iteration 3

Generated by Doxygen 1.8.11

Contents

Chapter 1

INDEX PAGE

1.1 Introduction

1.1.1 Team Name

Nostromo

1.1.2 TA Name

John Harwell

1.1.3 Discussion Section

02

1.1.4 Details

This document contains the details of different classes in scanner, parser and ast namespaces under the fcal namespace. Each class in this project can be viewed in this documentation, also the relationship between the classes has been shown. The intention of this project is to compile a file written in FCAL language and convert it to C/C++ code. In the present iteration, the goal is to create an AST and since the project is yet to be complete the documentation covers everything upto the present iteration which is the creation of the AST. All methods of classes with their arguements and details of some important methods with their description is included.This documentation should provide the reader with brief information regarding the work done so far in this project.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 fcal Namespace Reference

Namespaces.

Namespaces

- ast
- parser
- scanner

6.1.1 Detailed Description

Namespaces.

6.2 fcal::ast Namespace Reference

Classes

- class binaryExpr
- class boolExpr
- class bracketStmt
- class constantExpr
- class Decl
- class declStmt
- class emptyStmts
- class equalsStmt
- class Expr
- class ifElseStmt
- class ifExpr
- class ifStmt
- class letExpr
- class matrixDecl

- class matrixExpr
- class nestedOrExpr
- class Node
- class notExpr
- class parenthesisExpr
- class printStmt
- class repeatStmt
- class Root
- class semiColonStmt
- class seqStmts
- class Stmt
- class Stmts
- class varDecl
- class varName

  This class holds the lexeme details of a variable name.
- class whileStmt

Enumerations

- enum decType { int_, float_, string_, boolean_ }

6.2.1 Enumeration Type Documentation

6.2.1.1 enum fcal::ast::decType

decType has four possible values based on declaration types.

Enumerator

int_

float_

string_

boolean_

6.3 fcal::parser Namespace Reference

Classes

- class Parser
- class ParseResult

6.4 fcal::scanner Namespace Reference

Classes

- class CharConstToken

  Char Const.
- class DashToken

  Dash.
- class EndOfFileToken

  End of File.
- class ExtToken
- class FalseKwdToken

  False Kwd.
- class FloatConstToken

  Float Const.
- class ForwardSlashToken

  ForwardSlash.
- class IfToken
- class IntConstToken

  Int Const.
- class LeftParenToken

  Left Paren.
- class LetToken
- class NotOpToken
- class PlusSignToken

  Plus Sign.
- class RelationalOpToken

  Relational Op.
- class Scanner
- class StarToken

  Star.
- class StringConstToken

  String Const.
- class Token
- class TrueKwdToken

  True Kwd.
- class VariableNameToken

  Variable Name.

Typedefs

- typedef enum kTokenEnumType TokenType

Enumerations

- enum kTokenEnumType {
  kIntKwd, kFloatKwd, kBoolKwd, kTrueKwd,
  kFalseKwd, kStringKwd, kMatrixKwd, kLetKwd,
  kInKwd, kEndKwd, kIfKwd, kThenKwd,
  kElseKwd, kRepeatKwd, kWhileKwd, kPrintKwd,
  kToKwd, kIntConst, kFloatConst, kStringConst,
  kVariableName, kLeftParen, kRightParen, kLeftCurly,
  kRightCurly, kLeftSquare, kRightSquare, kSemiColon,
  kColon, kAssign, kPlusSign, kStar,
  kDash, kForwardSlash, kLessThan, kLessThanEqual,
  kGreaterThan, kGreaterThanEqual, kEqualsEquals, kNotEquals,
  kAndOp, kOrOp, kNotOp, kEndOfFile,
  kLexicalError }

Functions

- char * ReadInputFromFile (const char *filename)
- char * ReadInput (int argc, char **argv)
- regex_t * make_regex (const char *pattern)
- int match_regex (regex_t *re, const char *text)

Variables

- const int kRegexNSub = 1

6.4.1 Typedef Documentation

6.4.1.1 typedef enum kTokenEnumType fcal::scanner::TokenType

6.4.2 Enumeration Type Documentation

6.4.2.1 enum fcal::scanner::kTokenEnumType

Enumerator

kIntKwd

kFloatKwd

kBoolKwd

kTrueKwd

kFalseKwd

kStringKwd

kMatrixKwd

kLetKwd

kInKwd

kEndKwd

kIfKwd

kThenKwd

kElseKwd

kRepeatKwd

kWhileKwd

kPrintKwd

kToKwd

kIntConst

kFloatConst

kStringConst

kVariableName

kLeftParen

kRightParen

kLeftCurly

kRightCurly

kLeftSquare

kRightSquare

kSemiColon

kColon

kAssign

kPlusSign

kStar

kDash

kForwardSlash

kLessThan

kLessThanEqual

kGreaterThan

kGreaterThanEqual

kEqualsEquals

kNotEquals

kAndOp

kOrOp

kNotOp

kEndOfFile

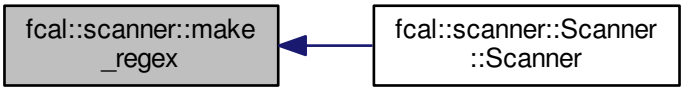
kLexicalError

6.4.3 Function Documentation

6.4.3.1 regex_t * fcal::scanner::make_regex (const char * *pattern*)

"Compile" the regular expression. This sets up the regex to do the matching specified by the regular expression given as a character string.

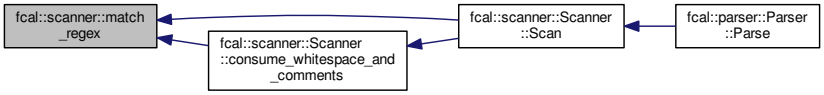
Here is the caller graph for this function:



6.4.3.2 int fcal::scanner::match_regex (regex_t * *re,* const char * *text*)

Execute the regular expression match against the text. If it matches, the beginning and ending of the matched text are stored in the first element of the matches array.

Here is the caller graph for this function:

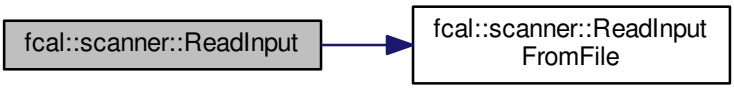


6.4.3.3 char * fcal::scanner::ReadInput (int *argc,* char ** *argv*)

ReadInput() - Read a file into a char buffer. The calling function is responsible for disposing of the return memory.

RETURN: char* - The buffer, or NULL if an error occurred.

Here is the call graph for this function:

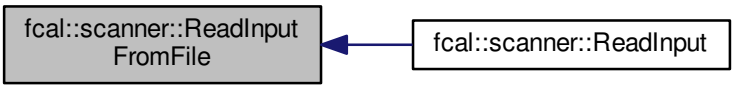


6.4.3.4 char * fcal::scanner::ReadInputFromFile (const char * *filename*)

ReadInputFromFile() - Do the actual reading of the file into the buffer

RETURN: char* - The buffer, or NULL if an error occurred.

Here is the caller graph for this function:



6.4.4 Variable Documentation

6.4.4.1 const int fcal::scanner::kRegexNSub = 1

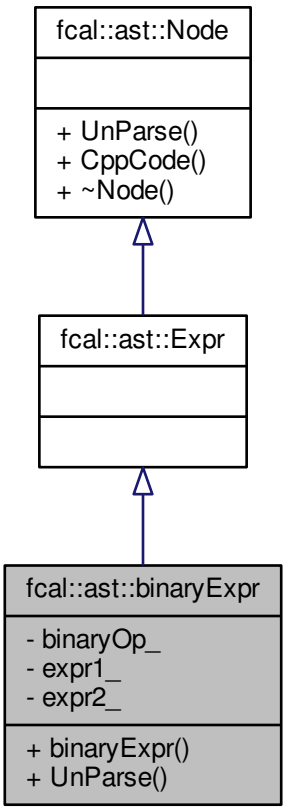
Chapter 7

Class Documentation

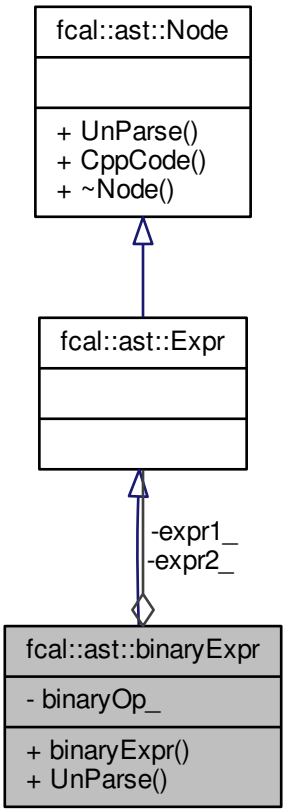
7.1 fcal::ast::binaryExpr Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::binaryExpr:



Collaboration diagram for fcal::ast::binaryExpr:



Public Member Functions

- binaryExpr (std::string binaryOp, Expr *expr1, Expr *expr2)
- std::string UnParse (void)

  Retruns the string : expr1_->UnParse() + binaryOp_ + expr2_->UnParse()

Private Attributes

- std::string binaryOp_

  *string representing +,/,-,**

- Expr * expr1_

  Expr representing Left Operand in this Binary Operation.

- Expr * expr2_

  Expr representing Right Operand in this Binary Operation.

7.1.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the productions,
Expr ::= Expr '∗' Expr
Expr ::= Expr '/' Expr
Expr ::= Expr '+' Expr
Expr ::= Expr '-' Expr

7.1.2 Constructor & Destructor Documentation

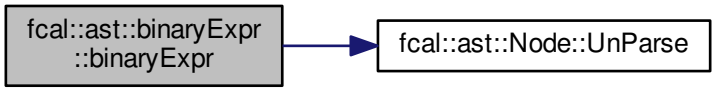
7.1.2.1 fcal::ast::binaryExpr::binaryExpr (std::string *binaryOp,* Expr ∗ *expr1,* Expr ∗ *expr2*) [inline]

This constructor takes three parameters

Parameters

binaryOp	string representing +,/,-,∗
expr1	left Expr of binaryOp
expr2	Right Expr of binaryOp

Here is the call graph for this function:



7.1.3 Member Function Documentation

7.1.3.1 std::string fcal::ast::binaryExpr::UnParse (void) [virtual]

Retruns the string : expr1_->UnParse() + binaryOp_ + expr2_->UnParse()

Reimplemented from fcal::ast::Node.

7.1.4 Member Data Documentation

7.1.4.1 std::string fcal::ast::binaryExpr::binaryOp_ [private]

string representing +,/,-,∗

7.1.4.2 Expr* fcal::ast::binaryExpr::expr1_ [private]

Expr representing Left Operand in this Binary Operation.

7.1.4.3 Expr* fcal::ast::binaryExpr::expr2_ [private]

Expr representing Right Operand in this Binary Operation.

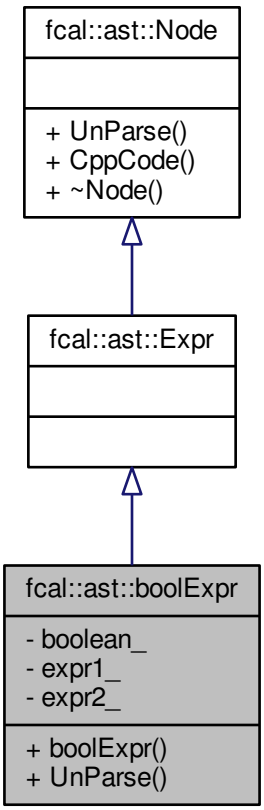
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

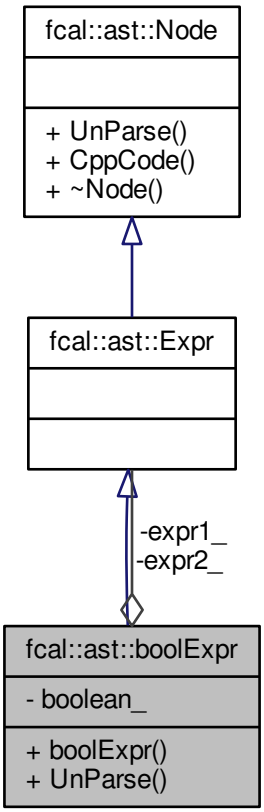
7.2 fcal::ast::boolExpr Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::boolExpr:



Collaboration diagram for fcal::ast::boolExpr:



Public Member Functions

- boolExpr (std::string boolean, Expr ∗expr1, Expr ∗expr2)
- std::string UnParse (void)

  Returns the string : expr1_->UnParse() + boolean_ + expr2_->UnParse()

Private Attributes

- std::string boolean_

  string representing relational operators
- Expr ∗ expr1_

  Expr representing Left Operand in this realtional Operation.
- Expr ∗ expr2_

  Expr representing Right Operand in this relational Operation.

7.2.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the productions,
Expr ::= Expr '>' Expr
Expr ::= Expr '>=' Expr
Expr ::= Expr '<' Expr
Expr ::= Expr '<=' Expr
Expr ::= Expr '==' Expr
Expr ::= Expr '!=' Expr
Expr ::= Expr '&&' Expr
Expr ::= Expr '||' Expr

7.2.2 Constructor & Destructor Documentation

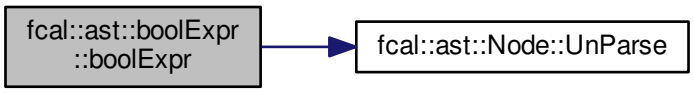
7.2.2.1 fcal::ast::boolExpr::boolExpr (std::string *boolean*, Expr * *expr1*, Expr * *expr2*) [inline]

This constructor takes three parameters

Parameters

boolean	string representing relational operators
expr1	Expr representing Left Operand
expr2	Expr representing Right Operand

Here is the call graph for this function:



7.2.3 Member Function Documentation

7.2.3.1 std::string fcal::ast::boolExpr::UnParse (void) [virtual]

Returns the string : expr1_->UnParse() + boolean_ + expr2_->UnParse()

Reimplemented from fcal::ast::Node.

7.2.4 Member Data Documentation

7.2.4.1 std::string fcal::ast::boolExpr::boolean_ [private]

string representing relational operators

7.2.4.2 Expr* fcal::ast::boolExpr::expr1_ [private]

Expr representing Left Operand in this realtional Operation.

7.2.4.3 Expr* fcal::ast::boolExpr::expr2_ [private]

Expr representing Right Operand in this relational Operation.

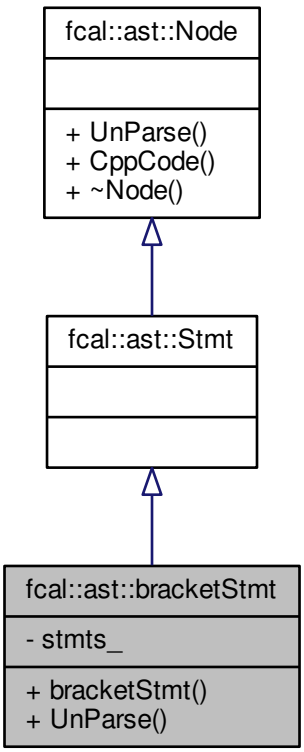
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

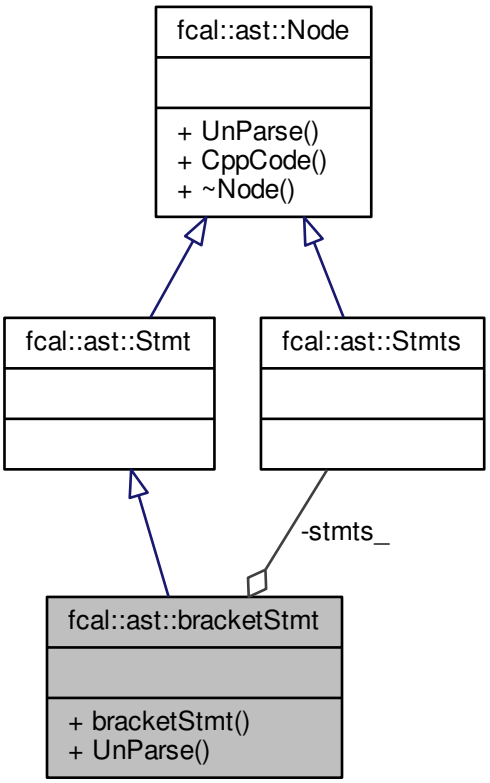
7.3 fcal::ast::bracketStmt Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::bracketStmt:



Collaboration diagram for fcal::ast::bracketStmt:



Public Member Functions

- bracketStmt (Stmts *stmts)
- std::string UnParse (void)

  Returns the string : " {" + stmts_->UnParse() + "}".

Private Attributes

- Stmts * stmts_

7.3.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Stmt ::= '{' Stmts '}'

7.3.2 Constructor & Destructor Documentation

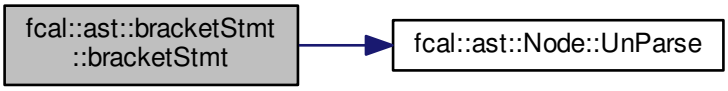
7.3.2.1 fcal::ast::bracketStmt::bracketStmt (Stmts * *stmts*) [inline], [explicit]

This constructor takes only one parameter

Parameters

smts	a Stmts* holding the address which contains details of Stmts under brackects '{' Stmts '}'

Here is the call graph for this function:



7.3.3 Member Function Documentation

7.3.3.1 std::string fcal::ast::bracketStmt::UnParse (void) [virtual]

Returns the string : " {" + stmts_->UnParse() + "}".

Reimplemented from fcal::ast::Node.

7.3.4 Member Data Documentation

7.3.4.1 Stmts* fcal::ast::bracketStmt::stmts_ [private]

stmts_ hols the address containing details of Stmts under brackets '{' Stmts '}'

The documentation for this class was generated from the following files:

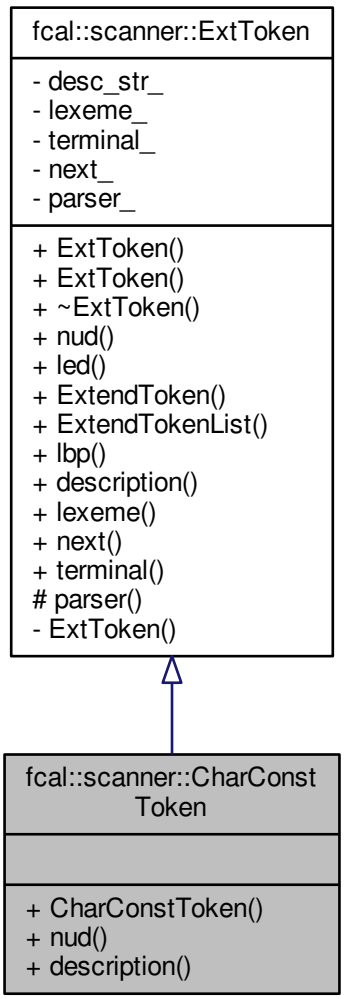
- include/ast.h
- src/ast.cc

7.4 fcal::scanner::CharConstToken Class Reference

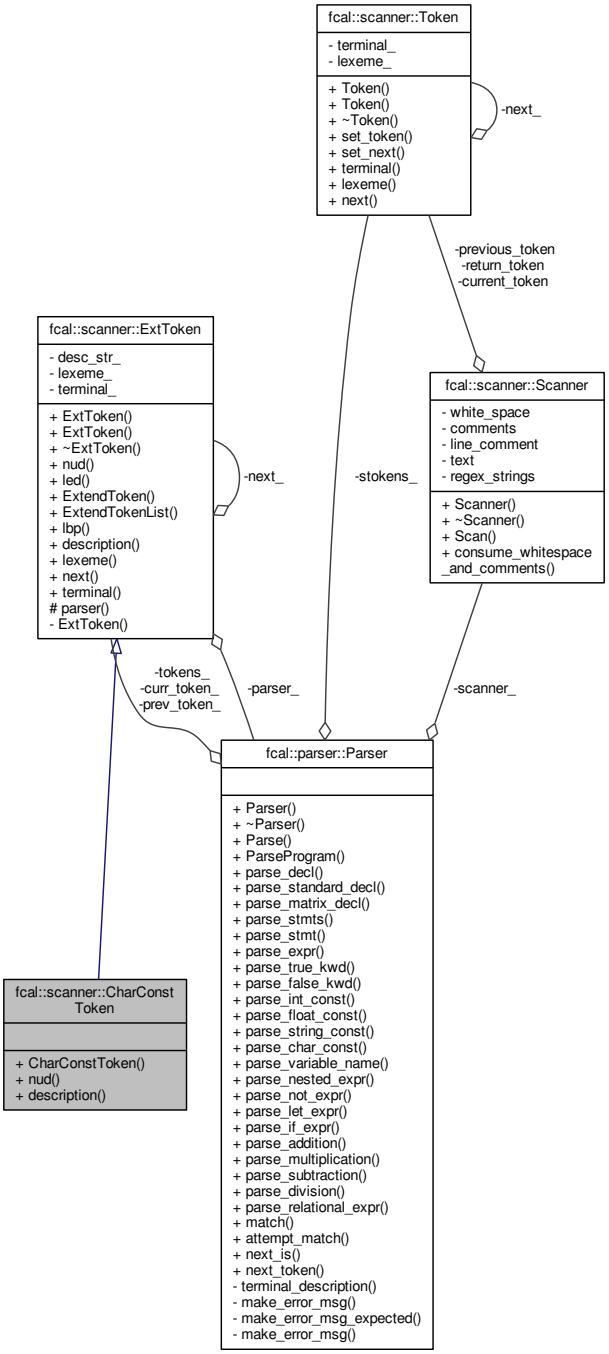
Char Const.

`#include <ext_token.h>`

Inheritance diagram for fcal::scanner::CharConstToken:



Collaboration diagram for fcal::scanner::CharConstToken:



Public Member Functions

- CharConstToken (parser::Parser ∗p, Token ∗t)
- parser::ParseResult nud ()
- std::string description ()

Additional Inherited Members

7.4.1 Detailed Description

Char Const.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 fcal::scanner::CharConstToken::CharConstToken (parser::Parser * *p*, Token * *t*) [inline]

7.4.3 Member Function Documentation

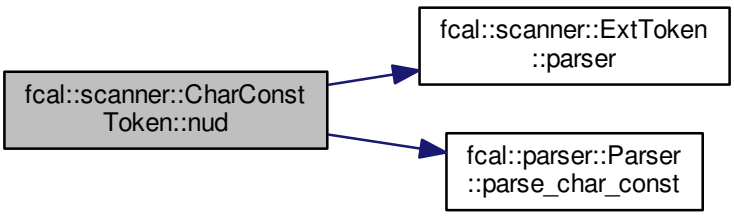
7.4.3.1 std::string fcal::scanner::CharConstToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.4.3.2 parser::ParseResult fcal::scanner::CharConstToken::nud (void) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



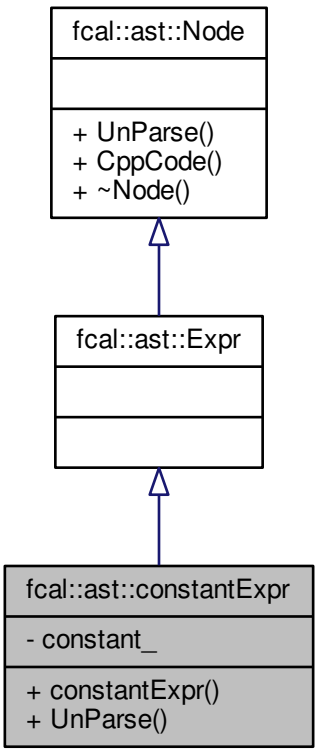
The documentation for this class was generated from the following file:

- include/ext_token.h

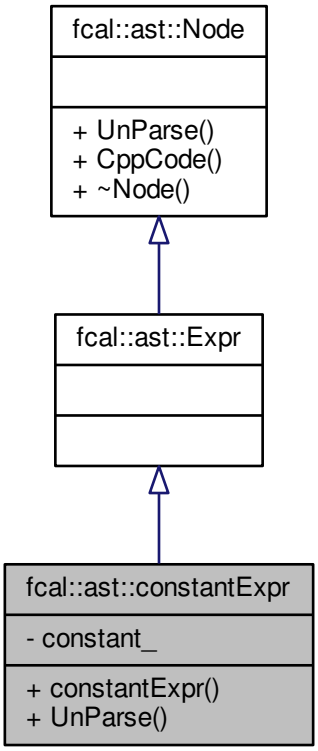
7.5 fcal::ast::constantExpr Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::constantExpr:



Collaboration diagram for fcal::ast::constantExpr:



Public Member Functions

- constantExpr (std::string const_)
- std::string UnParse (void)

  Returns the string : constant_.

Private Attributes

- std::string constant_

7.5.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the productions,
Expr ::= varName
Expr ::= integerConst | floatConst | stringConst
Expr ::= 'True' | 'False'

7.5.2 Constructor & Destructor Documentation

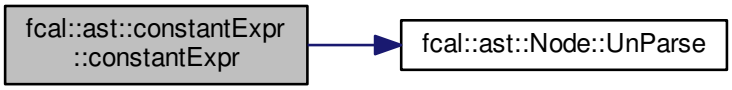
7.5.2.1 fcal::ast::constantExpr::constantExpr (std::string *const_*) [inline], [explicit]

This constructor takes one parameter

Parameters

const↩ _	takes a string constant representing a varName , integer, float , string , True or False

Here is the call graph for this function:



7.5.3 Member Function Documentation

7.5.3.1 std::string fcal::ast::constantExpr::UnParse (void) [virtual]

Returns the string : constant_.

Reimplemented from fcal::ast::Node.

7.5.4 Member Data Documentation

7.5.4.1 std::string fcal::ast::constantExpr::constant_ [private]

a string constant representing either one of the following : varName , integer, float , string , True , False

The documentation for this class was generated from the following files:

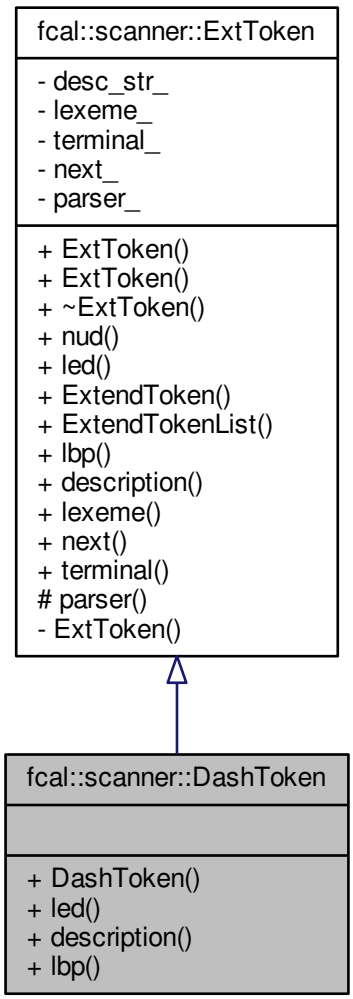
- include/ast.h
- src/ast.cc

7.6 fcal::scanner::DashToken Class Reference

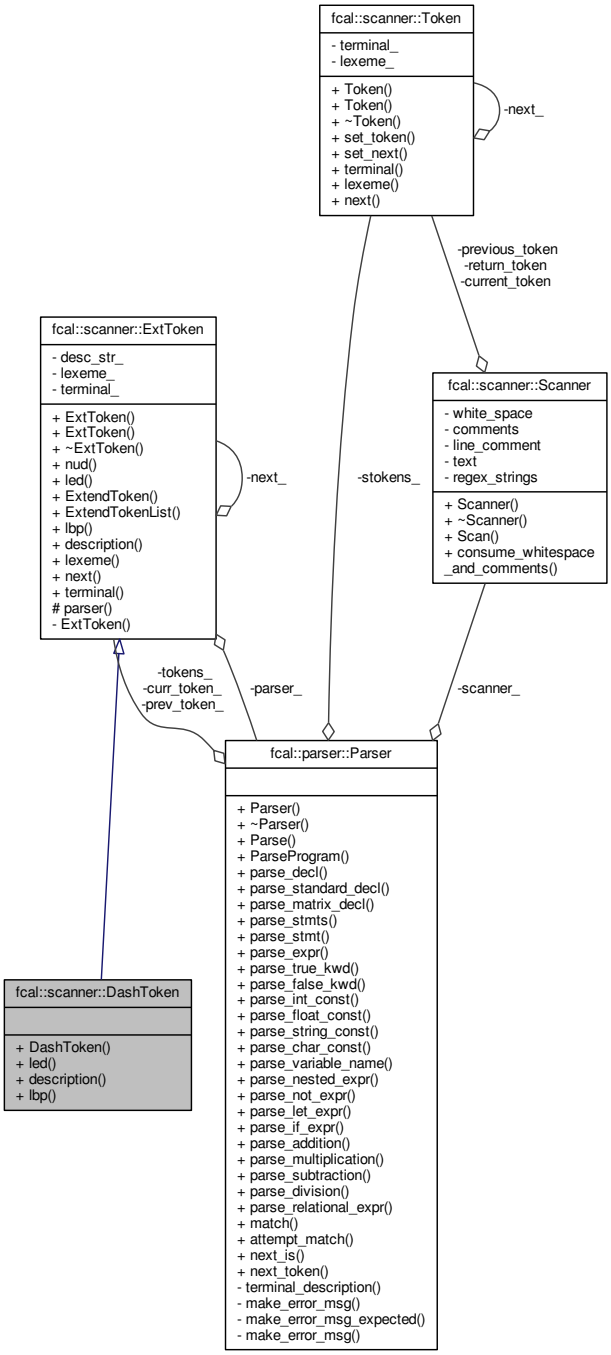
Dash.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::DashToken:



Collaboration diagram for fcal::scanner::DashToken:



Public Member Functions

- DashToken (parser::Parser ∗p, Token ∗t)
- parser::ParseResult led (parser::ParseResult left)
- std::string description ()
- int lbp ()

Additional Inherited Members

7.6.1 Detailed Description

Dash.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 fcal::scanner::DashToken::DashToken (parser::Parser * *p*, Token * *t*) [inline]

7.6.3 Member Function Documentation

7.6.3.1 std::string fcal::scanner::DashToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

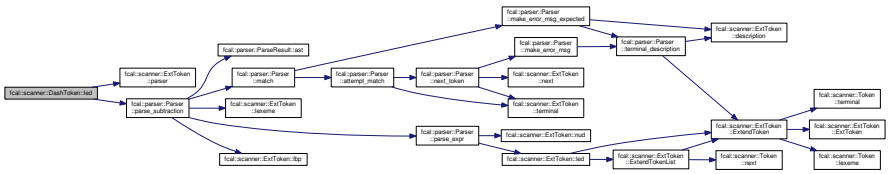
7.6.3.2 int fcal::scanner::DashToken::lbp ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.6.3.3 parser::ParseResult fcal::scanner::DashToken::led (parser::ParseResult *left*) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



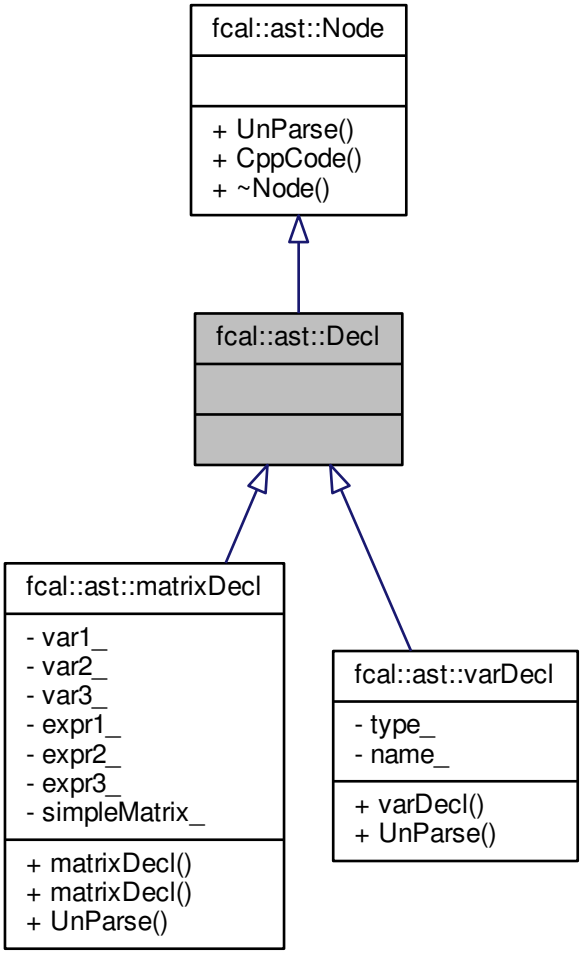
The documentation for this class was generated from the following file:

- include/ext_token.h

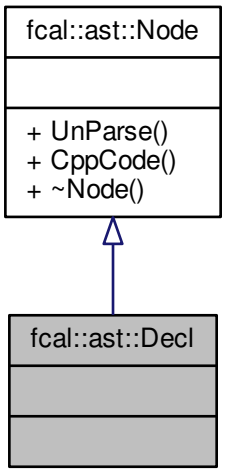
7.7 fcal::ast::Decl Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::Decl:



Collaboration diagram for fcal::ast::Decl:



Additional Inherited Members

7.7.1 Detailed Description

This is the abstract class which inherits from Node. It has no implementation for any functions derived from Node. It is the base class for classes which implement the productions which are derived from the nonterminal 'Decl' in the FCAL grammar.

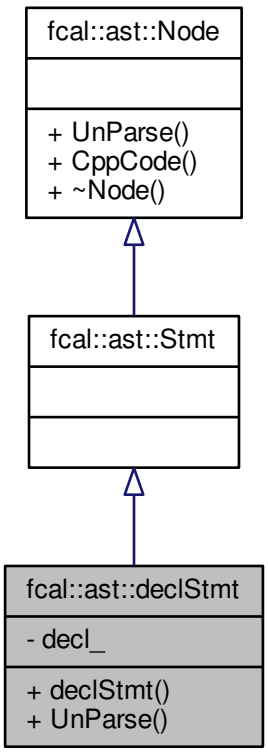
The documentation for this class was generated from the following file:

- include/ast.h

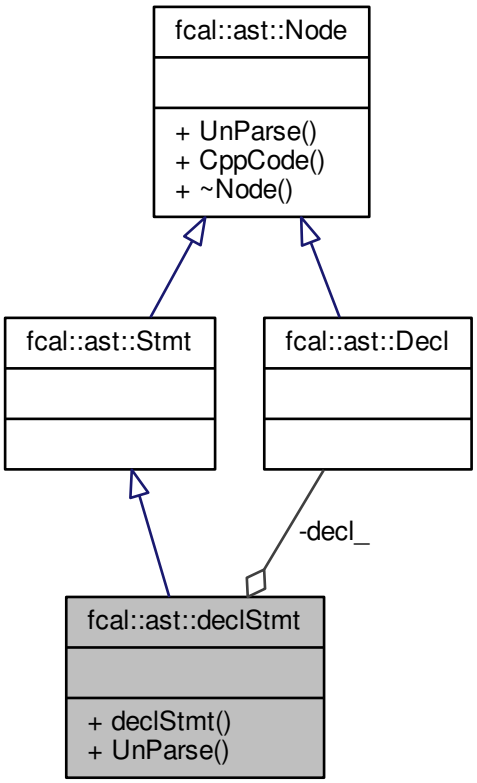
7.8 fcal::ast::declStmt Class Reference

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

Inheritance diagram for fcal::ast::declStmt:



Collaboration diagram for fcal::ast::declStmt:



Public Member Functions

- declStmt (Decl *decl)
- std::string UnParse (void)

  Returns the string : decl_->UnParse()

Private Attributes

- Decl * decl_

7.8.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Stmts ::= Stmt Stmts

7.8.2 Constructor & Destructor Documentation

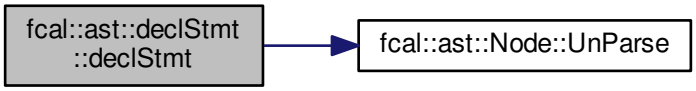
7.8.2.1 fcal::ast::declStmt::declStmt (Decl * *decl*) [inline], [explicit]

This constructor takes only one parameter

Parameters

decl	a Decl* holding the address which contains information reagarding a declaration

Here is the call graph for this function:



7.8.3 Member Function Documentation

7.8.3.1 std::string fcal::ast::declStmt::UnParse (void) [virtual]

Returns the string : decl_->UnParse()

Reimplemented from fcal::ast::Node.

7.8.4 Member Data Documentation

7.8.4.1 Decl* fcal::ast::declStmt::decl_ [private]

decl_ hols the address which contains details regarding the type of Declaration

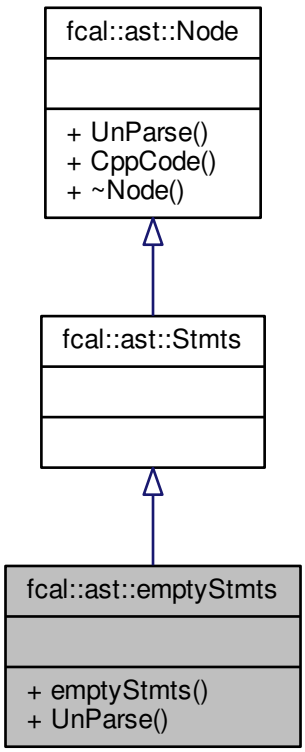
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

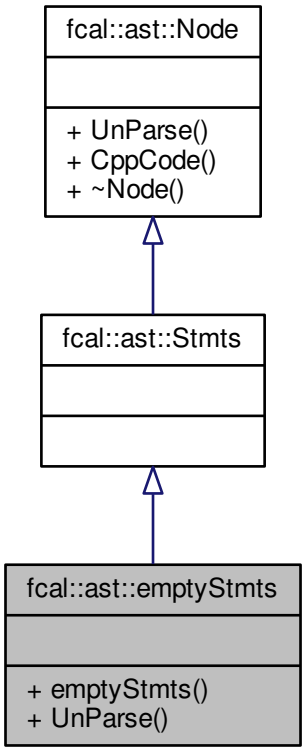
7.9 fcal::ast::emptyStmts Class Reference

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

Inheritance diagram for fcal::ast::emptyStmts:



Collaboration diagram for fcal::ast::emptyStmts:



Public Member Functions

- emptyStmts (void)

  Constructor for emptyStmts, it does nothing.
- std::string UnParse (void)

  Returns an empty String.

7.9.1 Detailed Description

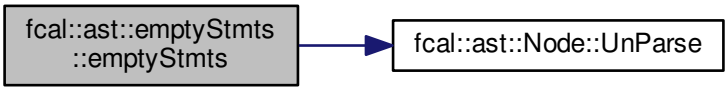
This is a concrete class in the ast class heirarchy. It implements the production,
Stmts ::= <<empty>>

7.9.2 Constructor & Destructor Documentation

7.9.2.1 fcal::ast::emptyStmts::emptyStmts (void) [inline]

Constructor for emptyStmts, it does nothing.

Here is the call graph for this function:



7.9.3 Member Function Documentation

7.9.3.1 std::string fcal::ast::emptyStmts::UnParse (void) [virtual]

Returns an empty String.

Returns the string : "".

Reimplemented from fcal::ast::Node.

