

NPC

Generated by Doxygen 1.8.17

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_result Struct Reference	6
3.3 node Struct Reference	6
3.4 node_array Struct Reference	7
3.5 parser_result Struct Reference	7
3.6 scanner_result Struct Reference	8
3.7 symbol_table Struct Reference	9
3.8 three_address_code Struct Reference	9
3.9 three_address_code_entry Struct Reference	10
3.10 three_address_code_entry_address Struct Reference	10
3.11 typetable Struct Reference	11
3.12 v_table Struct Reference	11
4 File Documentation	13
4.1 /home/max/Npc/src/ast.h File Reference	13
4.1.1 Detailed Description	15
4.1.2 Function Documentation	15
4.1.2.1 ast_add()	15
4.2 /home/max/Npc/src/char_utils.h File Reference	15
4.2.1 Detailed Description	16
Index	17

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_result	6
node	6
node_array	7
parser_result	7
scanner_result	8
symbol_table	9
three_address_code	9
three_address_code_entry	10
three_address_code_entry_address	10
typetable	11
v_table	11

Chapter 2

File Index

2.1 File List

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

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

Chapter 3

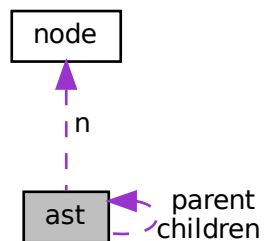
Class Documentation

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

- `node n`
- `ast ** children`
- `ast * parent`
- long `used`
- long `size`

3.1.1 Detailed Description

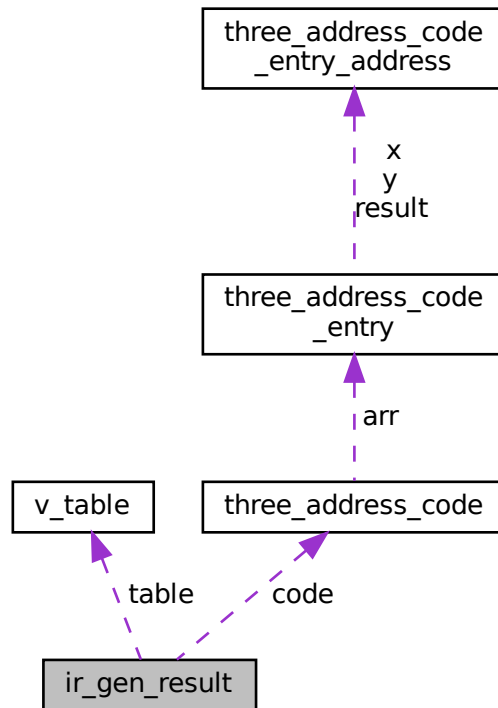
The abstract syntax tree is a tree representation of the source program.

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

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

3.2 ir_gen_result Struct Reference

Collaboration diagram for ir_gen_result:



Public Attributes

- `three_address_code` * **code**
- `v_table` * **table**

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

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

3.3 node Struct Reference

Public Attributes

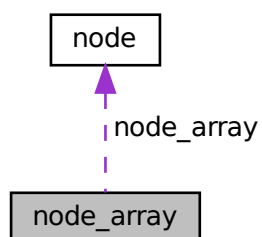
- `node_type` **type**
- `node_type_class` **type_class**
- `long` **value**

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

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

3.4 node_array Struct Reference

Collaboration diagram for node_array:



Public Attributes

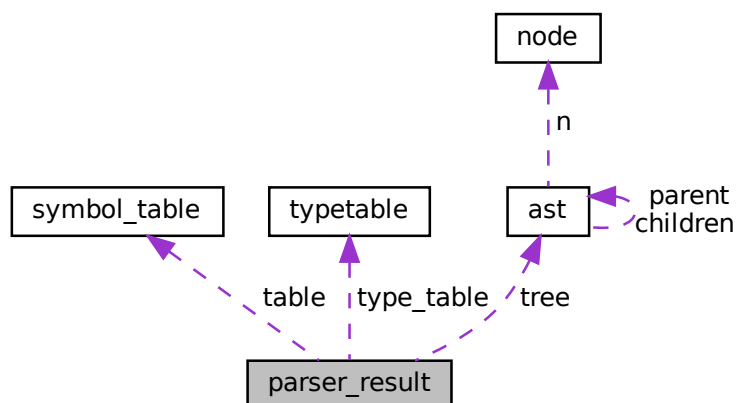
- long **used**
- long **size**

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

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

3.5 parser_result Struct Reference

Collaboration diagram for parser_result:



