

NPC

Generated by Doxygen 1.9.3

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 ast Struct Reference	5
3.1.1 Detailed Description	5
3.2 ir_gen Struct Reference	5
3.3 ir_gen_result Struct Reference	6
3.3.1 Detailed Description	6
3.4 parser Struct Reference	6
3.5 parser_result Struct Reference	7
3.6 scanner_result Struct Reference	7
3.7 symbol_table Struct Reference	7
3.8 three_address_code Struct Reference	7
3.9 three_address_code_entry Struct Reference	8
3.10 three_address_code_entry_address Struct Reference	8
3.11 token Struct Reference	8
3.12 token_array Struct Reference	9
3.13 typetable Struct Reference	9
3.13.1 Detailed Description	9
3.14 v_table Struct Reference	9
4 File Documentation	11
4.1 /home/max/Npc/Npc/src/ast.h File Reference	11
4.2 ast.h	11
4.3 /home/max/Npc/Npc/src/char_utils.h File Reference	12
4.3.1 Detailed Description	12
4.4 char_utils.h	12
4.5 ir_gen.h	13
4.6 log.h	13
4.7 npc.h	13
4.8 npclib.h	13
4.9 parser.h	13
4.10 scanner.h	14
4.11 symbol_table.h	15
4.12 three_address_code.h	15
4.13 token.h	16
4.14 typetable.h	18
Index	19

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ast	The abstract syntax tree is a tree representation of the source program	5
ir_gen	5
ir_gen_result	Holds the output of the intermediate repr. generation, consists of the corresponding three address code and the	6
parser	6
parser_result	7
scanner_result	7
symbol_table	7
three_address_code	7
three_address_code_entry	8
three_address_code_entry_address	8
token	8
token_array	9
typetable	Contains all types available, consists of a string and the size in bytes	9
v_table	9

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

/home/max/Npc/Npc/src/ ast.h	
Ast contains the type and prototypes for working with abstract syntax trees	11
/home/max/Npc/Npc/src/ char_utils.h	
An utility class for scanning, should be selfexplanatory	12
/home/max/Npc/Npc/src/ ir_gen.h	13
/home/max/Npc/Npc/src/ log.h	13
/home/max/Npc/Npc/src/ npc.h	13
/home/max/Npc/Npc/src/ npclib.h	13
/home/max/Npc/Npc/src/ parser.h	13
/home/max/Npc/Npc/src/ scanner.h	14
/home/max/Npc/Npc/src/ symbol_table.h	15
/home/max/Npc/Npc/src/ three_address_code.h	15
/home/max/Npc/Npc/src/ token.h	16
/home/max/Npc/Npc/src/ typetable.h	18

Chapter 3

Class Documentation

3.1 ast Struct Reference

The abstract syntax tree is a tree representation of the source program.

```
#include <ast.h>
```

Collaboration diagram for ast:

Public Attributes

- [token](#) **n**
- [ast](#) ** **children**
- [ast](#) * **parent**
- long **used**
- long **size**

3.1.1 Detailed Description

The abstract syntax tree is a tree representation of the source program.

The documentation for this struct was generated from the following file:

- [/home/max/Npc/Npc/src/ast.h](#)

3.2 ir_gen Struct Reference

Collaboration diagram for ir_gen:

Public Attributes

- [parser_result](#) **parser_result**
- [v_table](#) **v_table**
- [three_address_code](#) **code**

The documentation for this struct was generated from the following file:

- `/home/max/Npc/Npc/src/ir_gen.h`

3.3 ir_gen_result Struct Reference

Holds the output of the intermediate repr. generation, consists of the corresponding three address code and the.

```
#include <ir_gen.h>
```

Collaboration diagram for `ir_gen_result`:

Public Attributes

- [three_address_code](#) * **code**
- [v_table](#) * **table**

3.3.1 Detailed Description

Holds the output of the intermediate repr. generation, consists of the corresponding three address code and the.