The documentation for this class was generated from the following files:

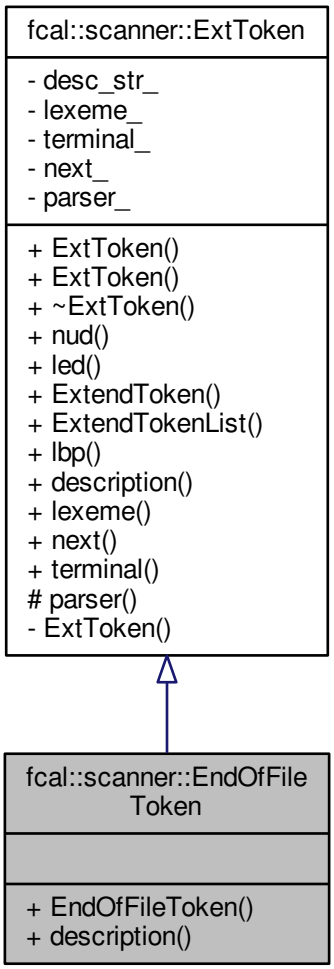
- include/ast.h
- src/ast.cc

7.10 fcal::scanner::EndOfFileToken Class Reference

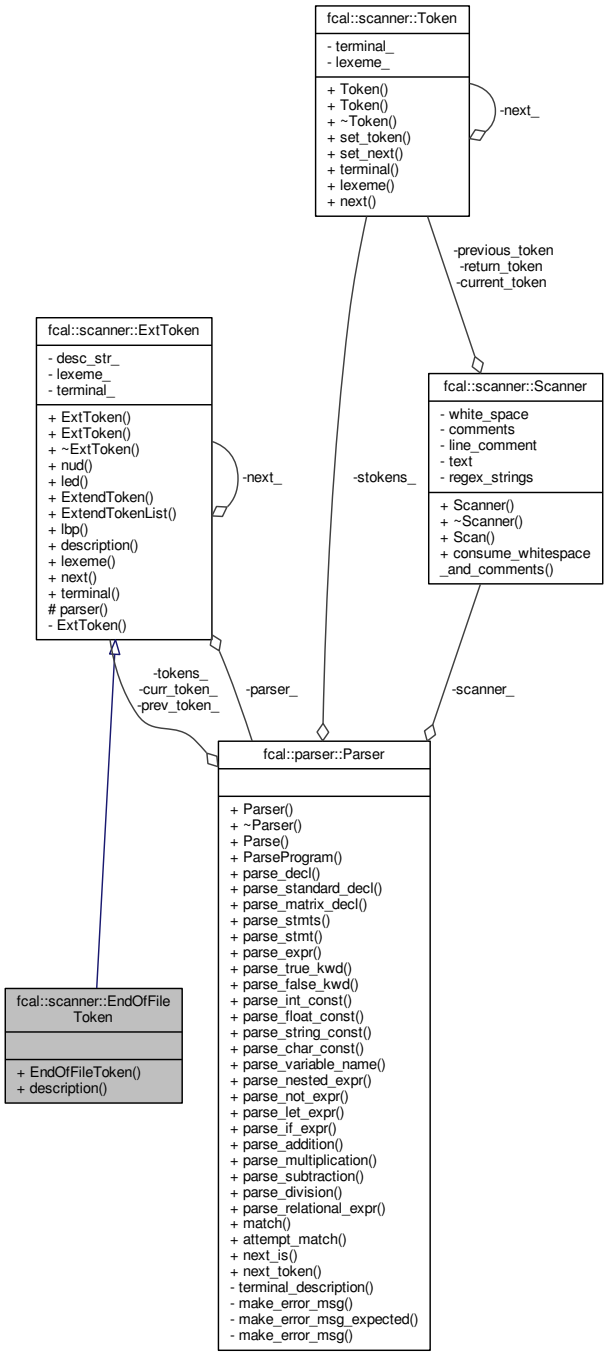
End of File.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::EndOfFileToken:



Collaboration diagram for fcal::scanner::EndOfFileToken:



Public Member Functions

- EndOfFileToken (parser::Parser *p, Token *t)
- std::string description ()

Additional Inherited Members

7.10.1 Detailed Description

End of File.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 fcal::scanner::EndOfFileToken::EndOfFileToken (parser::Parser * *p*, Token * *t*) [inline]

7.10.3 Member Function Documentation

7.10.3.1 std::string fcal::scanner::EndOfFileToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

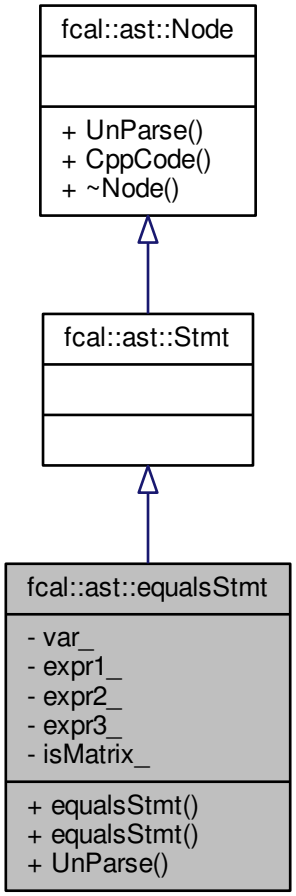
The documentation for this class was generated from the following file:

- include/ext_token.h

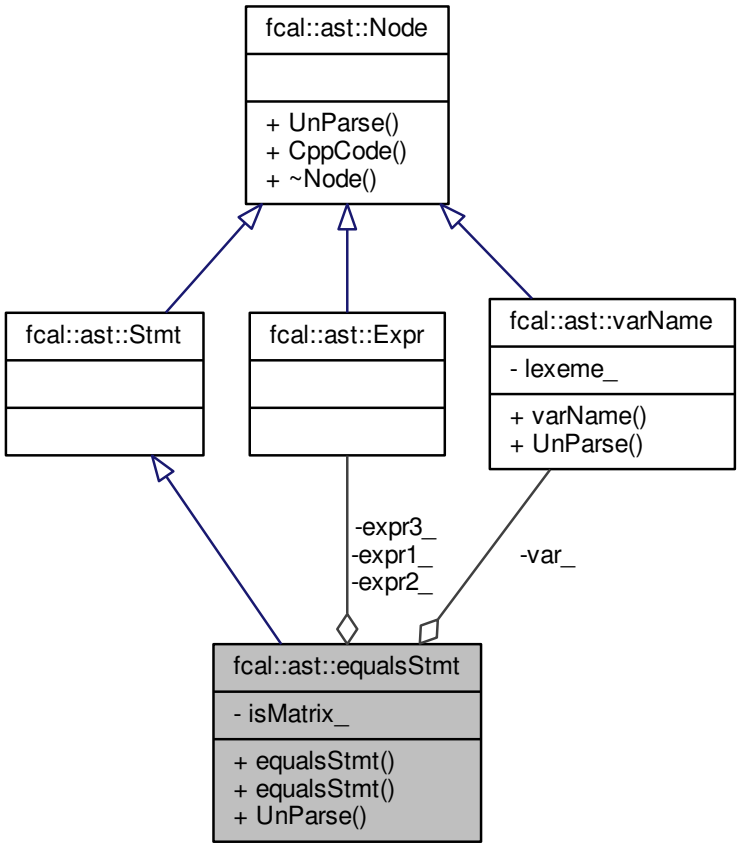
7.11 fcal::ast::equalsStmt Class Reference

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

Inheritance diagram for fcal::ast::equalsStmt:



Collaboration diagram for fcal::ast::equalsStmt:



Public Member Functions

- equalsStmt (varName ∗var, Expr ∗expr1, bool isMatrix)
- equalsStmt (varName ∗var, Expr ∗expr1, Expr ∗expr2, Expr ∗expr3, bool isMatrix)
- std::string UnParse (void)

Private Attributes

- varName ∗ var_

  variable name or that of a matrix
- Expr ∗ expr1_
- Expr ∗ expr2_

  Expr representing second element of Matrix if isMatrix is True.
- Expr ∗ expr3_
- bool isMatrix_

  isMatrix distinguishes between regular variable or matrix constructor

7.11.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Stmt ::= varName '=' Expr ';' | varName '[' Expr ':' Expr ']' '=' Expr ';'

7.11.2 Constructor & Destructor Documentation

7.11.2.1 fcal::ast::equalsStmt::equalsStmt (varName * *var*, Expr * *expr1*, bool *isMatrix*) [inline]

This constructor takes three parameters

Parameters

var	a varName* which holds the address containing the information of varName representing an Expr
expr1	a Expr* holding the address which contains the information of an Expr whose value will be assigned to a varName
isMatrix	bool value which is True if asignment is a Martix and False otherwise

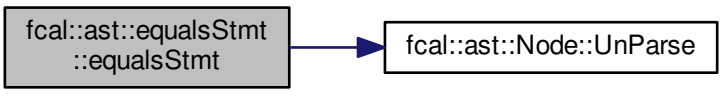
7.11.2.2 fcal::ast::equalsStmt::equalsStmt (varName * *var*, Expr * *expr1*, Expr * *expr2*, Expr * *expr3*, bool *isMatrix*) [inline]

This constructor takes five parameters

Parameters

var	varName of a matrix
expr1	Expr representing rows of matrix
expr2	Expr representing columns of matrix
expr3	Expr whise value will be assigned to a particular index in the matrix
isMatrix	bool value which is True if asignment is a Martix and False otherwise

Here is the call graph for this function:



7.11.3 Member Function Documentation

7.11.3.1 std::string fcal::ast::equalsStmt::UnParse (void) [virtual]

Returns the string : var_->UnParse() + " = " + expr1_->UnParse() + ";\n" , if isMatrix_ is False.
Returns the string : var_->UnParse() + " [" + expr1_->UnParse() + ":"

- expr2_->UnParse() + + "] " + expr3_->UnParse() + ";\n" , if isMatrix_ is True.

Reimplemented from fcal::ast::Node.

7.11.4 Member Data Documentation

7.11.4.1 Expr* fcal::ast::equalsStmt::expr1_ [private]

Expr representing first element of Matrix if isMatrix is True else it represents an Expr to be assigned to a variable

7.11.4.2 Expr* fcal::ast::equalsStmt::expr2_ [private]

Expr representing second element of Matrix if isMatrix is True.

7.11.4.3 Expr* fcal::ast::equalsStmt::expr3_ [private]

Expr which will b assigned to a position in the Matrix if isMatrix is True

7.11.4.4 bool fcal::ast::equalsStmt::isMatrix_ [private]

isMatrix distinguishes between regular variable or matrix constructor

7.11.4.5 varName* fcal::ast::equalsStmt::var_ [private]

variable name or that of a matrix

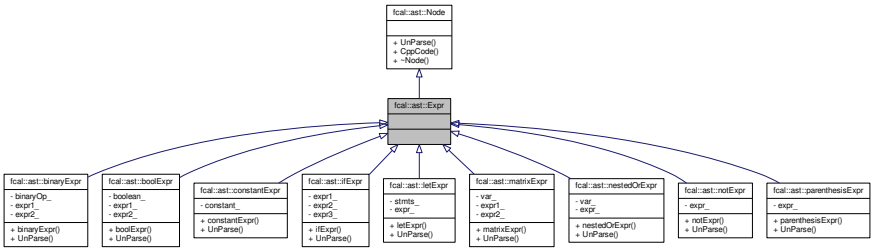
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

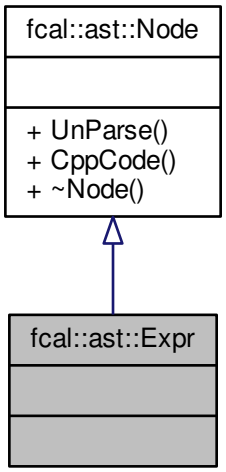
7.12 fcal::ast::Expr Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::Expr:



Collaboration diagram for fcal::ast::Expr:



Additional Inherited Members

7.12.1 Detailed Description

This is the abstract class which inherits from Node. It has no implementation for any functions derived from Node. It is the base class for classes which implement the productions which are derived from the nonterminal 'Expr' in the FCAL grammar.

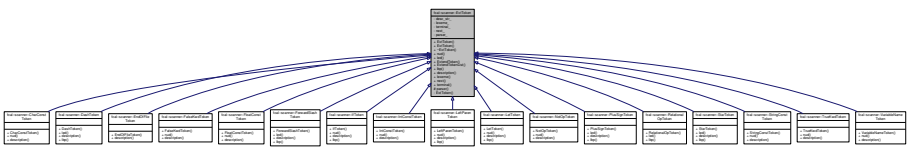
The documentation for this class was generated from the following file:

- include/ast.h

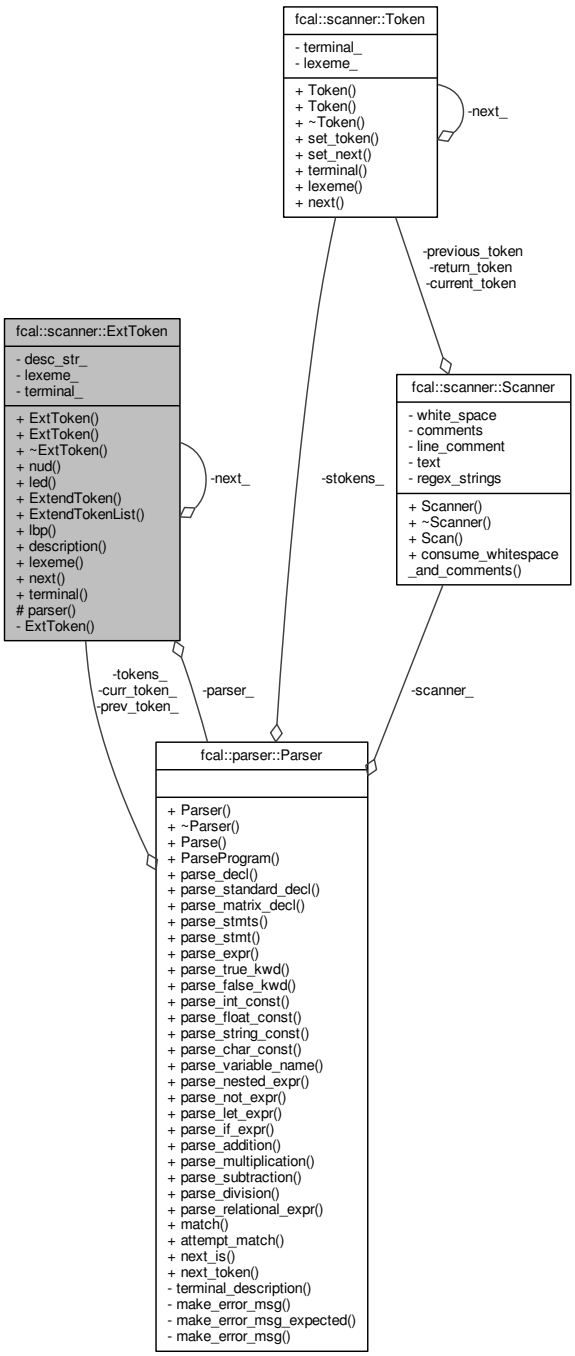
7.13 fcal::scanner::ExtToken Class Reference

`#include <ext_token.h>`

Inheritance diagram for fcal::scanner::ExtToken:



Collaboration diagram for fcal::scanner::ExtToken:



Public Member Functions

- ExtToken (parser::Parser ∗p, Token ∗t)
- ExtToken (parser::Parser ∗p, Token ∗t, std::string d)
- virtual ∼ExtToken ()
- virtual parser::ParseResult nud (void)
- virtual parser::ParseResult led (parser::ParseResult left)

- ExtToken ∗ ExtendToken (parser::Parser ∗p, Token ∗tokens)
- ExtToken ∗ ExtendTokenList (parser::Parser ∗p, Token ∗tokens)
- virtual int lbp ()
- virtual std::string description ()
- std::string lexeme (void) const
- ExtToken ∗ next (void) const
- scanner::TokenType terminal (void) const

Protected Member Functions

- parser::Parser ∗ parser (void)

Private Member Functions

- ExtToken (void)

Private Attributes

- std::string desc_str_
- std::string lexeme_
- scanner::TokenType terminal_
- ExtToken ∗ next_
- parser::Parser ∗ parser_

7.13.1 Constructor & Destructor Documentation

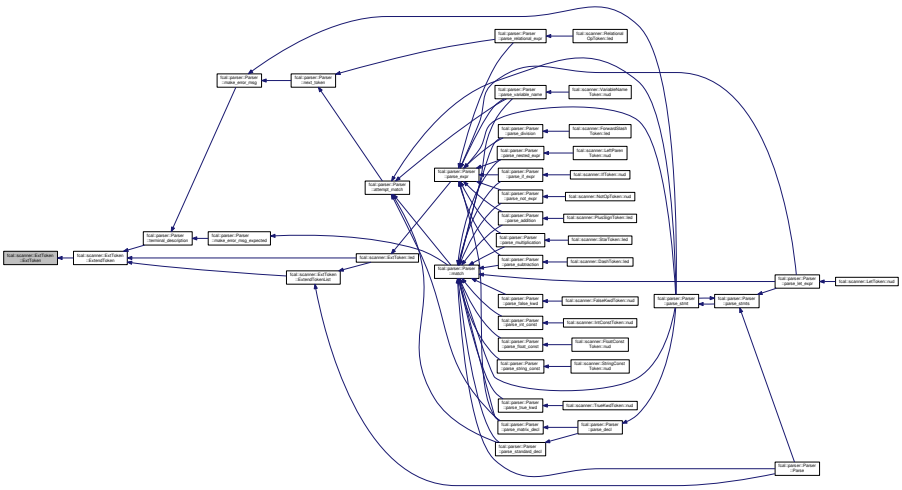
7.13.1.1 fcal::scanner::ExtToken::ExtToken (parser::Parser ∗ *p*, Token ∗ *t*) [inline]

7.13.1.2 fcal::scanner::ExtToken::ExtToken (parser::Parser ∗ *p*, Token ∗ *t*, std::string *d*) [inline]

7.13.1.3 virtual fcal::scanner::ExtToken::∼ExtToken ( ) [inline], [virtual]

7.13.1.4 fcal::scanner::ExtToken::ExtToken (void) [inline], [private]

Here is the caller graph for this function:

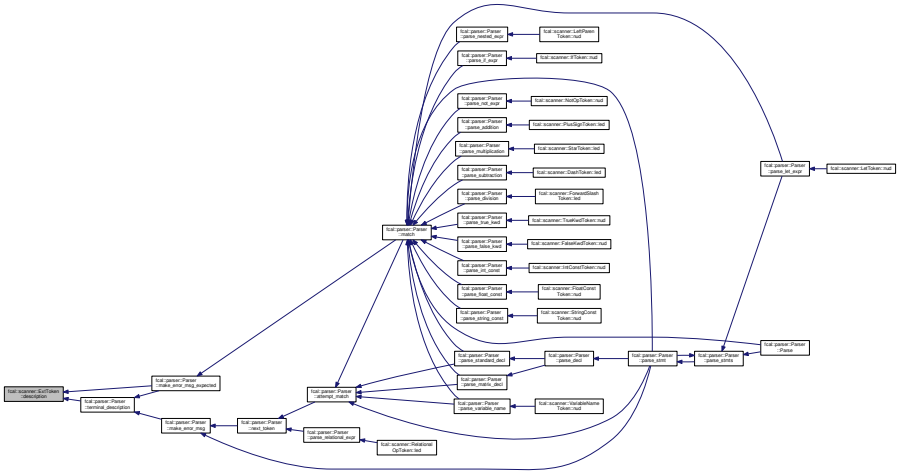


7.13.2 Member Function Documentation

7.13.2.1 virtual std::string fcal::scanner::ExtToken::description ( ) [inline], [virtual]

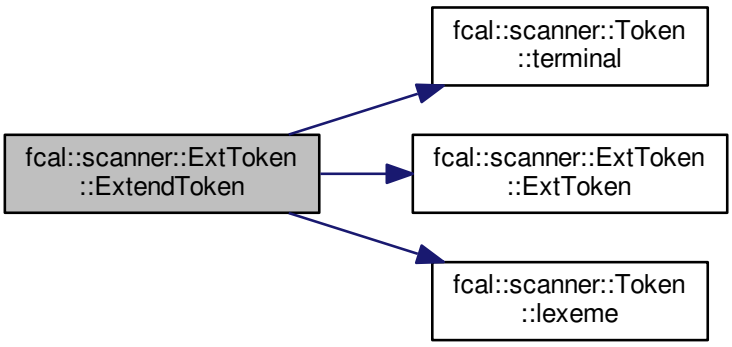
Reimplemented in fcal::scanner::EndOfFileToken, fcal::scanner::ForwardSlashToken, fcal::scanner::DashToken, fcal::scanner::StarToken, fcal::scanner::PlusSignToken, fcal::scanner::LeftParenToken, fcal::scanner::LetToken, fcal::scanner::IfToken, fcal::scanner::VariableNameToken, fcal::scanner::CharConstToken, fcal::scanner::StringConstToken, fcal::scanner::FloatConstToken, fcal::scanner::IntConstToken, fcal::scanner::FalseKwdToken, fcal::scanner::TrueKwdToken, and fcal::scanner::NotOpToken.

Here is the caller graph for this function:

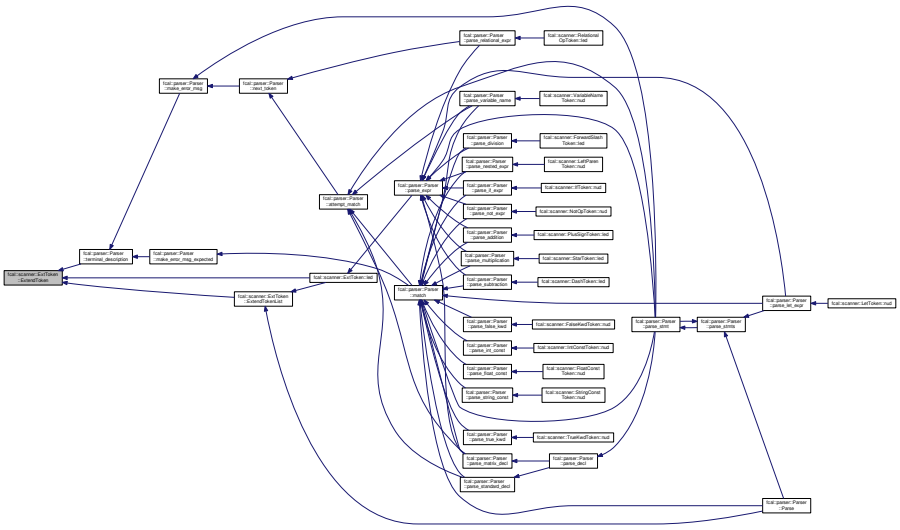


7.13.2.2 ExtToken * fcal::scanner::ExtToken::ExtendToken (parser::Parser * *p*, Token * *tokens*)

Here is the call graph for this function:

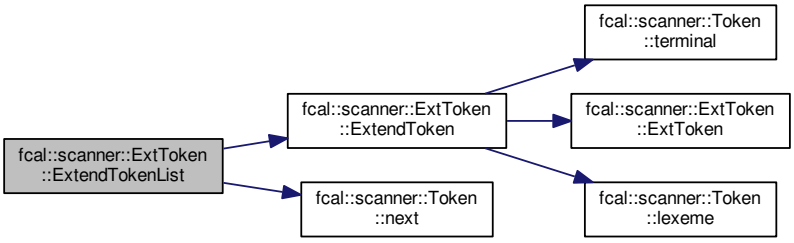


Here is the caller graph for this function:

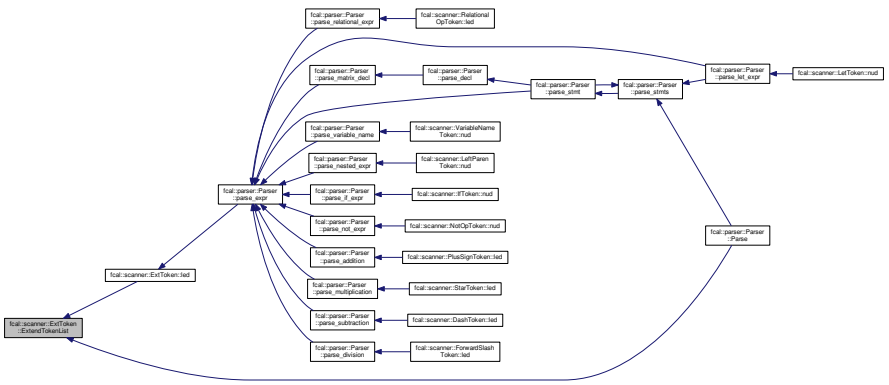


7.13.2.3 ExtToken * fcal::scanner::ExtToken::ExtendTokenList (parser::Parser * *p*, Token * *tokens*)

Here is the call graph for this function:



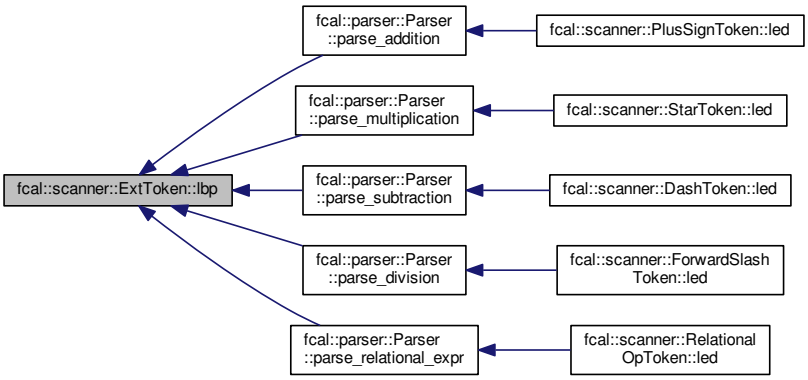
Here is the caller graph for this function:



7.13.2.4 virtual int fcal::scanner::ExtToken::lbp ( ) [inline], [virtual]

Reimplemented in fcal::scanner::RelationalOpToken, fcal::scanner::ForwardSlashToken, fcal::scanner::DashToken, fcal::scanner::StarToken, fcal::scanner::PlusSignToken, fcal::scanner::LeftParenToken, fcal::scanner::LetToken, and fcal::scanner::IfToken.

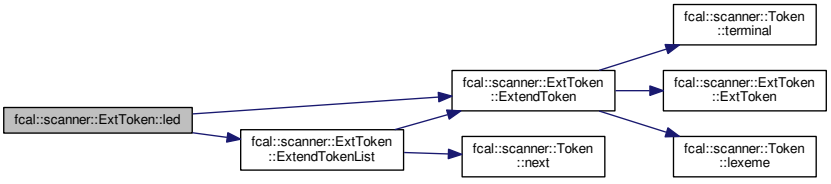
Here is the caller graph for this function:



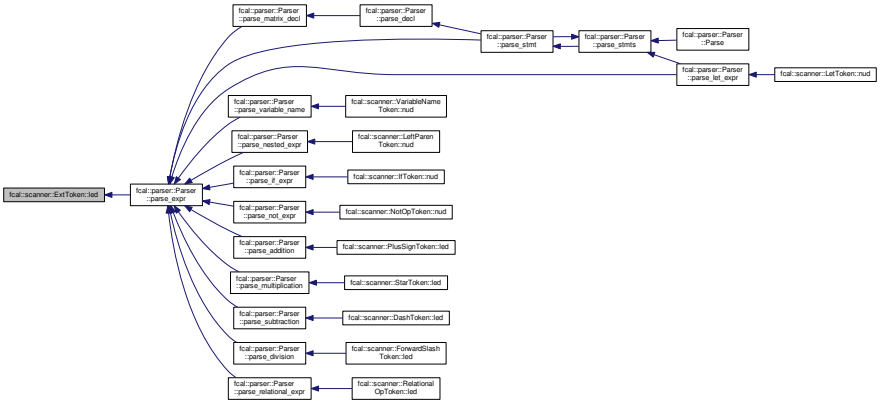
7.13.2.5 virtual parser::ParseResult fcal::scanner::ExtToken::led (parser::ParseResult *left*) [inline], [virtual]

Reimplemented in fcal::scanner::RelationalOpToken, fcal::scanner::ForwardSlashToken, fcal::scanner::DashToken, fcal::scanner::StarToken, and fcal::scanner::PlusSignToken.

Here is the call graph for this function:

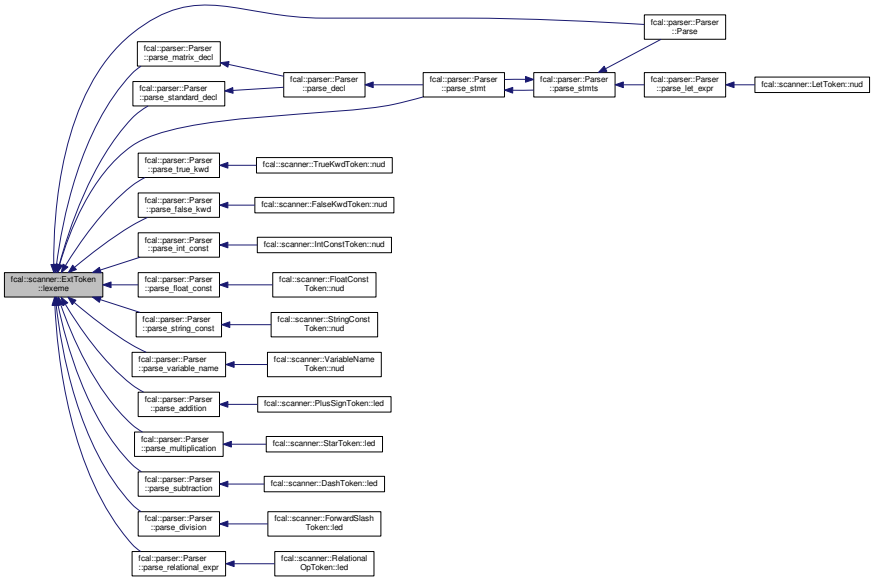


Here is the caller graph for this function:



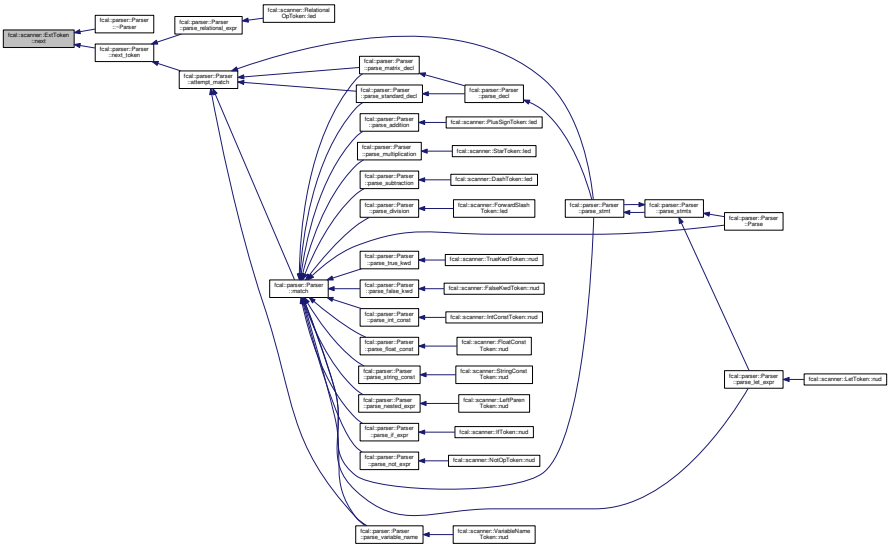
7.13.2.6 std::string fcal::scanner::ExtToken::lexeme (void) const [inline]

Here is the caller graph for this function:



7.13.2.7 ExtToken* fcal::scanner::ExtToken::next (void) const [inline]

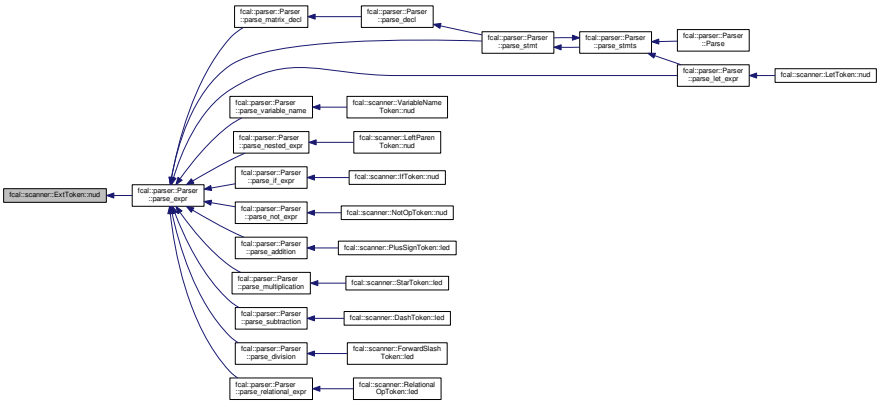
Here is the caller graph for this function:



7.13.2.8 virtual parser::ParseResult fcal::scanner::ExtToken::nud (void) [inline], [virtual]

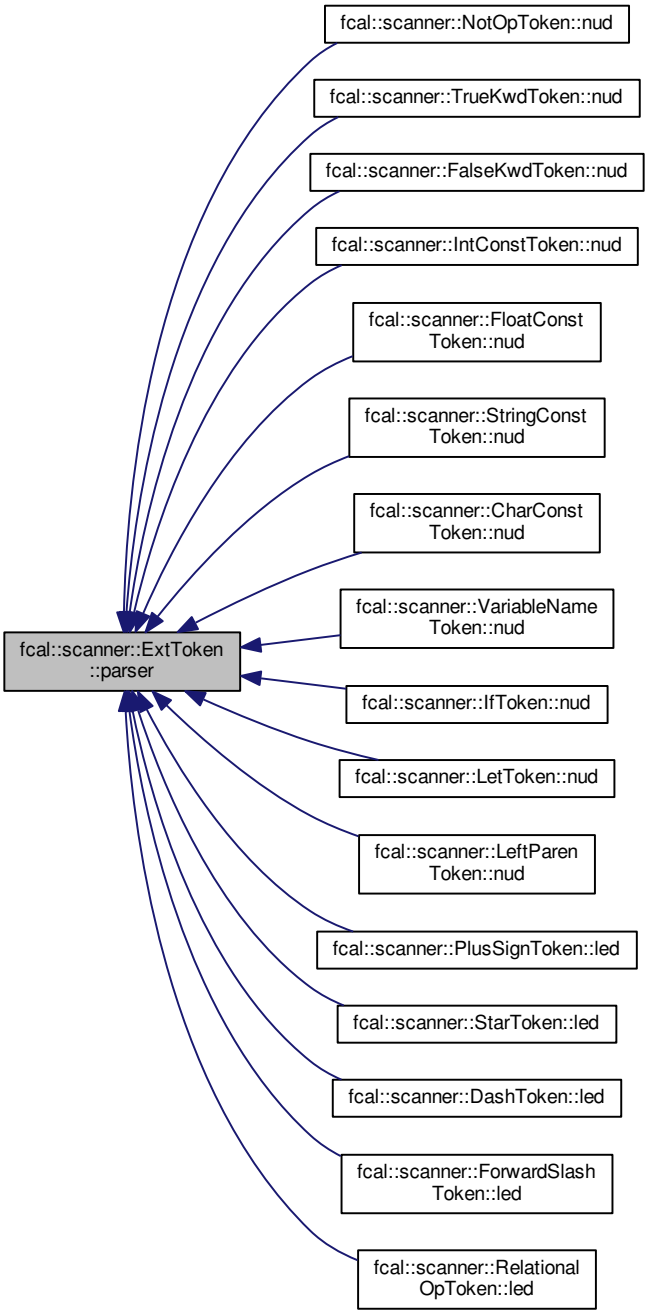
Reimplemented in fcal::scanner::LeftParenToken, fcal::scanner::LetToken, fcal::scanner::IfToken, fcal::scanner::VariableNameToken, fcal::scanner::CharConstToken, fcal::scanner::StringConstToken, fcal::scanner::FloatConstToken, fcal::scanner::IntConstToken, fcal::scanner::FalseKwdToken, fcal::scanner::TrueKwdToken, and fcal::scanner::NotOpToken.

Here is the caller graph for this function:



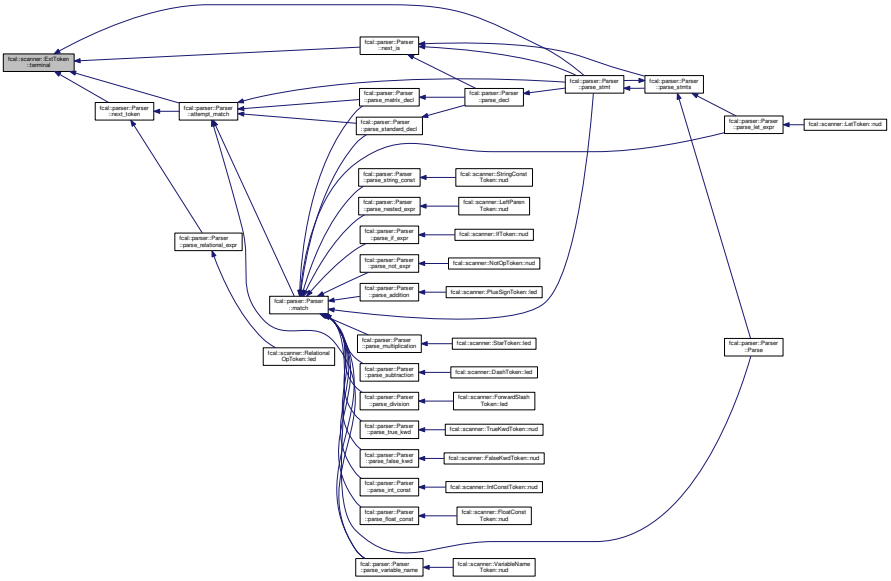
7.13.2.9 parser::Parser* fcal::scanner::ExtToken::parser (void) [inline], [protected]

Here is the caller graph for this function:



7.13.2.10 scanner::TokenType fcal::scanner::ExtToken::terminal (void) const [inline]

Here is the caller graph for this function:



7.13.3 Member Data Documentation

7.13.3.1 std::string fcal::scanner::ExtToken::desc_str_ [private]

7.13.3.2 std::string fcal::scanner::ExtToken::lexeme_ [private]

7.13.3.3 ExtToken* fcal::scanner::ExtToken::next_ [private]

7.13.3.4 parser::Parser* fcal::scanner::ExtToken::parser_ [private]

7.13.3.5 scanner::TokenType fcal::scanner::ExtToken::terminal_ [private]

The documentation for this class was generated from the following files:

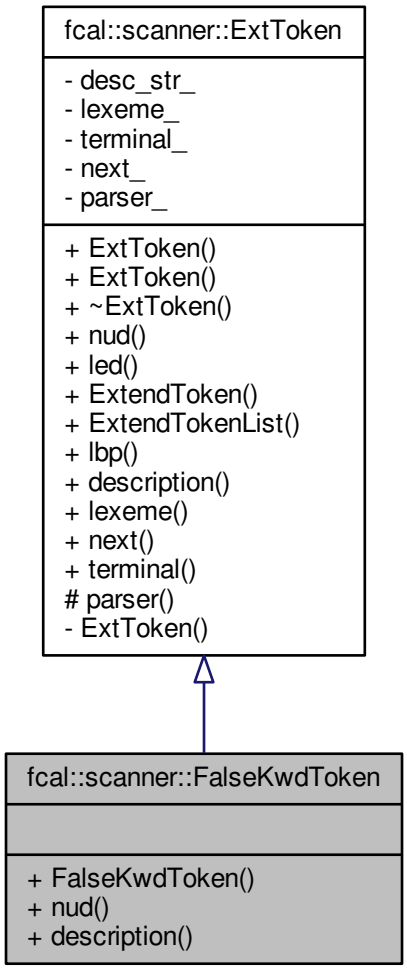
- include/ext_token.h
- src/ext_token.cc

7.14 fcal::scanner::FalseKwdToken Class Reference

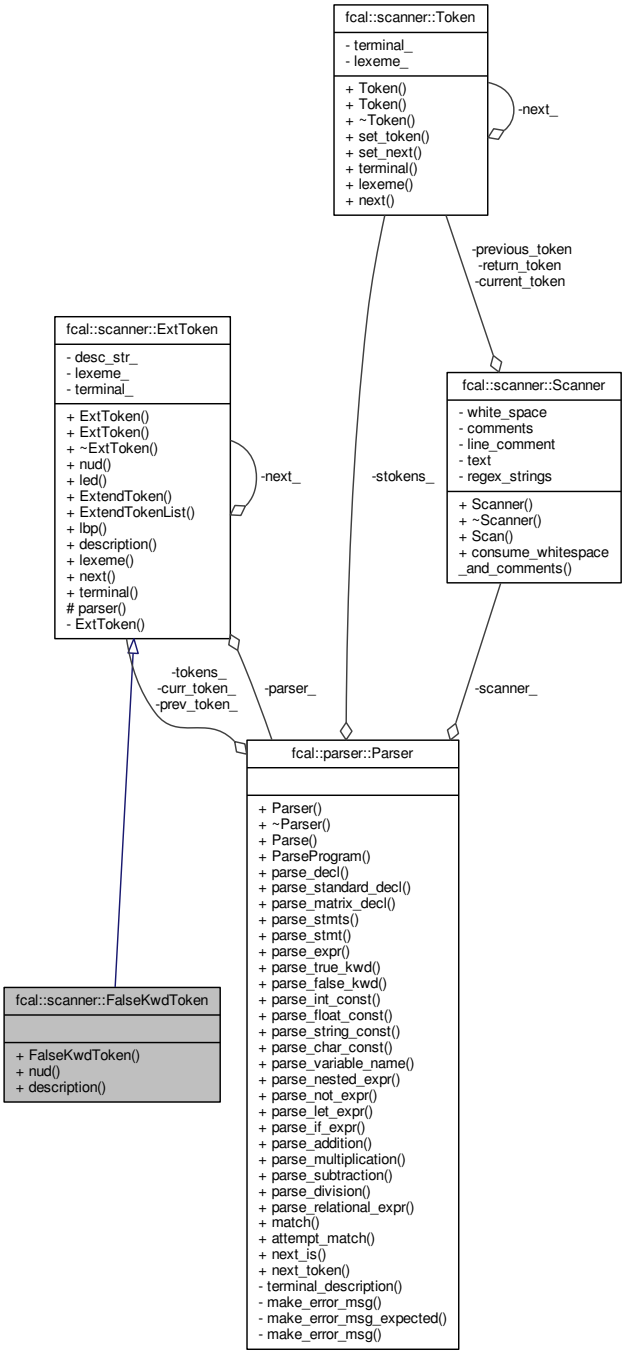
False Kwd.

