Outils de Compilation

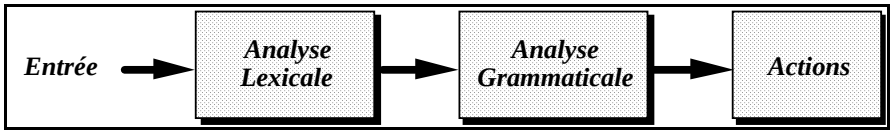
Lex et Yacc

Didier Donsez

ISTV / UVHC

Outils de Génération de Compilateurs/Interpréteurs

Rappel sur la compilation / interprétation



Analyse Lexicale

Analyse Grammaticale

Actions

- **Immédiates : Interpréteurs**
- **Différées: Compilation**

Lex

un générateur d'analyseur lexical

un homologue : GNU flex

génère une fonction `yylex()`

spécification lex

Lex - Exemple de spécification

Yacc

un générateur d'analyseur syntaxtique

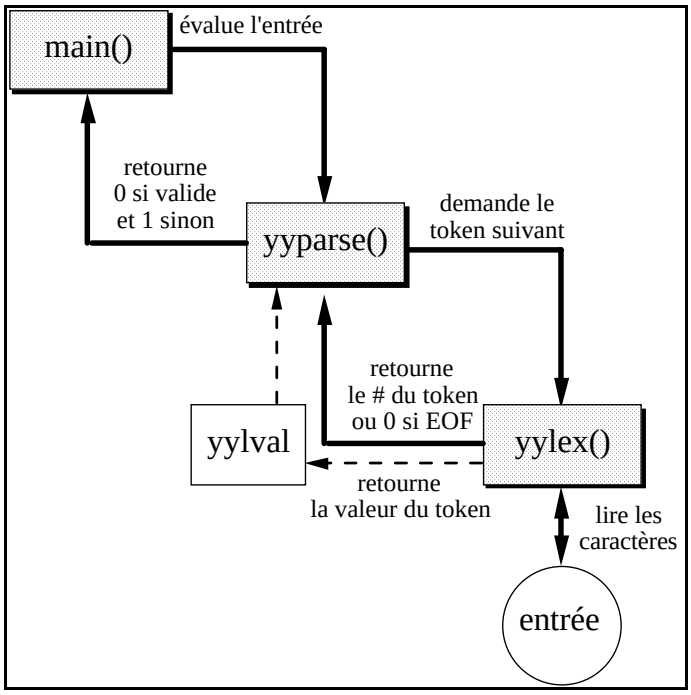
un homologue : GNU bison

génère une fonction `yyparse()`

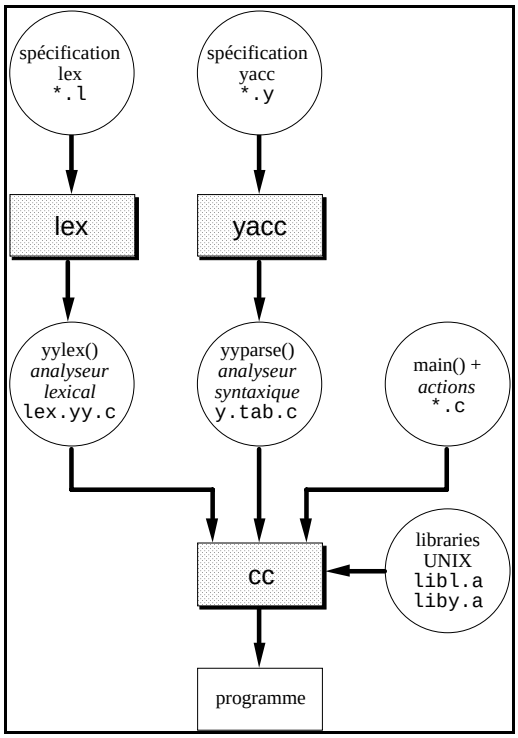
spécification yacc

Yacc - Exemple de spécification

Collaboration des Lex et Yacc



Utilisation de Lex et Yacc



Lex - Expressions Régulières

.	tout caractère seul sauf newline	
[ ]	un des caractères spécifiés	**[abcdABCD0123]**
-	un intervalle de caractère si dans [ ]	**[a-dA-D0-3]**
?	option	**10?9** est **109** ou **19**
*	répétition (zéro ou plus)	[ab]*
+	répétition (au moins une)	**[ab]+** est **[ab][ab]***
|	alternative	**000|110|101|011**
( )	groupement d'expressions	**(0|2|4|8)***
/	condition sur la suite	**etu/.univ-compiegne** est **etu**
{ }	permet des répétitions	**if(def)?** est **if(def)\{0,1\}**
$	la fin de ligne	
^	le début de ligne ou le complement dans [ ]	**\"[^\"\n]*\"**

