NPC

Generated by Doxygen 1.9.3

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

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.

```

#ifndef AST_H
#define AST_H

#include "token.h"
#include <stdlib.h>
#define AST_INIT_SIZE 10
typedef struct ast ast;
struct ast {
    token n;
    // Pointer to the array of children
    ast **children;
    ast *parent;
    long used;
    long size;
};

ast *ast_make();

void ast_add(ast *parent, ast *tree);

// get the child at position x
ast *ast_get_child(ast *tree, long id);

// set the value of the ast
void ast_set_token(ast *tree, token *n);

// get the last child
ast *ast_get_last(ast *tree);

// get the parent
ast *ast_get_parent(ast *tree);

ast *ast_get_root(ast *tree);

void ast_append(ast *tree, token *token);

#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.

```
#ifndef CHAR_UTILS_H
#define CHAR_UTILS_H

// general purpose library useful for working with strings
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);

#endif
```

4.5 ir_gen.h

```c
#ifndef IR_GEN_H
#define IR_GEN_H
#define V_TABLE_INIT_SIZE 10
#define MAXIMUM_LABEL 9223372036854775807
#define UNDEFINED -1
#include "parser.h"
#include "three_address_code.h"
typedef struct v_table {
    char **name;
     size_t size;
     size_t used;
} v_table;

typedef struct ir_gen {
     parser_result parser_result;
     v_table v_table;
     three_address_code code;
} ir_gen;

typedef struct ir_gen_result {
     three_address_code *code;
     v_table *table;
} ir_gen_result;

v_table *v_table_make();
size_t v_table_add(v_table *table, const char *str);
const char *v_table_get(v_table *table, size_t id);
ir_gen_result generate(parser_result parser_out);

three_address_code_op get_op(token_type type);
long new_var();
#endif
```

4.6 log.h

```c
#ifndef LOG_H
#define LOG_H
typedef enum log_level {
    log_info,
    log_warning,
    log_error,
    log_bad,
    log_debug,
    log_intern
} log_level;
int npc_log(log_level level, const char *message);
int npc_debug_log(int is_debug, const char *message);
#endif
```

4.7 npc.h

```c
#ifndef npc_H
#define npc_H
#include <stdio.h>

char *read_program(FILE *fp);

#endif
```

4.8 npclib.h

```c
#ifndef NPCLIB_H
#define NPCLIB_H
#endif
```

4.9 parser.h

```c
#ifndef PARSER_H
```

```
#define PARSER_H

#include "ast.h"
#include "token.h"
#include "scanner.h"
#include "symbol_table.h"
#include "typetable.h"

typedef struct parser_result {
    ast *tree;
    symbol_table *table;
    typetable *type_table;
} parser_result;

typedef struct parser {
    ast *tree;
    symbol_table *table;
    token_array *arr;
    size_t position;
    int debug;
} parser;

parser_result *parse_program(scanner_result res, int debug);
parser_result *parser_result_make(ast *tree, symbol_table *table,
                                  typetable *type_table);
parser *parser_make(ast *tree, symbol_table *table, int debug, token_array *arr);

void parse_syntax_err(parser *parser, char *err);

void program(parser *parser);

void parameter_list(parser *parser);

void function(parser *parser);

void program_directive(parser *parser);

void secondary_directive_list(parser *parser);

void match(parser *parser, token_type type);

void match_no_append(parser *parser, token_type type);

void type(parser *parser);

void functions(parser *parser);

void match_by_class(parser *parser, token_type_class type);

void match_by_class_no_append(parser *parser, token_type_class type);

void include_directive_select(parser *parser);

void var(parser *parser);

void print_tree(ast *tree, int depth);

void declaration(parser *parser);

void factor(parser *parser);
void expression(parser *parser);
void simple_expression(parser *parser);

void block(parser *parser);

void argument_list(parser *parser);

void term(parser *parser);
void fun_call(parser *parser);

void return_statement(parser *parser);

void for_statement(parser *parser);

void secondary_directives(parser *parser);

#endif
```

4.10 scanner.h

```
#ifndef SCANNER_H
#define SCANNER_H
#include "token.h"
```

```
#include "symbol_table.h"
typedef struct {
    token_array *token_array;
    symbol_table *table;
} scanner_result;

void lexing_error(size_t position, size_t line, char *code, size_t length);

scanner_result lex(char *code, int debug, int export_symbol);

#endif
```

4.11 symbol_table.h

```
#ifndef SYMBOL_TABLE_H
#define SYMBOL_TABLE_H
#define SYMBOL_TABLE_INIT_SIZE 10
#include <stdlib.h>
#include <stdio.h>
typedef struct {
    size_t *position;
    size_t *line;
    char **value;
     size_t size;
     size_t used;
} symbol_table;

symbol_table *symbol_table_make();

void symbol_table_add(symbol_table *table, size_t position, size_t line,
                      char *value, size_t val_len);

long symbol_table_get_position(symbol_table *table, size_t id);

long symbol_table_get_line(symbol_table *table, size_t id);

char *symbol_table_get_value(symbol_table *table, size_t id);
void write_symbol_table(FILE * file, symbol_table *table);
#endif
```

4.12 three_address_code.h