`#include <ext_token.h>`

Inheritance diagram for fcal::scanner::FalseKwdToken:



Collaboration diagram for fcal::scanner::FalseKwdToken:



Public Member Functions

- FalseKwdToken (parser::Parser *p, Token *t)
- parser::ParseResult nud ()
- std::string description ()

Additional Inherited Members

7.14.1 Detailed Description

False Kwd.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 fcal::scanner::FalseKwdToken::FalseKwdToken (parser::Parser * *p*, Token * *t*) [inline]

7.14.3 Member Function Documentation

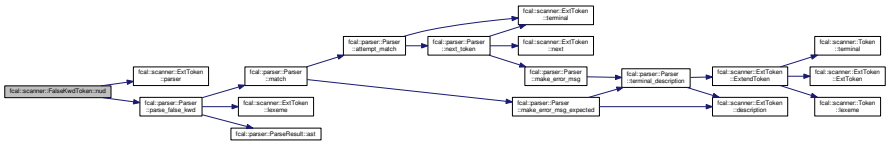
7.14.3.1 std::string fcal::scanner::FalseKwdToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.14.3.2 parser::ParseResult fcal::scanner::FalseKwdToken::nud (void) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



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

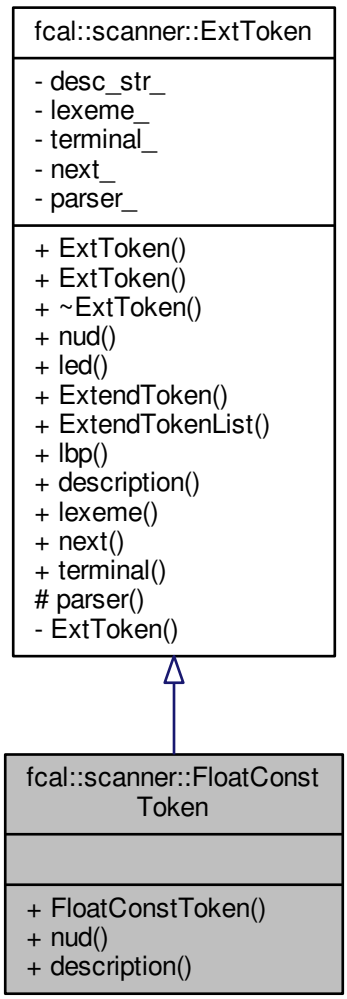
- include/ext_token.h

7.15 fcal::scanner::FloatConstToken Class Reference

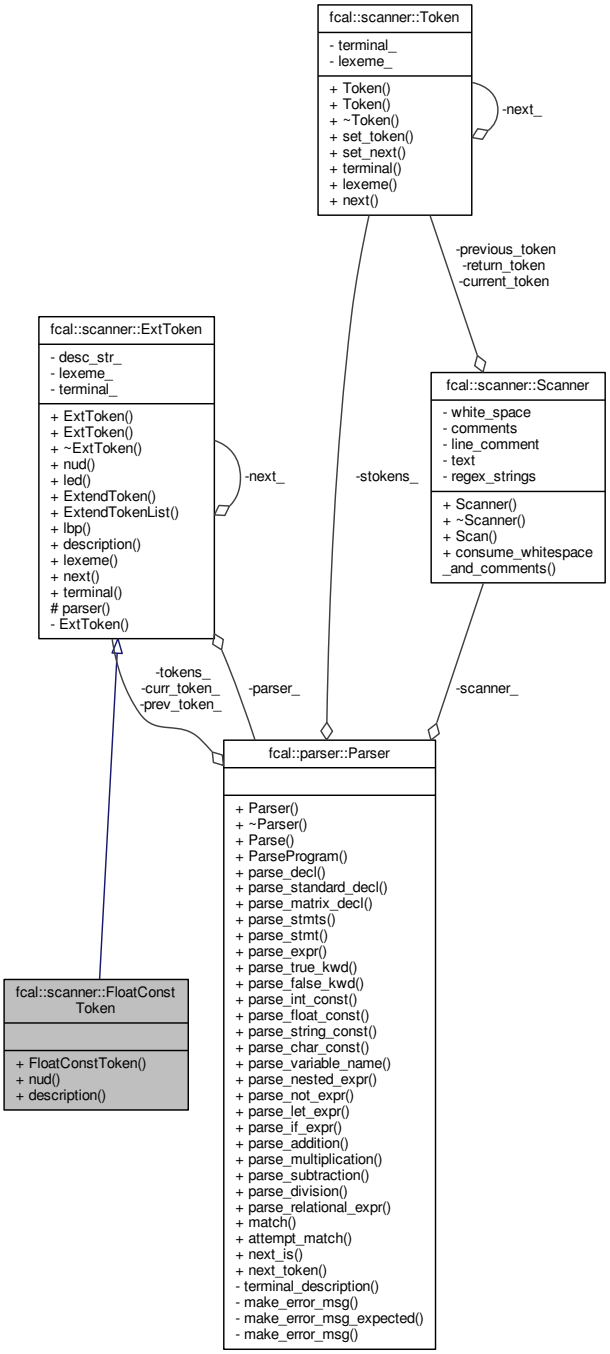
Float Const.

`#include <ext_token.h>`

Inheritance diagram for fcal::scanner::FloatConstToken:



Collaboration diagram for fcal::scanner::FloatConstToken:



Public Member Functions

- FloatConstToken (parser::Parser ∗p, Token ∗t)
- parser::ParseResult nud ()
- std::string description ()

Additional Inherited Members

7.15.1 Detailed Description

Float Const.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 fcal::scanner::FloatConstToken::FloatConstToken (parser::Parser * *p*, Token * *t*) [inline]

7.15.3 Member Function Documentation

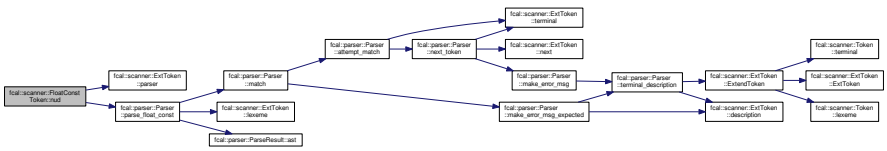
7.15.3.1 std::string fcal::scanner::FloatConstToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.15.3.2 parser::ParseResult fcal::scanner::FloatConstToken::nud (void) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



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

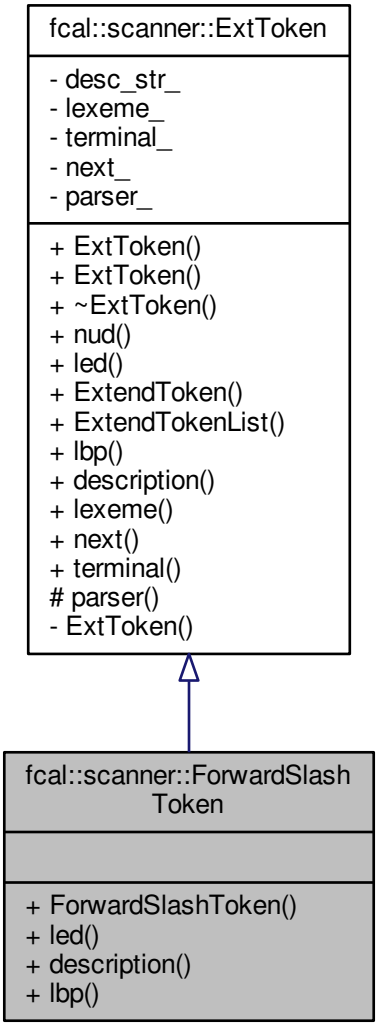
- include/ext_token.h

7.16 fcal::scanner::ForwardSlashToken Class Reference

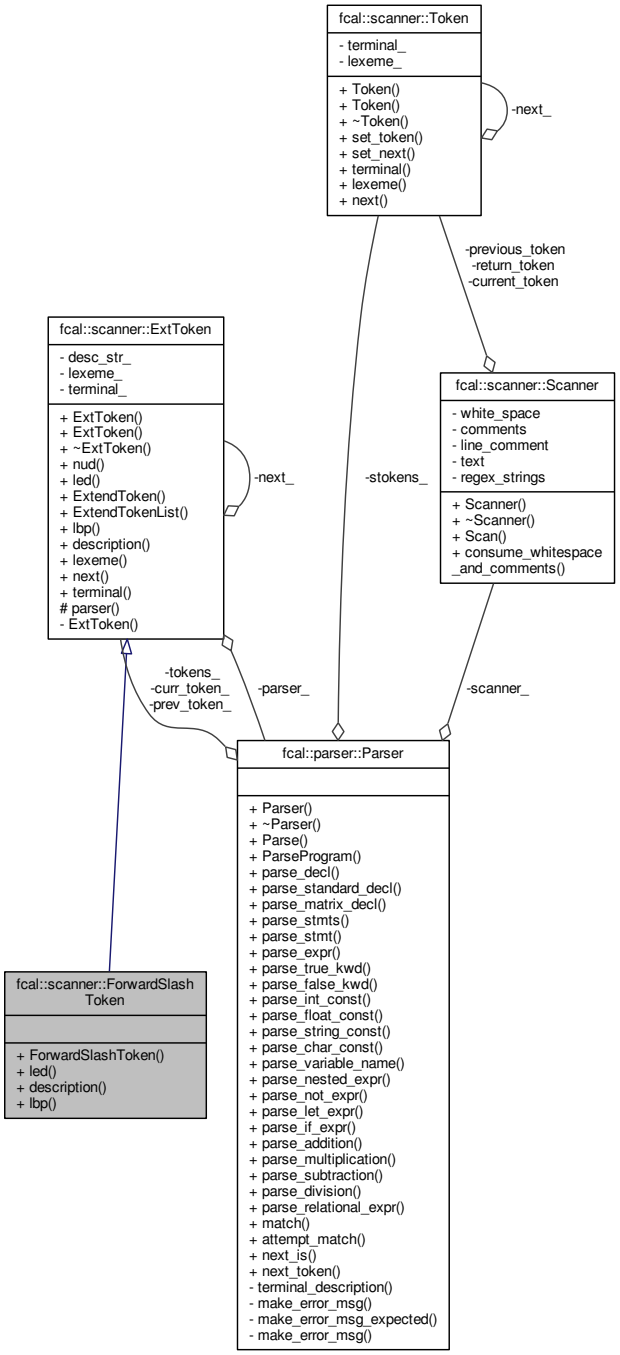
ForwardSlash.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::ForwardSlashToken:



Collaboration diagram for fcal::scanner::ForwardSlashToken:



Public Member Functions

- ForwardSlashToken (parser::Parser *p, Token *t)
- parser::ParseResult led (parser::ParseResult left)
- std::string description ()
- int lbp ()

Additional Inherited Members

7.16.1 Detailed Description

ForwardSlash.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 fcal::scanner::ForwardSlashToken::ForwardSlashToken (parser::Parser * *p*, Token * *t*) [inline]

7.16.3 Member Function Documentation

7.16.3.1 std::string fcal::scanner::ForwardSlashToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

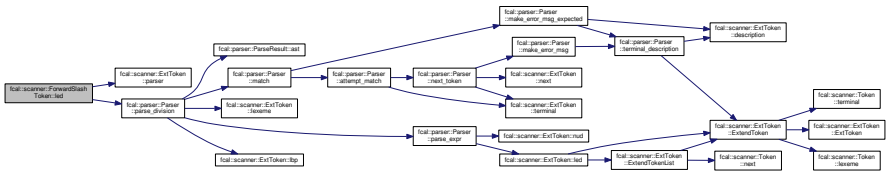
7.16.3.2 int fcal::scanner::ForwardSlashToken::lbp ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.16.3.3 parser::ParseResult fcal::scanner::ForwardSlashToken::led (parser::ParseResult *left*) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



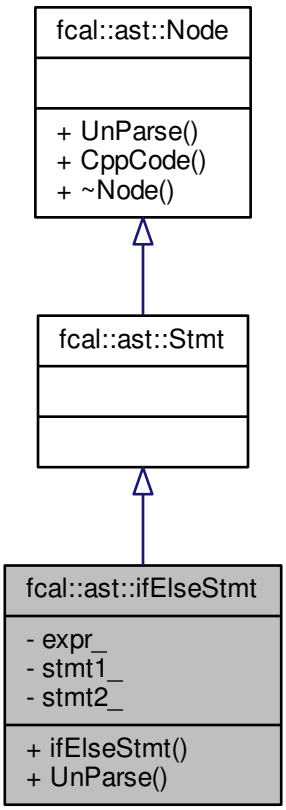
The documentation for this class was generated from the following file:

- include/ext_token.h

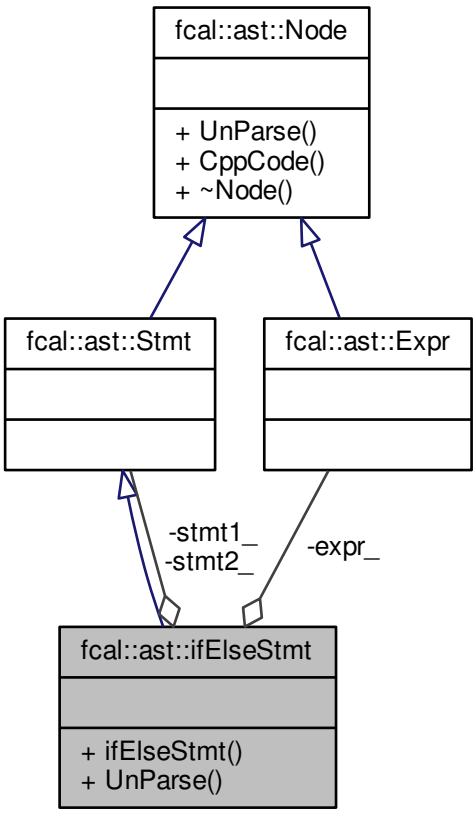
7.17 fcal::ast::ifElseStmt Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::ifElseStmt:



Collaboration diagram for fcal::ast::ifElseStmt:



Public Member Functions

- ifElseStmt (Expr *expr, Stmt *stmt1, Stmt *stmt2)
- std::string UnParse (void)

Private Attributes

- Expr * expr_

  Expr to be evaluated as a condition for if-else.
- Stmt * stmt1_

  Stmt to be executed if expr_ is True.
- Stmt * stmt2_

  Stmt to be executed if expr_ is False.

7.17.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Stmt ::= 'if' '(' Expr ')' Stmt 'else' Stmt

7.17.2 Constructor & Destructor Documentation

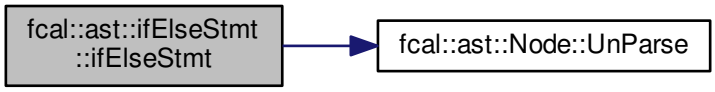
7.17.2.1 fcal::ast::ifElseStmt::ifElseStmt (Expr * *expr*, Stmt * *stmt1*, Stmt * *stmt2*) [inline]

This constructor takes two parameters

Parameters

expr	a Expr* holding the addresss which contains information regarding an Expr to be evaluated as condition for if-else Stmt
stmt1	a Stmt* holding the address which contains information regarding a Stmt to be executed if expr is True
stmt2	a Stmt* holding the address which contains the information regarding a Stmt to be executed if expr is False

Here is the call graph for this function:



7.17.3 Member Function Documentation

7.17.3.1 std::string fcal::ast::ifElseStmt::UnParse (void) [virtual]

Returns the string : "if (" + expr_->UnParse() + ")\n"

- stmt1_->UnParse() + "else " + stmt2_->UnParse()

Reimplemented from fcal::ast::Node.

7.17.4 Member Data Documentation

7.17.4.1 Expr* fcal::ast::ifElseStmt::expr_ [private]

Expr to be evaluated as a condition for if-else.

7.17.4.2 Stmt* fcal::ast::ifElseStmt::stmt1_ [private]

Stmt to be executed if expr_ is True.

7.17.4.3 Stmt* fcal::ast::ifElseStmt::stmt2_ [private]

Stmt to be executed if expr_ is False.

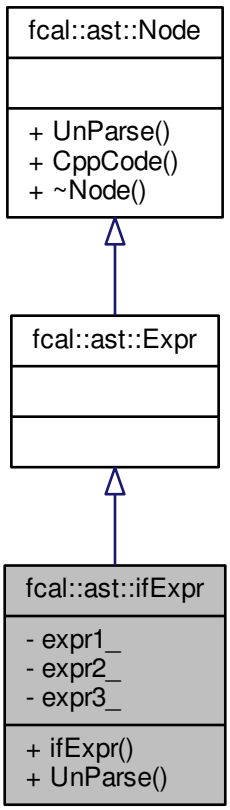
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

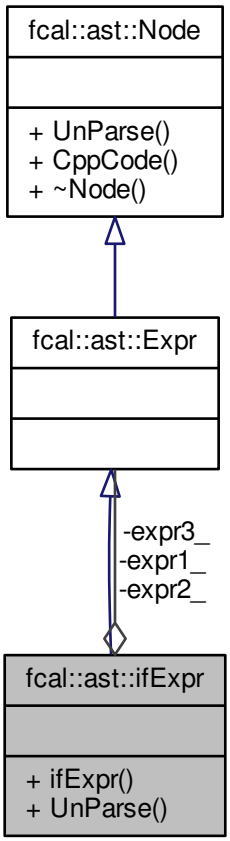
7.18 fcal::ast::ifExpr Class Reference

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

Inheritance diagram for fcal::ast::ifExpr:



Collaboration diagram for fcal::ast::ifExpr:



Public Member Functions

- ifExpr (Expr ∗expr1, Expr ∗expr2, Expr ∗expr3)
- std::string UnParse (void)

Private Attributes

- Expr ∗ expr1_

  Expr to be evaluated as a condition.
- Expr ∗ expr2_

  Expr to be evaluated if expr1_ is True.
- Expr ∗ expr3_

  Expr to be evaluated if expr1_ is False.

7.18.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Expr ::= 'if' Expr 'then' Expr 'else' Expr

7.18.2 Constructor & Destructor Documentation

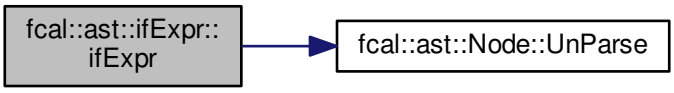
7.18.2.1 fcal::ast::ifExpr::ifExpr (Expr * *expr1*, Expr * *expr2*, Expr * *expr3*) [inline]

This constructor takes three parameters

Parameters

expr1	Expr to be evaluated as a condition
expr2	Expr to be evaluated if expr1 is True
expr3	Expr to be evaluated if expr1 is False

Here is the call graph for this function:



7.18.3 Member Function Documentation

7.18.3.1 std::string fcal::ast::ifExpr::UnParse (void) [virtual]

Returns the string : "if " + expr1_->UnParse() + " then " + expr2_->UnParse() + " else " + expr3_->UnParse();

Reimplemented from fcal::ast::Node.

7.18.4 Member Data Documentation

7.18.4.1 Expr* fcal::ast::ifExpr::expr1_ [private]

Expr to be evaluated as a condition.

7.18.4.2 Expr* fcal::ast::ifExpr::expr2_ [private]

Expr to be evaluated if expr1_ is True.

7.18.4.3 Expr* fcal::ast::ifExpr::expr3_ [private]

Expr to be evaluated if expr1_ is False.

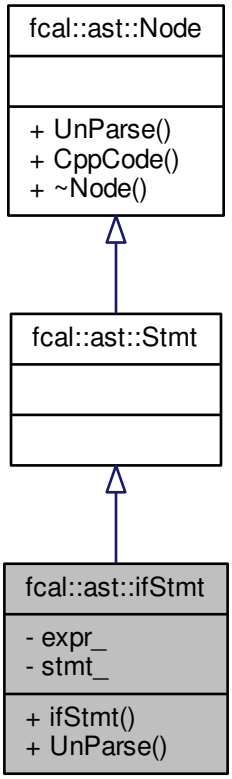
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

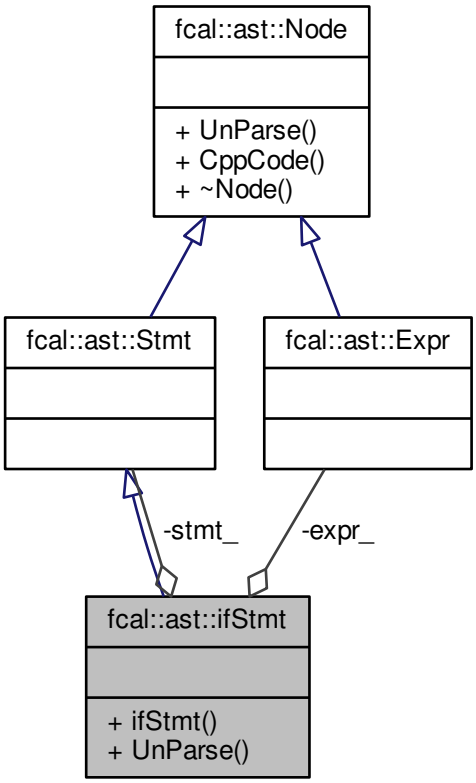
7.19 fcal::ast::ifStmt Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::ifStmt:



Collaboration diagram for fcal::ast::ifStmt:



Public Member Functions

- ifStmt (Expr *expr, Stmt *stmt)
- std::string UnParse (void)

  Returns the string : "if (" + expr_->UnParse() + ")" + stmt_->UnParse()

Private Attributes

- Expr * expr_

  Expr to be evaluated inside the if condition.
- Stmt * stmt_

  Stmt to be executed if expr_ is True.

7.19.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Stmt ::= 'if' '(' Expr ')' Stmt

7.19.2 Constructor & Destructor Documentation

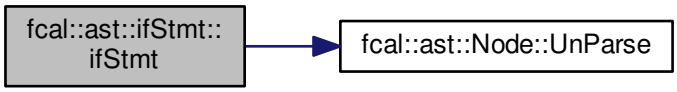
7.19.2.1 fcal::ast::ifStmt::ifStmt (Expr * *expr*, Stmt * *stmt*) [inline]

This constructor takes two parameters

Parameters

expr	a Expr* holding address which contains details of an Expr inside an if condition
stmt	a Stmt* holding address which contains details of a Stmt which is executed if expr is True

Here is the call graph for this function:



7.19.3 Member Function Documentation

7.19.3.1 std::string fcal::ast::ifStmt::UnParse (void) [virtual]

Returns the string : "if (" + expr_->UnParse() + ")" + stmt_->UnParse()

Reimplemented from fcal::ast::Node.

7.19.4 Member Data Documentation

7.19.4.1 Expr* fcal::ast::ifStmt::expr_ [private]

Expr to be evaluated inside the if condition.

7.19.4.2 Stmt* fcal::ast::ifStmt::stmt_ [private]

Stmt to be executed if expr_ is True.

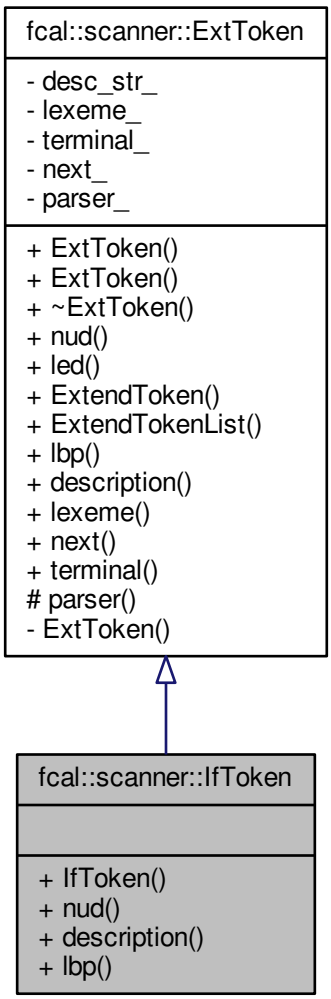
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

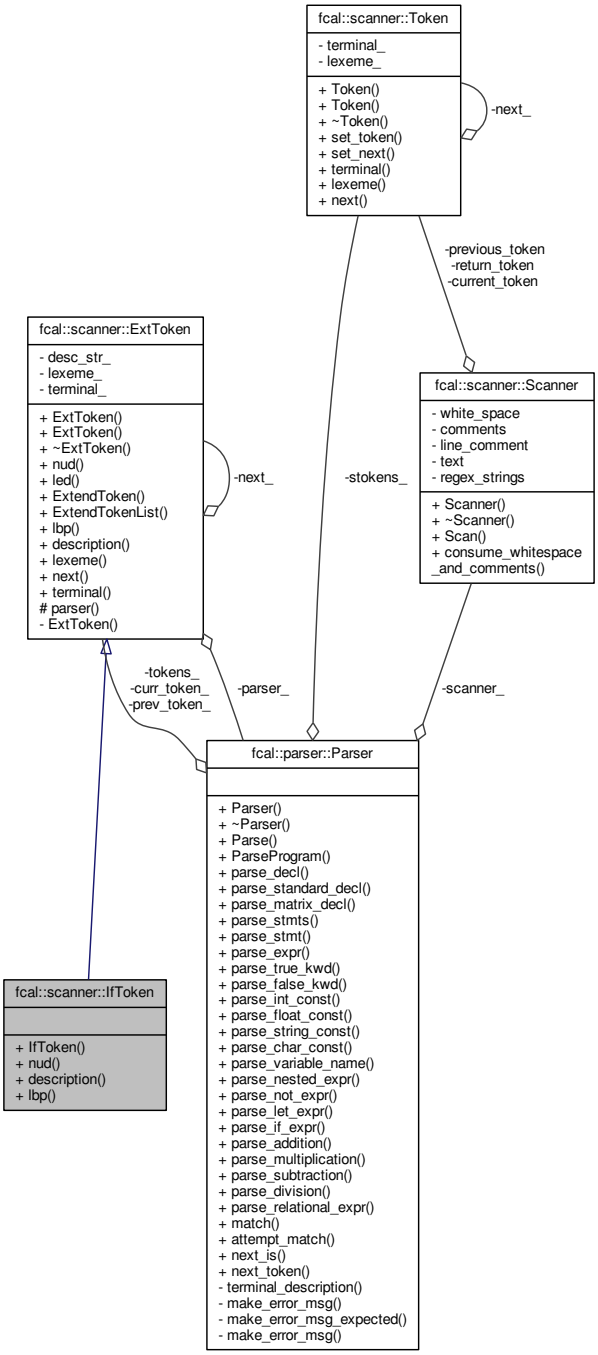
7.20 fcal::scanner::IfToken Class Reference

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::IfToken:



Collaboration diagram for fcal::scanner::IfToken:



Public Member Functions

- IfToken (parser::Parser ∗p, Token ∗t)
- parser::ParseResult nud ()
- std::string description ()
- int lbp ()

Additional Inherited Members

7.20.1 Constructor & Destructor Documentation

7.20.1.1 fcal::scanner::IfToken::IfToken (parser::Parser * *p*, Token * *t*) [inline]

7.20.2 Member Function Documentation

7.20.2.1 std::string fcal::scanner::IfToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

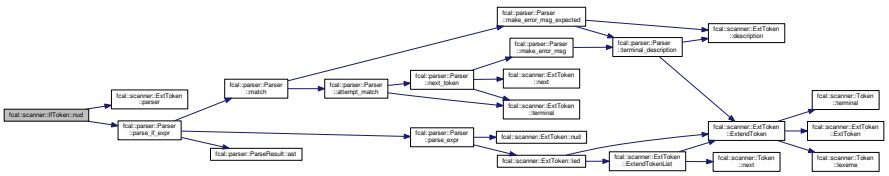
7.20.2.2 int fcal::scanner::IfToken::lbp ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.20.2.3 parser::ParseResult fcal::scanner::IfToken::nud (void) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



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

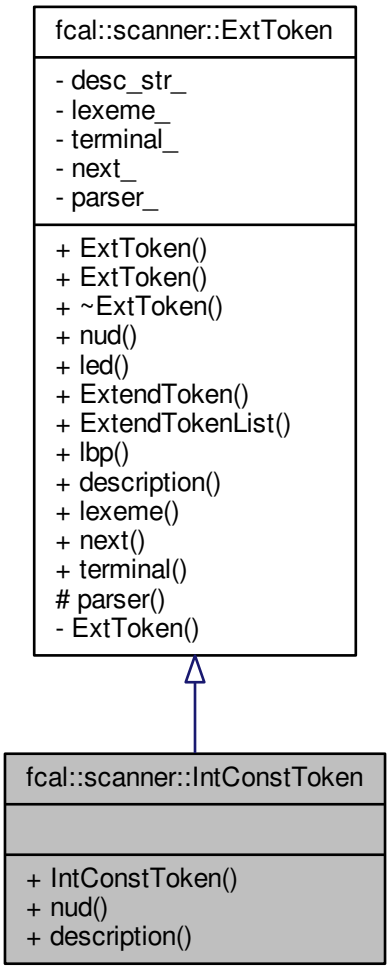
- include/ext_token.h

7.21 fcal::scanner::IntConstToken Class Reference

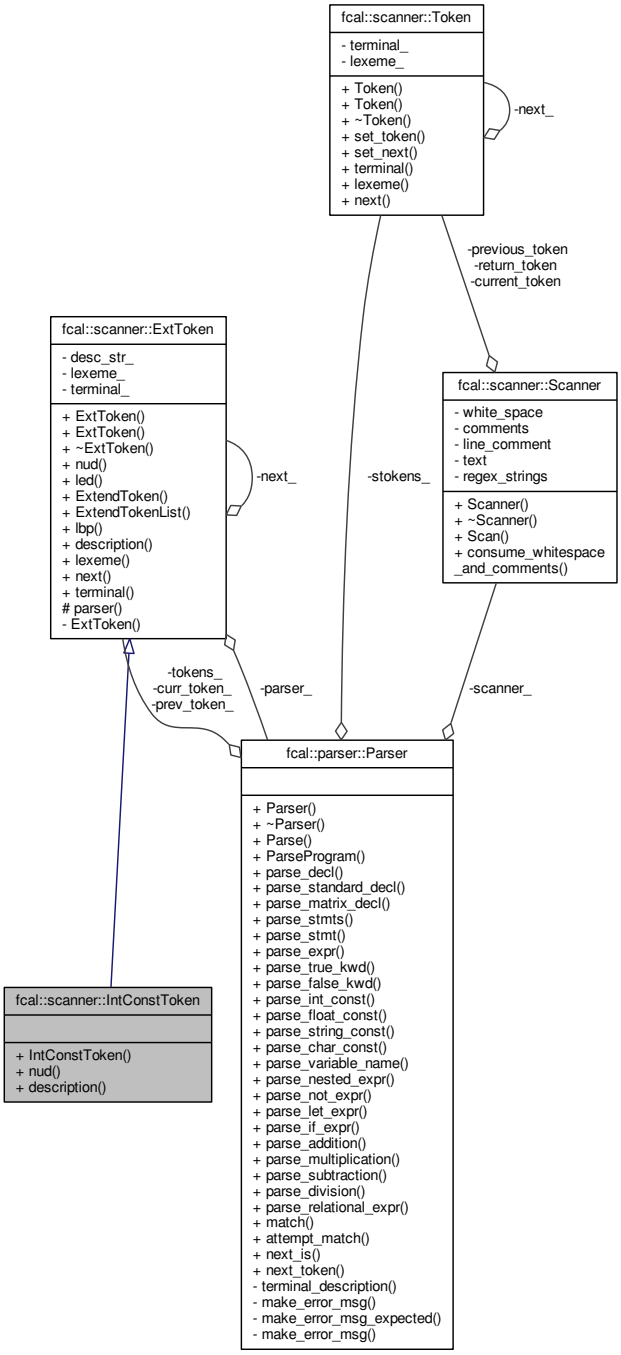
Int Const.

`#include <ext_token.h>`

Inheritance diagram for fcal::scanner::IntConstToken:



Collaboration diagram for fcal::scanner::IntConstToken:



Public Member Functions

- IntConstToken (parser::Parser *p, Token *t)
- parser::ParseResult nud ()
- std::string description ()

Additional Inherited Members

7.21.1 Detailed Description

Int Const.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 fcal::scanner::IntConstToken::IntConstToken (parser::Parser * *p*, Token * *t*) [inline]

7.21.3 Member Function Documentation

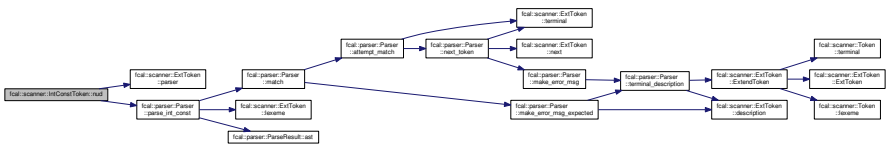
7.21.3.1 std::string fcal::scanner::IntConstToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.21.3.2 parser::ParseResult fcal::scanner::IntConstToken::nud (void) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



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

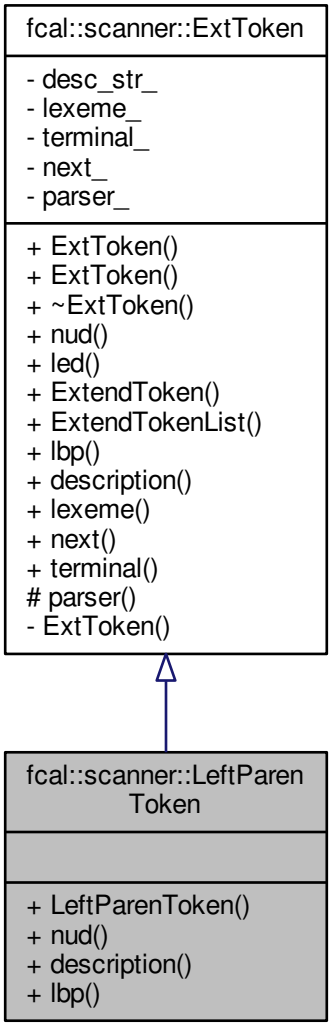
- include/ext_token.h

7.22 fcal::scanner::LeftParenToken Class Reference

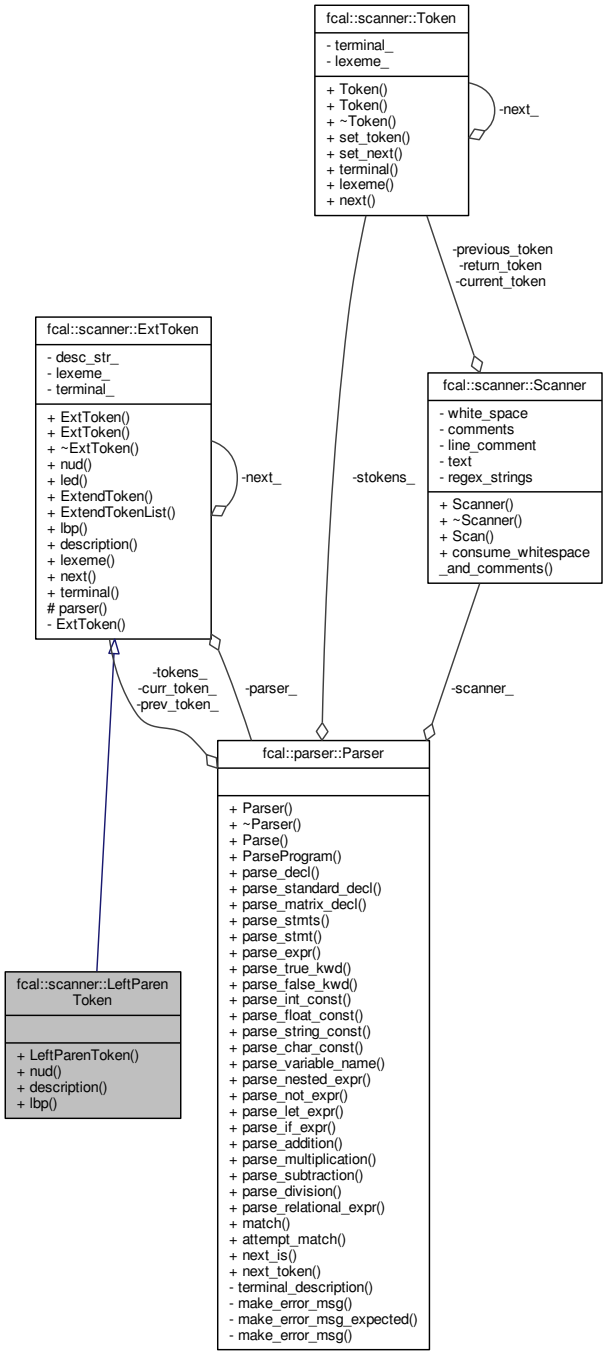
Left Paren.

`#include <ext_token.h>`

Inheritance diagram for fcal::scanner::LeftParenToken:



Collaboration diagram for fcal::scanner::LeftParenToken:



Public Member Functions

- LeftParenToken (parser::Parser *p, Token *t)
- parser::ParseResult nud ()
- std::string description ()
- int lbp ()

Additional Inherited Members

7.22.1 Detailed Description

Left Paren.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 fcal::scanner::LeftParenToken::LeftParenToken (parser::Parser * p, Token * t) [inline]

7.22.3 Member Function Documentation

7.22.3.1 std::string fcal::scanner::LeftParenToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

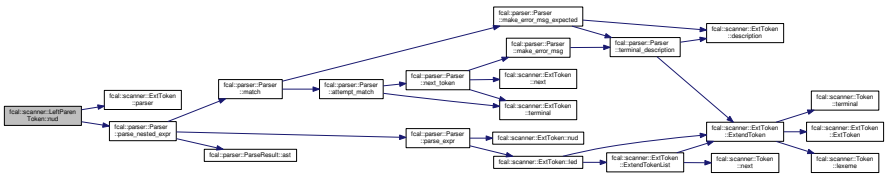
7.22.3.2 int fcal::scanner::LeftParenToken::lbp ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.22.3.3 parser::ParseResult fcal::scanner::LeftParenToken::nud (void) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



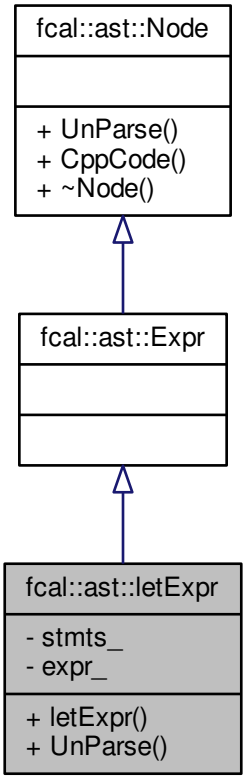
The documentation for this class was generated from the following file:

- include/ext_token.h

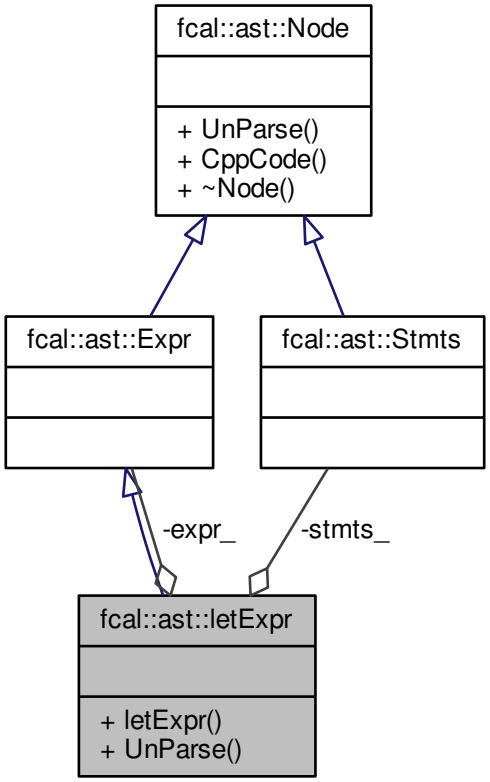
7.23 fcal::ast::letExpr Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::letExpr:



Collaboration diagram for fcal::ast::letExpr:



Public Member Functions

- letExpr (Stmts ∗stmts, Expr ∗expr)
- std::string UnParse (void)

Private Attributes

- Stmts ∗ stmts_

  Stmts in a Let.
- Expr ∗ expr_

  Expr in a let.