The documentation for this struct was generated from the following file:

- `/home/max/Npc/Npc/src/ir_gen.h`

3.4 parser Struct Reference

Collaboration diagram for `parser`:

Public Attributes

- [ast](#) * **tree**
- [symbol_table](#) * **table**
- [token_array](#) * **arr**
- `size_t` **position**
- `int` **debug**

The documentation for this struct was generated from the following file:

- `/home/max/Npc/Npc/src/parser.h`

3.5 parser_result Struct Reference

Collaboration diagram for parser_result:

Public Attributes

- [ast](#) * **tree**
- [symbol_table](#) * **table**
- [typetable](#) * **type_table**

The documentation for this struct was generated from the following file:

- /home/max/Npc/Npc/src/parser.h

3.6 scanner_result Struct Reference

Collaboration diagram for scanner_result:

Public Attributes

- [token_array](#) * **token_array**
- [symbol_table](#) * **table**

The documentation for this struct was generated from the following file:

- /home/max/Npc/Npc/src/scanner.h

3.7 symbol_table Struct Reference

Public Attributes

- [size_t](#) * **position**
- [size_t](#) * **line**
- [char](#) ** **value**
- [size_t](#) **size**
- [size_t](#) **used**

The documentation for this struct was generated from the following file:

- /home/max/Npc/Npc/src/symbol_table.h

3.8 three_address_code Struct Reference

Collaboration diagram for three_address_code:

Public Attributes

- `size_t` **used**
- `size_t` **size**
- `three_address_code_entry` * **arr**

The documentation for this struct was generated from the following file:

- `/home/max/Npc/Npc/src/three_address_code.h`

3.9 three_address_code_entry Struct Reference

Collaboration diagram for `three_address_code_entry`:

Public Attributes

- `long` **label**
- `three_address_code_op` **operation**
- `three_address_code_entry_address` **result**
- `three_address_code_entry_address` **x**
- `three_address_code_entry_address` **y**

The documentation for this struct was generated from the following file:

- `/home/max/Npc/Npc/src/three_address_code.h`

3.10 three_address_code_entry_address Struct Reference

Public Attributes

- `address_type` **type**
- `long` **value**

The documentation for this struct was generated from the following file:

- `/home/max/Npc/Npc/src/three_address_code.h`

3.11 token Struct Reference

Public Attributes

- `token_type` **type**
- `token_type_class` **type_class**
- `size_t` **position**

The documentation for this struct was generated from the following file:

- `/home/max/Npc/Npc/src/token.h`

3.12 token_array Struct Reference

Collaboration diagram for token_array:

Public Attributes

- size_t **used**
- size_t **size**

The documentation for this struct was generated from the following file:

- /home/max/Npc/Npc/src/token.h

3.13 typetable Struct Reference

Contains all types available, consists of a string and the size in bytes.

```
#include <typetable.h>
```

Public Attributes

- char ** **name**
- size_t * **type_size**
- size_t **used**
- size_t **size**

3.13.1 Detailed Description

Contains all types available, consists of a string and the size in bytes.

The documentation for this struct was generated from the following file:

- /home/max/Npc/Npc/src/typetable.h

3.14 v_table Struct Reference

Public Attributes

- char ** **name**
- size_t **size**
- size_t **used**

The documentation for this struct was generated from the following file:

- /home/max/Npc/Npc/src/ir_gen.h

Chapter 4

File Documentation

4.1 /home/max/Npc/Npc/src/ast.h File Reference

Ast contains the type and prototypes for working with abstract syntax trees.

```
#include "token.h"
#include <stdlib.h>
Include dependency graph for ast.h:
```

4.2 ast.h

[Go to the documentation of this file.](#)

```
1
12 #ifndef AST_H
13 #define AST_H
14
15 #include "token.h"
16 #include <stdlib.h>
17 #define AST_INIT_SIZE 10
18 typedef struct ast ast;
24 struct ast {
25     token n;
26     // Pointer to the array of children
27     ast **children;
28     ast *parent;
29     long used;
30     long size;
31 };
32
33 ast *ast_make();
34
41 void ast_add(ast *parent, ast *tree);
42
43 // get the child at position x
44 ast *ast_get_child(ast *tree, long id);
45
46 // set the value of the ast
47 void ast_set_token(ast *tree, token *n);
48
49 // get the last child
50 ast *ast_get_last(ast *tree);
51
52 // get the parent
53 ast *ast_get_parent(ast *tree);
54
55 ast *ast_get_root(ast *tree);
56
57 void ast_append(ast *tree, token *token);
58
59 #endif
```

4.3 /home/max/Npc/Npc/src/char_utils.h File Reference

An utility class for scanning, should be selfexplanatory.

Functions

- int **is_space** (char *ptr)
- int **is_tab** (char *ptr)
- int **is_whitespace** (char *ptr)
- int **is_newline** (char *ptr)
- int **is_latin** (char *ptr)
- int **is_number** (char *ptr)
- int **is_underscore** (char *ptr)

4.3.1 Detailed Description

An utility class for scanning, should be selfexplanatory.