```
#ifndef THREE_ADDRESS_CODE_H
#define THREE_ADDRESS_CODE_H
#define THREE_ADDRESS_CODE_INIT_SIZE 10
#include <stdlib.h>

typedef enum three_address_code_op {
    // Jumps
    unconditional_jump,
    conditional_jump,
     conditional_jump_inversed,

     // copying
     copy_op,
     mem_copy_op,
     indexed_copy_op,

     // unary
     minus_op,
     log_negation_op,
     conversion_op,
     inc_op,
     dec_op,
     // binary_instructions

     add_op,
     subtract_op,
     multiply_op,
     divide_op,
     modulo_op,
     pot_op,

     // relop
     gt_op,
     le_op,
     lt_op,
     ge_op,
     eq_op,
    ne_op,

    funheader
} three_address_code_op;

typedef enum {
    address_int_literal_token,
    address_float_literal_token,
    address_char_literal_token,
    address_string_literal_token,
    address_variable,
    address_function,
    address_temporary,
    address_undefined,
    address_memory,
    address_size
} address_type;

typedef struct three_address_code_entry_address {
    address_type type;
    long value;
} three_address_code_entry_address;

typedef struct three_address_code_entry {
    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;
} three_address_code_entry;

typedef struct three_address_code {
    size_t used;
    size_t size;
    three_address_code_entry *arr;
} three_address_code;

three_address_code *three_address_code_make();

void three_address_code_add(three_address_code *code, long label,
                            three_address_code_op op, address_type x_type,
                            long x, address_type y_type, long y,
                            address_type res_type, long res);

#endif
```

4.13 token.h

```
#ifndef token_H
#define token_H
#define token_array_INIT_SIZE 10
#include <stdlib.h>

typedef enum token_type {
    identifier_token,
    assignment_token,

    /* operators += /= *= -= */
    imm_minus_operator_token,
    imm_plus_operator_token,
    imm_mul_operator_token,
    imm_division_operator_token,

    selector_token,
    semicolon_token,
    colon_token,
    comma_token,

    // DIRECTIVES
    program_directive_token,
    end_directive_token,
    module_directive_token,
    include_directive_token,
    macro_directive_token,

    // BINARY OPERATORS
    plus_operator_token,
    minus_operator_token,
    multiplication_operator_token,
    division_operator_token,
    mod_operator_token,
    pot_operator_token,
    gt_operator_token,
```

```c
    lt_operator_token,
    le_operator_token,
    ge_operator_token,
    floor_div_operator_token,

    // UNARY OPERATORS
    increment_operator_token,
    not_token,
    decrement_operator_token,

    // STRUCTURE //
    opening_bracket_token,
    closing_bracket_token,

    opening_s_bracket_token,
    closing_s_bracket_token,

    opening_c_bracket_token,
    closing_c_bracket_token,

    // Literals
    string_literal_token,
    char_literal_token,
    int_literal_token,
    float_literal_token,

    // types
    string_type_token,
    char_type_token,
    int_type_token,
    float_type_token,
    long_type_token,

    return_keyword_token,
    for_keyword_token,
    while_keyword_token,
    if_keyword_token,
    else_keyword_token,
    elif_keyword_token,
    function_token,

    // Ntm
    if_statement_n,
    return_statement_n,
    for_statement_n,
    expression_n,
    factor_n,
    term_n,
    program_n,
    declaration_n,
    functioncall_n,
    argument_n,
    argument_list_n,
    block_n,
    var_n,
    module_n,
    secondarydirective_n,
    secondarydirective_list_n,
    include_directive_n,
    include_directive_subselect_n,
    program_directive_n,
    module_directive_n,
    macro_directive_n,
    function_n,
    functions_n,
    unop_n,
    binop_n,
    parameter_n,
    parameter_list_n,
    type_n,
    statement_n,
    simple_expression_n,
    function_call_n

} token_type;

typedef enum token_type_class {
    unop_c,
    binop_c,
    assign_c,
    sec_directive_c,
    prim_directive_c,
    literal_c,
    type_c,
    nont_c,
    keyword_c,
    nac_c,
    bracket_c,
    punctuation_c,
    directive_c,
    relop_c

} token_type_class;

typedef struct token {
    token_type type;
    token_type_class type_class;
    size_t position;
} token;

typedef struct token_array {
    token *token_array;
    size_t used;
    size_t size;
} token_array;

token_array *token_array_make();
void token_array_add(token_array *arr, token_type type,
                     token_type_class type_class, size_t position);

token_type token_array_get_token_type(token_array *arr, size_t index);
token_type_class token_array_get_token_type_class(token_array *arr,
                                                   size_t index);

token *token_array_get_token(token_array *arr, size_t index);

void print_tokens(token_array *arr);

token *token_make(token_type type, token_type_class type_class,
                  size_t position);

char *token_type_get_canonial(token_type type);

char *token_type_get_class(token_type_class type);

#endif
```

4.14 typetable.h

```c
#ifndef TYPETABLE_H
#define TYPETABLE_H
#define TYPETABLE_INIT_SIZE 10

#include <stdlib.h>

typedef struct typetable {
    char **name;
    size_t *type_size;
    size_t used;
    size_t size;
} typetable;

size_t typetable_get_size(typetable *table, size_t id);

typetable *typetable_make();

void typetable_add(typetable *table, char *name, size_t size);

int typetable_exists(typetable *table, char *name);

#endif
```

Index