7.23.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Expr ::= 'let' Stmts 'in' Expr 'end'

7.23.2 Constructor & Destructor Documentation

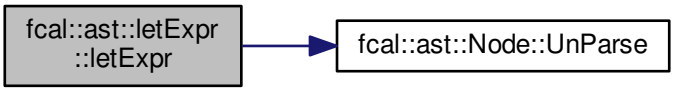
7.23.2.1 fcal::ast::letExpr::letExpr (Stmts * *stmts*, Expr * *expr*) [inline]

This constructor takes two parameters

Parameters

stmts	Stmts in a Let
expr	Expr in a Let

Here is the call graph for this function:



7.23.3 Member Function Documentation

7.23.3.1 std::string fcal::ast::letExpr::UnParse (void) [virtual]

Returns the string : "let " + stmts_->UnParse() + " in " + expr_->UnParse() + " end"

Reimplemented from fcal::ast::Node.

7.23.4 Member Data Documentation

7.23.4.1 Expr* fcal::ast::letExpr::expr_ [private]

Expr in a let.

7.23.4.2 Stmts* fcal::ast::letExpr::stmts_ [private]

Stmts in a Let.

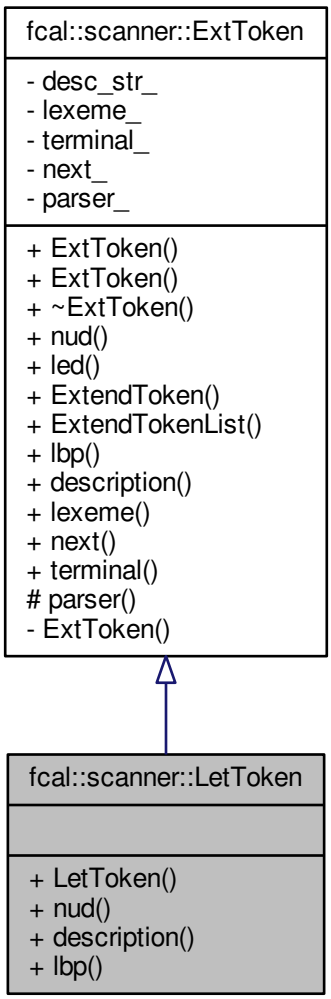
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

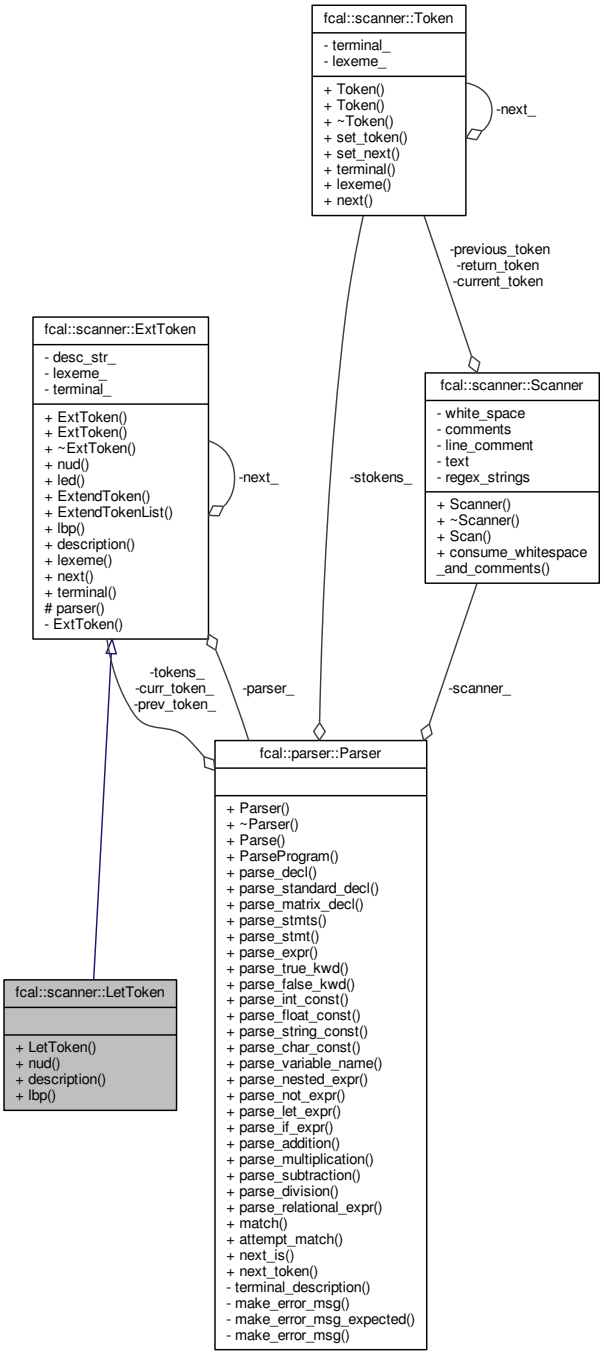
7.24 fcal::scanner::LetToken Class Reference

`#include <ext_token.h>`

Inheritance diagram for fcal::scanner::LetToken:



Collaboration diagram for fcal::scanner::LetToken:



Public Member Functions

- LetToken (parser::Parser *p, Token *t)
- parser::ParseResult nud ()
- std::string description ()
- int lbp ()

Additional Inherited Members

7.24.1 Constructor & Destructor Documentation

7.24.1.1 fcal::scanner::LetToken::LetToken (parser::Parser * *p*, Token * *t*) [inline]

7.24.2 Member Function Documentation

7.24.2.1 std::string fcal::scanner::LetToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

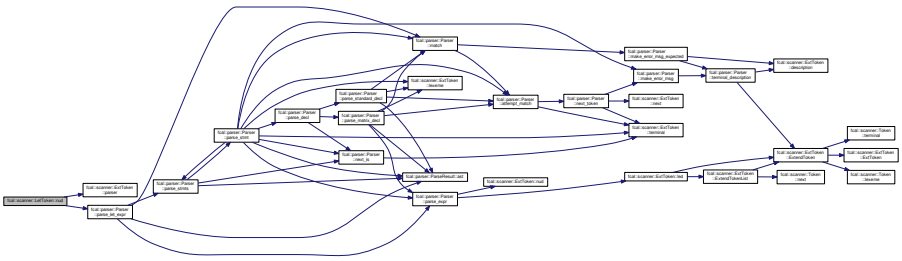
7.24.2.2 int fcal::scanner::LetToken::lbp ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.24.2.3 parser::ParseResult fcal::scanner::LetToken::nud (void) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



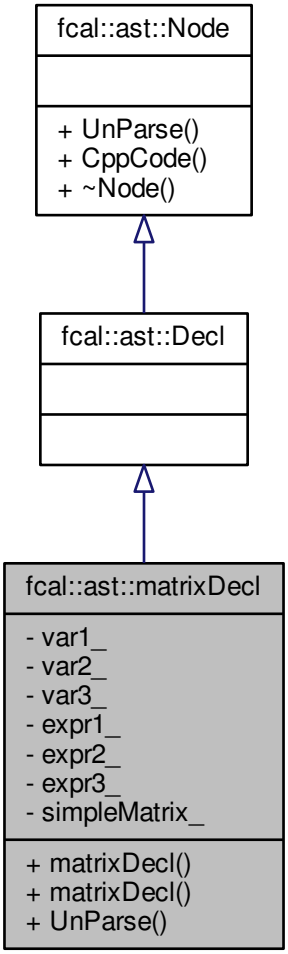
The documentation for this class was generated from the following file:

- include/ext_token.h

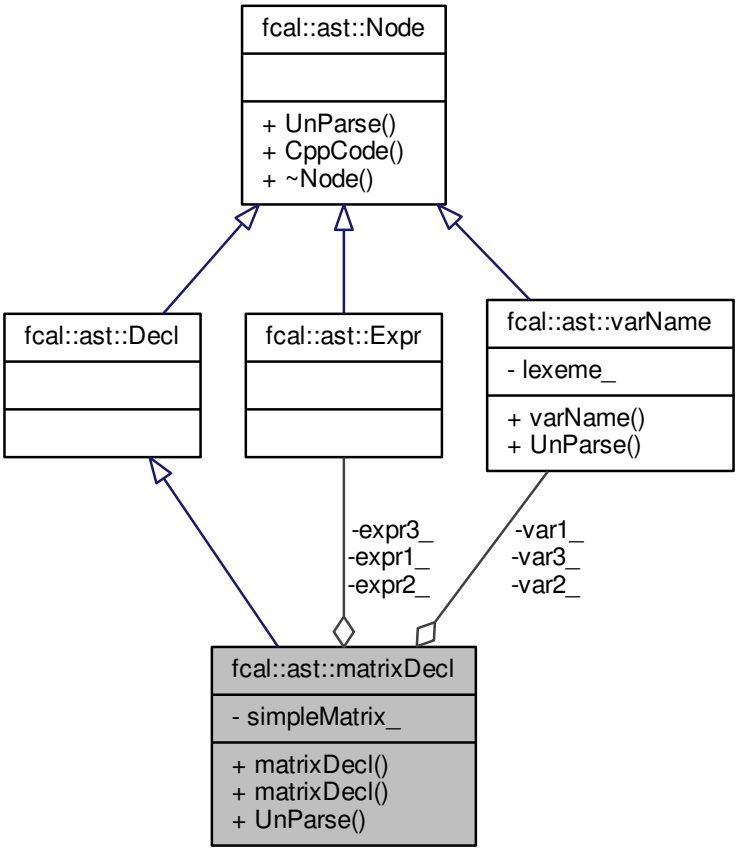
7.25 fcal::ast::matrixDecl Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::matrixDecl:



Collaboration diagram for fcal::ast::matrixDecl:



Public Member Functions

- matrixDecl (varName ∗var1, Expr ∗expr1, bool simpleMatrix)
- matrixDecl (varName ∗var1, varName ∗var2, varName ∗var3, Expr ∗expr1, Expr ∗expr2, Expr ∗expr3, bool simpleMatrix)
- std::string UnParse (void)

Private Attributes

- varName ∗ var1_

  varName representing Matrix name
- varName ∗ var2_

  varName representing Matrix row if simpleMatrix_ is False
- varName ∗ var3_

  varName representing Matrix column if simpleMatrix_ is False
- Expr ∗ expr1_
- Expr ∗ expr2_

expr assigned to column of Matrix if simpleMatrix_ is False

- Expr * expr3_

  expr assigned to matrix if simpleMatrix_ is False

- bool simpleMatrix_

  simpleMatrix distinguishes which constructor is used

7.25.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Decl ::= 'matrix' varName '[' Expr ':' Expr ']' varName ':' varName '=' Expr ';'
Decl ::= 'matrix' varName '=' Expr ';'

7.25.2 Constructor & Destructor Documentation

7.25.2.1 fcal::ast::matrixDecl::matrixDecl (varName * *var1*, Expr * *expr1*, bool *simpleMatrix*) [inline]

This constructor takes three parameters

Parameters

var1	a varName repersenting the matrix
expr1	expression assigned to a matrix
simpleMatrix	bool which is True if Matrix Declaration is Decl ::= 'matrix' varName '=' Expr ';'

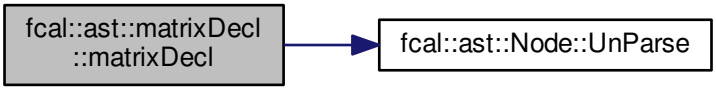
7.25.2.2 fcal::ast::matrixDecl::matrixDecl (varName * *var1*, varName * *var2*, varName * *var3*, Expr * *expr1*, Expr * *expr2*, Expr * *expr3*, bool *simpleMatrix*) [inline]

This constructor takes seven parameters

Parameters

var1	a varName repersenting the matrix
var2	a varName repersenting row of matrix
var3	a varName repersenting column of matrix
expr1	expression for row of Matrix
expr2	expression for column of Matrix
expr3	expression assigned to the matrix
simpleMatrix	bool which is False if Matrix Declaration is Decl ::= 'matrix' varName '[' Expr ':' Expr ']' varName ':' varName '=' Expr ';'

Here is the call graph for this function:



7.25.3 Member Function Documentation

7.25.3.1 std::string fcal::ast::matrixDecl::UnParse (void) [virtual]

If simpleMatrix_ is True then return string : "matrix " + var1_->UnParse() + " = " + expr1_->UnParse() + ";"
Else return : "matrix " + var1_->UnParse() + "[" + expr1_->UnParse() + ":"

- expr2_->UnParse() + "] " + var2_->UnParse() + ":" + var3_->UnParse() + " = "
- expr3_->UnParse() + ";"

Reimplemented from fcal::ast::Node.

7.25.4 Member Data Documentation

7.25.4.1 Expr* fcal::ast::matrixDecl::expr1_ [private]

expr assigned to matrix if simpleMatrix_ is True expr assigned to row of Matrix if simpleMatrix_ is False

7.25.4.2 Expr* fcal::ast::matrixDecl::expr2_ [private]

expr assigned to column of Matrix if simpleMatrix_ is False

7.25.4.3 Expr* fcal::ast::matrixDecl::expr3_ [private]

expr assigned to matrix if simpleMatrix_ is False

7.25.4.4 bool fcal::ast::matrixDecl::simpleMatrix_ [private]

simpleMatrix distinguishes which constructor is used

7.25.4.5 varName* fcal::ast::matrixDecl::var1_ [private]

varName representing Matrix name

7.25.4.6 varName* fcal::ast::matrixDecl::var2_ [private]

varName representing Matrix row if simpleMatrix_ is False

7.25.4.7 varName* fcal::ast::matrixDecl::var3_ [private]

varName representing Matrix column if simpleMatrix_ is False

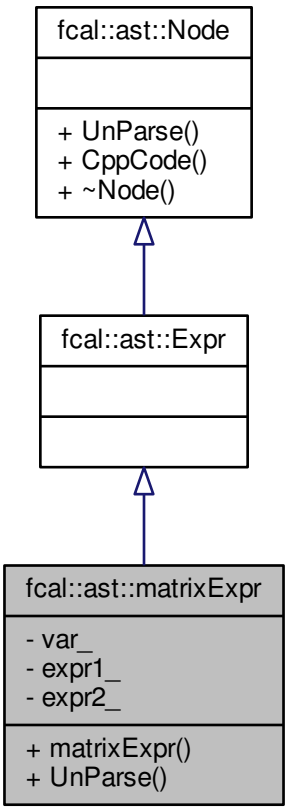
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

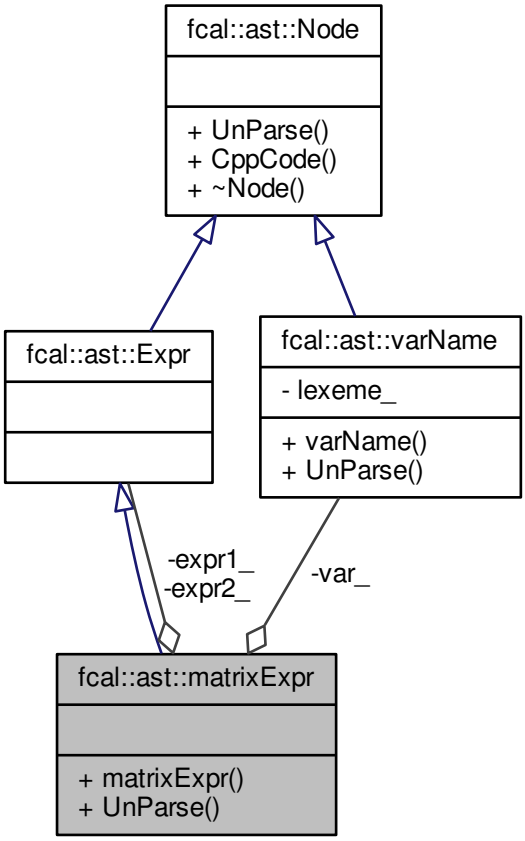
7.26 fcal::ast::matrixExpr Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::matrixExpr:



Collaboration diagram for fcal::ast::matrixExpr:



Public Member Functions

- matrixExpr (varName ∗var, Expr ∗expr1, Expr ∗expr2)
- std::string UnParse (void)

Private Attributes

- varName ∗ var_

  varName representing matrix name
- Expr ∗ expr1_

  Expr which represents row of Matrix.
- Expr ∗ expr2_

  Expr which represents column of Matrix.

7.26.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,

Expr ::= varName '[' Expr ':' Expr ']'

7.26.2 Constructor & Destructor Documentation

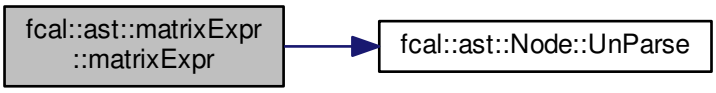
7.26.2.1 fcal::ast::matrixExpr::matrixExpr (varName * *var*, Expr * *expr1*, Expr * *expr2*) [inline]

This constructor takes three parameters

Parameters

var	varName representing matrix name
expr1	Expr which represents row of Matrix
expr2	Expr which represents column of Matrix

Here is the call graph for this function:



7.26.3 Member Function Documentation

7.26.3.1 std::string fcal::ast::matrixExpr::UnParse (void) [virtual]

Returns the string : var_->UnParse() + " [" + expr1_->UnParse() + ":" + expr2_->UnParse() + "]"

Reimplemented from fcal::ast::Node.

7.26.4 Member Data Documentation

7.26.4.1 Expr* fcal::ast::matrixExpr::expr1_ [private]

Expr which represents row of Matrix.

7.26.4.2 Expr* fcal::ast::matrixExpr::expr2_ [private]

Expr which represents column of Matrix.

7.26.4.3 varName* fcal::ast::matrixExpr::var_ [private]

varName representing matrix name

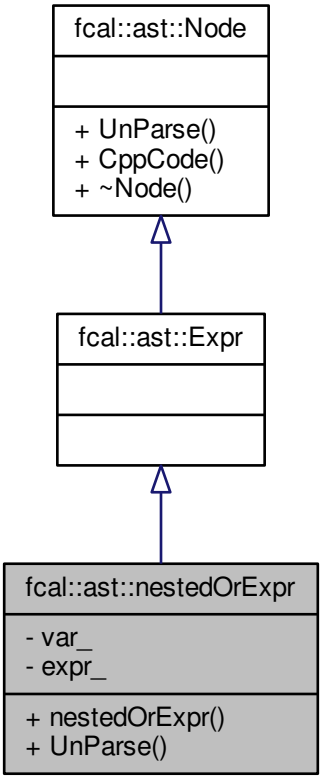
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

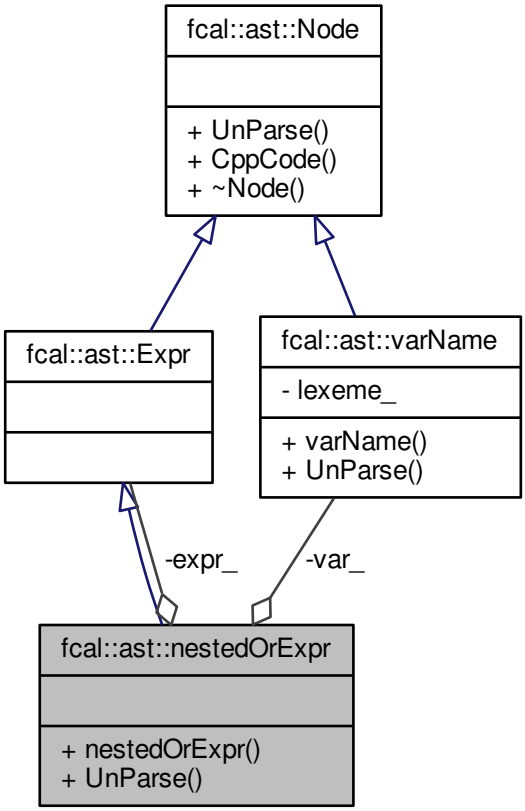
7.27 fcal::ast::nestedOrExpr Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::nestedOrExpr:



Collaboration diagram for fcal::ast::nestedOrExpr:



Public Member Functions

- nestedOrExpr (varName ∗var, Expr ∗expr)
- std::string UnParse (void)

  Returns the string : var_->UnParse() + " (" + expr_->UnParse() + ")".

Private Attributes

- varName ∗ var_

  varName before Expr enclosed in paranthesis
- Expr ∗ expr_

  Expr contained in paranthesis.

7.27.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Expr ::= varName '(' Expr ')'

7.27.2 Constructor & Destructor Documentation

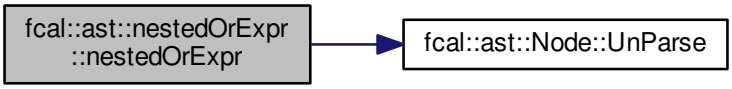
7.27.2.1 fcal::ast::nestedOrExpr::nestedOrExpr (varName * *var*, Expr * *expr*) [inline]

This constructor takes 2 parameters

Parameters

var	varName before Expr enclosed in paranthesis
expr	Expr contained in paranthesis

Here is the call graph for this function:



7.27.3 Member Function Documentation

7.27.3.1 std::string fcal::ast::nestedOrExpr::UnParse (void) [virtual]

Returns the string : var_->UnParse() + " (" + expr_->UnParse() + ")".

Reimplemented from fcal::ast::Node.

7.27.4 Member Data Documentation

7.27.4.1 Expr* fcal::ast::nestedOrExpr::expr_ [private]

Expr contained in paranthesis.

7.27.4.2 varName* fcal::ast::nestedOrExpr::var_ [private]

varName before Expr enclosed in paranthesis

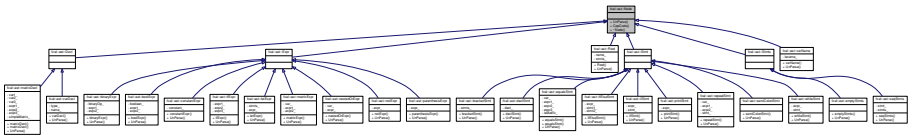
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

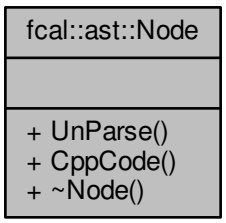
7.28 fcal::ast::Node Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::Node:



Collaboration diagram for fcal::ast::Node:



Public Member Functions

- virtual std::string UnParse (void)
- virtual std::string CppCode (void)
- virtual ~Node (void)

  virtual destructor for polymorphism

7.28.1 Detailed Description

This is the abstract class for ast heirarchy. Root, Stmts, Stmt, Decl, Expr and varName are derived from this class.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 virtual fcal::ast::Node::~Node (void) [inline], [virtual]

virtual destructor for polymorphism

7.28.3 Member Function Documentation

7.28.3.1 virtual std::string fcal::ast::Node::CppCode (void) [inline], [virtual]

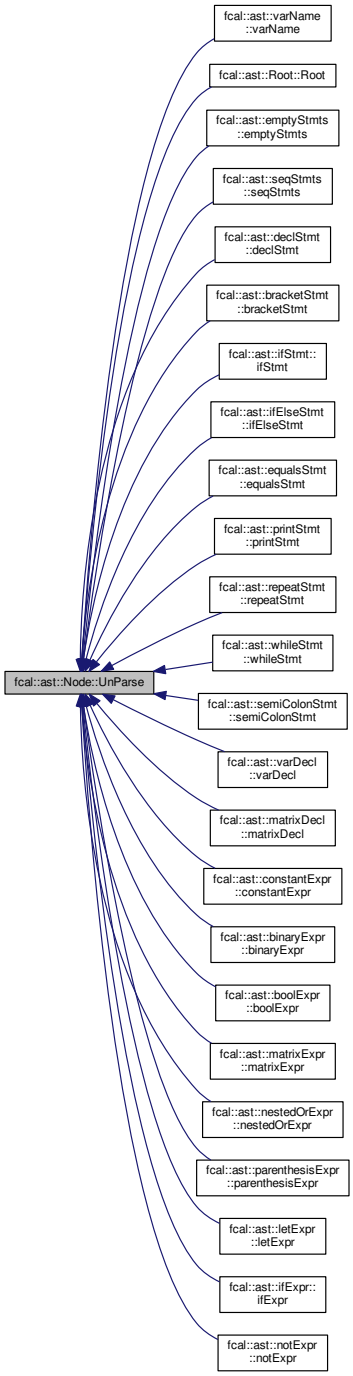
CppCode is a polymorphic function which is implemented by concrete classes in the ast.

7.28.3.2 virtual std::string fcal::ast::Node::UnParse (void) [inline], [virtual]

UnParse is a polymorphic function which is implemented by concrete classes in the ast.

Reimplemented in fcal::ast::notExpr, fcal::ast::ifExpr, fcal::ast::letExpr, fcal::ast::parenthesisExpr, fcal::ast::nestedOrExpr, fcal::ast::matrixExpr, fcal::ast::boolExpr, fcal::ast::binaryExpr, fcal::ast::constantExpr, fcal::ast::matrixDecl, fcal::ast::varDecl, fcal::ast::semiColonStmt, fcal::ast::whileStmt, fcal::ast::repeatStmt, fcal::ast::printStmt, fcal::ast::equalsStmt, fcal::ast::ifElseStmt, fcal::ast::ifStmt, fcal::ast::bracketStmt, fcal::ast::declStmt, fcal::ast::seqStmts, fcal::ast::emptyStmts, fcal::ast::Root, and fcal::ast::varName.

Here is the caller graph for this function:



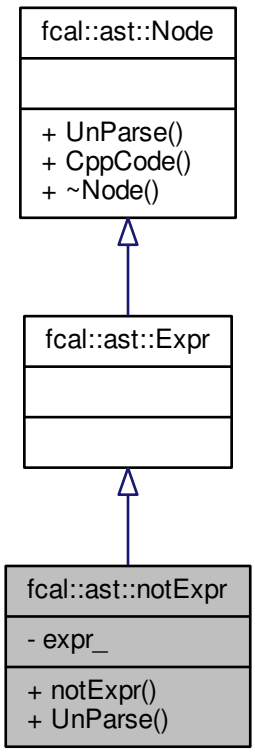
The documentation for this class was generated from the following file:

- include/ast.h

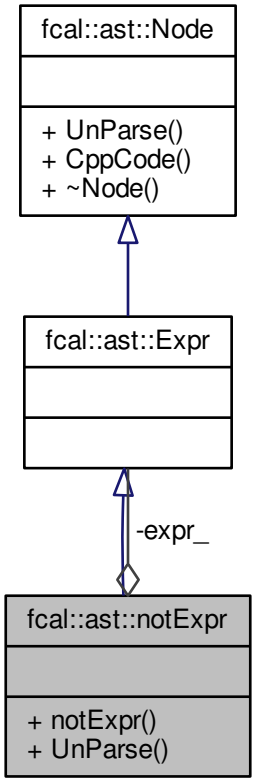
7.29 fcal::ast::notExpr Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::notExpr:



Collaboration diagram for fcal::ast::notExpr:



Public Member Functions

- notExpr (Expr *expr)
- std::string UnParse (void)

  Returns the string : "!" + expr_->UnParse();.

Private Attributes

- Expr * expr_

  Expr whose boolean result will be negated.

7.29.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,

Expr ::= '!' Expr

7.29.2 Constructor & Destructor Documentation

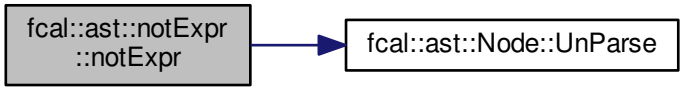
7.29.2.1 fcal::ast::notExpr::notExpr (Expr * *expr*) [inline], [explicit]

This constructor takes one parameter

Parameters

expr	Expr whose boolean result will be negated

Here is the call graph for this function:



7.29.3 Member Function Documentation

7.29.3.1 std::string fcal::ast::notExpr::UnParse (void) [virtual]

Returns the string : "!" + expr_->UnParse();.

Reimplemented from fcal::ast::Node.

7.29.4 Member Data Documentation

7.29.4.1 Expr* fcal::ast::notExpr::expr_ [private]

Expr whose boolean result will be negated.

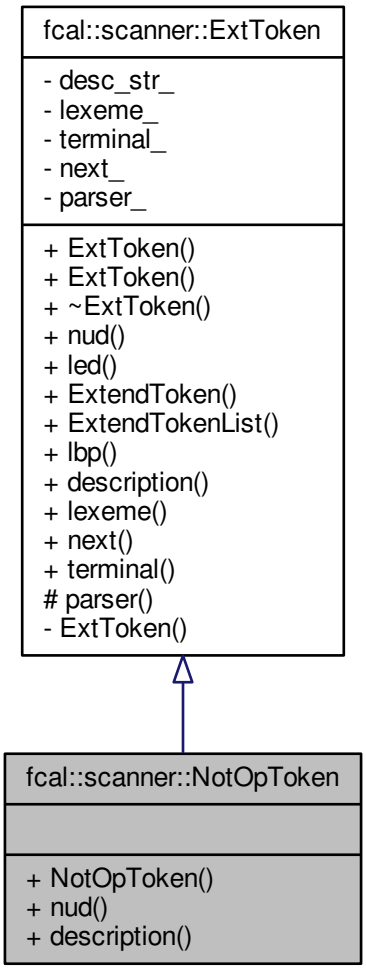
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

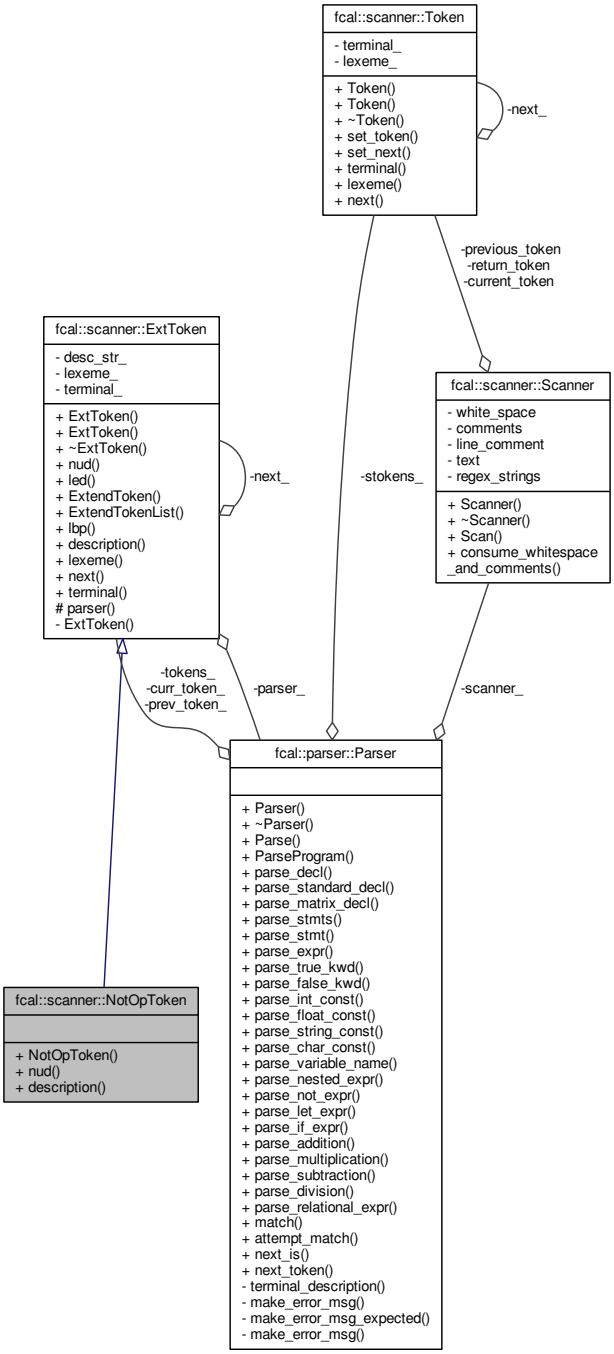
7.30 fcal::scanner::NotOpToken Class Reference

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::NotOpToken:



Collaboration diagram for fcal::scanner::NotOpToken:



Public Member Functions

- NotOpToken (parser::Parser ∗p, Token ∗t)
- parser::ParseResult nud ()
- std::string description ()

Additional Inherited Members

7.30.1 Detailed Description

For each terminal symbol that will play some unique role in the semantic analysis of the program, we need a unique subclass of ExtToken.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 fcal::scanner::NotOpToken::NotOpToken (parser::Parser * *p*, Token * *t*) [inline]

7.30.3 Member Function Documentation

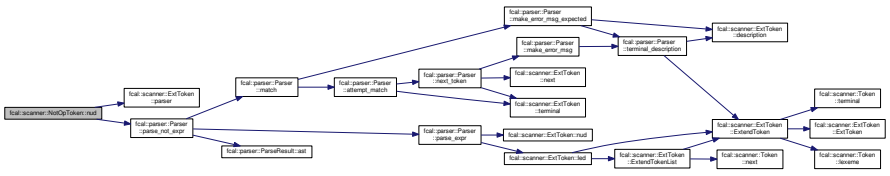
7.30.3.1 std::string fcal::scanner::NotOpToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.30.3.2 parser::ParseResult fcal::scanner::NotOpToken::nud (void) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



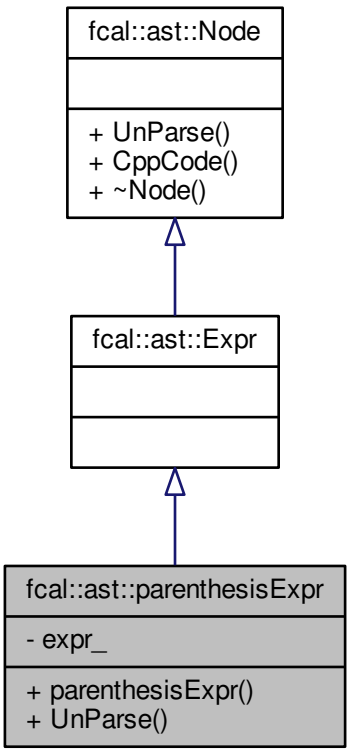
The documentation for this class was generated from the following file:

- include/ext_token.h

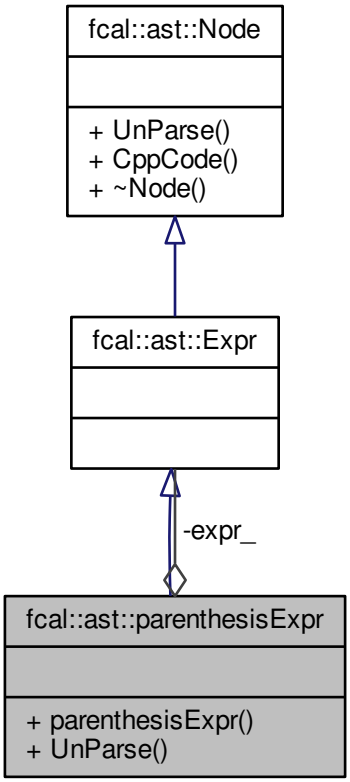
7.31 fcal::ast::parenthesisExpr Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::parenthesisExpr:



Collaboration diagram for fcal::ast::parenthesisExpr:



Public Member Functions

- parenthesisExpr (Expr *expr)
- std::string UnParse (void)

  Returns the string : "(" + expr_->UnParse() + ")".

Private Attributes

- Expr * expr_

  an Expr enclosed in parenthesis

7.31.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Expr ::= '(' Expr ')'

7.31.2 Constructor & Destructor Documentation

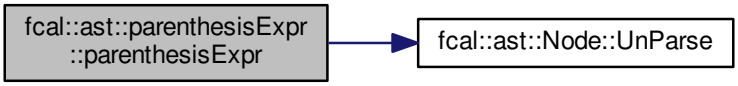
7.31.2.1 fcal::ast::parenthesisExpr::parenthesisExpr (Expr * *expr*) [inline], [explicit]

This constructor takes one paramters

Parameters

expr	an Expr enclosed in parenthesis

Here is the call graph for this function:



7.31.3 Member Function Documentation

7.31.3.1 std::string fcal::ast::parenthesisExpr::UnParse (void) [virtual]

Returns the string : "(" + expr_->UnParse() + ")".

Reimplemented from fcal::ast::Node.

7.31.4 Member Data Documentation

7.31.4.1 Expr* fcal::ast::parenthesisExpr::expr_ [private]

an Expr enclosed in parenthesis

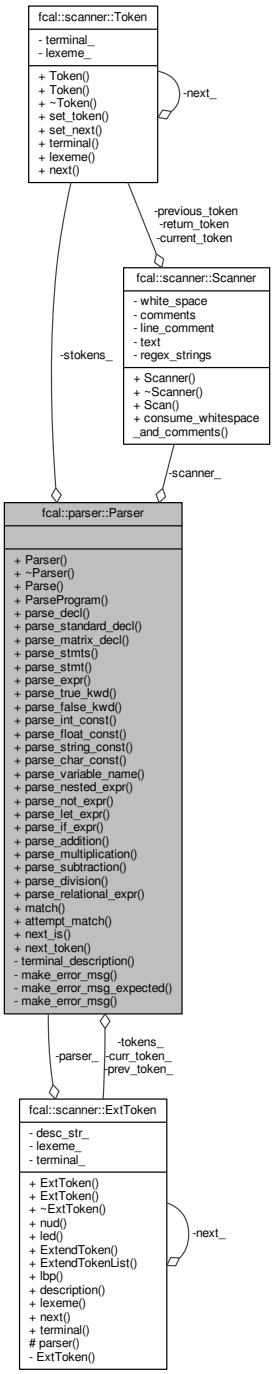
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

7.32 fcal::parser::Parser Class Reference

```
#include <parser.h>
```

Collaboration diagram for fcal::parser::Parser:



Public Member Functions

- Parser (void)
- ∼Parser (void)

  Parser constructors/destructors.
- ParseResult Parse (const char ∗text)
- ParseResult ParseProgram ()

Parser methods for the nonterminals:

- ParseResult parse_decl ()

  Parser::standardDecl.

- ParseResult parse_standard_decl ()

  Parser::matrixDecl.

- ParseResult parse_matrix_decl ()

  Parser::ParseProgram()

- ParseResult parse_stmts ()

  Parse general statements.

- ParseResult parse_stmt ()

  Parse a general statement.

- ParseResult parse_expr (int rbp)

  Parser::parse_stmt.

- ParseResult parse_true_kwd ()

  methods for parsing productions for Expr

- ParseResult parse_false_kwd ()

  Expr ::= falseKwd.

- ParseResult parse_int_const ()

  Expr ::= intConst.

- ParseResult parse_float_const ()

  Expr ::= floatConst.

- ParseResult parse_string_const ()

  Expr ::= stringConst.

- ParseResult parse_char_const ()
- ParseResult parse_variable_name ()

  Expr ::= variableName .....

- ParseResult parse_nested_expr ()

  Expr ::= leftParen Expr rightParen.

