Projet de compilation : Compilateur LLVM

BEZAMAT Jérémy
PARPAITE Thibault

Bordeaux INP
ENSEIRB
MATMECA

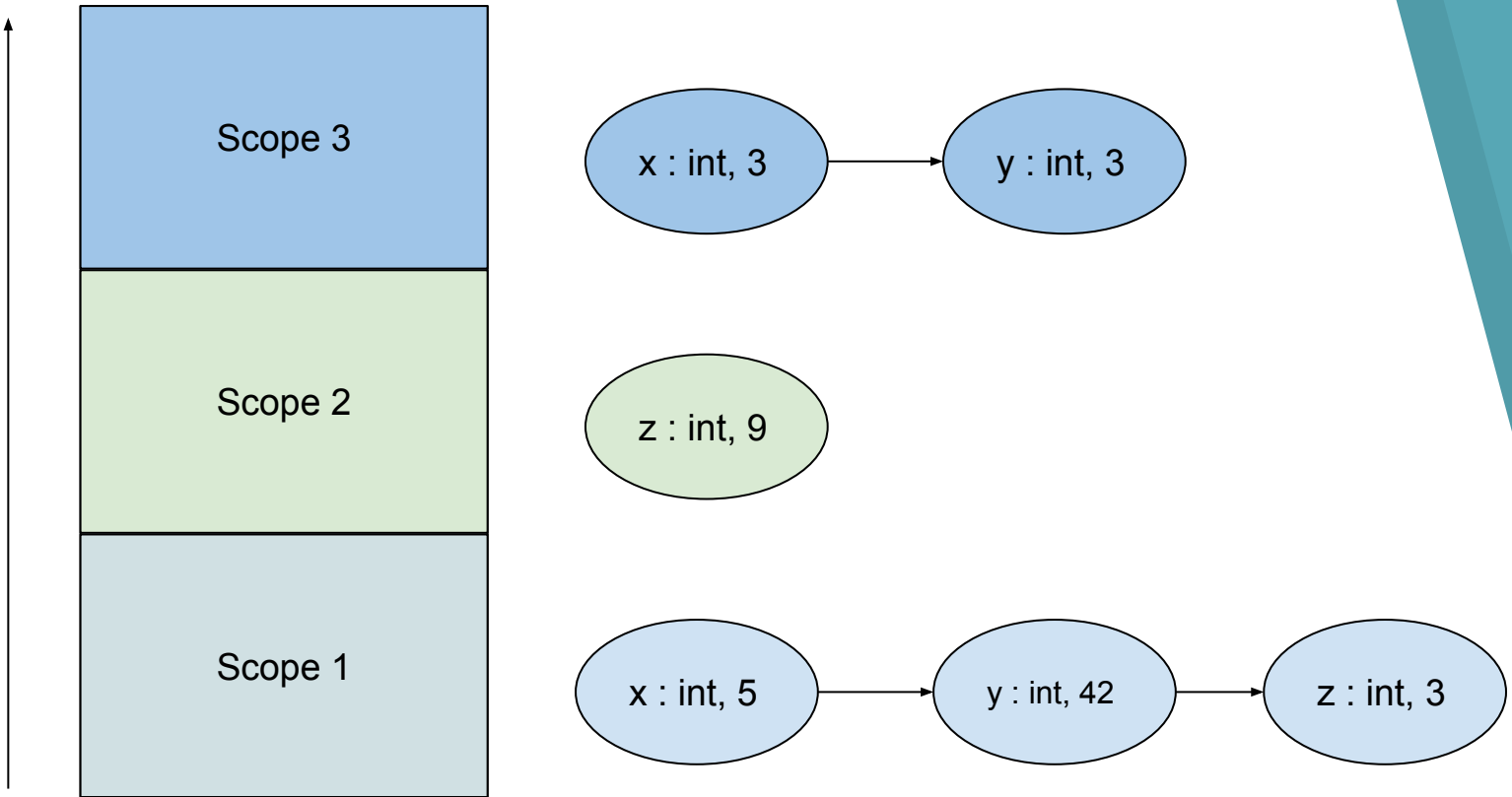
S7
18/12/2017

Présentation générale du projet

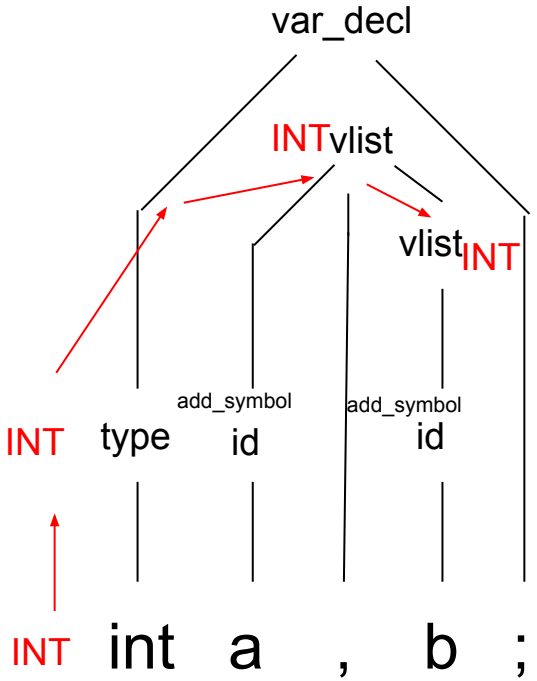
- Lexer.l (*analyse lexicale*)
- Parser.y (*analyse syntaxique*)
- utile.h
- utilitaire.h / utilitaire.c
- Table des symboles
- Liste générique (avec itérateur)
- Tests : .c / .ll / .exp
- CMake

Table des symboles, portée des variables

get_symbol / set_symbol /increase_scope / decrease_scope



Liste de déclarations, attributs hérités




```
var_decl : type vlist           /* On transmet par héritage la valeur de type en utilisant la pile (parcours LR unique), équivalent de $2 = $1 */


vlist : ID vir { $<t_type>$ = $<t_type>0; set_type($1, $<t_type>$); } vlist     /* On utilise un attribut hérité (le type) qu'on récupère sur la pile $0       */
      | ID     { set_type($1, $<t_type>0); }                                    /* L'action $$ = $0 est nécessaire pour remettre l'attribut type comme dernier */
      ;                                                                         /* élément de la pile (pour que la suite de la liste puisse également hériter) */
```

Évaluation paresseuse, if-then-else, boucle

On génère un attribut de type label, puis on utilise la pile pour récupérer les informations au moment voulu.

Eval paresseuse : dès qu'on a FAUX (and) ou VRAI (or), on jump vers THEN

```
cond : if bool_cond inst %prec UNA     { printf("  br label L%i\n", $1.two); printf("L%i:\n", $1.two); }
     | if bool_cond inst else inst     { printf("  br label L%i\n", $4.one); printf("L%i:\n", $4.one); } /* Label qu'on vient de generer dans else */
     ;


