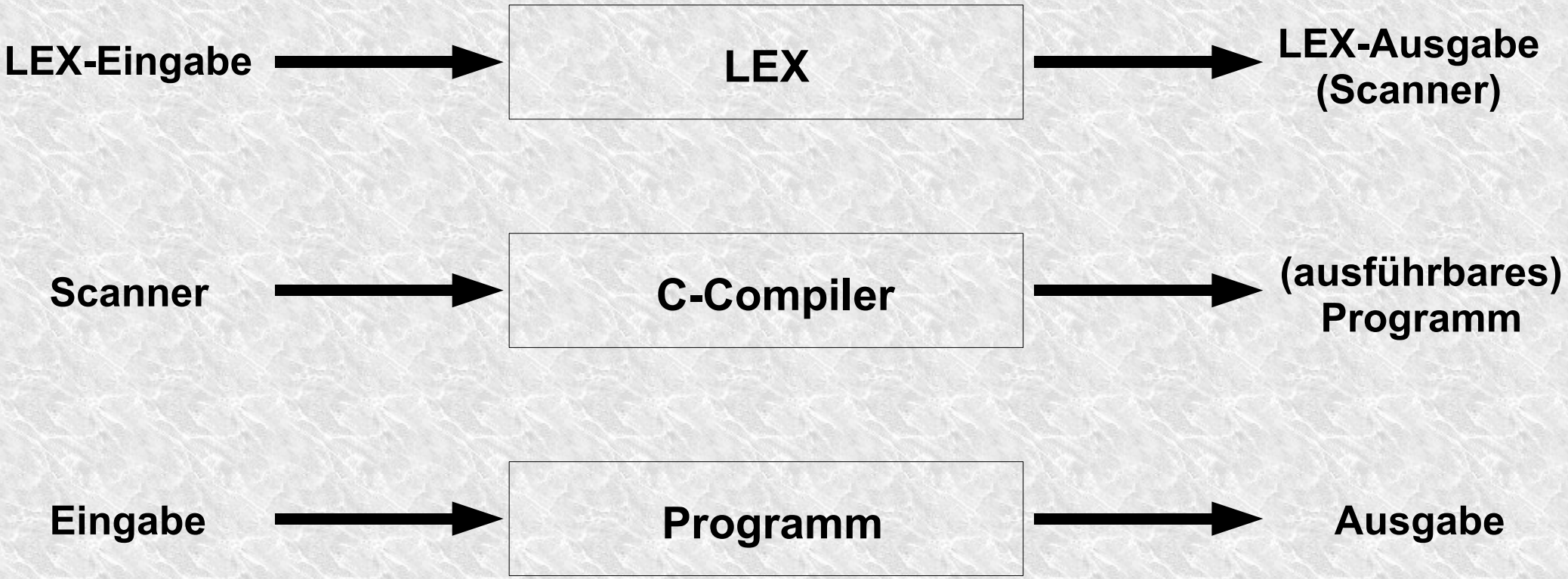
- LEX benötigt eine Spezifikationsdatei.
- Aufbau der Spezifikationsdatei:

Definitionen
%%
Regeln
%%
Funktionen



- Aus der Spezifikation generiert LEX den Scanner.

Arbeitsweise von LEX



Beispiel: Artikel entfernen

Als Beispiel nehmen wir eine Datei **`artikel.l`**, die Artikel filtern soll:

```
%%
der     |
die     |
das     |
ein     |
eine    ;
.       |
\n      ECHO;
```

Beispiel: Artikel entfernen (2)

```
$ lex artikel.l
$ cc -o artikel lex.yy.c -ll
$ echo "das Haus" | ./artikel
 Haus
$ _
```

- **`artikel.l`**: LEX-Eingabe
- **`lex.yy.c`**: von LEX erzeugte Datei (Scanner)
- **`-ll`**: Compileroption zum Binden der LEX-Bibliothek

Überlappende Regeln und **REJECT**

LEX bietet die Möglichkeit eine Regel zurückzuweisen und somit überlappende Regeln zu realisieren.

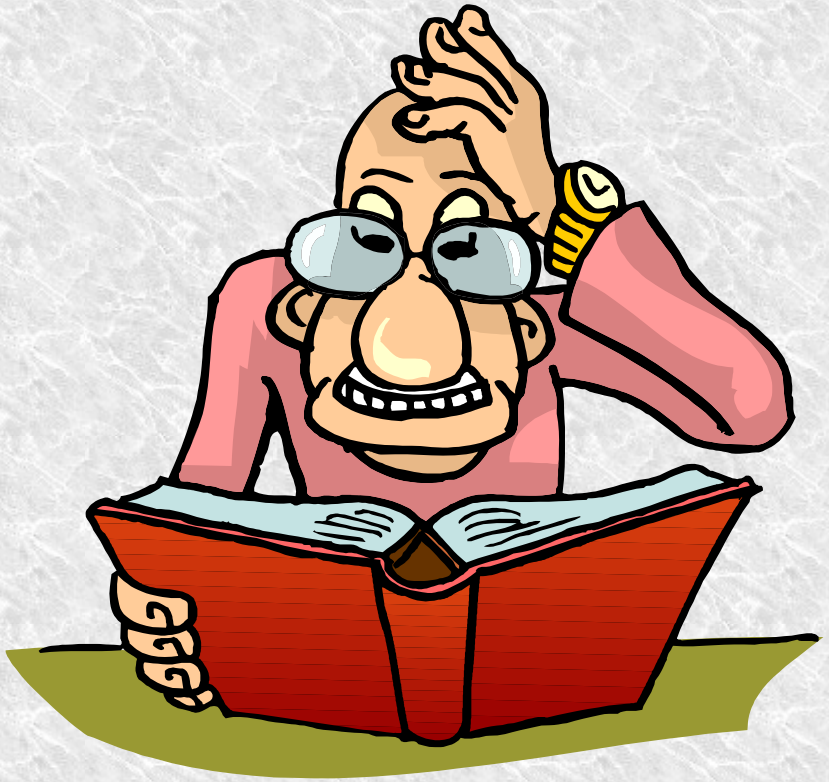
Beispiel (Datei **reject.l**):

```
%%
.    printf("Ja %s\n", yytext);
... printf("Nein %s\n", yytext); REJECT;
win printf("Treffer %s\n", yytext);
\n  ;
```

In **yytext** ist die betrachtete Zeichenkette gespeichert.

Überlappende Regeln und **REJECT** (2)

```
$ lex reject.l
$ cc -o reject lex.yy.c -ll
$ echo iwinthat | ./reject
Nein iwi
Ja i
Nein win
Treffer win
Nein tha
Ja t
Nein hat
Ja h
Ja a
Ja t
$ _
```



Neugierig geworden?

Literatur:

Andreas Zeller, Jenk Krinke:
„Open-Source-Programmierwerkzeuge“
2. Auflage
dpunkt.verlag
ISBN 3-89864-226-7
Preis: 39,00 € (D), 40,10 € (A)
www.programmierwerkzeuge.de