Author

MaximilianHeim@protonmail.com

Version

0.1

Date

2022-04-27

Copyright

Copyright (c) 2022

4.4 char_utils.h

[Go to the documentation of this file.](#)

```
1
12 #ifndef CHAR_UTILS_H
13 #define CHAR_UTILS_H
14
15 // general purpose library useful for working with strings
16 int is_space(char *ptr);
17 int is_tab(char *ptr);
18 int is_whitespace(char *ptr);
19 int is_newline(char *ptr);
20 int is_latin(char *ptr);
21 int is_number(char *ptr);
22 int is_underscore(char *ptr);
23
24 #endif
```


4.5 ir_gen.h

```

1 #ifndef IR_GEN_H
2 #define IR_GEN_H
3 #define V_TABLE_INIT_SIZE 10
4 #define MAXIMUM_LABEL 9223372036854775807
5 #define UNDEFINED -1
6 #include "parser.h"
7 #include "three_address_code.h"
8 typedef struct v_table {
9     char **name;
10    size_t size;
11    size_t used;
12 } v_table;
13
14 typedef struct ir_gen {
15     parser_result parser_result;
16     v_table v_table;
17     three_address_code code;
18 } ir_gen;
19
20 typedef struct ir_gen_result {
21     three_address_code *code;
22     v_table *table;
23 } ir_gen_result;
24
25 v_table *v_table_make();
26 size_t v_table_add(v_table *table, const char *str);
27 const char *v_table_get(v_table *table, size_t id);
28 ir_gen_result generate(parser_result parser_out);
29
30 three_address_code_op get_op(token_type type);
31 long new_var();
32 #endif

```

4.6 log.h

```

1 #ifndef LOG_H
2 #define LOG_H
3 typedef enum log_level {
4     log_info,
5     log_warning,
6     log_error,
7     log_bad,
8     log_debug,
9     log_intern
10 } log_level;
11 int npc_log(log_level level, const char *message);
12 int npc_debug_log(int is_debug, const char *message);
13 #endif

```

4.7 npc.h

```

1 #ifndef npc_H
2 #define npc_H
3 #include <stdio.h>
4
5 char *read_program(FILE *fp);
6
7 #endif

```

4.8 npclib.h

```

1 #ifndef NPCLIB_H
2 #define NPCLIB_H
3 #endif

```

4.9 parser.h

```

1 #ifndef PARSER_H

```

```

2 #define PARSER_H
3
4 #include "ast.h"
5 #include "token.h"
6 #include "scanner.h"
7 #include "symbol_table.h"
8 #include "typetable.h"
9
10 typedef struct parser_result {
11     ast *tree;
12     symbol_table *table;
13     typetable *type_table;
14 } parser_result;
15
16 typedef struct parser {
17     ast *tree;
18     symbol_table *table;
19     token_array *arr;
20     size_t position;
21     int debug;
22 } parser;
23
24 parser_result *parse_program(scanner_result res, int debug);
25 parser_result *parser_result_make(ast *tree, symbol_table *table,
26                                   typetable *type_table);
27 parser *parser_make(ast *tree, symbol_table *table, int debug, token_array *arr);
28
29 void parse_syntax_err(parser *parser, char *err);
30
31 void program(parser *parser);
32
33 void parameter_list(parser *parser);
34
35 void function(parser *parser);
36
37 void program_directive(parser *parser);
38
39 void secondary_directive_list(parser *parser);
40
41 void match(parser *parser, token_type type);
42
43 void match_no_append(parser *parser, token_type type);
44
45 void type(parser *parser);
46
47 void functions(parser *parser);
48
49 void match_by_class(parser *parser, token_type_class type);
50
51 void match_by_class_no_append(parser *parser, token_type_class type);
52
53 void include_directive_select(parser *parser);
54
55 void var(parser *parser);
56
57 void print_tree(ast *tree, int depth);
58
59 void declaration(parser *parser);
60
61 void factor(parser *parser);
62 void expression(parser *parser);
63 void simple_expression(parser *parser);
64
65 void block(parser *parser);
66
67 void argument_list(parser *parser);
68
69 void term(parser *parser);
70 void fun_call(parser *parser);
71
72 void return_statement(parser *parser);
73
74 void for_statement(parser *parser);
75
76 void secondary_directives(parser *parser);
77
78 #endif

```

4.10 scanner.h

```

1 #ifndef SCANNER_H
2 #define SCANNER_H
3 #include "token.h"