Public Attributes

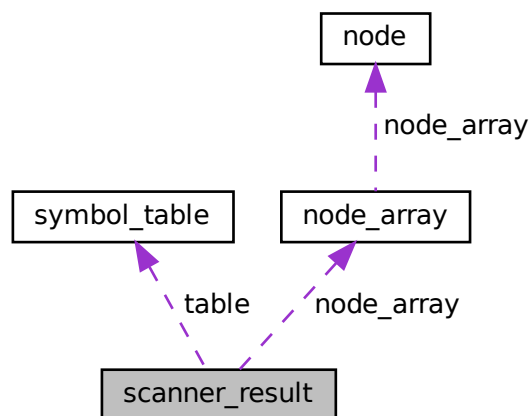
- [ast](#) * **tree**
- [symbol_table](#) * **table**
- [typetable](#) * **type_table**

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

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

3.6 scanner_result Struct Reference

Collaboration diagram for scanner_result:



Public Attributes

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

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

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

3.7 symbol_table Struct Reference

Public Attributes

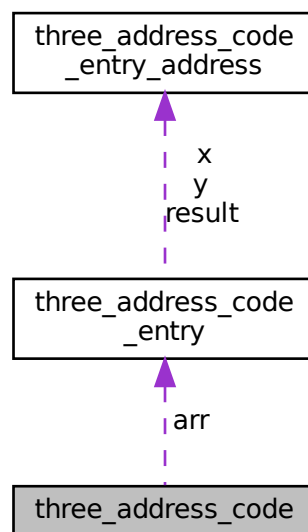
- `size_t * position`
- `size_t * line`
- `char ** value`
- `size_t size`
- `size_t used`

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

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

3.8 three_address_code Struct Reference

Collaboration diagram for `three_address_code`:



Public Attributes

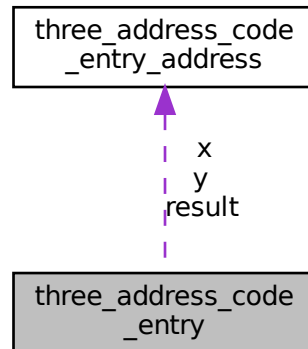
- `size_t used`
- `size_t size`
- `three_address_code_entry * arr`

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

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

3.11 `typetable` Struct Reference

Public Attributes

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

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

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

3.12 `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/src/ir_gen.h`

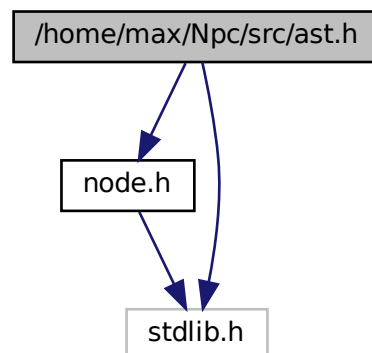
Chapter 4

File Documentation

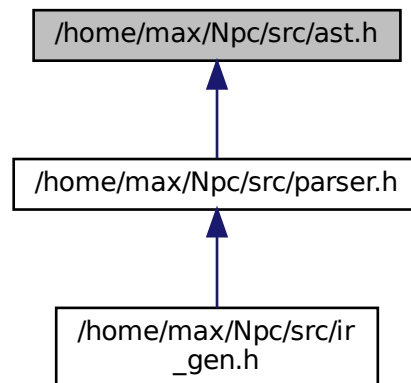
4.1 /home/max/Npc/src/ast.h File Reference

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

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



This graph shows which files directly or indirectly include this file:



Classes

- struct [ast](#)

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

Macros

- `#define AST_INIT_SIZE 10`

Typedefs

- `typedef struct ast ast`

Functions

- `ast * ast_make ()`
- `void ast_add (ast *parent, ast *tree)`
- `ast * ast_get_child (ast *tree, long id)`
- `void ast_set_node (ast *tree, node *n)`
- `ast * ast_get_last (ast *tree)`
- `ast * ast_get_parent (ast *tree)`

4.1.1 Detailed Description

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

Author

MaximilianHeim@protonmail.com

Version

0.1

Date

2022-04-26

Copyright

Copyright (c) 2022

4.1.2 Function Documentation

4.1.2.1 ast_add()

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

Parameters

<i>parent</i>	
<i>tree</i>	

4.2 /home/max/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)

4.2.1 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

Index

/home/max/Npc/src/ast.h, [13](#)

/home/max/Npc/src/char_utils.h, [15](#)

ast, [5](#)

ast.h

 ast_add, [15](#)

ast_add

 ast.h, [15](#)

ir_gen_result, [6](#)

node, [6](#)

node_array, [7](#)

parser_result, [7](#)

scanner_result, [8](#)

symbol_table, [9](#)

three_address_code, [9](#)

three_address_code_entry, [10](#)

three_address_code_entry_address, [10](#)

typetable, [11](#)

v_table, [11](#)