Konstruktory lexikálních analyzátorů

Thursday, May 30, 2013 8:39 AM

Lex

Program LEX (Lexical Analyzer Generator) slouží k tvorbě jiných programů, které mají cosi udělat se vstupním (textovým) souborem za pomoci lexikální analýzy. Tím se myslí analýza struktur, které se dají zapsat lineárními gramatikami, konečnými automaty nebo regulárními výrazy. Typické použití LEXu je dvojí: vytvořený program pracuje samostatně, nebo slouží jako vstupní filtr pro jiný (syntaktický) analyzátor, např. bison či yacc.

Lex umožňuje vytvořit lexikální analyzátor uvedením regulárních výrazů, které popisují vzory (patterns) pro tokeny. Vstupní notace pro Lex se nazývá *Lex language* a samotný nástroj je *Lex compiler*. Kompilátor Lexu transformuje vstupní vzory do přechodového diagramu (Jádrem toho všeho je *konečný automat*.) a generuje kód do souboru lex.yy.c, ve kterém je simulován přechodový diagram.

Vstupem Lexu je soubor, obvykle s koncovkou .l, např. *lex.l*, je napsaný v jazyce Lexu a popisuje lexikální analyzátor k vygenerování (rozpoznávaná slova a akce, které se mají po jejich rozpoznání provést). Slova (tokeny) se popisují regulárními výrazy, akce v cílovém programovacím jazyku. Výstup LEXu je zdrojový kód hotového programu, který se potom musí běžným způsobem přeložit. Pokud tedy používáme variantu LEXu, která generuje výstup v jazyku C, musí být i akce zapsané v jazyku C.

Lex kompilátor překlopí lex.l do programu v C, který je vždy uložen v souboru *lex.yy.c*. Pak je tento soubor zkompilován vždy do *a.out*. Výstupem je fungující lexikální analyzátor, který bere proud vstupních znaků a vytváří z nich proud tokenů.

Hodnoty atributů (= numerický kód/pointer do tabulky symbolů/nic) jsou umístěny v globální proměnné `yylval`, kterou sdílí lexikální analyzátor a parser.

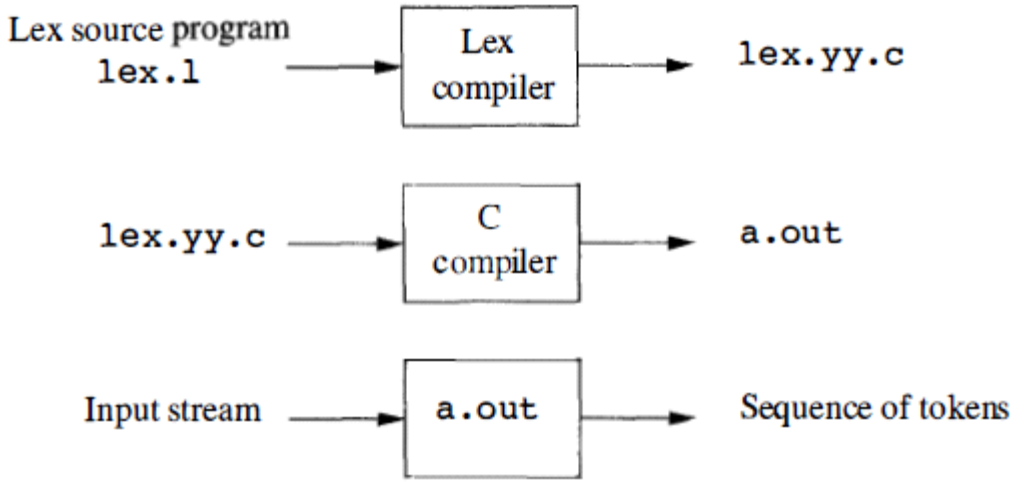


Figure 3.22: Creating a lexical analyzer with **Lex**

Struktura programu v Lexu

```
deklarace, definice
%%
popis slov a akcí
%%
další funkce (zapsané v cílovém jazyku)
```

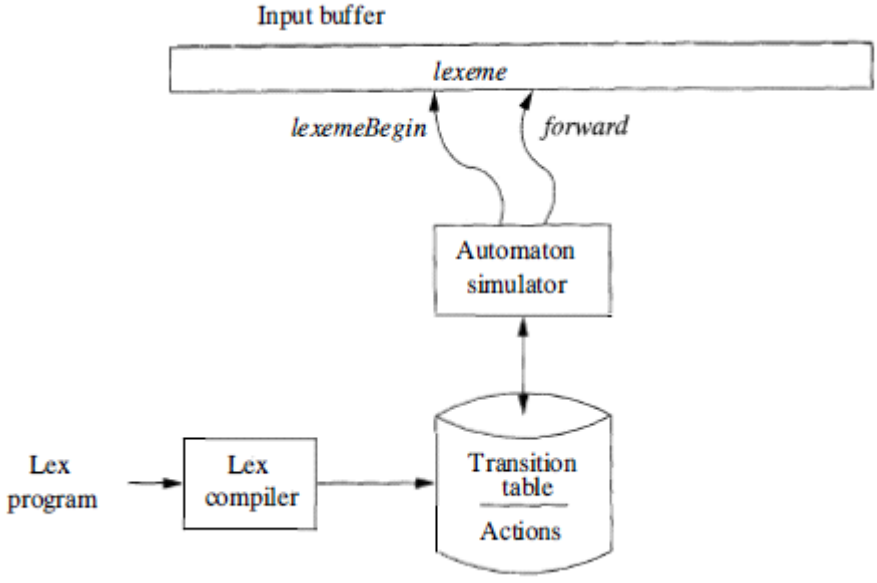
Příklad: program, který ve vstupním souboru nahradí všechny identifikátory slovem "IDENTIFIKATOR"

```
%%
[a-zA-Z_][0-9a-zA-Z_]*    printf("IDENTIFIKATOR");
%%
int main(void)
{
  yylex();
  return 0;
}
```

V popisu rozpoznávaných slov lze používat následující konstrukce:

x	znak "x"
[xy]	znak "x" nebo "y"
[x-z]	všechny znaky od "x" až k "z"
[^x]	jakýkoliv znak vyjma "x"
.	jakýkoliv znak, až na novou řádku
^x	znak "x", pokud se nachází na začátku řádky
x$	znak "x", pokud se nachází na konci řádky
x*	libovolný počet znaků "x"
x+	alespoň jeden znak "x"
x?	jeden nebo žádný znak "x"
x{m,n}	M až n výskytů znaku "x"
x|y	znak "x" nebo "y"
(x)	znak "x"
x/y	znak "x", je-li následován znakem "y"
{DEF}	doplnění definice z úvodní sekce
<y>x	znak "x", je-li splněna podmínka y

Architektura lexikálního analyzátoru generovaného Lexem



Lex z definovaných regulárních výrazů ze vstupního souboru → NKA → DKA; pravidlo: v případě konfliktů přiřazuje lexém vzoru dle nejdelšího prefixu.

Další varianty

- Flex – volně dostupná implementace Lexu, pro C.
- JLex – volně dostupná implementace Lexu, pro Javu.
- C# LEX - varianta JLex pro C#.

- PLY - implementace Lexu v Pythonu

From <https://d.docs.live.net/e3534876709763a3/Dokumenty/ZCU/Statnice/Statnice.docx>