```

```

4 #include "symbol_table.h"
5 typedef struct {
6     token_array *token_array;
7     symbol_table *table;
8 } scanner_result;
9
10 void lexing_error(size_t position, size_t line, char *code, size_t length);
11
12 scanner_result lex(char *code, int debug, int export_symbol);
13
14 #endif

```

4.11 symbol_table.h

```

1 #ifndef SYMBOL_TABLE_H
2 #define SYMBOL_TABLE_H
3 #define SYMBOL_TABLE_INIT_SIZE 10
4 #include <stdlib.h>
5 #include <stdio.h>
6 typedef struct {
7     size_t *position;
8     size_t *line;
9     char **value;
10    size_t size;
11    size_t used;
12 } symbol_table;
13
14 symbol_table *symbol_table_make();
15
16 void symbol_table_add(symbol_table *table, size_t position, size_t line,
17                      char *value, size_t val_len);
18
19 long symbol_table_get_position(symbol_table *table, size_t id);
20
21 long symbol_table_get_line(symbol_table *table, size_t id);
22
23 char *symbol_table_get_value(symbol_table *table, size_t id);
24 void write_symbol_table(FILE * file, symbol_table *table);
25 #endif

```

4.12 three_address_code.h

```

1 #ifndef THREE_ADDRESS_CODE_H
2 #define THREE_ADDRESS_CODE_H
3 #define THREE_ADDRESS_CODE_INIT_SIZE 10
4 #include <stdlib.h>
5
6 typedef enum three_address_code_op {
7     // Jumps
8     unconditional_jump,
9     conditional_jump,
10    conditional_jump_inversed,
11
12    // copying
13    copy_op,
14    mem_copy_op,
15    indexed_copy_op,
16
17    // unary
18    minus_op,
19    log_negation_op,
20    conversion_op,
21    inc_op,
22    dec_op,
23    // binary_instructions
24
25    add_op,
26    subtract_op,
27    multiply_op,
28    divide_op,
29    modulo_op,
30    pot_op,
31
32    // relop
33    gt_op,
34    le_op,
35    lt_op,
36    ge_op,
37    eq_op,

```

```

38     ne_op,
39
40     funheader
41 } three_address_code_op;
42
43 typedef enum {
44     address_int_literal_token,
45     address_float_literal_token,
46     address_char_literal_token,
47     address_string_literal_token,
48     address_variable,
49     address_function,
50     address_temporary,
51     address_undefined,
52     address_memory,
53     address_size
54 } address_type;
55
56 typedef struct three_address_code_entry_address {
57     address_type type;
58     long value;
59 } three_address_code_entry_address;
60
61 typedef struct three_address_code_entry {
62     long label;
63     three_address_code_op operation;
64     three_address_code_entry_address result;
65     three_address_code_entry_address x;
66     three_address_code_entry_address y;
67 } three_address_code_entry;
68
69 typedef struct three_address_code {
70     size_t used;
71     size_t size;
72     three_address_code_entry *arr;
73 } three_address_code;
74
75 three_address_code *three_address_code_make();
76
77 void three_address_code_add(three_address_code *code, long label,
78                             three_address_code_op op, address_type x_type,
79                             long x, address_type y_type, long y,
80                             address_type res_type, long res);
81
82 #endif

```

4.13 token.h

```

1  #ifndef token_H
2  #define token_H
3  #define token_array_INIT_SIZE 10
4  #include <stdlib.h>
5
12 typedef enum token_type {
13     identifier_token,
14     assignment_token,
15
17     /* operators += /= *= -= */
18     imm_minus_operator_token,
19     imm_plus_operator_token,
20     imm_mul_operator_token,
21     imm_division_operator_token,
22
23     selector_token,
24     semicolon_token,
25     colon_token,
26     comma_token,
27
28
29     // DIRECTIVES
30     program_directive_token,
31     end_directive_token,
32     module_directive_token,
33     include_directive_token,
34     macro_directive_token,
35
36     // BINARY OPERATORS
37     plus_operator_token,
38     minus_operator_token,
39     multiplication_operator_token,
40     division_operator_token,
41     mod_operator_token,
42     pot_operator_token,
43     gt_operator_token,