Lex - Variables, Macros, Routines

< > définit une condition de départ

```
#include  "sys/*foo.h"

\% start PREPROC
%%
^#          { ECHO; BEGIN PREPROC; }
\n          { ECHO; BEGIN 0; }
<PREPROC>" { ••• }
•••
```

Variables

yytext, yyleng

Macro

BEGIN	place lex dans un état,
ECHO	**printf(" %s", yytext);**
REJECT	teste d'autres actions
input, unput, output	permet de redéfinir l'entrée et la sortie de yylex()

Routines

yyless, yymore, yyreject, yywrap

Yacc - Modèle de Fonctionnement

Modèle du pointeur

```
%token A B C D E F
%%
```

après A B

```
start: x | y
x : A B ∧ C D
y : A B ∧ E F
```

après A B C

```
start: x | y
x : A B C ∧ D
y : A B E F
```

après A

```
start: x | y
x : A ∧ B z E
y : A ∧ B z F
z : C D
```

après A B C

```
start: x | y
x : A B z E
y : A B z F
z : C ∧ D
```

après A B C D

```
start: x | y
x : A B z ∧ E
y : A B z ∧ F
z : C D
```

Réduction d'une règle

quand un pointeur "passe"
le dernier symbole de la règle

```
start: x | y
x: A ∧
y: B
```

Définition d'un Conflit

si une règle est réduite
quand il y a plus d'un pointeur

```
start: x | y
x: A ∧
y: A ∧
```

Yacc - Conflits

Conflit *Reduce-Reduce*

```
start: x | y
x : A ^
y : A ^
```

Conflit *Shift-Reduce*

```
start : x C | y ;
x : A ;
y : A C ;
```

y.output *(yacc -v) fichier des conflits détectés par yacc*

```
...
4: reduce/reduce conflict
   (red'ns 3 and 4 ) on $end
state 4
   x :  A_    (3)
   y :  A_    (4)
```

```
...
4: shift/reduce conflict
   (shift 6, red'n 3) on C
state 4
   x :  A_    (3)
   y :  A_C
```

Remarques


```
start: x B | y C
x : A ^
y : A ^
```

pas de conflit : un token d'avance

```
start: x B C | y B D
x : A ^
y : A ^
```

conflit RR

Yacc - Règles ambigues

```
expr: INTEGER | expr '-' expr
```

Entrée: INTEGER - INTEGER - INTEGER
Interprétation1: (INTEGER - INTEGER) - INTEGER
Interprétation2: INTEGER - (INTEGER - INTEGER)

```
4: shift/reduce conflict
   (shift 3, red'n 1) on -
state 4
        expr :  expr_- expr
        expr :  expr - expr_    (1)
```

```
stmt : IF '(' cond ')' stmt
     | IF '(' cond ')' stmt ELSE stmt
     | INST ;
cond : COND ;
```

IF (COND) IF (COND) INST ELSE INST
IF (COND) { *IF (COND) INST* } ELSE INST
IF (COND) { *IF (COND) INST ELSE INST* }

Yacc - Règles récursives

Récursivité Gauche

```
list  : list objet
        { printf("R11 "); }
      | objet
        { printf("R12 "); }
      ;
objet : INTEGER
        { printf("R2(%d) ",$1); }
      ;
```

Entrée :

```
101 102 103 104 105 106
```

Sortie :

```
R2(101) R12 R2(102) R11
R2(103) R11 R2(104) R11
R2(105) R11 R2(106) R11
```

Récursivité Droite

```
list  : objet list
        { printf("R11 "); }
      | objet
        { printf("R12 "); }
      ;
objet : INTEGER
        { printf("R2(%d) ",$1); }
      ;
```

Entrée :

```
101 102 103 104 105 106
```

Sortie :

```
R2(101) R2(102) R2(103) R2(104)
R2(105) R2(106) R12 R11 R11 R11
R11 R11
```

parser plus gros
pile interne plus "haute"

Yacc - Misc

token error *manipulation des erreurs*

```
liste : objet
      | liste objet
      | error
            { warning("objet non connu", (char*)0);
              printf("resynchronisation sur \\n\n"); }
        '\n'
      ;
```

synchronisation = abandon de tous les tokens jusquà **'\n'**

yyerror() *appelée par yacc en cas d'erreur de syntaxte*

yylex() *écrite par vous ou généré par* **lex**

yydebug *comment opère l'automate ?*

```
main() {
#ifdef YYDEBUG
      extern yydebug;  yydebug = 1;
#end
      return yyparse();
}
```


Outil

C/C++

Lex et Yacc

Java

Byacc/Java (version de Yacc générant un parser en Java)

Bibliographie

Lex & Yacc, Ed. O'Reilly

le "Dragon Book" : Aho, Sethi, Ullmann, "Compilateurs : principes, techniques et outils", Ed. Interedition.