- ParseResult parse_not_expr ()

  Expr ::= '!' Expr.

- ParseResult parse_let_expr ()

  Expr ::= 'let' Stmts 'in' Expr 'end'.

- ParseResult parse_if_expr ()

  Expr ::= 'if' Expr 'then' Expr 'else' Expr.

- ParseResult parse_addition (ParseResult left)

  Expr ::= Expr plusSign Expr.

- ParseResult parse_multiplication (ParseResult left)

  Expr ::= Expr star Expr.

- ParseResult parse_subtraction (ParseResult left)
- ParseResult parse_division (ParseResult left)

  Expr ::= Expr forwardSlash Expr.

- ParseResult parse_relational_expr (ParseResult left)
- void match (const scanner::TokenType &tt)

  Helper function used by the parser.

- bool attempt_match (const scanner::TokenType &tt)
- bool next_is (const scanner::TokenType &tt)
- void next_token (void)

Private Member Functions

- std::string terminal_description (const scanner::TokenType &terminal)
- std::string make_error_msg (const scanner::TokenType &terminal)
- std::string make_error_msg_expected (const scanner::TokenType &terminal)
- std::string make_error_msg (const char *msg)

Private Attributes

- scanner::ExtToken * tokens_
- scanner::ExtToken * curr_token_
- scanner::ExtToken * prev_token_
- scanner::Token * stokens_
- scanner::Scanner * scanner_

7.32.1 Constructor & Destructor Documentation

7.32.1.1 fcal::parser::Parser::Parser (void) [inline]

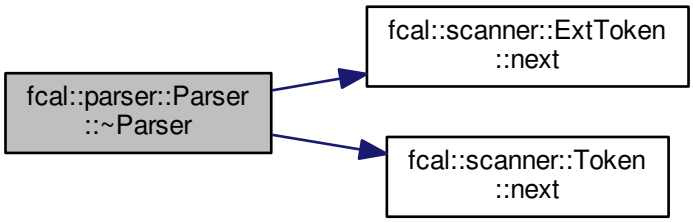
7.32.1.2 fcal::parser::Parser::~Parser (void)

Parser constructors/destructors.

while()

while()

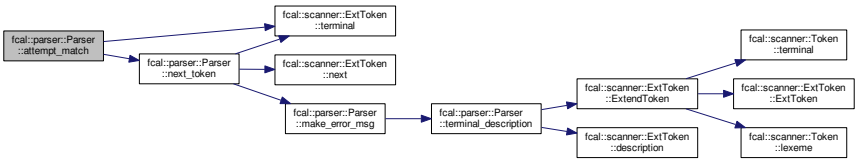
Here is the call graph for this function:



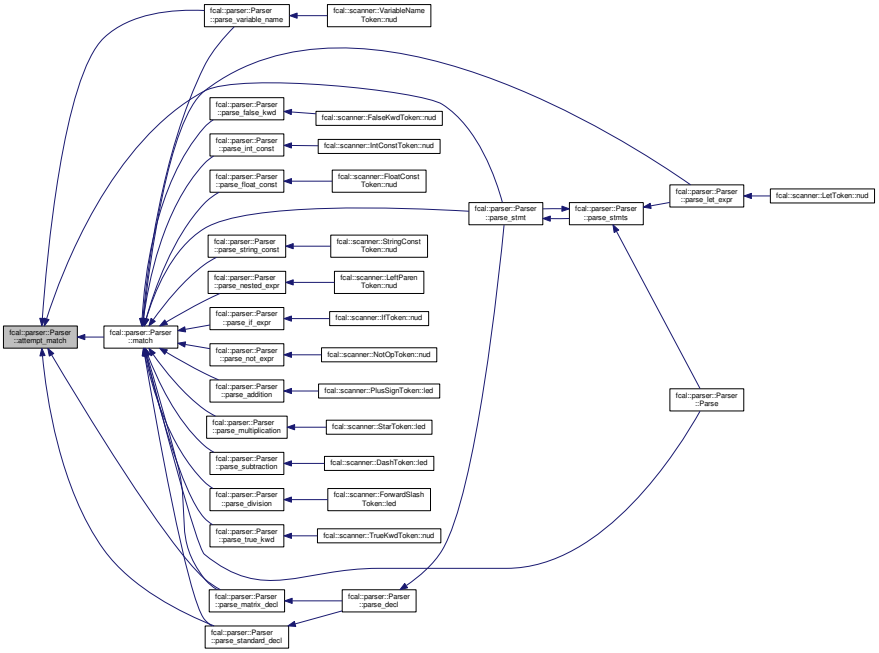
7.32.2 Member Function Documentation

7.32.2.1 bool fcal::parser::Parser::attempt_match (const scanner::TokenType & *tt*)

Here is the call graph for this function:

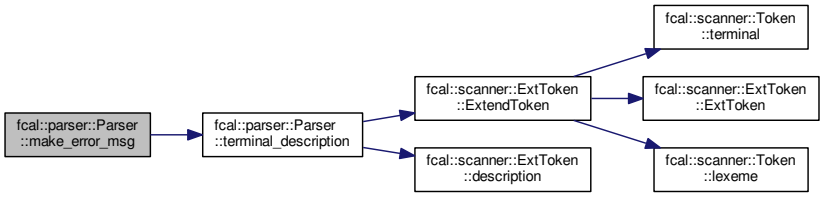


Here is the caller graph for this function:

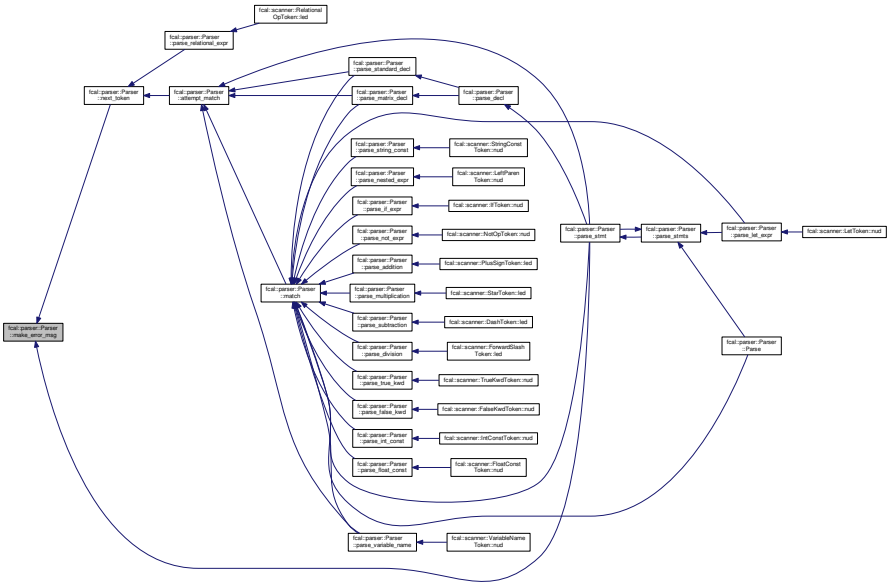


7.32.2.2 std::string fcal::parser::Parser::make_error_msg (const scanner::TokenType & *terminal*) [private]

Here is the call graph for this function:



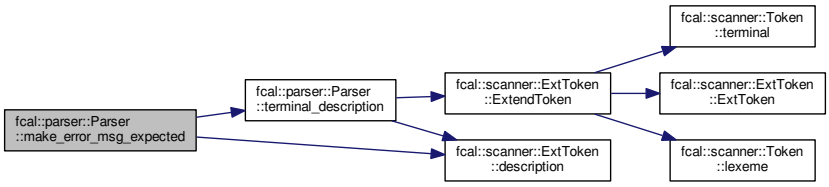
Here is the caller graph for this function:



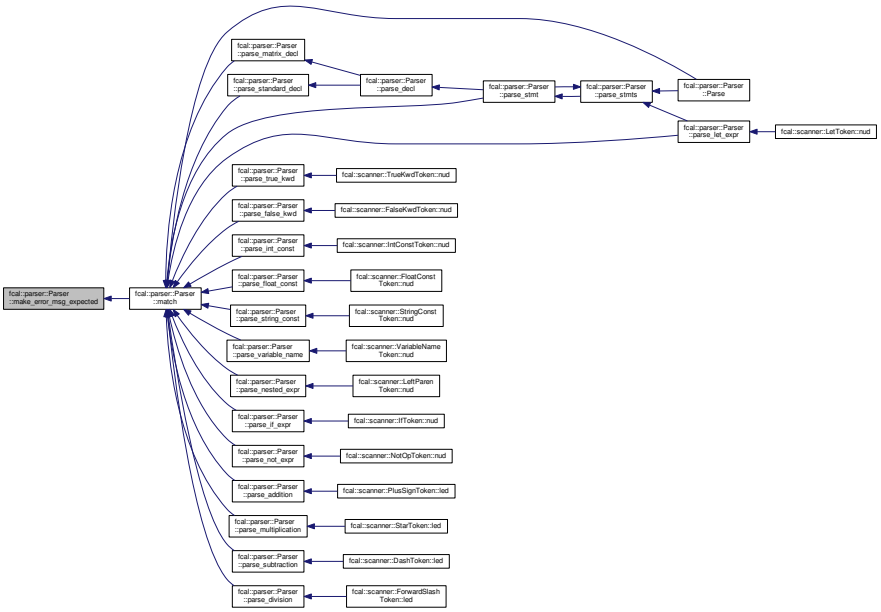
7.32.2.3 std::string fcal::parser::Parser::make_error_msg (const char * *msg*) [private]

7.32.2.4 std::string fcal::parser::Parser::make_error_msg_expected (const scanner::TokenType & *terminal*) [private]

Here is the call graph for this function:



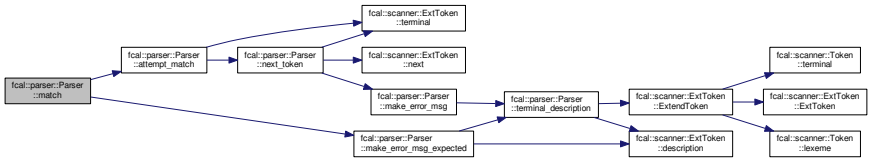
Here is the caller graph for this function:



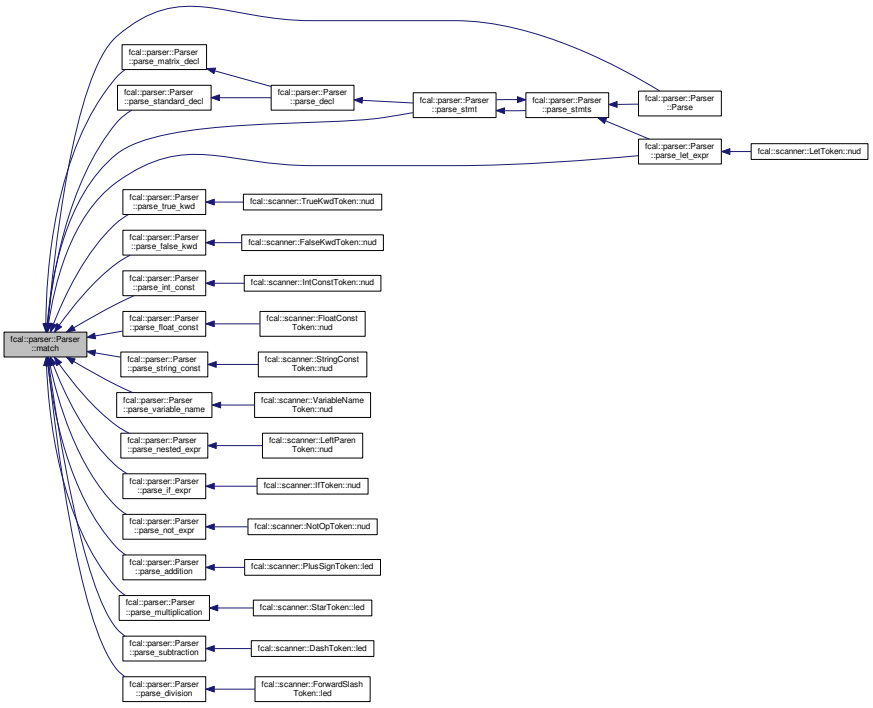
7.32.2.5 void fcal::parser::Parser::match (const scanner::TokenType & *tt*)

Helper function used by the parser.

Here is the call graph for this function:

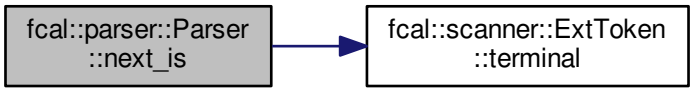


Here is the caller graph for this function:

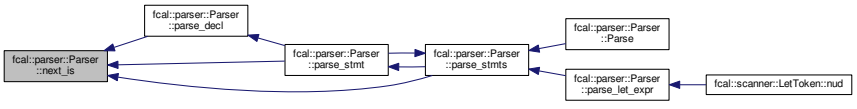


7.32.2.6 bool fcal::parser::Parser::next_is (const scanner::TokenType & *tt*)

Here is the call graph for this function:

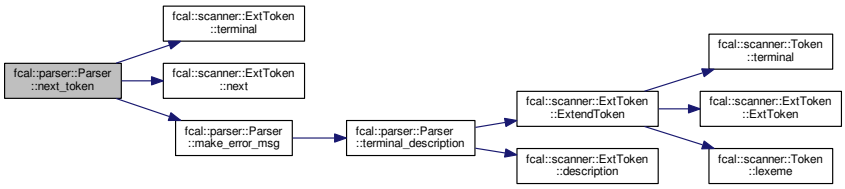


Here is the caller graph for this function:

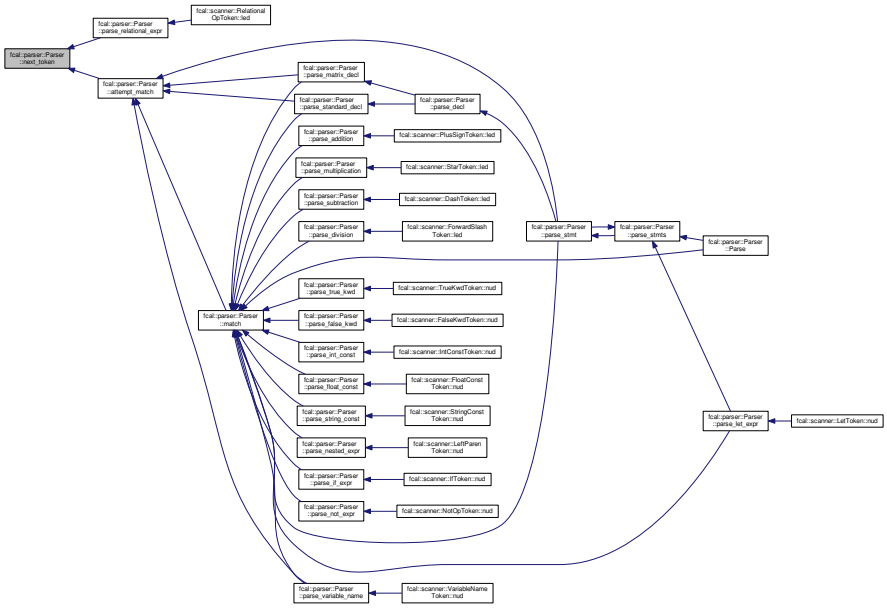


7.32.2.7 void fcal::parser::Parser::next_token (void)

Here is the call graph for this function:



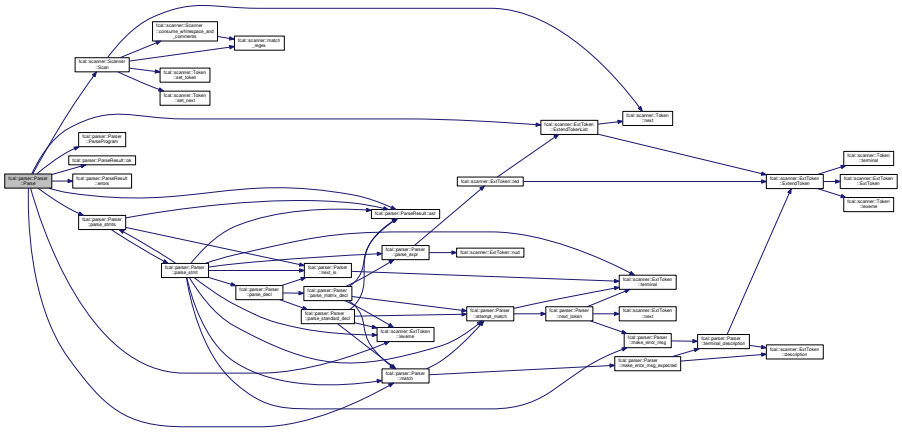
Here is the caller graph for this function:



7.32.2.8 ParseResult fcal::parser::Parser::Parse (const char * *text*)

Parser::~Parser()

Here is the call graph for this function:



7.32.2.9 ParseResult fcal::parser::Parser::parse_addition (ParseResult *left*)

Expr ::= Expr plusSign Expr.

parser has already matched left expression

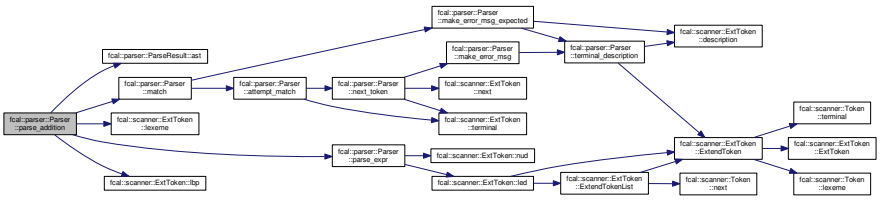
expr1

string +

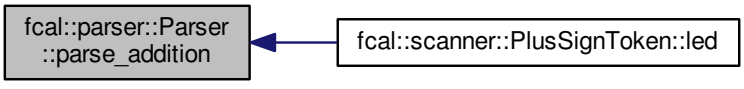
expr2

Create binaryExpr node

Here is the call graph for this function:

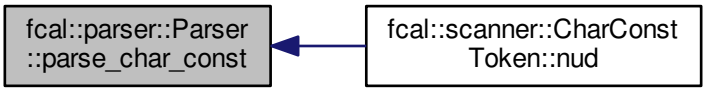


Here is the caller graph for this function:



7.32.2.10 ParseResult fcal::parser::Parser::parse_char_const ( )

Here is the caller graph for this function:



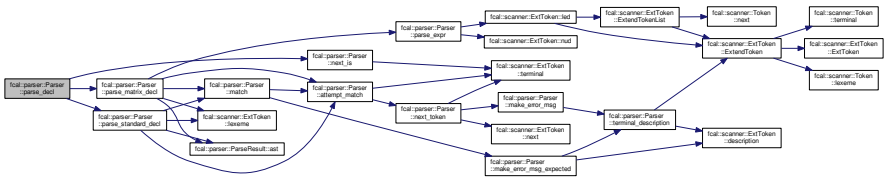
7.32.2.11 ParseResult fcal::parser::Parser::parse_decl ( )

Parser::standardDecl.

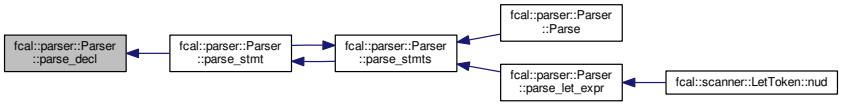
Determines whether declaration is a matrix or standard variable declaration. Decl :: matrix variableName ....

Decl ::= Type variableName semiColon

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.12 ParseResult fcal::parser::Parser::parse_division (ParseResult *left*)

Expr ::= Expr forwardSlash Expr.

parser has already matched left expression

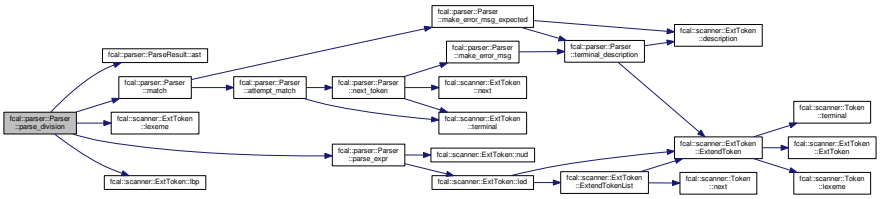
expr1

string /

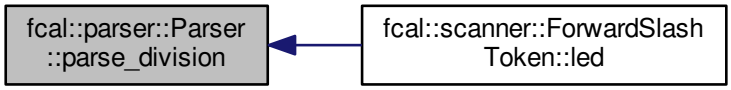
expr2

Create binaryExpr node

Here is the call graph for this function:



Here is the caller graph for this function:

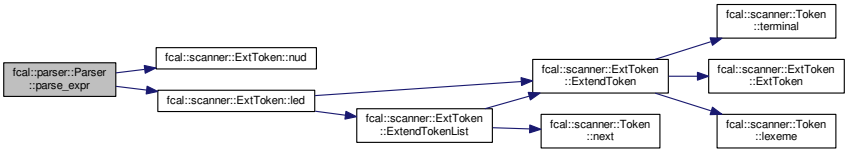


7.32.2.13 ParseResult fcal::parser::Parser::parse_expr (int *rbp*)

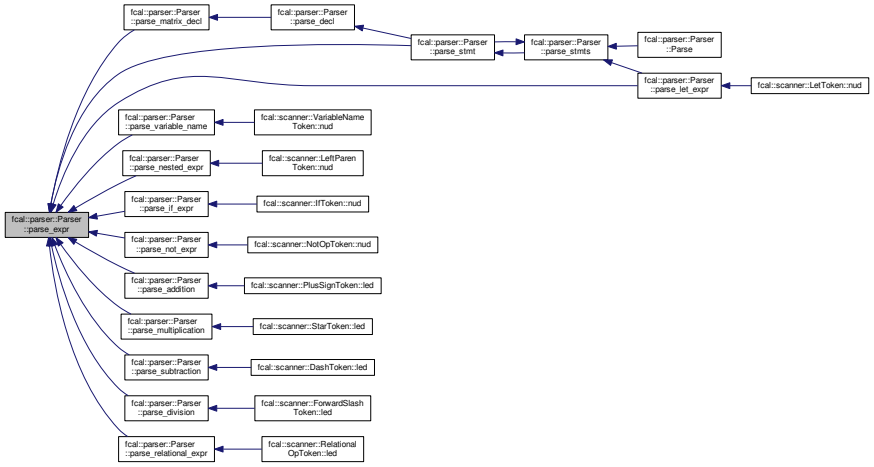
Parser::parse_stmt.

Expr

Here is the call graph for this function:



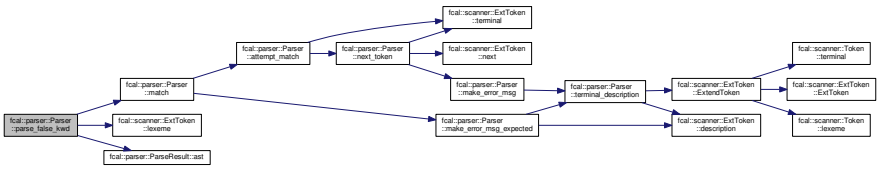
Here is the caller graph for this function:



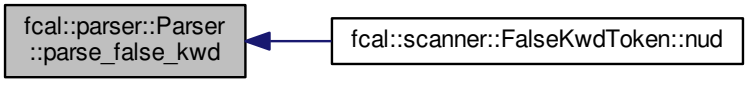
7.32.2.14 ParseResult fcal::parser::Parser::parse_false_kwd ( )

Expr ::= falseKwd.

Here is the call graph for this function:



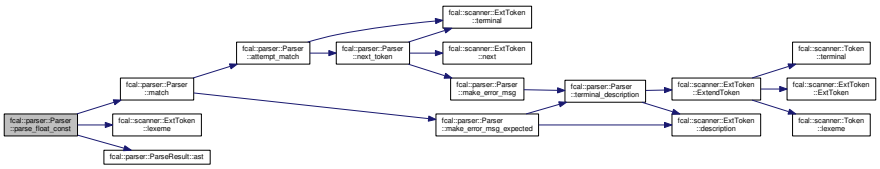
Here is the caller graph for this function:



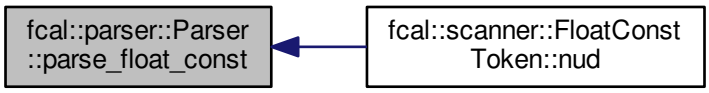
7.32.2.15 ParseResult fcal::parser::Parser::parse_float_const ( )

Expr ::= floatConst.

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.16 ParseResult fcal::parser::Parser::parse_if_expr ( )

Expr ::= 'if' Expr 'then' Expr 'else' Expr.

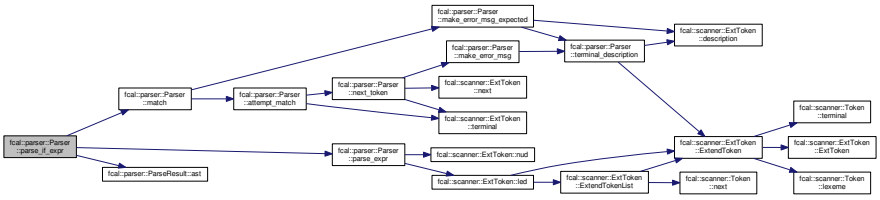
expr1

expr2

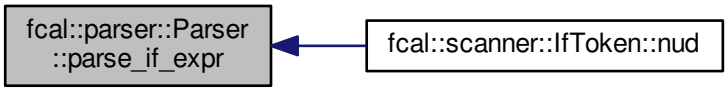
expr3

Create ifExpr node

Here is the call graph for this function:



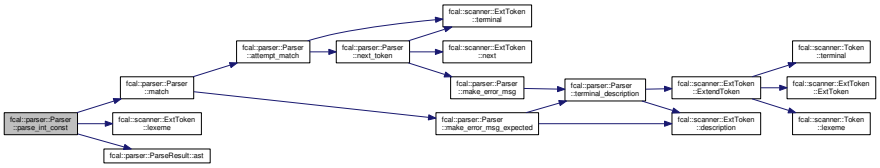
Here is the caller graph for this function:



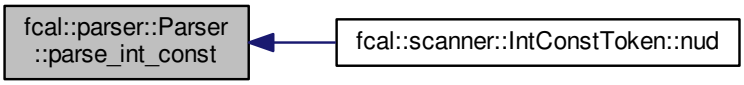
7.32.2.17 ParseResult fcal::parser::Parser::parse_int_const ( )

Expr ::= intConst.

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.18 ParseResult fcal::parser::Parser::parse_let_expr ( )

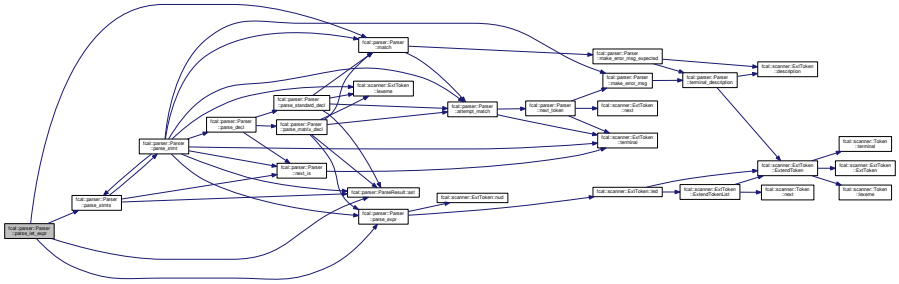
Expr ::= 'let' Stmts 'in' Expr 'end'.

stmts

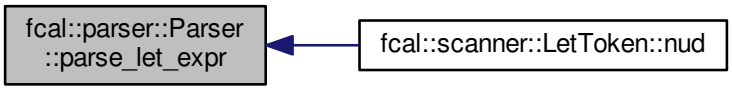
expr

Create letExpr node

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.19 ParseResult fcal::parser::Parser::parse_matrix_decl ( )

Parser::ParseProgram()

MatrixDecl simpleMatirx determines which grammar to use for Matrix.

Set the varName for the matrix declaration

First check if we have a full matrix declaration. Parse each part of the grammar below setting 3 variable names and 3 expressions. Decl ::= 'matrix' varName '[' Expr ':' Expr ']' varName ':' varName '=' Expr ';'

expr1

expr2

varName2

varName3

expr3

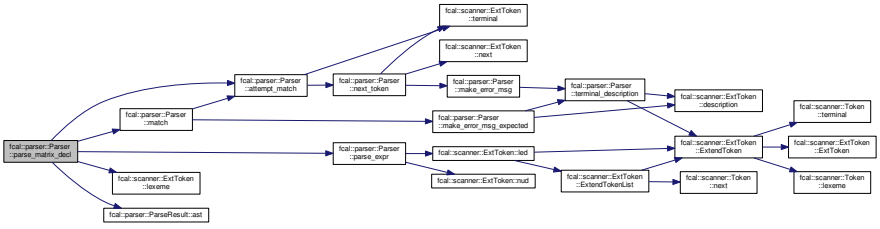
Create decl node for matrix.

If we have a simple matrix dexlaration, we only need to set the first variable name and 1 expression. Decl ::= 'matrix' varName '=' Expr ';'

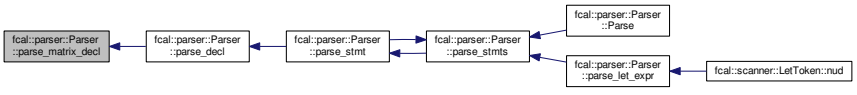
expr1

Create decl node for simple matrix.

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.20 ParseResult fcal::parser::Parser::parse_multiplication (ParseResult *left*)

Expr ::= Expr star Expr.

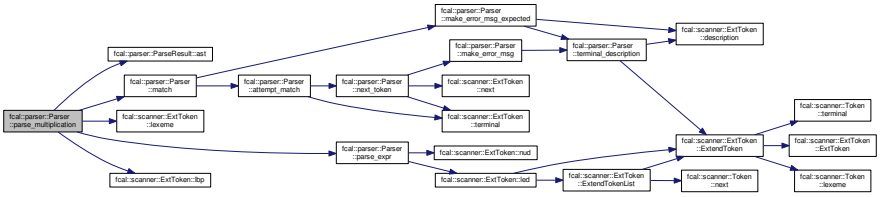
expr1

string ∗

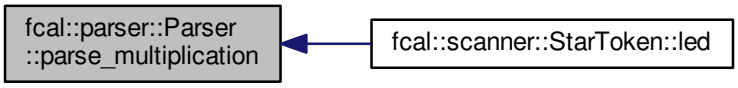
expr1

Create binaryExpr node

Here is the call graph for this function:



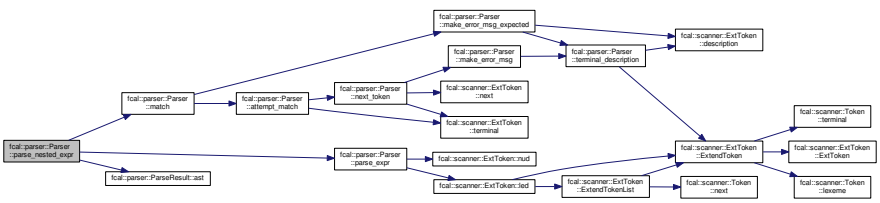
Here is the caller graph for this function:



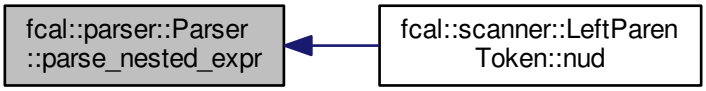
7.32.2.21 ParseResult fcal::parser::Parser::parse_nested_expr ( )

Expr ::= leftParen Expr rightParen.

Here is the call graph for this function:



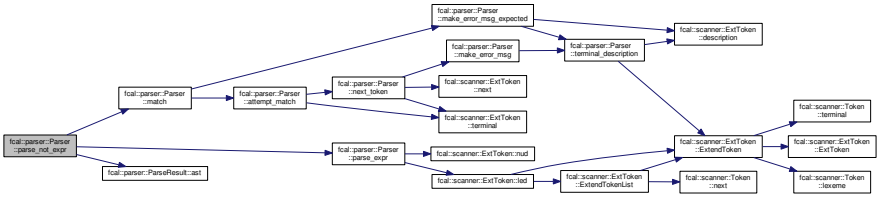
Here is the caller graph for this function:



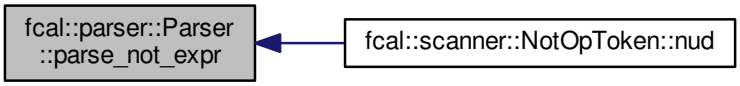
7.32.2.22 ParseResult fcal::parser::Parser::parse_not_expr ( )

Expr ::= '!' Expr.

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.23 ParseResult fcal::parser::Parser::parse_relational_expr (ParseResult *prLeft*)

Expr ::= Expr equalEquals Expr Expr ::= Expr lessThanEquals Expr Expr ::= Expr greaterThanEquals Expr Expr ::= Expr notEquals Expr Expr ::= Expr leftAngle Expr Expr ::= Expr rightAngle Expr Notice that for relational operators we use just one parse function. This shows another possible means for implementing expressions, as opposed to the method used for arithmetic expressions in which each operation has its own parse method. It will depend on what we do in iteration 3 in building an abstract syntax tree to decide which method is better. parser has already matched left expression

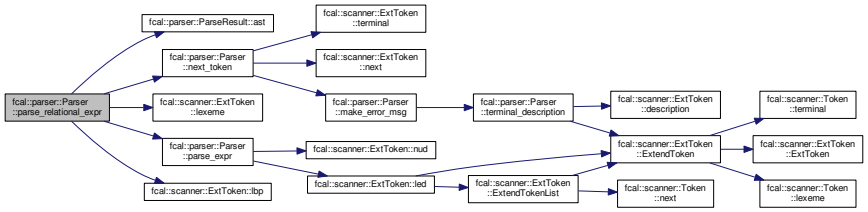
expr1

string boolean operation

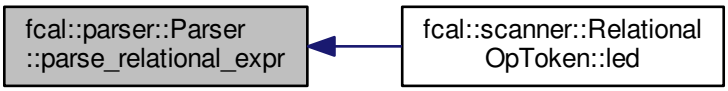
expr2

Create binaryExpr node

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.24 ParseResult fcal::parser::Parser::parse_standard_decl ( )

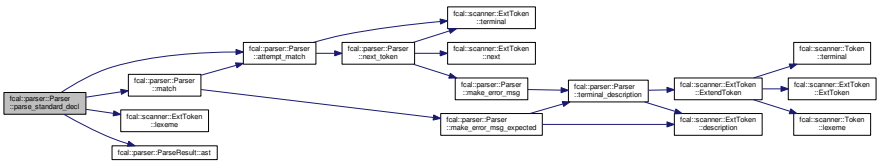
Parser::matrixDecl.

standardDecl Decl ::= integerKwd varName | floatKwd varName | stringKwd varName decType type is an enumerated type to show the variable type that is being declared.

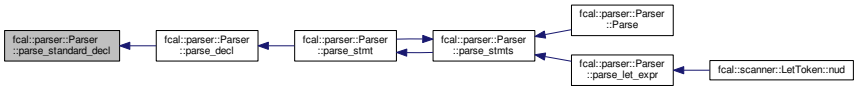
varName

Create decl node for variable declaration.

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.25 ParseResult fcal::parser::Parser::parse_stmt ( )

Parse a general statement.

If we have a declaration statement: Stmt ::= Decl

decl

Create new declStmt node

Stmt ::= '{' Stmts '}'

stmts

Create bracketStmt node

Stmt ::= 'if' '(' Expr ')' Stmt Stmt ::= 'if' '(' Expr ')' Stmt 'else' Stmt

expr

stmt

Create ifStmt node

It is if-else statment

elseStmt

Create ifElseStmt node

We have an assignment statement Stmt ::= varName '=' Expr ';' | varName '[' Expr ':' Expr ']' '=' Expr ';'

isMatrix is used to show matrix or standard assginment

varName

We have a matrix assignment

expr2

expr3

expr1

Create equalsStmt node for matrix or standard declaration

We have a print statement Stmt ::= 'print' '(' Expr ')' ';'

expr

Create printStmt node

We have a repeat statement Stmt ::= 'repeat' '(' varName '=' Expr 'to' Expr ')' Stmt

varName

expr1

expr2

stmt

Create repeatStmt node

We have a while statement Stmt ::= 'while' '(' Expr ')' Stmt

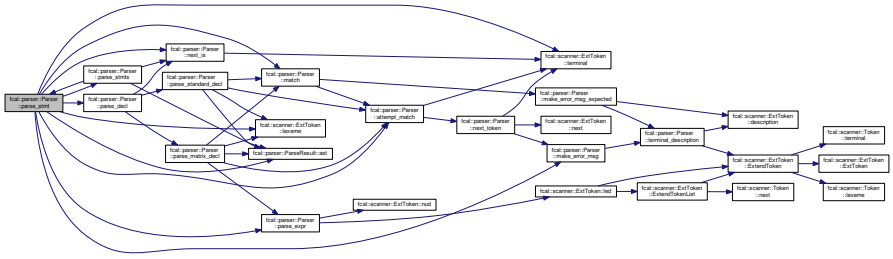
expr

stmt

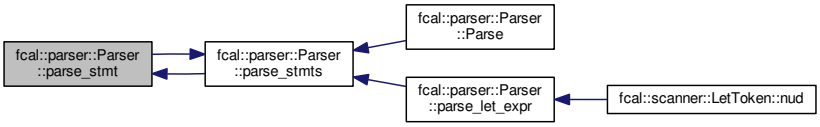
Create whileStmt node

Stmt ::= ';

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.26 ParseResult fcal::parser::Parser::parse_stmts ( )

Parse general statements.

Stmts ::= Stmt Stmts

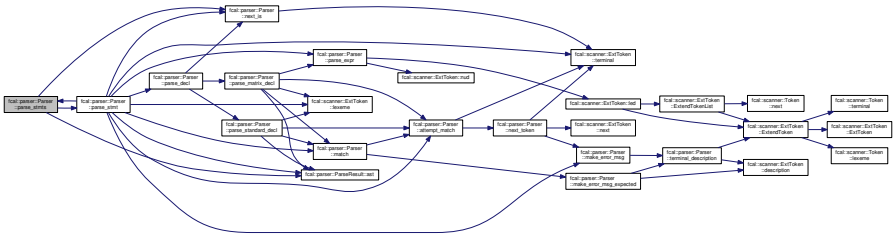
stmt

stmts

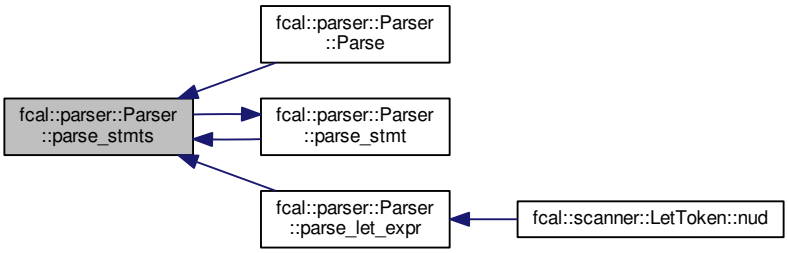
Create stmts node

No statements

Here is the call graph for this function:



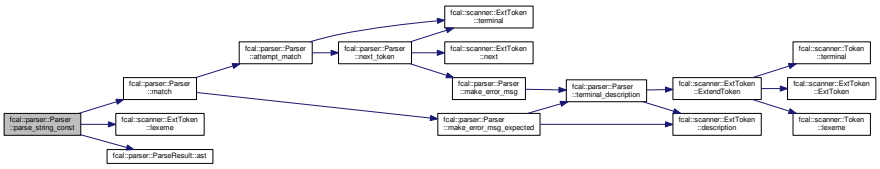
Here is the caller graph for this function:



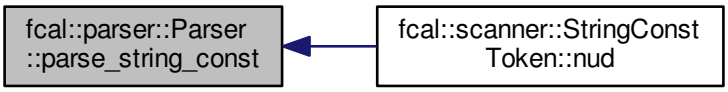
7.32.2.27 ParseResult fcal::parser::Parser::parse_string_const ( )

Expr ::= stringConst.