```

```

44     lt_operator_token,
45     le_operator_token,
46     ge_operator_token,
47     floor_div_operator_token,
48
49     // UNARY OPERATORS
50     increment_operator_token,
51     not_token,
52     decrement_operator_token,
53
54     // STRUCTURE //
55     opening_bracket_token,
56     closing_bracket_token,
57
58     opening_s_bracket_token,
59     closing_s_bracket_token,
60
61     opening_c_bracket_token,
62     closing_c_bracket_token,
63
64     // Literals
65     string_literal_token,
66     char_literal_token,
67     int_literal_token,
68     float_literal_token,
69
70     // types
71     string_type_token,
72     char_type_token,
73     int_type_token,
74     float_type_token,
75     long_type_token,
76
77     return_keyword_token,
78     for_keyword_token,
79     while_keyword_token,
80     if_keyword_token,
81     else_keyword_token,
82     elif_keyword_token,
83     function_token,
84
85     // Ntm
86     if_statement_n,
87     return_statement_n,
88     for_statement_n,
89     expression_n,
90     factor_n,
91     term_n,
92     program_n,
93     declaration_n,
94     functioncall_n,
95     argument_n,
96     argument_list_n,
97     block_n,
98     var_n,
99     module_n,
100    secondarydirective_n,
101    secondarydirective_list_n,
102    include_directive_n,
103    include_directive_subselect_n,
104    program_directive_n,
105    module_directive_n,
106    macro_directive_n,
107    function_n,
108    functions_n,
109    unop_n,
110    binop_n,
111    parameter_n,
112    parameter_list_n,
113    type_n,
114    statement_n,
115    simple_expression_n,
116    function_call_n
117
118 } token_type;
119
120 typedef enum token_type_class {
121     unop_c,
122     binop_c,
123     assign_c,
124     sec_directive_c,
125     prim_directive_c,
126     literal_c,
127     type_c,
128     nont_c,
129     keyword_c,
130     nac_c,

```

```

131     bracket_c,
132     punctuation_c,
133     directive_c,
134     relop_c
135 } token_type_class;
136 } token_type_class;
137
138 typedef struct token {
139     token_type type;
140     token_type_class type_class;
141     size_t position;
142 } token;
143
144 typedef struct token_array {
145     token *token_array;
146     size_t used;
147     size_t size;
148 } token_array;
149
150 token_array *token_array_make();
151 void token_array_add(token_array *arr, token_type type,
152                     token_type_class type_class, size_t position);
153
154 token_type token_array_get_token_type(token_array *arr, size_t index);
155 token_type_class token_array_get_token_type_class(token_array *arr,
156                                                  size_t index);
157
158 token *token_array_get_token(token_array *arr, size_t index);
159
160 void print_tokens(token_array *arr);
161
162 token *token_make(token_type type, token_type_class type_class,
163                  size_t position);
164
165 char *token_type_get_canonical(token_type type);
166
167 char *token_type_get_class(token_type_class type);
168
169 #endif

```

4.14 typetable.h

```

1  #ifndef TYPETABLE_H
2  #define TYPETABLE_H
3  #define TYPETABLE_INIT_SIZE 10
4
5  #include <stdlib.h>
6
12 typedef struct typetable {
13     char **name;
14     size_t *type_size;
15     size_t used;
16     size_t size;
17 } typetable;
18
19 size_t typetable_get_size(typetable *table, size_t id);
20
21 typetable *typetable_make();
22
23 void typetable_add(typetable *table, char *name, size_t size);
24
25 int typetable_exists(typetable *table, char *name);
26
27 #endif

```

Index

[/home/max/Npc/Npc/src/ast.h](#), [11](#)
[/home/max/Npc/Npc/src/char_utils.h](#), [12](#)
[/home/max/Npc/Npc/src/ir_gen.h](#), [13](#)
[/home/max/Npc/Npc/src/log.h](#), [13](#)
[/home/max/Npc/Npc/src/npc.h](#), [13](#)
[/home/max/Npc/Npc/src/npclib.h](#), [13](#)
[/home/max/Npc/Npc/src/parser.h](#), [13](#)
[/home/max/Npc/Npc/src/scanner.h](#), [14](#)
[/home/max/Npc/Npc/src/symbol_table.h](#), [15](#)
[/home/max/Npc/Npc/src/three_address_code.h](#), [15](#)
[/home/max/Npc/Npc/src/token.h](#), [16](#)
[/home/max/Npc/Npc/src/typetable.h](#), [18](#)

[ast](#), [5](#)

[ir_gen](#), [5](#)

[ir_gen_result](#), [6](#)

[parser](#), [6](#)

[parser_result](#), [7](#)

[scanner_result](#), [7](#)

[symbol_table](#), [7](#)

[three_address_code](#), [7](#)

[three_address_code_entry](#), [8](#)

[three_address_code_entry_address](#), [8](#)

[token](#), [8](#)

[token_array](#), [9](#)

[typetable](#), [9](#)

[v_table](#), [9](#)