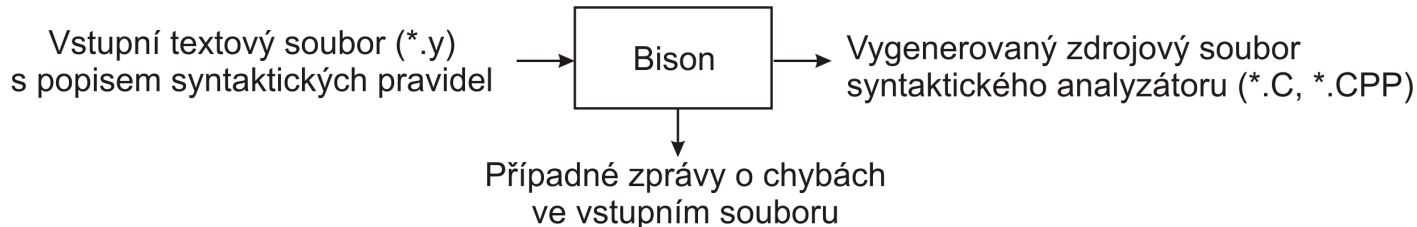


Bison – parser generator



Formát vstupního souboru:

```
%{  
    .. hlavičkové soubory, deklarace a jiné ..  
}%  
deklarace Bisonu  
%%  
gramatická pravidla  
%%  
uživatelský kód
```

Část, která je mezi symboly `%{` a `%}`, Bison nijak nevyužívá a jen ji překopíruje do výstupního souboru. Obsahuje jména hlavičkových souborů a různé deklarace, které jsou zapotřebí pro následný překlad vygenerovaného syntaktického analyzátoru.

V části deklarací Bisonu jsou popsány neterminální a terminální symboly bezkontextové gramatiky. Každému neterminálnímu symbolu může být přiřazen jeden syntetizující atribut. V deklaracích Bisonu se uvádí proměnná, jejíž datový typ určuje datový typ atributu. Tyto proměnné jsou deklarovány v datovém typu `YYSTYPE`. Ten je zpravidla vytvořen jako strukturovaný typ *union*.

V následujících zápisech jsou neterminální symboly psány malými písmeny a terminální symboly, které mají délku větší než 1 znak, jsou psány velkými písmeny (konvence používaná programy Flex a Bison).

Deklarace má zápis

```
%type <typ> symbol1 symbol2 ...
```

Příklad.

```
struct Uzel { /* ... */ };  
union YYSTYPE { int c; const char *r; Uzel *u; };  
%type <u> prikaz blok
```

Pokud některým neterminálním symbolům nepřičadíme žádný atribut, nemusí se deklarovat nebo deklarace má tvar

```
%type symbol1 symbol2 ...
```

Deklarace terminálních symbolů má zápis

```
%token <typ> symbol1 symbol2 ...
```

Nemá-li terminální symbol přiřazený atribut, deklarujeme ho

```
%token symbol1 symbol2 ...
```

U deklarace operátorů uvádíme jejich asociativitu. Operátory s levou asociativitou deklarujeme

%left *symbol*₁ *symbol*₂ ...

Operátory s pravou asociativitou deklarujeme

%right *symbol*₁ *symbol*₂ ...

Dále u operátorů uvádíme precedenci. Operátory se stejnou precedencí zapíšeme na jeden řádek. Řádky se zápisy operátorů seřadíme od operátorů s nejnižší precedencí po operátory s nejvyšší precedencí.

Příklad.

```
%right '='
%left '+' '-'
%left '*' '/' '%'
%right '!' '~'
```

Počáteční symbol gramatiky deklarujeme

%start *symbol*

V části gramatických pravidel jsou uvedena jednotlivá gramatická pravidla bezkontextové gramatiky. Pravidlo bezkontextové gramatiky

A → α

zde zapíšeme ve tvaru

A : α ;

Za pravidlem zpravidla bývá uveden blok se sémantikou. Zápis je pak ve tvaru

A : α { ... *sémantika* ... } ;

Více pravidel se stejnou levou stranou

A → α_1 | α_2 | ... | α_s

zapíšeme ve tvaru

```
A :     $\alpha_1$  { ... sémantika ... }
      |  $\alpha_2$  { ... sémantika ... }
      | ...
      |  $\alpha_s$  { ... sémantika ... } ;
```

Atributy přiřazené symbolům pravidla mají označení \$\$, \$1, \$2, ...

Přiřazení označení atributů pro pravidlo **A** → $X_1X_2...X_m$ je

```
A :    $X_1$     $X_2$    .....  $X_m$ 
      $$    $1    $2   ..... $m
```

Příklad.

```
enum Typ { INT,FLOAT,CHYBA } ;
```

```
union YYSTYPE { Typ typ; } ;
```

```
%type <typ> vyraz
```

```
vyraz: vyraz '+' vyraz { if ($1==INT && $3==INT) $$=INT;
                        else
                          if ($1==CHYBA || $3==CHYBA) $$=CHYBA;
                        else $$=FLOAT; }
```

```
| vyraz '&' vyraz { if ($1==INT && $3==INT) $$=INT;  
                    else $$=CHYBA; } ;
```

Hodnoty a funkce dostupné pro uživatele

`void yyerror(const char *)` ; – tuto funkci volá syntaktický analyzátor při nalezení chyby v analyzovaném programu. Parametrem funkce je popis chyby.