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.28 ParseResult fcal::parser::Parser::parse_subtraction (ParseResult *left*)

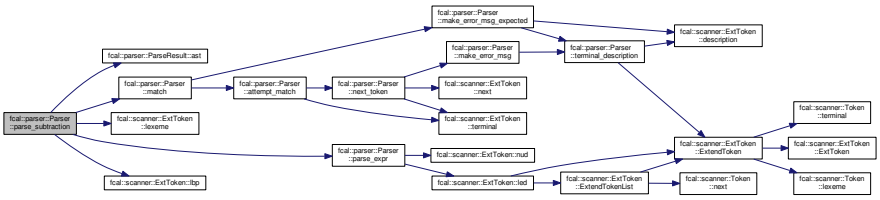
expr1

string -

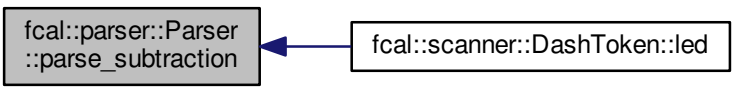
expr2

Create binaryExpr node

Here is the call graph for this function:



Here is the caller graph for this function:



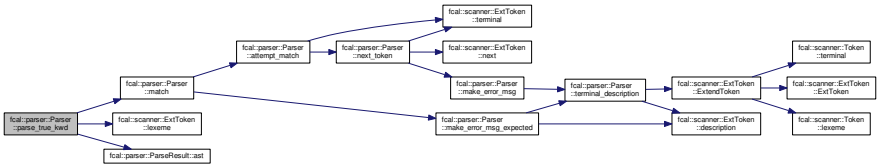
7.32.2.29 ParseResult fcal::parser::Parser::parse_true_kwd ( )

methods for parsing productions for Expr

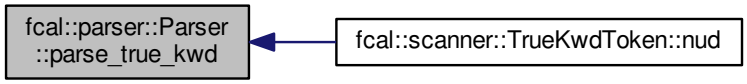
Parser::parse_expr.

Parse each specific expression and create the proper node in the AST. Expr ::= trueKwd

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.30 ParseResult fcal::parser::Parser::parse_variable_name ( )

Expr ::= variableName .....

Expr ::= matrix [Expr:Expr]

expr1

expr2

Create matrixExpr node

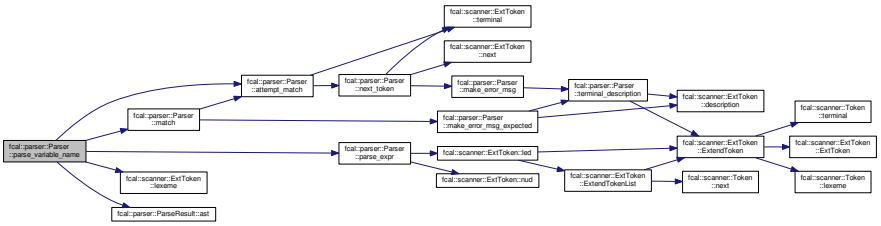
Expr ::= varableName '(' Expr ')'

expr

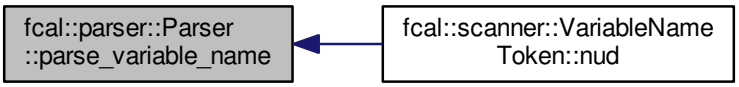
Create nestedExpr node

We have right hand side variable expression

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.2.31 ParseResult fcal::parser::Parser::ParseProgram ( )

Parser methods for the nonterminals:

Parser::parse()

parse methods for non-terminal symbols

Program – This is the root node where the entire program will be placed once everything has been parsed and placed in the proper node of the abstract syntax tree(AST). This function's comments describe the general structure of the overall program, and specifically how each function parses the program accoding to the grammar of that function and how it then creates the AST from that grammar. Each function will follow this same general structure. root – the following line shows the grammar that this function is intended to parse to create the specific node needed for the AST.

Program ::= varName '(' ')' '{' Stmts '}'

String name holds varName in grammar after finding kVariableName, setting varName class with name. This same form will continue throughout program for any function requiring a variable name in the grammar.

pr_stmts will hold all statements after recursively calling appropriate grammar. ParseResult pr_stmts, pr_expr, pr↩ _decl, or similar variations will follow the same form throughout the program calling each function recursively when a declaration, statement,or expression is needed according to the specific grammar that the current function requires. ParseResult pr_stmts = parse_stmts();

ast::Stmts stmts will hold the Node ∗ast value from pr_stmts. This same form will continue throughout the program to set the current Expr, Stmt, or Decl class ∗ast value. These ∗ast values are what will be used to set the values for the subclasses that will be used to create the nodes for the AST.

The following checks that pr_stmts ast value isn't null, and that stmts was properly assigned the value from Parse↩ Result's Node class ∗ast.

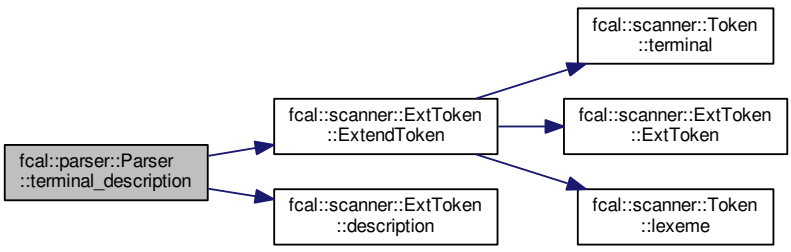
Set the Root node with the entire program, varName {statements}. This same form will be used throughout the program to set the respective variables, statements, expressions, and declarations to the proper node of the AST according to the grammar for the current function.

Set the ParseResults Node ∗ast with the final program once everthing is parsed. This same form will continue throughout the program, pr.ast(node) sets the primary ParseResult object with the node that holds the values acquired during the current function according to its grammar.

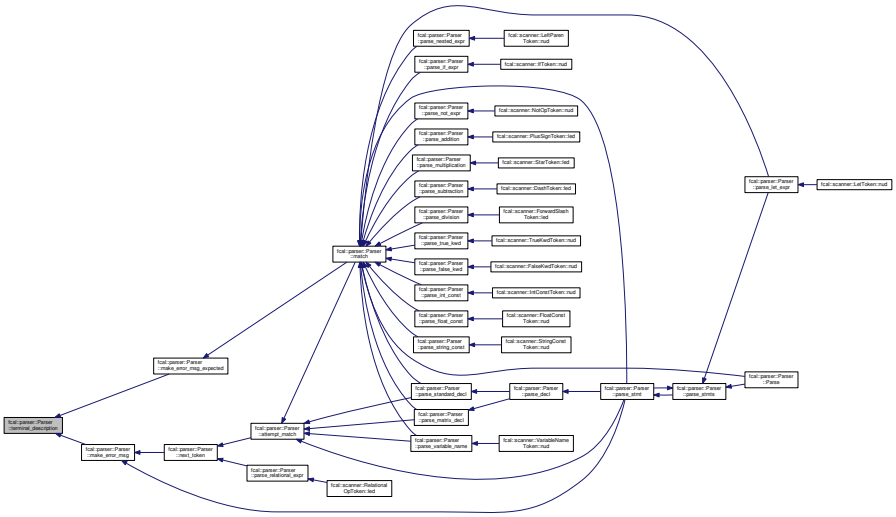
Here is the caller graph for this function:

7.32.2.32 std::string fcal::parser::Parser::terminal_description (const scanner::TokenType & *terminal*) [private]

Here is the call graph for this function:



Here is the caller graph for this function:



7.32.3 Member Data Documentation

7.32.3.1 scanner::ExtToken* fcal::parser::Parser::curr_token_ [private]

7.32.3.2 scanner::ExtToken* fcal::parser::Parser::prev_token_ [private]

7.32.3.3 scanner::Scanner* fcal::parser::Parser::scanner_ [private]

7.32.3.4 scanner::Token* fcal::parser::Parser::stokens_ [private]

7.32.3.5 scanner::ExtToken* fcal::parser::Parser::tokens_ [private]

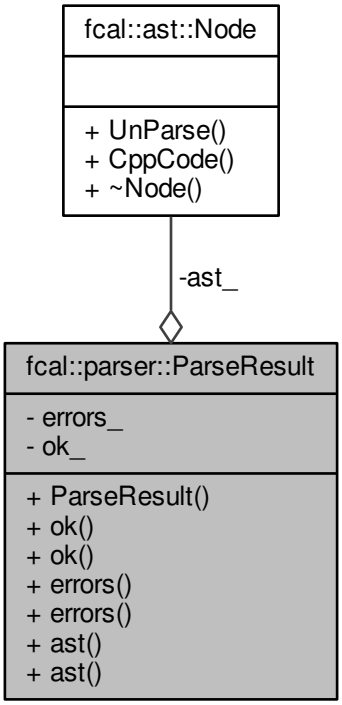
The documentation for this class was generated from the following files:

- include/parser.h
- src/parser.cc

7.33 fcal::parser::ParseResult Class Reference

```
#include <parse_result.h>
```

Collaboration diagram for fcal::parser::ParseResult:



Public Member Functions

- ParseResult (void)
- bool ok (void) const
- void ok (bool result_in)
- std::string errors (void) const
- void errors (const std::string str_in)
- ast::Node * ast (void)
- void ast (ast::Node *Node_ptr)

Private Attributes

- std::string errors_
- ast::Node * ast_
- bool ok_

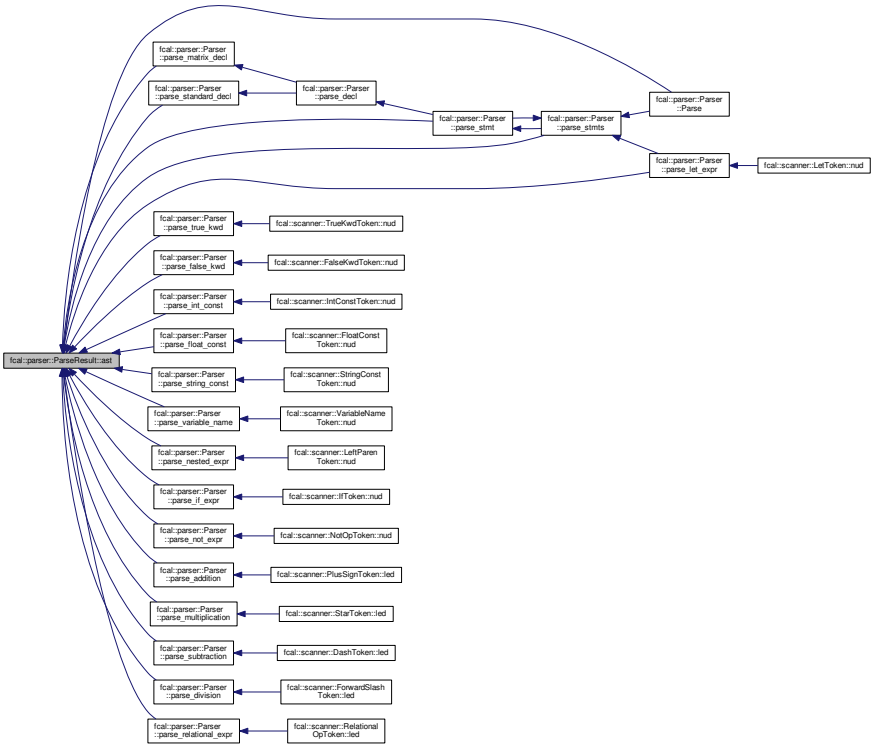
7.33.1 Constructor & Destructor Documentation

7.33.1.1 fcal::parser::ParseResult::ParseResult (void) [inline]

7.33.2 Member Function Documentation

7.33.2.1 ast::Node* fcal::parser::ParseResult::ast (void) [inline]

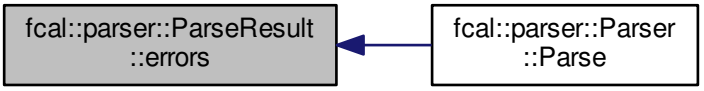
Here is the caller graph for this function:



7.33.2.2 void fcal::parser::ParseResult::ast (ast::Node * *Node_ptr*) [inline]

7.33.2.3 std::string fcal::parser::ParseResult::errors (void) const [inline]

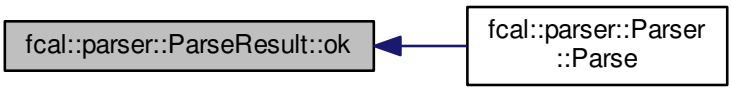
Here is the caller graph for this function:



7.33.2.4 void fcal::parser::ParseResult::errors (const std::string *str_in*) [inline]

7.33.2.5 bool fcal::parser::ParseResult::ok (void) const [inline]

Here is the caller graph for this function:



7.33.2.6 void fcal::parser::ParseResult::ok (bool *result_in*) [inline]

7.33.3 Member Data Documentation

7.33.3.1 ast::Node* fcal::parser::ParseResult::ast_ [private]

7.33.3.2 std::string fcal::parser::ParseResult::errors_ [private]

7.33.3.3 bool fcal::parser::ParseResult::ok_ [private]

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

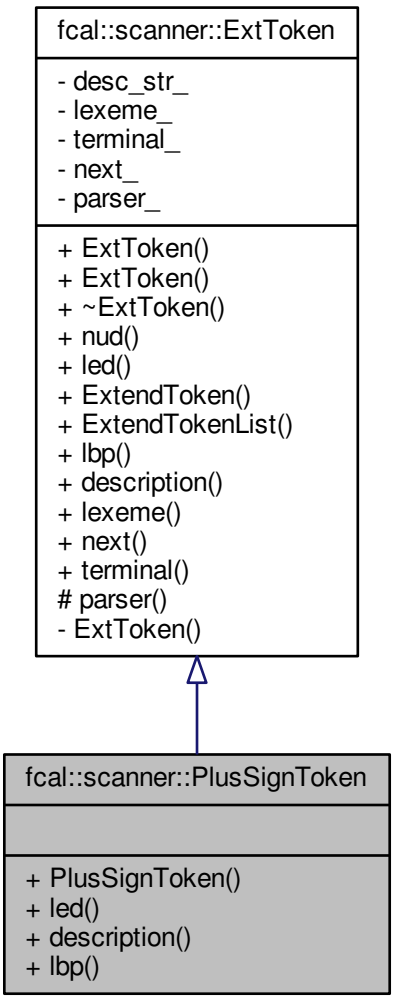
- include/parse_result.h

7.34 fcal::scanner::PlusSignToken Class Reference

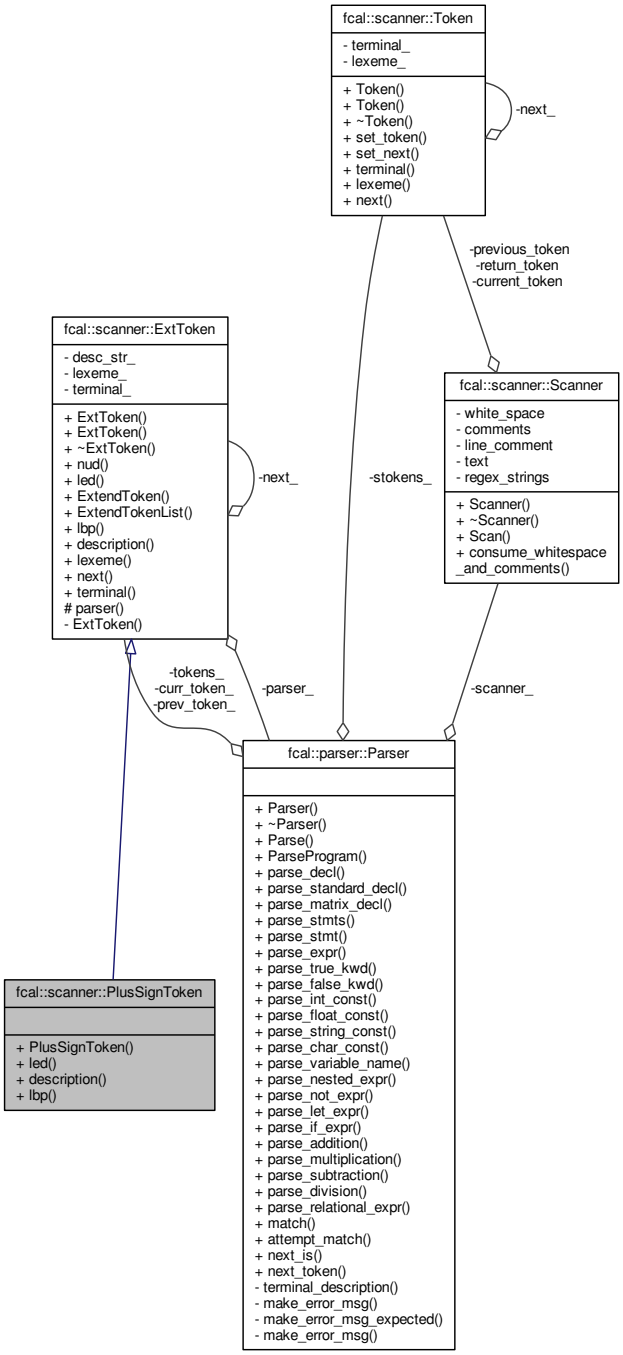
Plus Sign.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::PlusSignToken:



Collaboration diagram for fcal::scanner::PlusSignToken:



Public Member Functions

- PlusSignToken (parser::Parser ∗p, Token ∗t)
- parser::ParseResult led (parser::ParseResult left)
- std::string description ()
- int lbp ()

Additional Inherited Members

7.34.1 Detailed Description

Plus Sign.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 fcal::scanner::PlusSignToken::PlusSignToken (parser::Parser * *p*, Token * *t*) [inline]

7.34.3 Member Function Documentation

7.34.3.1 std::string fcal::scanner::PlusSignToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

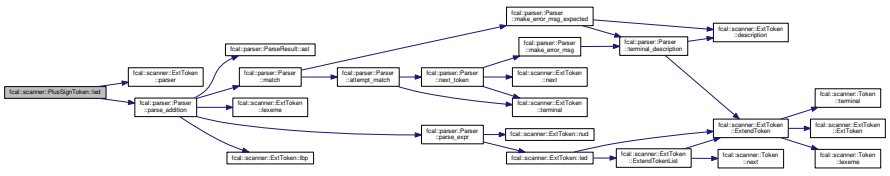
7.34.3.2 int fcal::scanner::PlusSignToken::lbp ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.34.3.3 parser::ParseResult fcal::scanner::PlusSignToken::led (parser::ParseResult *left*) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



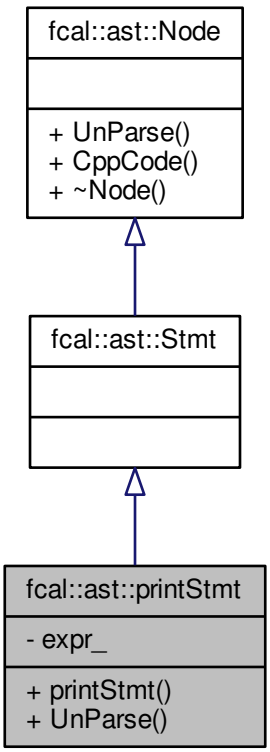
The documentation for this class was generated from the following file:

- include/ext_token.h

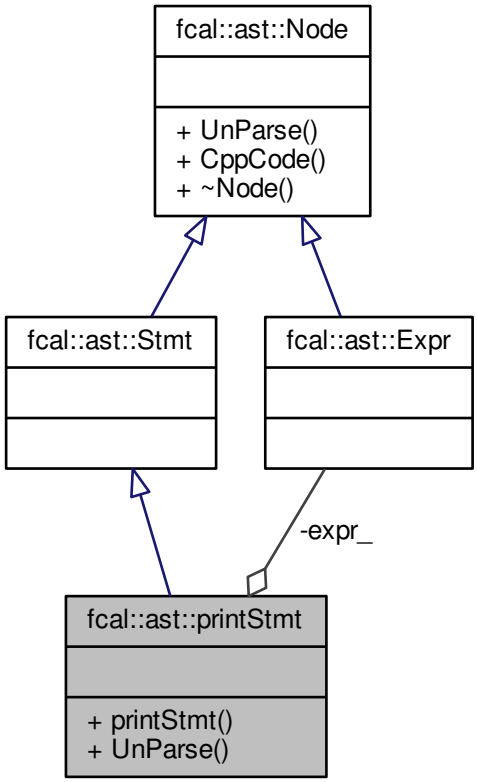
7.35 fcal::ast::printStmt Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::printStmt:



Collaboration diagram for fcal::ast::printStmt:



Public Member Functions

- printStmt (Expr *expr)
- std::string UnParse (void)

  Returns the string: "print(" + expr_->UnParse() + ");\n".

Private Attributes

- Expr * expr_

  An Expr which will be printed.

7.35.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Stmt ::= 'print' '(' Expr ')' ';'

7.35.2 Constructor & Destructor Documentation

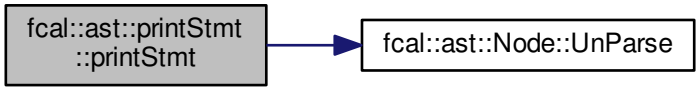
7.35.2.1 fcal::ast::printStmt::printStmt (Expr * *expr*) [inline], [explicit]

This constructor takes one arguement

Parameters

expr	an expression which should be printed

Here is the call graph for this function:



7.35.3 Member Function Documentation

7.35.3.1 std::string fcal::ast::printStmt::UnParse (void) [virtual]

Returns the string: "print(" + expr_->UnParse() + ");\n".

Reimplemented from fcal::ast::Node.

7.35.4 Member Data Documentation

7.35.4.1 Expr* fcal::ast::printStmt::expr_ [private]

An Expr which will be printed.

The documentation for this class was generated from the following files:

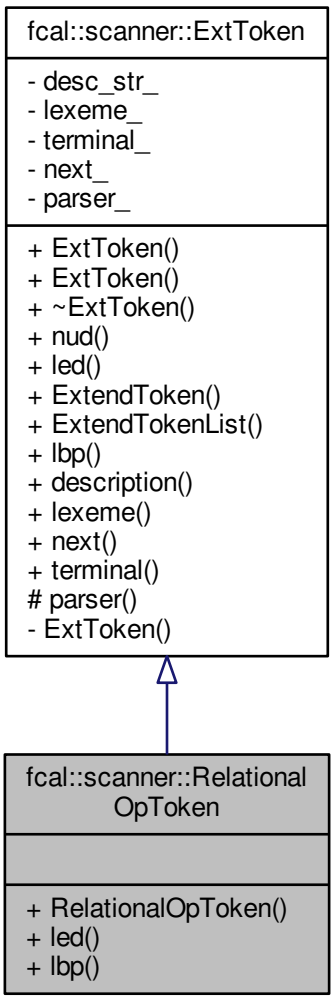
- include/ast.h
- src/ast.cc

7.36 fcal::scanner::RelationalOpToken Class Reference

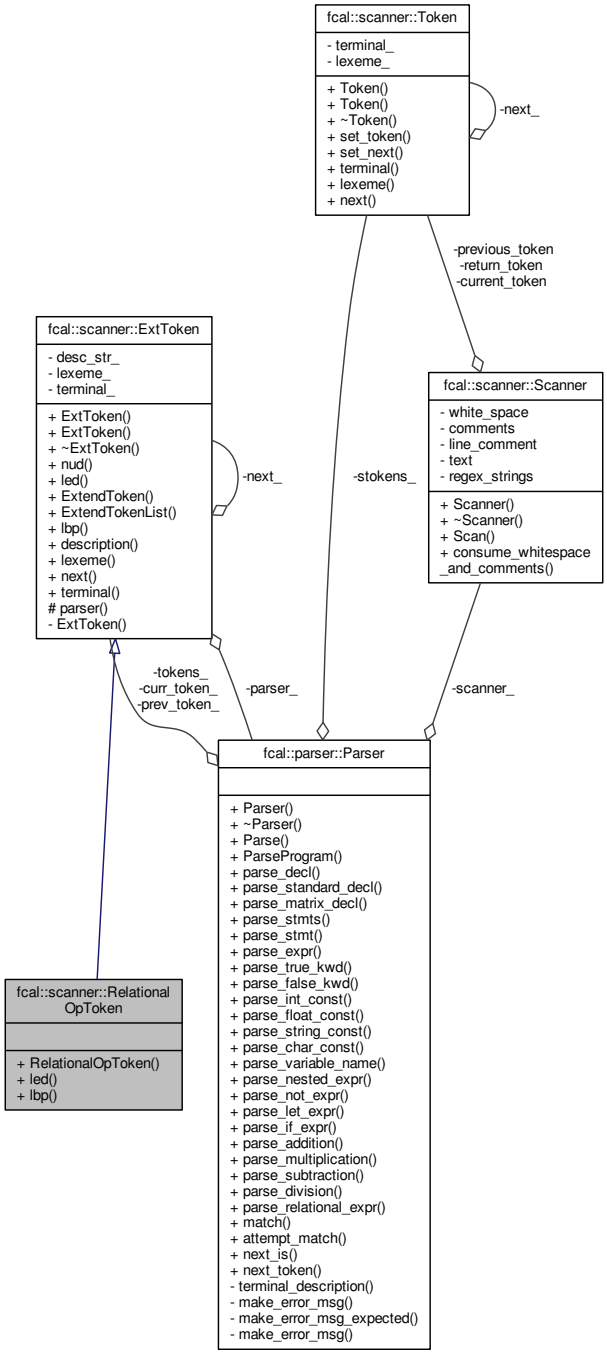
Relational Op.

`#include <ext_token.h>`

Inheritance diagram for fcal::scanner::RelationalOpToken:



Collaboration diagram for fcal::scanner::RelationalOpToken:



Public Member Functions

- RelationalOpToken (parser::Parser ∗p, Token ∗t, std::string d)
- parser::ParseResult led (parser::ParseResult left)
- int lbp ()

Additional Inherited Members

7.36.1 Detailed Description

Relational Op.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 fcal::scanner::RelationalOpToken::RelationalOpToken (parser::Parser * *p*, Token * *t*, std::string *d*) [inline]

7.36.3 Member Function Documentation

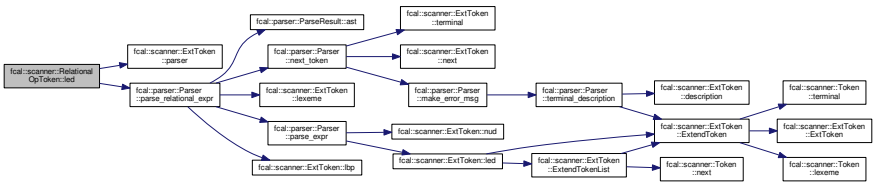
7.36.3.1 int fcal::scanner::RelationalOpToken::lbp ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.36.3.2 parser::ParseResult fcal::scanner::RelationalOpToken::led (parser::ParseResult *left*) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



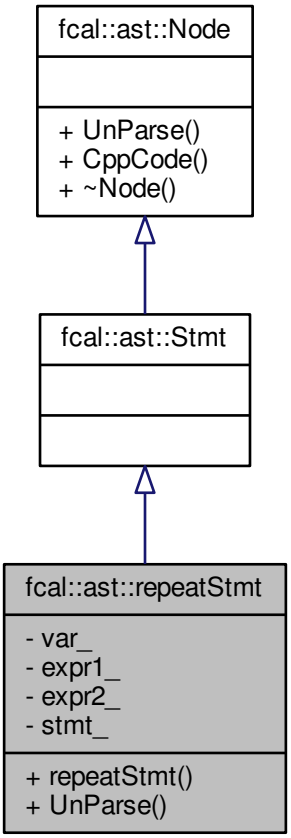
The documentation for this class was generated from the following file:

- include/ext_token.h

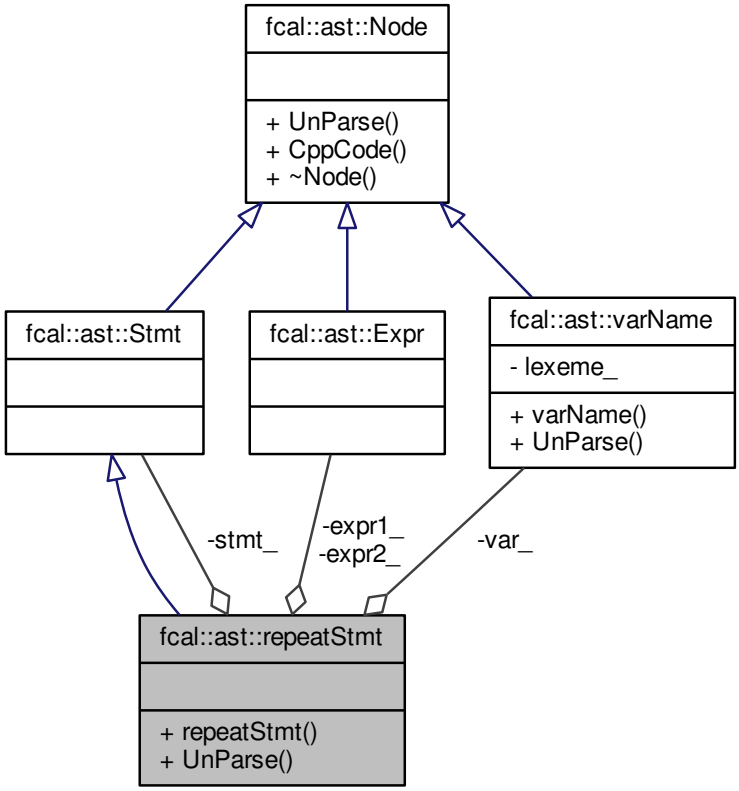
7.37 fcal::ast::repeatStmt Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::repeatStmt:



Collaboration diagram for fcal::ast::repeatStmt:



Public Member Functions

- repeatStmt (varName *var, Expr *expr1, Expr *expr2, Stmt *stmt)
- std::string UnParse (void)

Private Attributes

- varName * var_

  a variable representing the index
- Expr * expr1_

  an Expr which is the lower bound of repeat loop
- Expr * expr2_

  an Expr which is the upper bound of repeat loop
- Stmt * stmt_

7.37.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Stmt ::= 'repeat' '(' varName '=' Expr 'to' Expr ')' Stmt

7.37.2 Constructor & Destructor Documentation

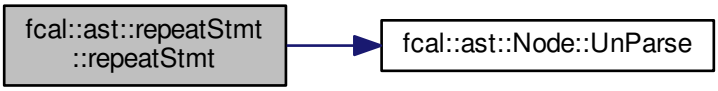
7.37.2.1 fcal::ast::repeatStmt::repeatStmt (varName * *var*, Expr * *expr1*, Expr * *expr2*, Stmt * *stmt*) [inline]

This constructor takes four parameters

Parameters

var	a variable representing the index
expr1	an Expr which is the lower bound of repeat loop
expr2	an Expr which is the upper bound of repeat loop
stmt	a Stmt which is executed while the index is within the limits specified by the expr1 and expr2

Here is the call graph for this function:



7.37.3 Member Function Documentation

7.37.3.1 std::string fcal::ast::repeatStmt::UnParse (void) [virtual]

Returns the string : "repeat (" + var_->UnParse() + " = "

- expr1_->UnParse() + " to " + expr2_->UnParse() + ")\n" + stmt_->UnParse()

Reimplemented from fcal::ast::Node.

7.37.4 Member Data Documentation

7.37.4.1 Expr* fcal::ast::repeatStmt::expr1_ [private]

an Expr which is the lower bound of repeat loop

7.37.4.2 Expr* fcal::ast::repeatStmt::expr2_ [private]

an Expr which is the upper bound of repeat loop

7.37.4.3 Stmt* fcal::ast::repeatStmt::stmt_ [private]

a Stmt which is executed while the index is within the limits specified by the expr1 and expr2

7.37.4.4 varName* fcal::ast::repeatStmt::var_ [private]

a variable representing the index

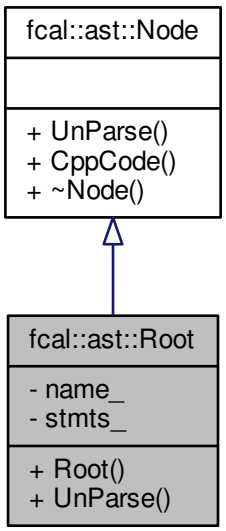
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

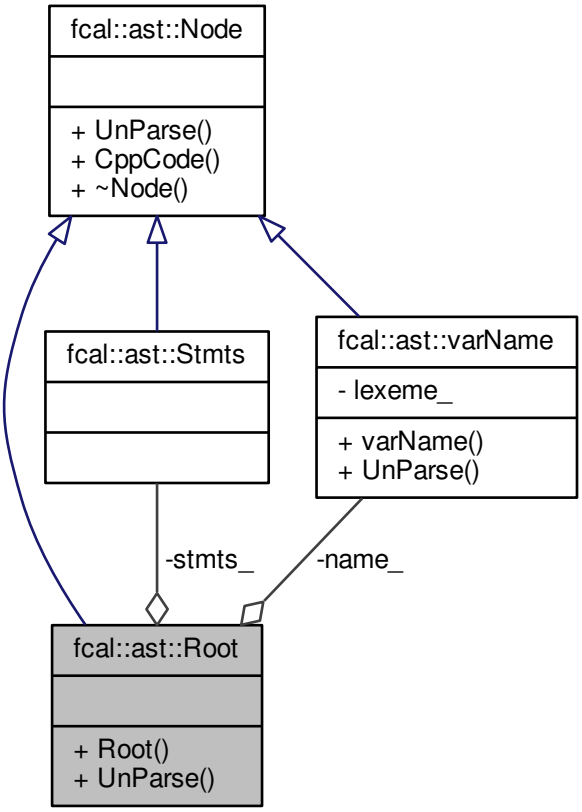
7.38 fcal::ast::Root Class Reference

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

Inheritance diagram for fcal::ast::Root:



Collaboration diagram for fcal::ast::Root:



Public Member Functions

- Root (varName ∗name, Stmts ∗stmts)
- std::string UnParse (void)

Private Attributes

- varName ∗ name_

  name_ holds the address containing details of program name
- Stmts ∗ stmts_

7.38.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Program ::= varName '(' ')' '{' Stmts '}' This class is the root of the ast tree created by the parser.

7.38.2 Constructor & Destructor Documentation

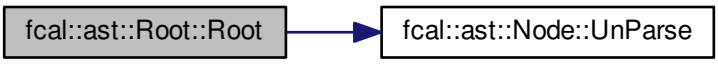
7.38.2.1 fcal::ast::Root::Root (varName * *name*, Stmts * *stmts*) [inline]

Constructor for the Root class takes parameters

Parameters

name	a varName∗ holding a reference to the varName representing the program
stmts	a Stmts∗ holding a reference to the Stmts in the program

Here is the call graph for this function:



7.38.3 Member Function Documentation

7.38.3.1 std::string fcal::ast::Root::UnParse (void) [virtual]

Returns the string : name_->UnParse() + " () " + "{\n" + stmts_->UnParse() + "\n}\n"

Reimplemented from fcal::ast::Node.

7.38.4 Member Data Documentation

7.38.4.1 varName∗ fcal::ast::Root::name_ [private]

name_ holds the address containing details of program name

7.38.4.2 Stmts∗ fcal::ast::Root::stmts_ [private]

stmts_ hols the address containing details of Stmts in the program

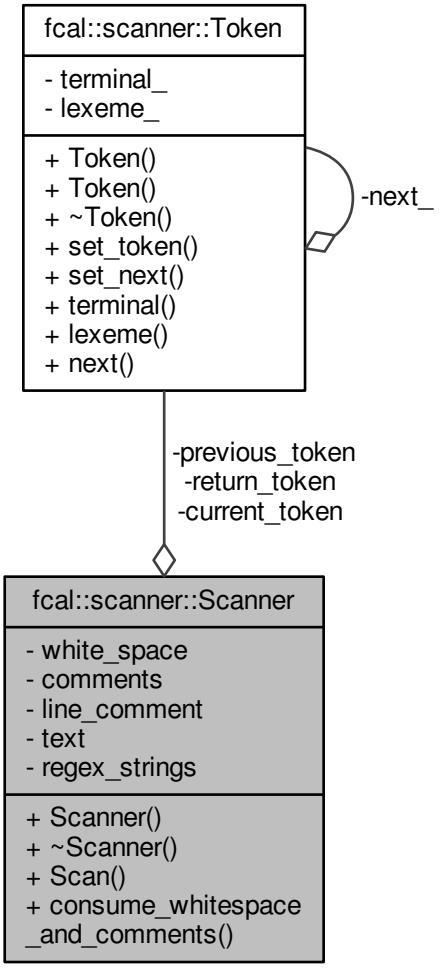
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

7.39 fcal::scanner::Scanner Class Reference

`#include <scanner_class.h>`

Collaboration diagram for fcal::scanner::Scanner:



Public Member Functions

- Scanner ()

  Default constructor for Scanner.
- ~Scanner ()
- Token * Scan (const char *)
- int consume_whitespace_and_comments (regex_t *, regex_t *, regex_t *, const char *)

  end scan

Private Attributes

- Token * return_token

  Token representing front of the list.*
- Token * previous_token

  Token representing previous Token in the list.*
- Token * current_token

  Token representing current Token in the list.*
- regex_t * white_space

  A regex which matches white space.
- regex_t * comments

  A regex which matches comments.
- regex_t * line_comment

  A regex which matches line comments.
- const char * text

  String representing contents of the File.
- regex_t * regex_strings [1+static_cast< int >(kNotOp)]

  A regex array created for matching lexeme of Tokens.

7.39.1 Detailed Description

This is the Scanner class which implements the scanner of the FCAL program. Scanner has the method Scan which returns the front of the List of Tokens generated from scanning a file. It implements the first stage of the parsing process.

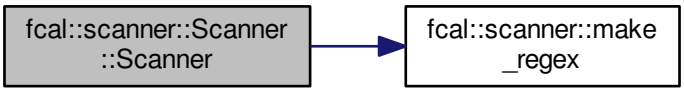
7.39.2 Constructor & Destructor Documentation

7.39.2.1 fcal::scanner::Scanner::Scanner ( )

Default constructor for Scanner.

Initialize token_strings, white_space, comments, line_comment and regex_strings

Here is the call graph for this function:



7.39.2.2 fcal::scanner::Scanner::~Scanner ( )

7.39.3 Member Function Documentation

7.39.3.1 int fcal::scanner::Scanner::consume_whitespace_and_comments (regex_t * *white_space*, regex_t * *block_comment*, regex_t * *line_comment*, const char * *text*)

end scan

Consumes white spaces and comments

Parameters

white_space	regex for matching white space
block_comment	regex for matching block comment
line_comment	regex for matching line comment
text	A string representing File input

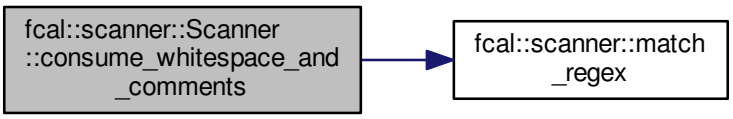
exit loop if not reset by a match

Try to match white space

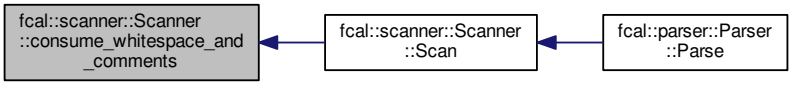
Try to match block comments

Try to match line comment comments

Here is the call graph for this function:



Here is the caller graph for this function:



7.39.3.2 Token * fcal::scanner::Scanner::Scan (const char * *s*)

Scans the input string and return's the front of the list of Token's.

Parameters

s	input string representing the contents of a file to be scanned

Token pointers to create list of tokens

consume white_space and comments

Check the text for qualified tokens or lexical errors

current Token is where we store token information

Check if next text string matches a token, if so add terminal and lexeme null pointer. If a token isn't found add lexeme and lexical error.

