

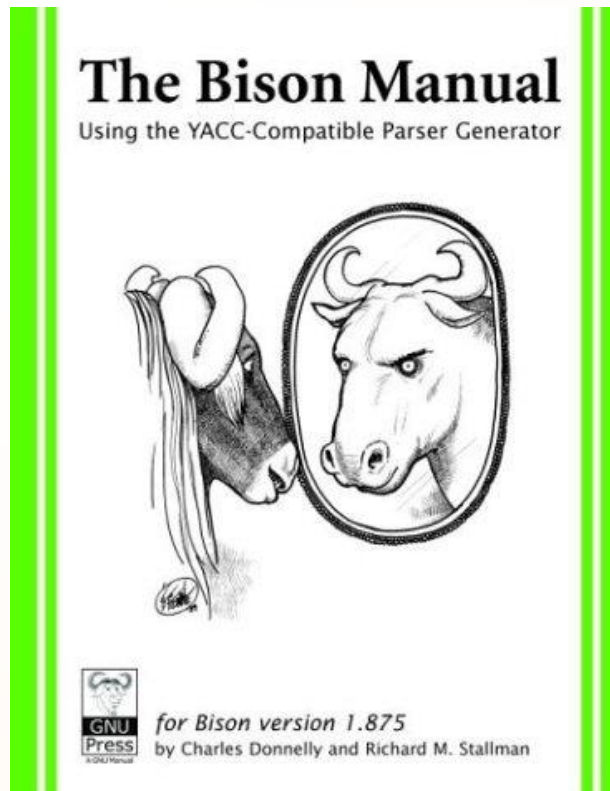
yacc / bison

•



yacc / bison

generatoare de analizoare sintactice



- **yacc** = Yet Another Compiler Compiler
- **bison** =
“ GNU Project parser generator (yacc replacement)”

“general-purpose parser generator
that converts a context-free grammar ...
into a C program to parse that grammar “

<http://dinosaur.compilertools.net/bison/index.html>

<http://www.gnu.org/software/bison/manual/bison.html>



- **yacc** = Yet Another Compiler Compiler
- **bison** =
“ GNU Project parser generator (yacc replacement)”
gramatică bazată pe LR(1) → cod C

sub UNIX

```
$ info bison
```

```
$ bison -V
```

```
bison (GNU Bison) 2.5 2.4.1
```

```
Written by Robert Corbett and Richard Stallman
```

```
~ 1988 - 1990
```

•

Structura fișierului de intrare

```
% {  
declarații C  
% }
```

```
declarații
```

```
% %  
reguli ale gramaticii  
% %
```

```
Cod C
```

Secțiunea *reguli ale gramaticii*:

- regulile multiple pt. același *rezultat* pot fi scrise separat sau pot fi unificate prin " | " :

```
rezultat:      regula1-componente...  
              | regula2-componente...  
              ...  
              ;
```

De exemplu:

```
exp:      exp exp '+'  
          | exp exp '*'  
          ;
```

Secțiunea *reguli ale gramaticii*:

- pentru fiecare componenta pot fi specificate *acțiuni* (semantica regulii):

{instrucțiuni C}

```
rezultat:   regula1-componente... {instr. C 1}  
           | regula2-componente... {instr. C 2}  
           ...  
           ;
```

- acțiunea poate fi vida.

Observatii (1)

- presupune ca exista functiile:
 int yylex()
 - returneaza codul unui atom
 (\leq **%token DIGIT**)
 yyerror
 - apelata atunci cand se detecteaza o eroare de sintaxa
- genereaza functia
 - **int yyparse(void)**
- nu genereaza functia main

Restul fisierului

- yacc generează o funcție `yyparse()`
- erorile sintactice sunt raportate apelând `yyerror()`

```
%%  
yylex()  
{  
    ...  
}  
main()  
{  
    yyparse();  
}  
  
yyerror()  
{  
    printf("syntax error\n");  
    exit(1);  
}
```

Folosire bison

- `$ bison fisier.y`
 `=> fisier.tab.c`
 `... gcc ...`

Folosirea bison + flex

- ?? constantele asociate atomilor lexicali
- \$ bison -d fisier.y
 - => fisier.tab.h
 - => fisier.tab.c
- fisierul *.lxi (ce urmeaza sa fie compilat cu flex)
 - #include “fisier.tab.h”

Valori semantice ale neterminalelor

- permite ca actiunea semantica asociata unei reguli de productie sa fie descrisa in functie de valorile semantice ale neterminalelor
- exemplu:
`expr: expr '+' expr { $$ = $1 + $3; }`
- `$$` - valoarea semantica asociata net. din membrul stang
- `$i` - valoarea semantica asociata
celui de-al i-lea net. din membrul drept
- **yylval** - valoarea semantica a unui atom
`<= yylex !!`

Legatura cu yylex

- yacc apelează `yylex()` pt. a obține următorul atom
- “valoarea” unui atom se memoreaza în var. globală `yylval`
- tipul implicit este `int`

```
%%  
yylex()  
{  
  
    int c;  
  
    c = getchar();  
  
    if (isdigit(c)) {  
        yylval = c - '0';  
        return DIGIT;  
    }  
    return c;  
}
```

Interpreter expresii

```
%%  
line : expr '\n'          { printf("%d\n", $1); }  
    ;  
expr  : expr '+' expr      { $$ = $1 + $3; }  
    |  expr '*' expr      { $$ = $1 * $3; }  
    |  '(' expr ')'        { $$ = $2; }  
    |  DIGIT               { $$ = $1; }  
    ;  
%%
```

Gramatica ambigua !

- atunci cand construieste arborele de derivare – care alegere se face?
- eroare: “conflict shift/reduce”

Precedența operatorilor

prioritate
de sus
(mică)
în jos
(mare)

```
%token DIGIT
```

```
%left '+'
```

declaratie de atom !!

```
%left '*'
```

```
%%
```

```
line : expr '\n'      { printf("%d\n", $1) ; }  
      ;
```

```
expr : expr '+' expr   { $$ = $1 + $3 ; }  
      | expr '*' expr   { $$ = $1 * $3 ; }  
      | '(' expr ')'    { $$ = $2 ; }  
      | DIGIT           { $$ = $1 ; }  
      ;
```

```
%%
```