bool_cond : PO bool PF                 { printf("  br i1 %%r%d, label %%L%d, label %%L%d\n", $2->num_reg, $<lab>0.one, $<lab>0.two);
                                         printf("L%i:\n", $<lab>0.one); }
// L'attribut du if est juste avant sur la pile.


if : IF                                { $$ = new_label(0); }


else : ELSE                            { $$ = new_label(0); printf("  br label L%i\n", $$.one); printf("L%i:\n", $<lab>-2.two); }
// L'attribut du if se trouve à trois niveau en dessus, sur la pile,
// en effet, le else apparait sur la pile toujorus trois coups après le if (voir règle du cond)
```

Définition de fonction

```
fun_decl : type fun
         ;

fun : fun_head fun_body


fun_head : ID PO PF                { display_funhead($<t_type>0, $1, list_param_create()); }
         | ID PO param_list PF     { display_funhead($<t_type>0, $1, $3); }
         ;
/* Le type de retour de la fonction est le dernier elem sur la pile */


fun_body : ao { reset_label(); } block af  { printf("}\n") ;}


param_list : type ID vir param_list          { $$ = $4; list_param_add_head($$, $1, $2);   }
           | type ID                         { $$ = list_param_create(); list_param_add_head($$, $1, $2); }
           ;

ao: AO                             { ST_increase_scope(); }


af: AF                             { ST_decrease_scope(); }
```

Appel de fonction

```
fun_app : ID PO args PF                { $$ = mk_type_exp(get_return_type($1)); display_funapp($$, $1, $3); }


args : arglist                         { $$ = $1; }
     |                                 { $$ = list_arg_create(); }
     ;


arglist : exp VIR arglist              { $$ = $3; list_add_head($$, (void *) $1); }
        | exp                          { $$ = list_arg_create(); list_add_head($$, (void *) $1); }
        ;
```