If no token match, set lexical error

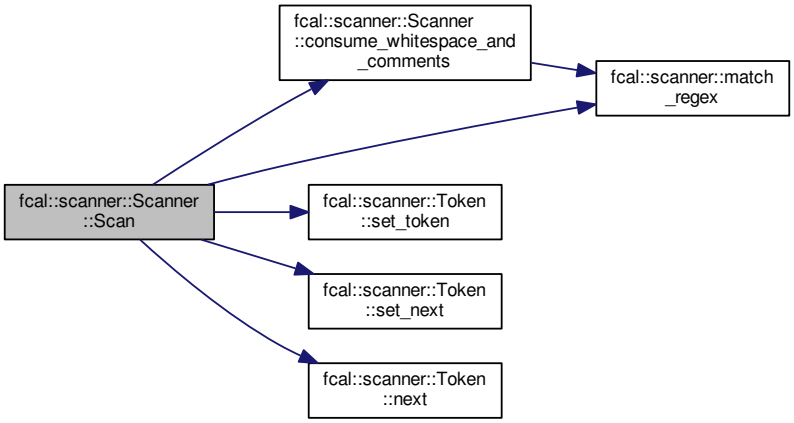
Find the current lexeme using the number of matched characters

Set the current token values

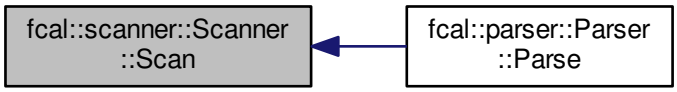
Set previous token to point to current

Add end of file token once all matches are found

Here is the call graph for this function:



Here is the caller graph for this function:



7.39.4 Member Data Documentation

7.39.4.1 regex_t* fcal::scanner::Scanner::comments [private]

A regex which matches comments.

7.39.4.2 Token* fcal::scanner::Scanner::current_token [private]

Token* representing current Token in the list.

7.39.4.3 regex_t* fcal::scanner::Scanner::line_comment [private]

A regex which matches line comments.

7.39.4.4 Token* fcal::scanner::Scanner::previous_token [private]

Token* representing previous Token in the list.

7.39.4.5 regex_t* fcal::scanner::Scanner::regex_strings[1+static_cast< int >(kNotOp)] [private]

A regex array created for matching lexeme of Tokens.

7.39.4.6 Token* fcal::scanner::Scanner::return_token [private]

Token* representing front of the list.

7.39.4.7 const char* fcal::scanner::Scanner::text [private]

String representing contents of the File.

7.39.4.8 regex_t* fcal::scanner::Scanner::white_space [private]

A regex which matches white space.

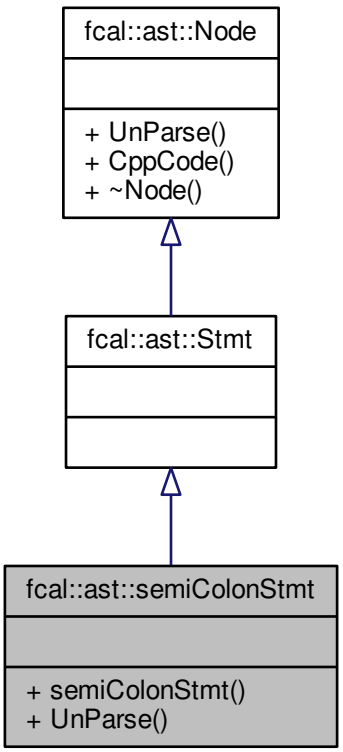
The documentation for this class was generated from the following files:

- include/scanner_class.h
- src/scanner_class.cc

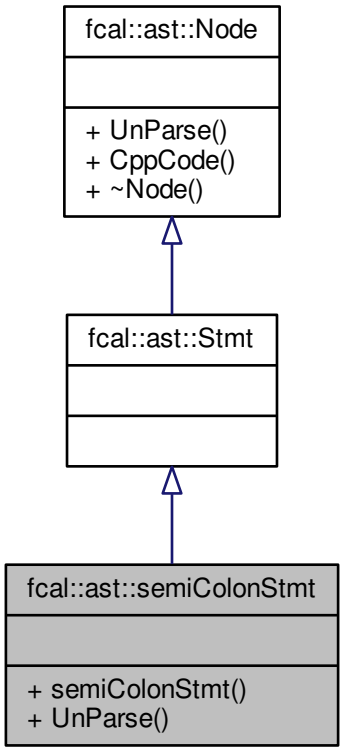
7.40 fcal::ast::semiColonStmt Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::semiColonStmt:



Collaboration diagram for fcal::ast::semiColonStmt:



Public Member Functions

- semiColonStmt ()

  This is a default constructor and does nothing.
- std::string UnParse (void)

  Returns the string : ";".

7.40.1 Detailed Description

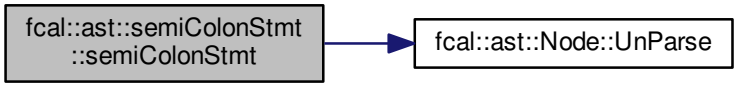
This is a concrete class in the ast class heirarchy. It implements the production,
Stmt ::= ';'

7.40.2 Constructor & Destructor Documentation

7.40.2.1 fcal::ast::semiColonStmt::semiColonStmt ( ) [inline]

This is a default constructor and does nothing.

Here is the call graph for this function:



7.40.3 Member Function Documentation

7.40.3.1 std::string fcal::ast::semiColonStmt::UnParse (void) [virtual]

Returns the string : ";".

Reimplemented from fcal::ast::Node.

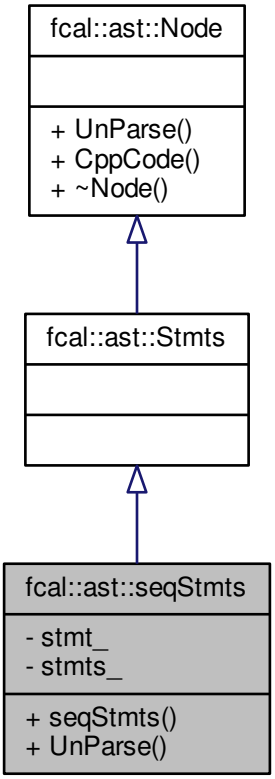
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

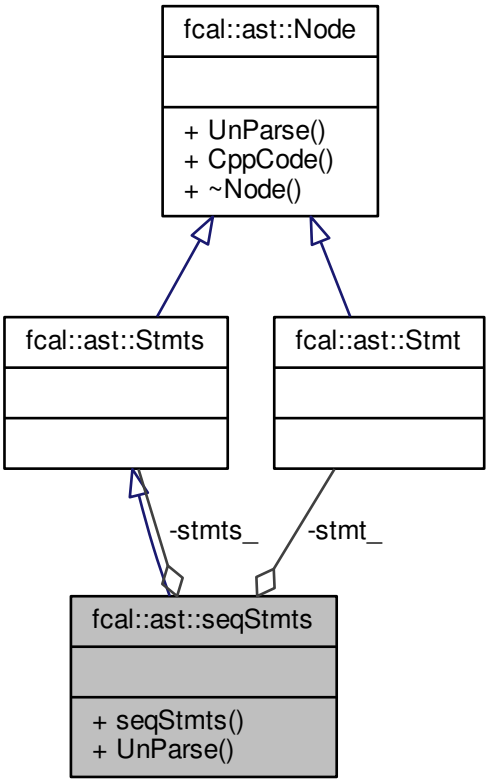
7.41 fcal::ast::seqStmts Class Reference

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

Inheritance diagram for fcal::ast::seqStmts:



Collaboration diagram for fcal::ast::seqStmts:



Public Member Functions

- seqStmts (Stmt ∗stmt, Stmts ∗stmts)
- std::string UnParse (void)

  Returns a string representing a sequence of Stmts.

Private Attributes

- Stmt ∗ stmt_
- Stmts ∗ stmts_

7.41.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Stmts ::= Stmt Stmts

7.41.2 Constructor & Destructor Documentation

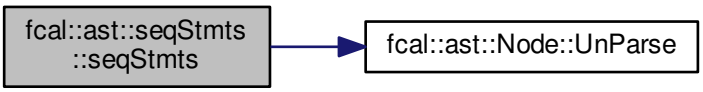
7.41.2.1 fcal::ast::seqStmts::seqStmts (Stmt ∗ *stmt,* Stmts ∗ *stmts*) [inline]

Constructor which takes parameters

Parameters

stmt	a Stmt* which holds the address which contains details of first Stmt in a sequence of Stmts
stmts	a Stmts* which holds the address which contains details of rest of the Stmts after the first Stmt

Here is the call graph for this function:



7.41.3 Member Function Documentation

7.41.3.1 std::string fcal::ast::seqStmts::UnParse (void) [virtual]

Returns a string representing a sequence of Stmts.

Returns the string : stmt_->UnParse() + stmts_->UnParse()

Reimplemented from fcal::ast::Node.

7.41.4 Member Data Documentation

7.41.4.1 Stmt* fcal::ast::seqStmts::stmt_ [private]

stmt_ hols the address which contains details of first Stmt in a sequence of Stmts

7.41.4.2 Stmts* fcal::ast::seqStmts::stmts_ [private]

stmts_ hols the address which contains details of rest of Stmts after the first Stmt

The documentation for this class was generated from the following files:

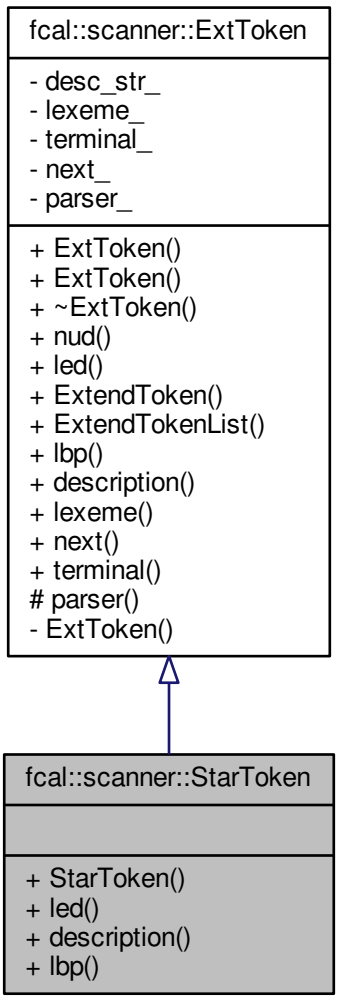
- include/ast.h
- src/ast.cc

7.42 fcal::scanner::StarToken Class Reference

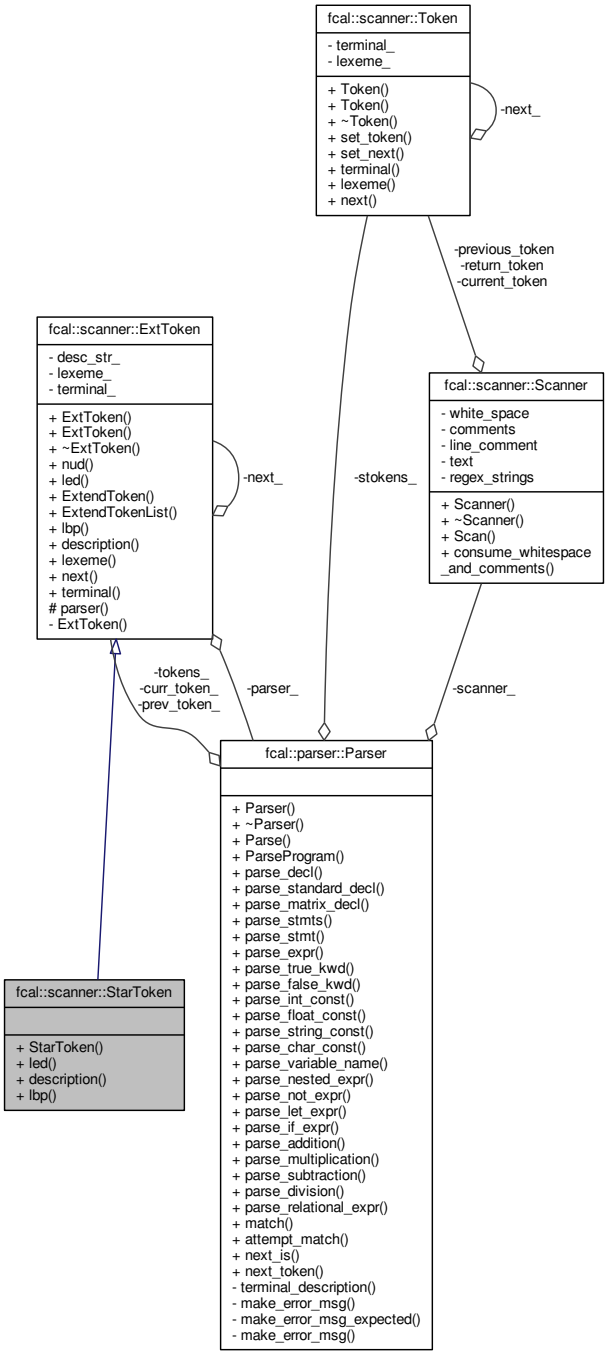
Star.

`#include <ext_token.h>`

Inheritance diagram for fcal::scanner::StarToken:



Collaboration diagram for fcal::scanner::StarToken:



Public Member Functions

- StarToken (parser::Parser ∗p, Token ∗t)
- parser::ParseResult led (parser::ParseResult left)
- std::string description ()
- int lbp ()

Additional Inherited Members

7.42.1 Detailed Description

Star.

7.42.2 Constructor & Destructor Documentation

7.42.2.1 fcal::scanner::StarToken::StarToken (parser::Parser * *p*, Token * *t*) [inline]

7.42.3 Member Function Documentation

7.42.3.1 std::string fcal::scanner::StarToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

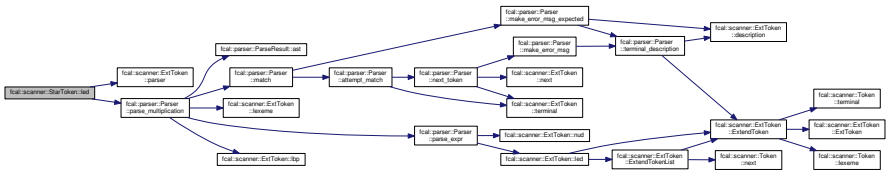
7.42.3.2 int fcal::scanner::StarToken::lbp ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.42.3.3 parser::ParseResult fcal::scanner::StarToken::led (parser::ParseResult *left*) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



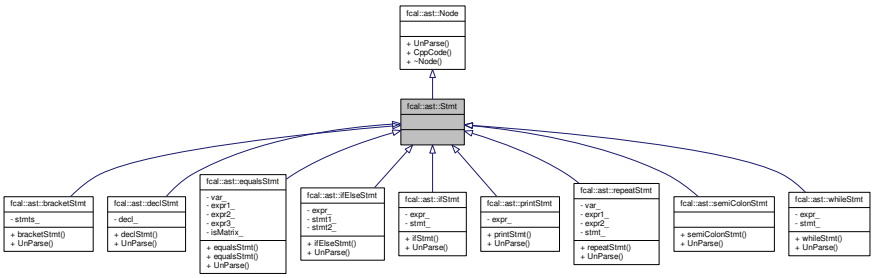
The documentation for this class was generated from the following file:

- include/ext_token.h

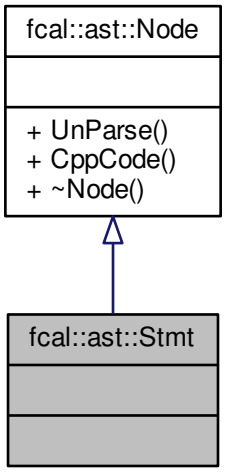
7.43 fcal::ast::Stmt Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::Stmt:



Collaboration diagram for fcal::ast::Stmt:



Additional Inherited Members

7.43.1 Detailed Description

This is the abstract class which inherits from Node. It has no implementation for any functions derived from Node. It is the base class for classes which implement the productions which are derived from the nonterminal 'Stmt' in the FCAL grammar.

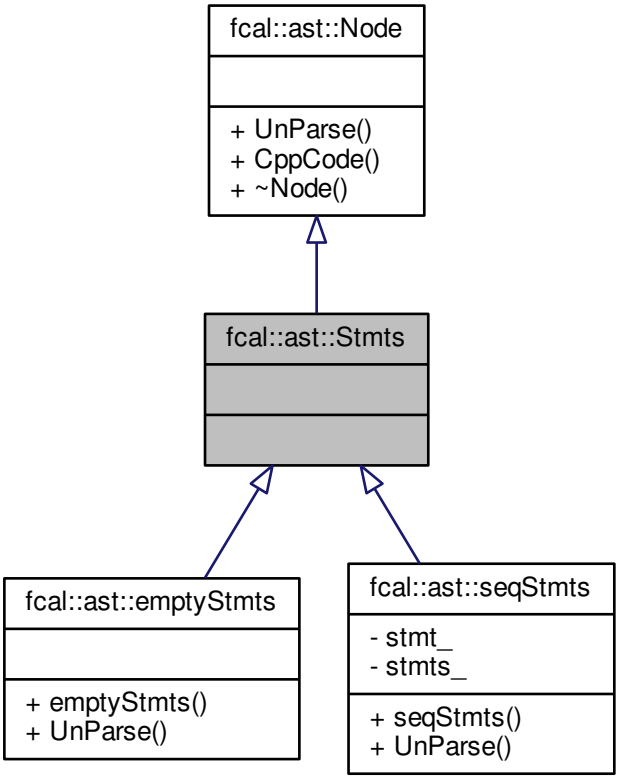
The documentation for this class was generated from the following file:

- include/ast.h

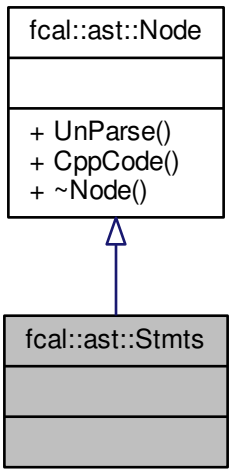
7.44 fcal::ast::Stmts Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::Stmts:



Collaboration diagram for fcal::ast::Stmts:



Additional Inherited Members

7.44.1 Detailed Description

This is the abstract class which inherits from Node. It has no implementation for any functions derived from Node. It is the base class for classes which implement the productions which are derived from the nonterminal 'Stmts' in the FCAL grammar.

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

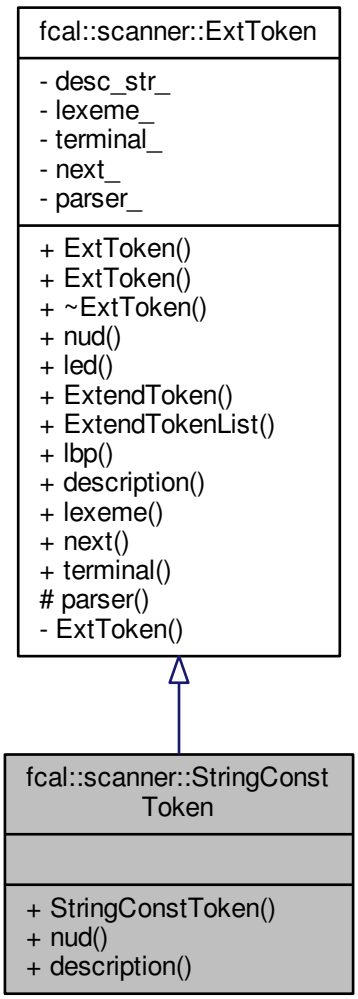
- include/ast.h

7.45 fcal::scanner::StringConstToken Class Reference

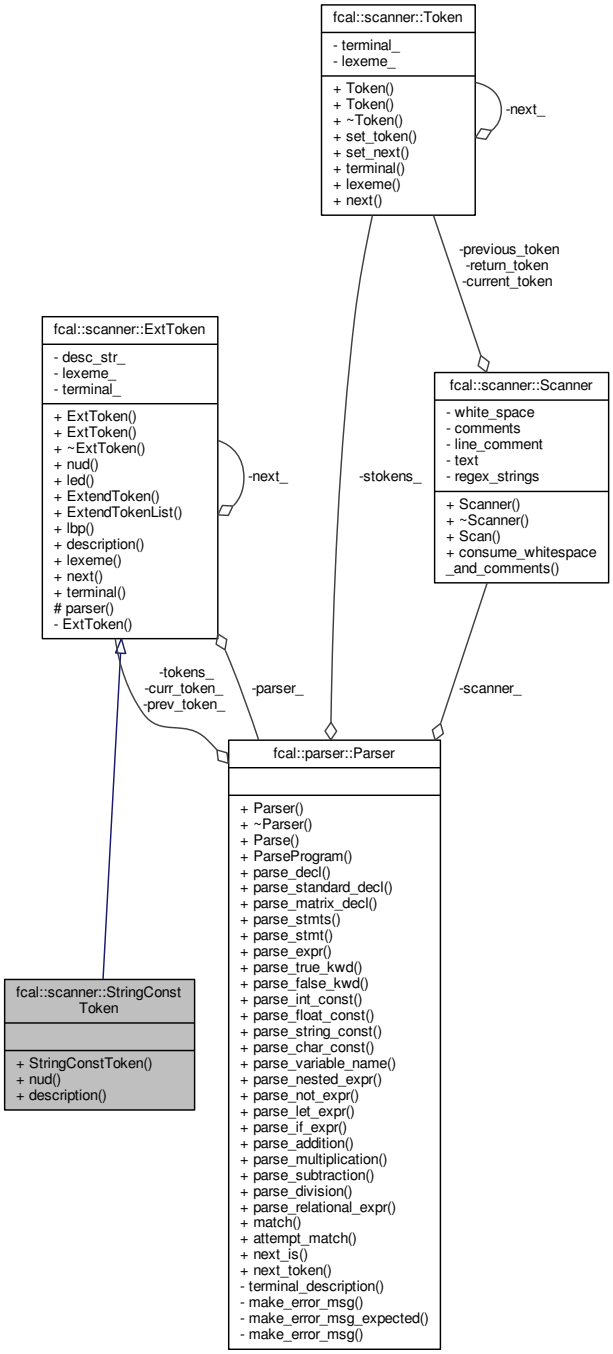
String Const.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::StringConstToken:



Collaboration diagram for fcal::scanner::StringConstToken:



Public Member Functions

- StringConstToken (parser::Parser ∗p, Token ∗t)
- parser::ParseResult nud ()
- std::string description ()

Additional Inherited Members

7.45.1 Detailed Description

String Const.

7.45.2 Constructor & Destructor Documentation

7.45.2.1 fcal::scanner::StringConstToken::StringConstToken (parser::Parser * *p*, Token * *t*) [inline]

7.45.3 Member Function Documentation

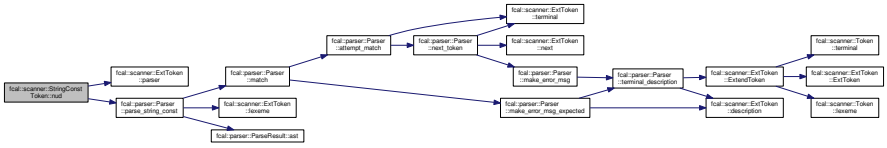
7.45.3.1 std::string fcal::scanner::StringConstToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.45.3.2 parser::ParseResult fcal::scanner::StringConstToken::nud (void) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



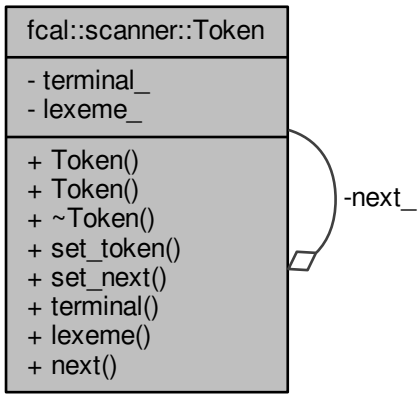
The documentation for this class was generated from the following file:

- include/ext_token.h

7.46 fcal::scanner::Token Class Reference

`#include <token_class.h>`

Collaboration diagram for fcal::scanner::Token:



Public Member Functions

- Token ()
  Default constructor for Token.
- Token (std::string, const TokenType, Token ∗)
  Constructor which initializes all attributes of Token.
- ∼Token ()
- void set_token (TokenType t, std::string lex)
  Set the values of the current token.
- void set_next (Token ∗current)
  end set_token
- TokenType terminal ()
  end set_next
- std::string lexeme ()
  Get the value of lexeme.
- Token ∗ next ()
  Get the value of next.

Private Attributes

- TokenType terminal_
  Type of the Token.
- std::string lexeme_
  lexeme of the Token
- Token ∗ next_
  Next Token in the Token Array returned by Scan of Scanner.

7.46.1 Detailed Description

This class holds information about a Token.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 fcal::scanner::Token::Token ( )

Default constructor for Token.

7.46.2.2 fcal::scanner::Token::Token (std::string *a*, const TokenType *b*, Token * *c*)

Constructor which initializes all attributes of Token.

Parameters

a	string representing the lexeme of the Token
b	TokenType of the Token
c	Token* which holds address of the next Token in the Token list returned by Scan method of Scanner

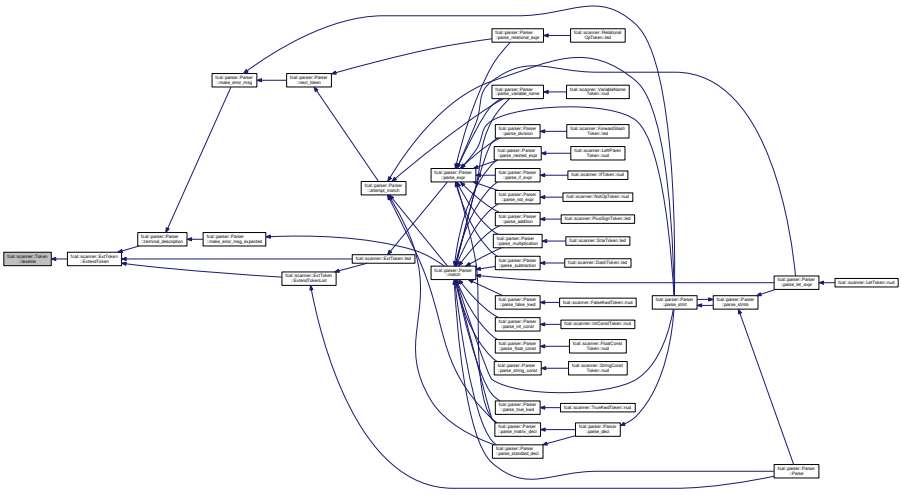
7.46.2.3 fcal::scanner::Token::~Token ( )

7.46.3 Member Function Documentation

7.46.3.1 std::string fcal::scanner::Token::lexeme (void)

Get the value of lexeme.

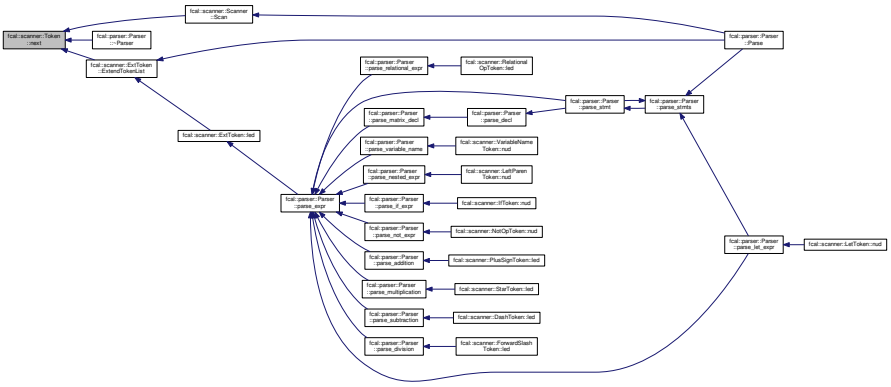
Here is the caller graph for this function:



7.46.3.2 Token * fcal::scanner::Token::next (void)

Get the value of next.

Here is the caller graph for this function:

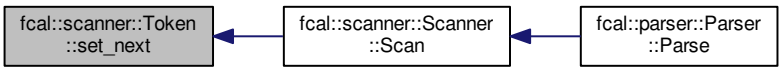


7.46.3.3 void fcal::scanner::Token::set_next (Token * *current*)

end set_token

Set the value of the previous token to point to the current token

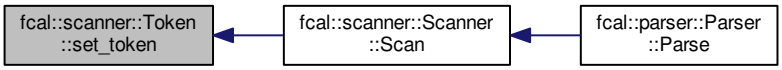
Here is the caller graph for this function:



7.46.3.4 void fcal::scanner::Token::set_token (TokenType *t*, std::string *lex*)

Set the values of the current token.

Here is the caller graph for this function:

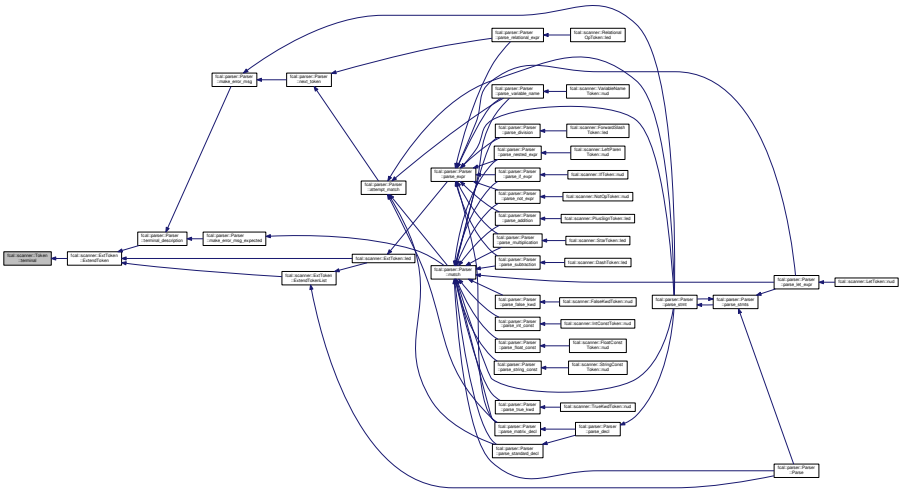


7.46.3.5 TokenType fcal::scanner::Token::terminal (void)

end set_next

Get the value of terminal

Here is the caller graph for this function:



7.46.4 Member Data Documentation

7.46.4.1 std::string fcal::scanner::Token::lexeme_ [private]

lexeme of the Token

7.46.4.2 Token* fcal::scanner::Token::next_ [private]

Next Token in the Token Array returned by Scan of Scanner.

7.46.4.3 TokenType fcal::scanner::Token::terminal_ [private]

Type of the Token.

The documentation for this class was generated from the following files:

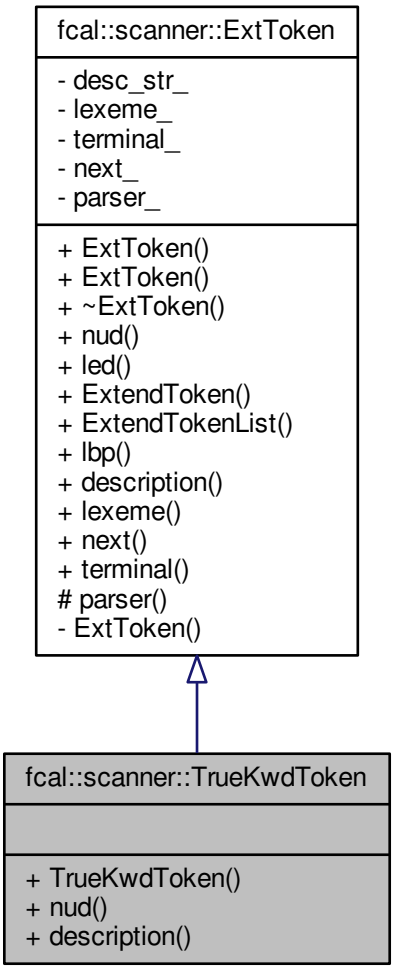
- include/token_class.h
- src/token_class.cc

7.47 fcal::scanner::TrueKwdToken Class Reference

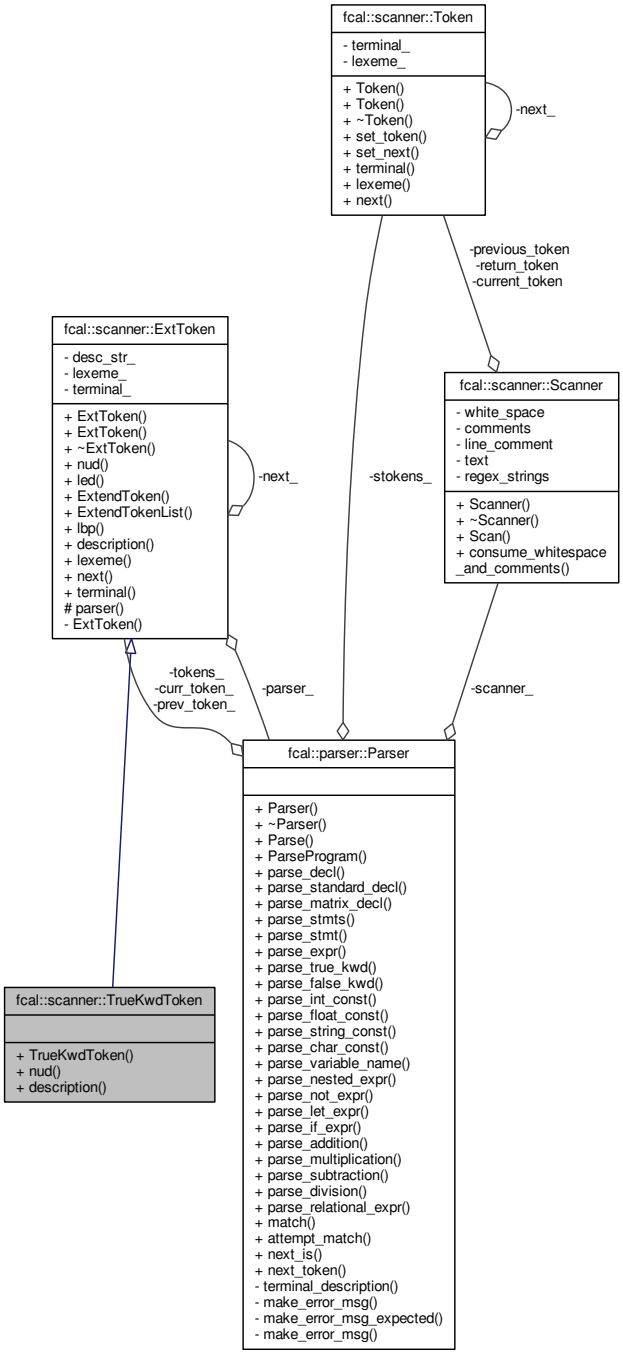
True Kwd.

```
#include <ext_token.h>
```

Inheritance diagram for fcal::scanner::TrueKwdToken:



Collaboration diagram for fcal::scanner::TrueKwdToken:



Public Member Functions

- TrueKwdToken (parser::Parser ∗p, Token ∗t)
- parser::ParseResult nud ()
- std::string description ()

Additional Inherited Members

7.47.1 Detailed Description

True Kwd.

7.47.2 Constructor & Destructor Documentation

7.47.2.1 fcal::scanner::TrueKwdToken::TrueKwdToken (parser::Parser * *p*, Token * *t*) [inline]

7.47.3 Member Function Documentation

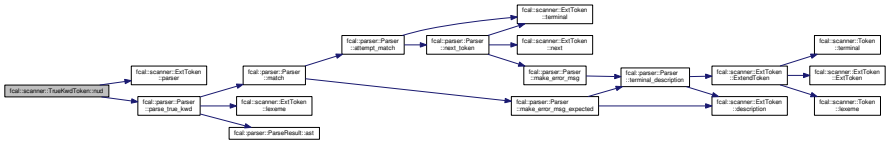
7.47.3.1 std::string fcal::scanner::TrueKwdToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.47.3.2 parser::ParseResult fcal::scanner::TrueKwdToken::nud (void) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



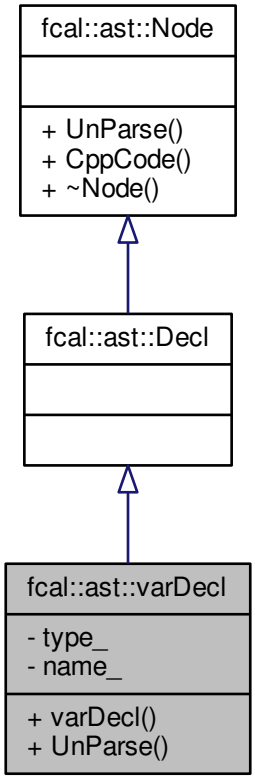
The documentation for this class was generated from the following file:

- include/ext_token.h

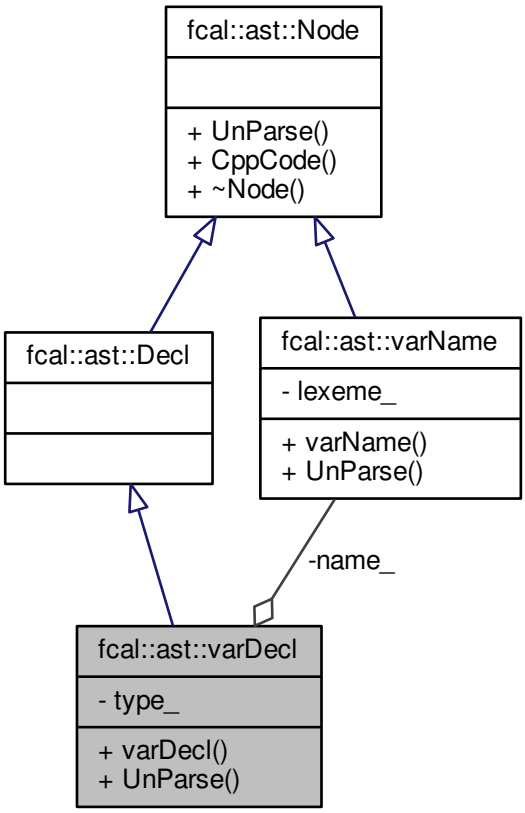
7.48 fcal::ast::varDecl Class Reference

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

Inheritance diagram for fcal::ast::varDecl:



Collaboration diagram for fcal::ast::varDecl:



Public Member Functions

- varDecl (decType type, varName *name)
- std::string UnParse (void)

Private Attributes

- decType type_

  type of declaration
- varName * name_

  *varName declared as a type of type_*

7.48.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the productions,
Decl ::= 'int' varName ';'
Decl ::= 'float' varName ';'
Decl ::= 'string' varName ';'
Decl ::= 'boolean' varName ';'

7.48.2 Constructor & Destructor Documentation

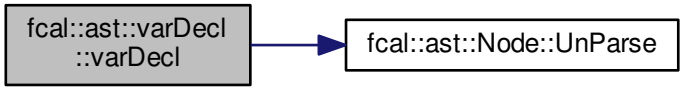
7.48.2.1 fcal::ast::varDecl::varDecl (decType *type*, varName * *name*) [inline]

This constructor takes two parameters

Parameters

type	a decType which tells the type of declaration
name	a varName used as a declaration

Here is the call graph for this function:



7.48.3 Member Function Documentation

7.48.3.1 std::string fcal::ast::varDecl::UnParse (void) [virtual]

Returns a string depending on the value of type_.
If type_ = int_ then return string : "int " + name_->UnParse() + ";"
If type_ = float_ then return string : "float " + name_->UnParse() + ";"
If type_ = string_ then return string : "string " + name_->UnParse() + ";"
If type_ = boolean_ then return string : "boolean " + name_->UnParse() + ";"

Reimplemented from fcal::ast::Node.

