## 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
	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
	15
	16
	18
Index	19

# **Class Index**

## 1.1 Class List

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

st	
The abstract syntax tree is a tree representation of the source program	5
_gen	5
_gen_result	
Holds the output of the intermediate repr. generation, consists of the corresponding three ad-	
dress code and the	6
arser	6
arser_result	7
canner_result	7
ymbol_table	7
rree_address_code	7
rree_address_code_entry	8
nree_address_code_entry_address	8
oken	8
oken_array	9
petable	
Contains all types available, consists of a string and the size in bytes	9
table	9

2 Class Index

# 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/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/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

File Index

## **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:

6 Class Documentation

#### **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:

8 Class Documentation

#### **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

10 Class Documentation

## **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.

```
12 #ifndef AST_H
13 #define AST_H
15 #include "token.h"
16 #include <stdlib.h>
17 #define AST_INIT_SIZE 10
18 typedef struct ast ast;
24 struct ast {
token n;
token n;

// Pointer to the array of children
ast **children;
ast **parent;
long used;
30
        long size;
31 };
33 ast *ast_make();
41 void ast_add(ast *parent, ast *tree);
43 // get the child at position x
44 ast *ast_get_child(ast *tree, long id);
46 // set the value of the ast
47 void ast_set_token(ast *tree, token *n);
49 // get the last child
50 ast *ast_get_last(ast *tree);
51
52 // get the parent
53 ast *ast_get_parent(ast *tree);
55 ast *ast_get_root(ast *tree);
57 void ast_append(ast *tree, token *token);
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 13

## 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 {
      char **name;
      size_t size;
size_t used;
10
11
12 } v_table;
14 typedef struct ir_gen {
     parser_result parser_result;
1.5
        v table v table;
16
        three_address_code code;
18 } ir_gen;
19
23 typedef struct ir_gen_result {
2.4
       three_address_code *code;
25
        v_table *table;
26 } ir_gen_result;
28 v_table *v_table_make();
29 size_t v_table_add(v_table *table, const char *str);
30 const char *v_table_get(v_table *table, size_t id);
31 ir_gen_result generate(parser_result parser_out);
33 three_address_code_op get_op(token_type type);
34 long new_var();
35 #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
4 #include "ast.h"
5 #include "token.h"
6 #include "scanner.h"
7 #include "symbol_table.h"
8 #include "typetable.h"
10 typedef struct parser_result {
     ast *tree;
symbol_table *table;
typetable *type_table;
11
12
13
14 } parser_result;
15
16 typedef struct parser {
     ast *tree;
17
18
       symbol_table *table;
19
       token array *arr;
20
      size_t position;
       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);
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);
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);
53 void include_directive_select(parser *parser);
54
55 void var(parser *parser);
56
57 void print tree(ast *tree, int depth);
59 void declaration(parser *parser);
60
61 void factor(parser *parser);
62 void expression(parser *parser);
63 void simple_expression(parser *parser);
65 void block(parser *parser);
66
67 void argument_list(parser *parser);
68
69 void term(parser *parser);
70 void fun_call(parser *parser);
72 void return_statement(parser *parser);
73
74 void for_statement(parser *parser);
75
76 void secondary directives (parser *parser);
78 #endif
```

#### 4.10 scanner.h

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

4.11 symbol\_table.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 {
      size_t *position;
size_t *line;
9
       char **value;
1.0
      size_t size;
11 size_t used;
12 } symbol_table;
14 symbol_table *symbol_table_make();
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);
21 long symbol_table_get_line(symbol_table *table, size_t id);
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>
6 typedef enum three_address_code_op {
       // Jumps
       unconditional_jump,
8
       conditional_jump,
  conditional_jump_inversed,
10
12
       // copying
13
14
       mem_copy_op,
15
       indexed_copy_op,
16
        // unary
17
18
        minus_op,
19
        log_negation_op,
2.0
        conversion_op,
21
        inc_op,
22
       dec_op,
       // binary_instructions
24
        add_op,
25
26
        subtract_op,
2.7
        multiply_op,
28
        divide_op,
29
       modulo_op,
       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 {
       address_int_literal_token,
44
45
       address_float_literal_token,
46
       address_char_literal_token,
47
       address_string_literal_token,
48
       address_variable, address_function,
49
       address_temporary,
50
51
       address_undefined,
52
       address_memory,
53
       address_size
54 } address_type;
55
56 typedef struct three_address_code_entry_address {
       address_type type;
       long value;
59 } three_address_code_entry_address;
60
61 typedef struct three_address_code_entry {
       long label;
62
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();
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);
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>
12 typedef enum token_type {
13
      identifier_token,
14
       assignment_token,
15
       /* operators += /= *= -= */
18
       imm_minus_operator_token,
19
       imm_plus_operator_token,
20
       imm_mul_operator_token,
21
       imm_division_operator_token,
23
24
       selector_token,
25
       semicolon_token,
26
       colon_token,
27
       comma_token,
2.8
       // DIRECTIVES
29
30
       program_directive_token,
31
       end_directive_token,
32
       module_directive_token,
33
       include_directive_token,
34
       macro_directive_token,
35
       // BINARY OPERATORS
36
       plus_operator_token,
38
       minus_operator_token,
39
       multiplication_operator_token,
40
       division_operator_token,
41
       mod_operator_token,
42
       pot operator token,
       gt_operator_token,
```

4.13 token.h 17

```
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
       // STRUCTURE //
54
       opening_bracket_token,
55
56
       closing_bracket_token,
57
58
       opening_s_bracket_token,
59
       closing_s_bracket_token,
60
       opening_c_bracket_token,
61
       closing_c_bracket_token,
62
63
64
       // Literals
65
       string_literal_token,
66
       char_literal_token,
67
       int literal token,
68
       float_literal_token,
69
       // types
70
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,
       else_keyword_token,
81
       elif_keyword_token,
83
       function_token,
84
8.5
       // Ntm
       if_statement_n,
86
87
       return_statement_n,
       for_statement_n,
88
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
        {\tt 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;
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
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,
                                                             size_t index);
156
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,
                        size_t position);
164
165 char *token_type_get_canonial(token_type type);
166
167 char *token_type_get_class(token_type_class type);
169 #endif
```

## 4.14 typetable.h

```
1 #ifndef TYPETABLE H
2 #define TYPETABLE_H
3 #define TYPETABLE_INIT_SIZE 10
5 #include <stdlib.h>
12 typedef struct typetable {
     char **name;
13
      size_t *type_size;
14
      size_t used;
15
17 } typetable;
18
19 size_t typetable_get_size(typetable *table, size_t id);
20
21 typetable *typetable_make();
23 void typetable_add(typetable *table, char *name, size_t size);
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
```