7.48.4 Member Data Documentation

7.48.4.1 varName* fcal::ast::varDecl::name_ [private]

varName declared as a type of type_

7.48.4.2 decType fcal::ast::varDecl::type_ [private]

type of declaration

The documentation for this class was generated from the following files:

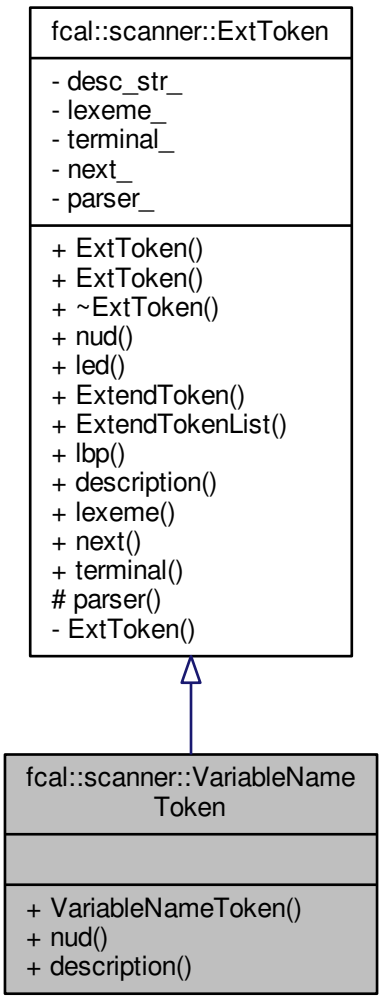
- include/ast.h
- src/ast.cc

7.49 fcal::scanner::VariableNameToken Class Reference

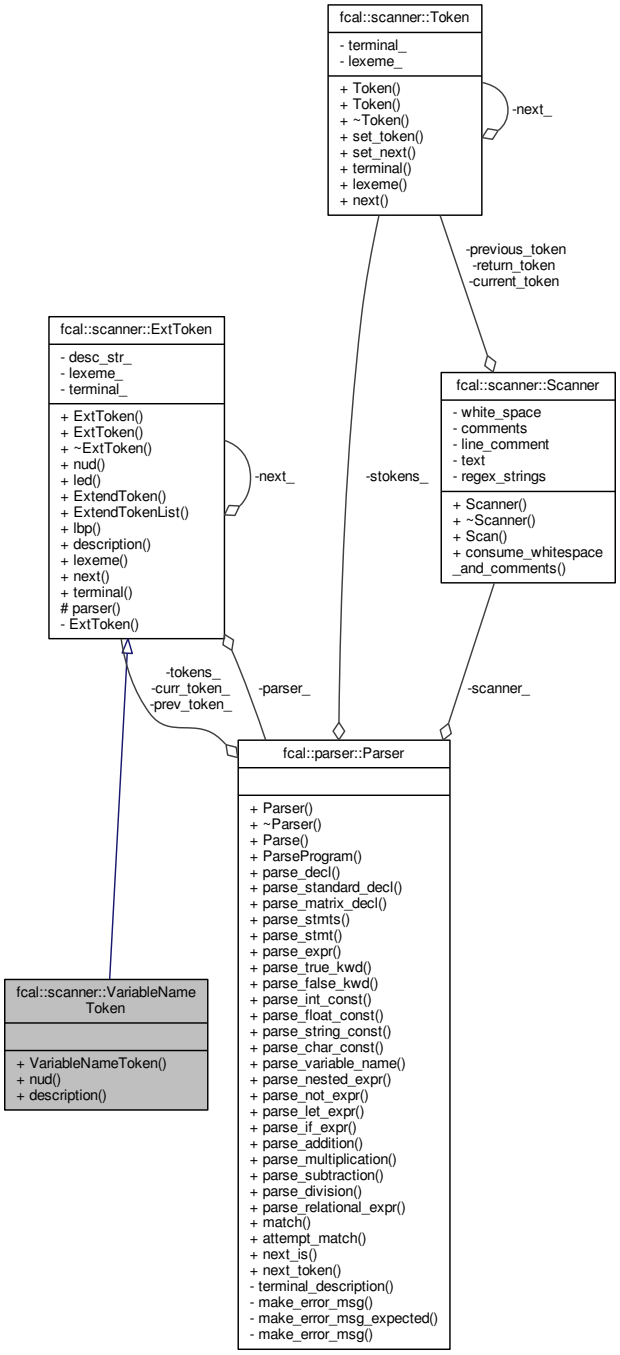
Variable Name.

`#include <ext_token.h>`

Inheritance diagram for fcal::scanner::VariableNameToken:



Collaboration diagram for fcal::scanner::VariableNameToken:



Public Member Functions

- VariableNameToken (parser::Parser ∗p, Token ∗t)
- parser::ParseResult nud ()
- std::string description ()

Additional Inherited Members

7.49.1 Detailed Description

Variable Name.

7.49.2 Constructor & Destructor Documentation

7.49.2.1 fcal::scanner::VariableNameToken::VariableNameToken (parser::Parser * *p*, Token * *t*) [inline]

7.49.3 Member Function Documentation

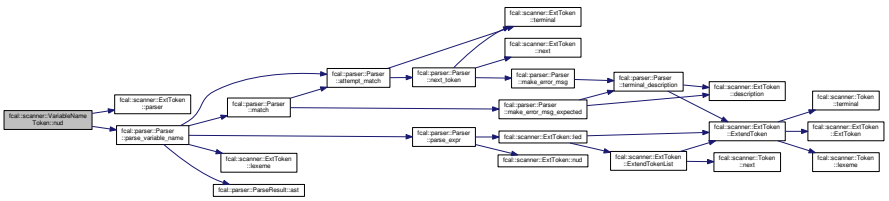
7.49.3.1 std::string fcal::scanner::VariableNameToken::description ( ) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

7.49.3.2 parser::ParseResult fcal::scanner::VariableNameToken::nud (void) [inline], [virtual]

Reimplemented from fcal::scanner::ExtToken.

Here is the call graph for this function:



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

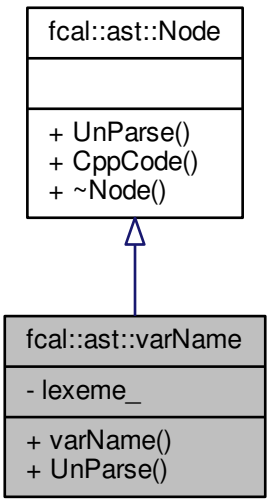
- include/ext_token.h

7.50 fcal::ast::varName Class Reference

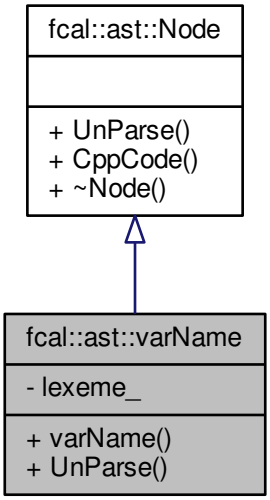
This class holds the lexeme details of a variable name.

`#include <ast.h>`

Inheritance diagram for fcal::ast::varName:



Collaboration diagram for fcal::ast::varName:



Public Member Functions

- varName (std::string lexeme)
- std::string UnParse (void)

  This function returns the variable name's lexeme.

Private Attributes

- std::string lexeme_

  A string holding the lexeme of varName.

7.50.1 Detailed Description

This class holds the lexeme details of a variable name.

7.50.2 Constructor & Destructor Documentation

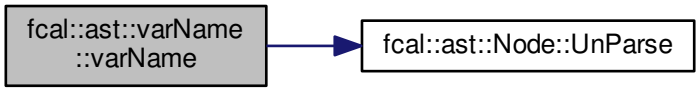
7.50.2.1 fcal::ast::varName::varName (std::string *lexeme*) [inline],[explicit]

This constructor requires one parameter: lexeme

Parameters

lexeme	a string which is the variable name

Here is the call graph for this function:



7.50.3 Member Function Documentation

7.50.3.1 std::string fcal::ast::varName::UnParse (void) [virtual]

This function returns the variable name's lexeme.

returns the string : lexeme_

Reimplemented from fcal::ast::Node.

7.50.4 Member Data Documentation

7.50.4.1 std::string fcal::ast::varName::lexeme_ [private]

A string holding the lexeme of varName.

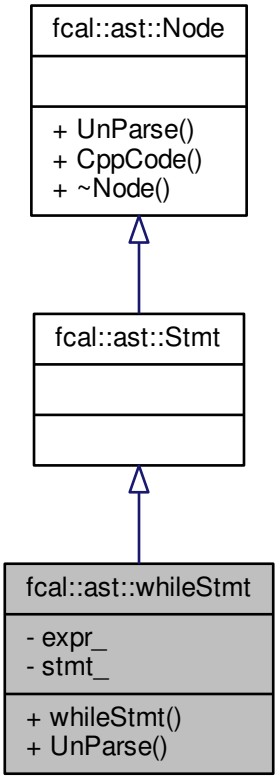
The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

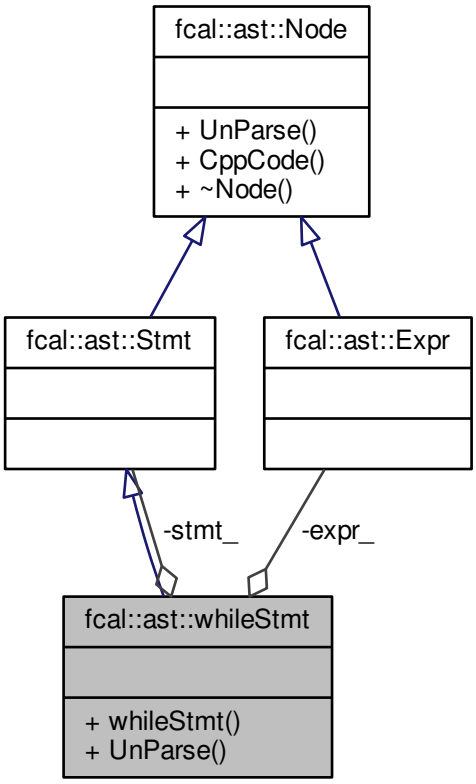
7.51 fcal::ast::whileStmt Class Reference

`#include <ast.h>`

Inheritance diagram for fcal::ast::whileStmt:



Collaboration diagram for fcal::ast::whileStmt:



Public Member Functions

- whileStmt (Expr ∗expr, Stmt ∗stmt)
- std::string UnParse (void)

  Returns the string : "while (" + expr_->UnParse() + ")\n" + stmt_->UnParse()

Private Attributes

- Expr ∗ expr_

  an Expr evluated as a condition in while
- Stmt ∗ stmt_

  stmt_ executed while expr_ is still True

7.51.1 Detailed Description

This is a concrete class in the ast class heirarchy. It implements the production,
Stmt ::= 'while' '(' Expr ')' Stmt

7.51.2 Constructor & Destructor Documentation

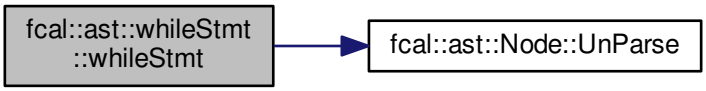
7.51.2.1 fcal::ast::whileStmt::whileStmt (Expr * *expr*, Stmt * *stmt*) [inline]

The constructor takes two arguements

Parameters

expr	an Expr evluated as a condition in while
stmt	a Stmt executed while the Expr is still True

Here is the call graph for this function:



7.51.3 Member Function Documentation

7.51.3.1 std::string fcal::ast::whileStmt::UnParse (void) [virtual]

Returns the string : "while (" + expr_->UnParse() + ")\n" + stmt_->UnParse()

Reimplemented from fcal::ast::Node.

7.51.4 Member Data Documentation

7.51.4.1 Expr* fcal::ast::whileStmt::expr_ [private]

an Expr evluated as a condition in while

7.51.4.2 Stmt* fcal::ast::whileStmt::stmt_ [private]

stmt_ executed while expr_ is still True

The documentation for this class was generated from the following files:

- include/ast.h
- src/ast.cc

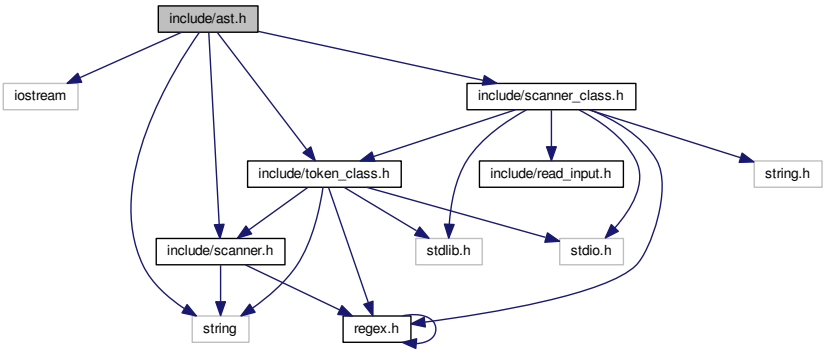
Chapter 8

File Documentation

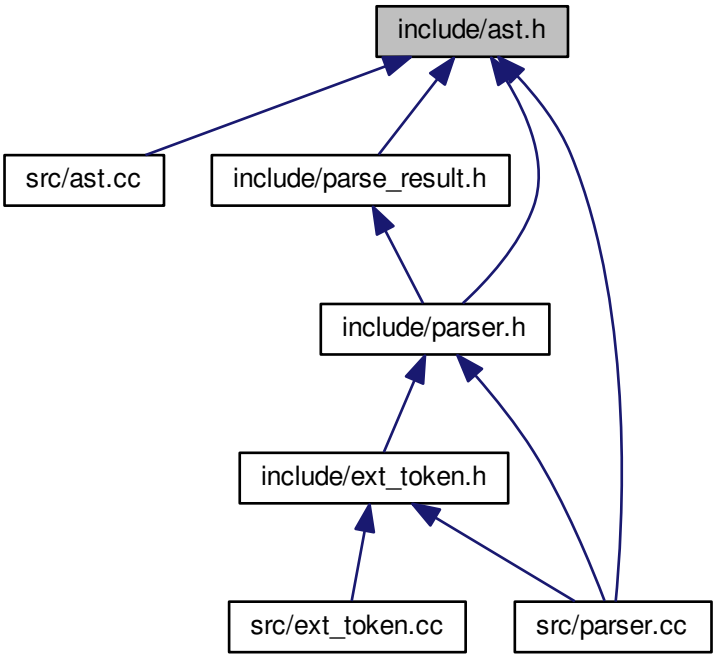
8.1 include/ast.h File Reference

```
#include <iostream>
#include <string>
#include "include/scanner.h"
#include "include/scanner_class.h"
#include "include/token_class.h"
```

Include dependency graph for ast.h:



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



Classes

- class fcal::ast::Node
- class fcal::ast::varName

  This class holds the lexeme details of a variable name.
- class fcal::ast::Stmts
- class fcal::ast::Stmt
- class fcal::ast::Decl
- class fcal::ast::Expr
- class fcal::ast::Root
- class fcal::ast::emptyStmts
- class fcal::ast::seqStmts
- class fcal::ast::declStmt
- class fcal::ast::bracketStmt
- class fcal::ast::ifStmt
- class fcal::ast::ifElseStmt
- class fcal::ast::equalsStmt
- class fcal::ast::printStmt
- class fcal::ast::repeatStmt
- class fcal::ast::whileStmt
- class fcal::ast::semiColonStmt
- class fcal::ast::varDecl
- class fcal::ast::matrixDecl
- class fcal::ast::constantExpr

- class fcal::ast::binaryExpr
- class fcal::ast::boolExpr
- class fcal::ast::matrixExpr
- class fcal::ast::nestedOrExpr
- class fcal::ast::parenthesisExpr
- class fcal::ast::letExpr
- class fcal::ast::ifExpr
- class fcal::ast::notExpr

Namespaces

- fcal

  Namespaces.
- fcal::ast

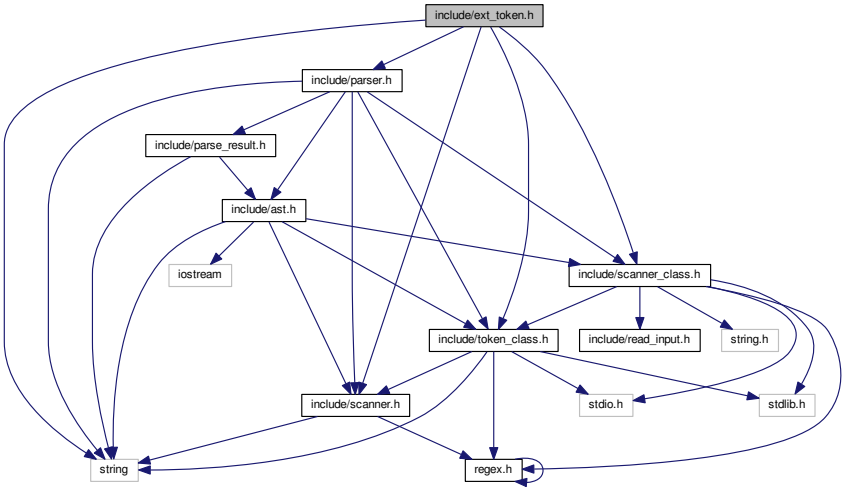
Enumerations

- enum fcal::ast::decType { fcal::ast::int_, fcal::ast::float_, fcal::ast::string_, fcal::ast::boolean_ }

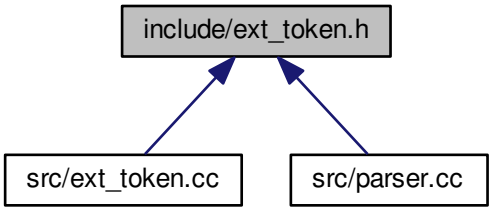
8.2 include/ext_token.h File Reference

```
#include <string>
#include "include/parser.h"
#include "include/scanner.h"
#include "include/token_class.h"
#include "include/scanner_class.h"
```

Include dependency graph for ext_token.h:



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



Classes

- class fcal::scanner::ExtToken
- class fcal::scanner::NotOpToken
- class fcal::scanner::TrueKwdToken

  True Kwd.
- class fcal::scanner::FalseKwdToken

  False Kwd.
- class fcal::scanner::IntConstToken

  Int Const.
- class fcal::scanner::FloatConstToken

  Float Const.
- class fcal::scanner::StringConstToken

  String Const.
- class fcal::scanner::CharConstToken

  Char Const.
- class fcal::scanner::VariableNameToken

  Variable Name.
- class fcal::scanner::IfToken
- class fcal::scanner::LetToken
- class fcal::scanner::LeftParenToken

  Left Paren.
- class fcal::scanner::PlusSignToken

  Plus Sign.
- class fcal::scanner::StarToken

  Star.
- class fcal::scanner::DashToken

  Dash.
- class fcal::scanner::ForwardSlashToken

  ForwardSlash.
- class fcal::scanner::RelationalOpToken

  Relational Op.
- class fcal::scanner::EndOfFileToken

  End of File.

Namespaces

- fcal

  Namespaces.

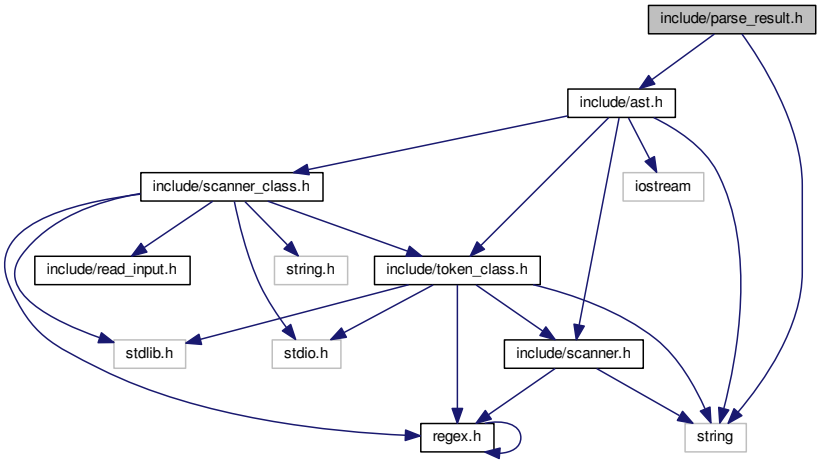
- fcal::scanner

8.3 include/mainpage.h File Reference

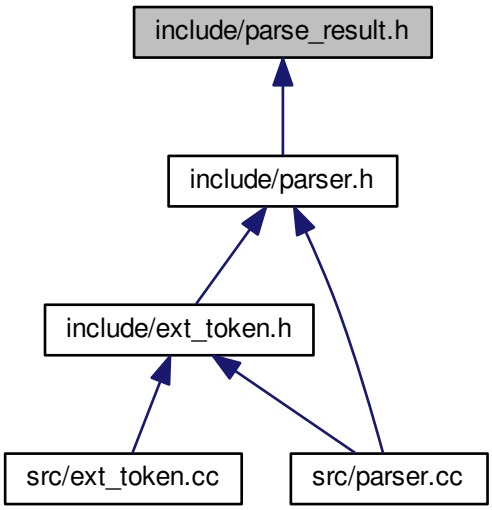
8.4 include/parse_result.h File Reference

```
#include <string>
#include "include/ast.h"
```

Include dependency graph for parse_result.h:



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



Classes

- class fcal::parser::ParseResult

Namespaces

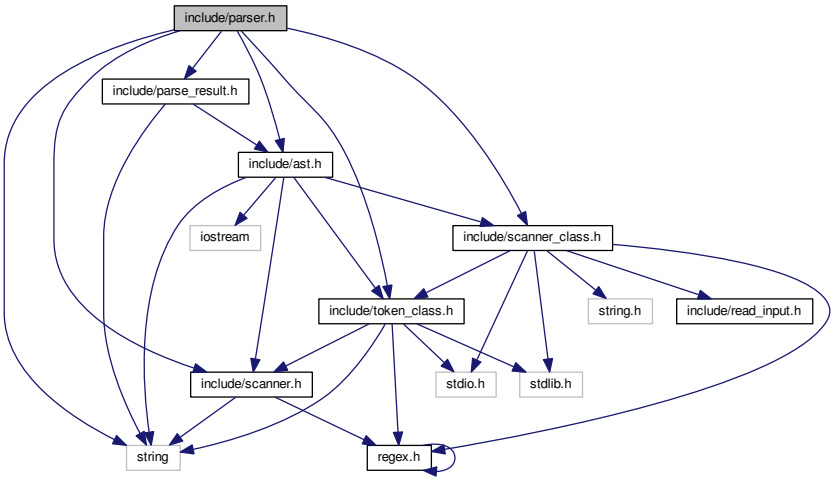
- fcal

  Namespaces.
- fcal::parser

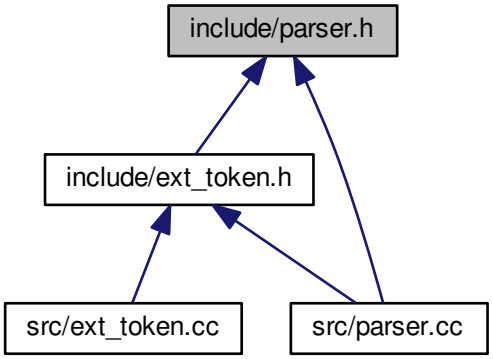
8.5 include/parser.h File Reference

```
#include <string>
#include "include/parse_result.h"
#include "include/scanner.h"
#include "include/scanner_class.h"
#include "include/token_class.h"
#include "include/ast.h"
```

Include dependency graph for parser.h:



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



Classes

- class fcal::parser::Parser

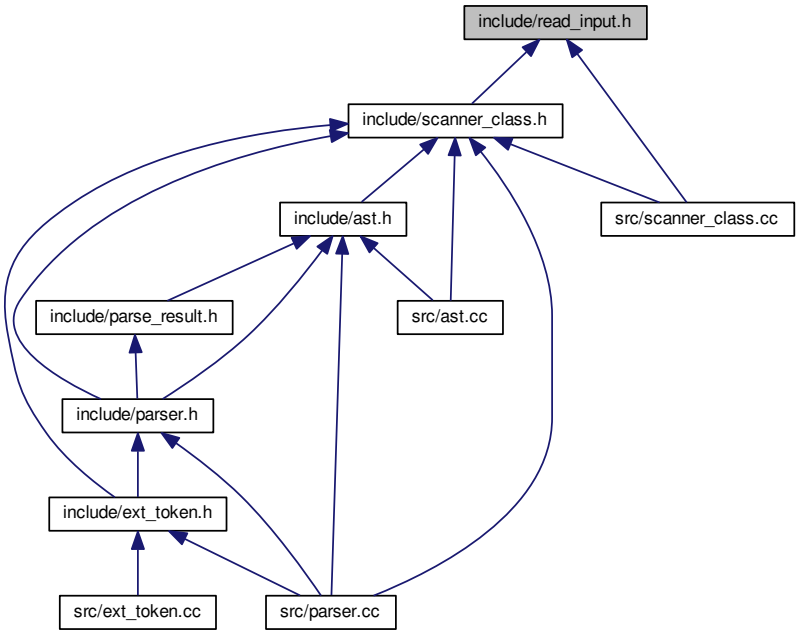
Namespaces

- fcal

  Namespaces.
- fcal::scanner
- fcal::parser

8.6 include/read_input.h File Reference

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



Namespaces

- fcal

  Namespaces.
- fcal::scanner

Functions

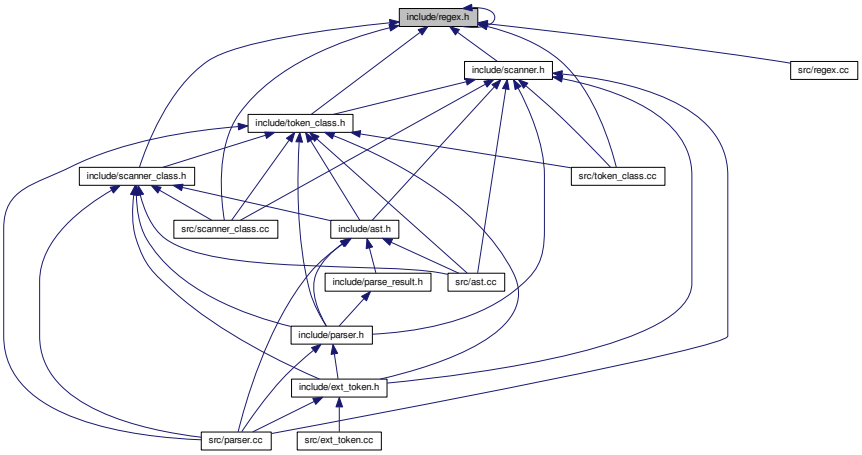
- char * fcal::scanner::ReadInput (int argc, char **argv)
- char * fcal::scanner::ReadInputFromFile (const char *filename)

8.7 include/regex.h File Reference

```
#include <regex.h>
```

Include dependency graph for regex.h:

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



Namespaces

- fcal

  Namespaces.
- fcal::scanner

Functions

- regex_t ∗ fcal::scanner::make_regex (const char ∗pattern)
- int fcal::scanner::match_regex (regex_t ∗re, const char ∗text)

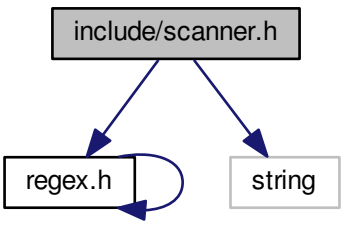
Variables

- const int fcal::scanner::kRegexNSub = 1

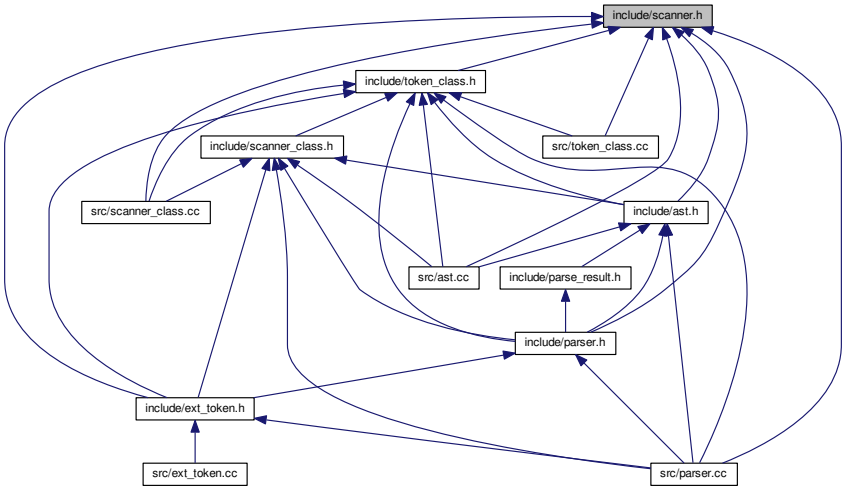
8.8 include/scanner.h File Reference

```
#include <regex.h>
#include <string>
```

Include dependency graph for scanner.h:



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



Namespaces

- fcal

  Namespaces.
- fcal::scanner

Typedefs

- typedef enum kTokenEnumType fcal::scanner::TokenType

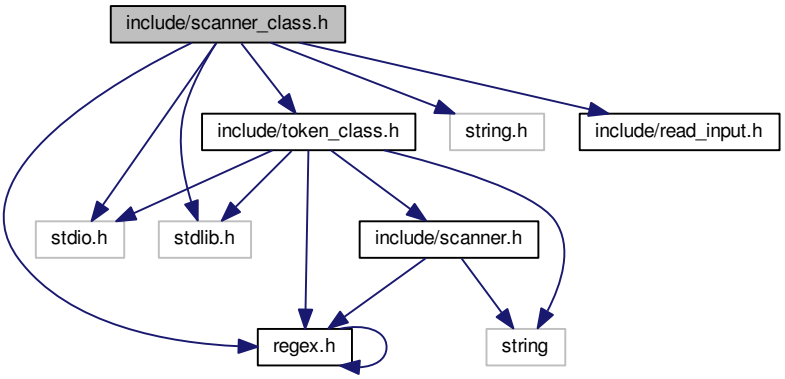
Enumerations

- enum fcal::scanner::kTokenEnumType {
  fcal::scanner::kIntKwd, fcal::scanner::kFloatKwd, fcal::scanner::kBoolKwd, fcal::scanner::kTrueKwd,
  fcal::scanner::kFalseKwd, fcal::scanner::kStringKwd, fcal::scanner::kMatrixKwd, fcal::scanner::kLetKwd,
  fcal::scanner::kInKwd, fcal::scanner::kEndKwd, fcal::scanner::kIfKwd, fcal::scanner::kThenKwd,
  fcal::scanner::kElseKwd, fcal::scanner::kRepeatKwd, fcal::scanner::kWhileKwd, fcal::scanner::kPrintKwd,
  fcal::scanner::kToKwd, fcal::scanner::kIntConst, fcal::scanner::kFloatConst, fcal::scanner::kStringConst,
  fcal::scanner::kVariableName, fcal::scanner::kLeftParen, fcal::scanner::kRightParen, fcal::scanner::kLeft↩
  Curly,
  fcal::scanner::kRightCurly, fcal::scanner::kLeftSquare, fcal::scanner::kRightSquare, fcal::scanner::kSemi↩
  Colon,
  fcal::scanner::kColon, fcal::scanner::kAssign, fcal::scanner::kPlusSign, fcal::scanner::kStar,
  fcal::scanner::kDash, fcal::scanner::kForwardSlash, fcal::scanner::kLessThan, fcal::scanner::kLessThan↩
  Equal,
  fcal::scanner::kGreaterThan, fcal::scanner::kGreaterThanEqual, fcal::scanner::kEqualsEquals, fcal↩
  ::scanner::kNotEquals,
  fcal::scanner::kAndOp, fcal::scanner::kOrOp, fcal::scanner::kNotOp, fcal::scanner::kEndOfFile,
  fcal::scanner::kLexicalError }

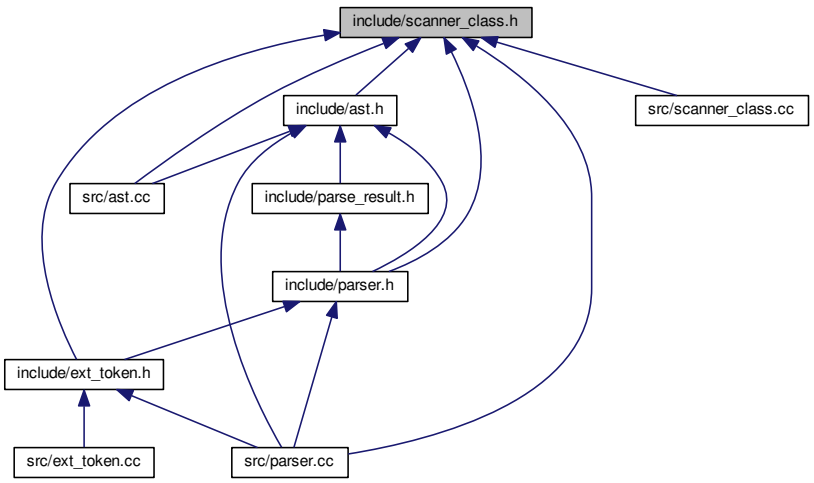
8.9 include/scanner_class.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "include/token_class.h"
#include "include/read_input.h"
#include "include/regex.h"
```

Include dependency graph for scanner_class.h:



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



Classes

- class fcal::scanner::Scanner

Namespaces

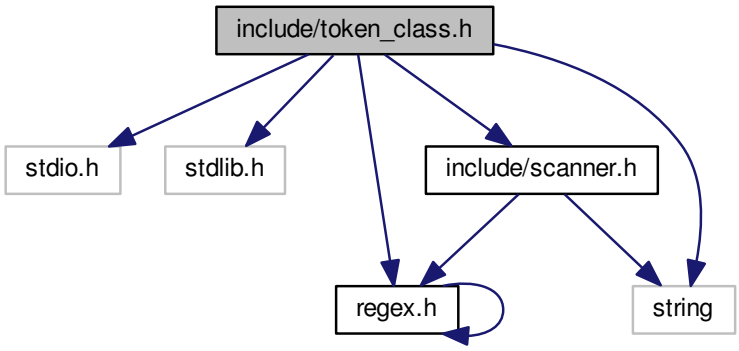
- fcal

  Namespaces.
- fcal::scanner

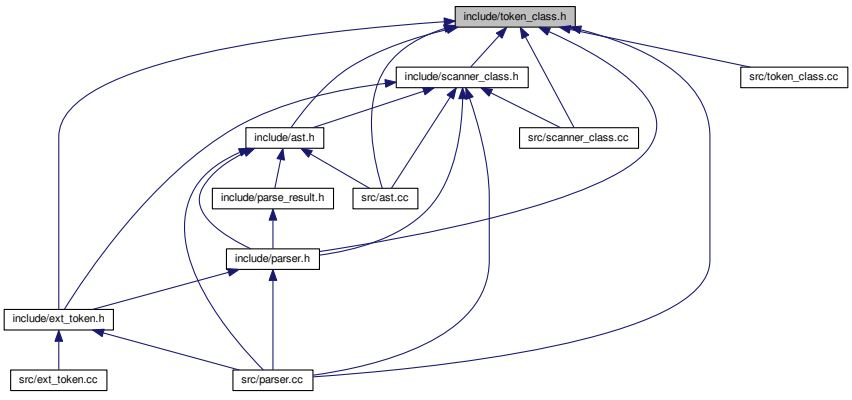
8.10 include/token_class.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <regex.h>
#include <string>
#include "include/scanner.h"
```

Include dependency graph for token_class.h:



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



Classes

- class fcal::scanner::Token

Namespaces

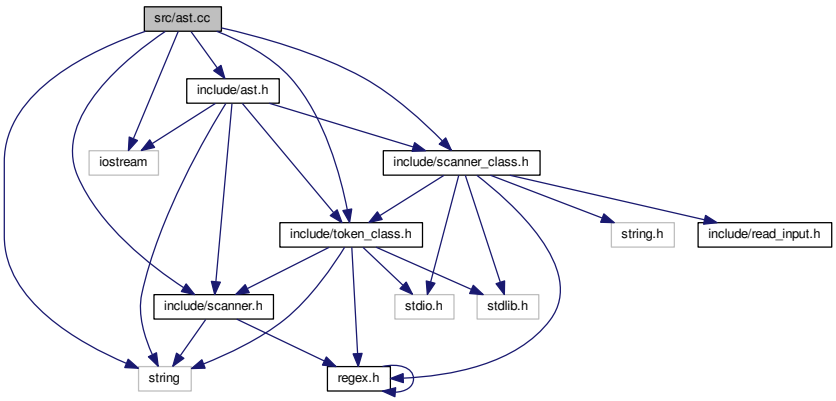
- fcal

  Namespaces.
- fcal::scanner

8.11 src/ast.cc File Reference

```
#include <iostream>
#include <string>
#include "include/scanner.h"
#include "include/scanner_class.h"
#include "include/token_class.h"
#include "include/ast.h"
```

Include dependency graph for ast.cc:



Namespaces

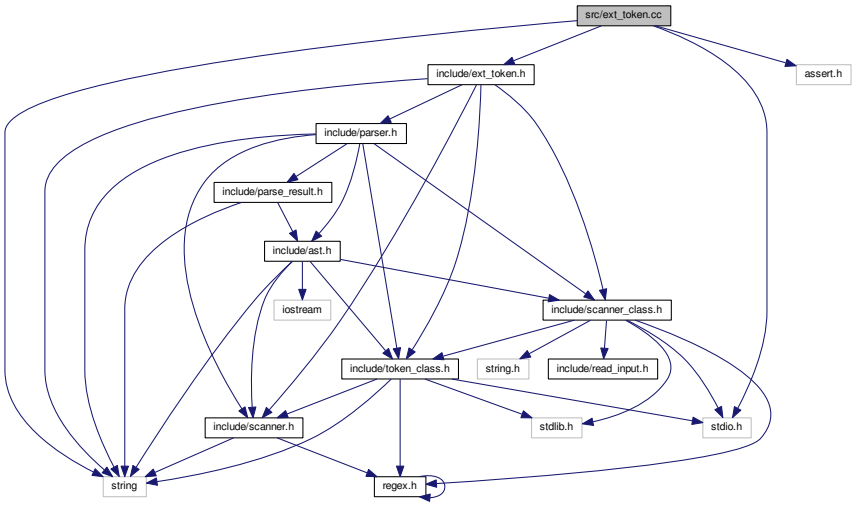
- fcal

  Namespaces.
- fcal::ast

8.12 src/ext_token.cc File Reference

```
#include "include/ext_token.h"
#include <assert.h>
#include <stdio.h>
#include <string>
```

Include dependency graph for ext_token.cc:



Namespaces

- fcal

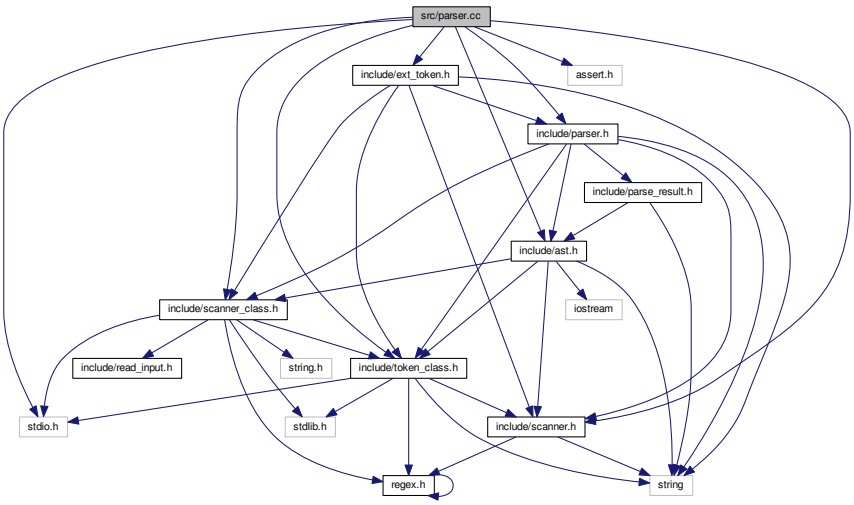
  Namespaces.

- fcal::scanner

8.13 src/parser.cc File Reference

```
#include "include/parser.h"
#include <assert.h>
#include <stdio.h>
#include "include/ext_token.h"
#include "include/scanner.h"
#include "include/scanner_class.h"
#include "include/token_class.h"
#include "include/ast.h"
```

Include dependency graph for parser.cc:



Namespaces

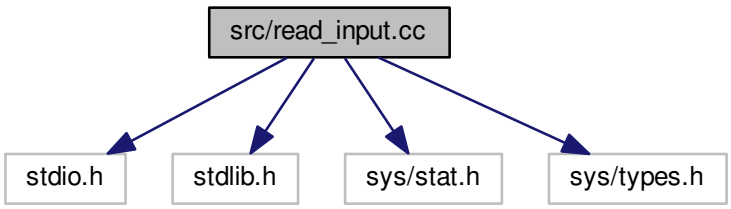
- fcal

  Namespaces.
- fcal::parser

8.14 src/read_input.cc File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
```

Include dependency graph for read_input.cc:



Namespaces

- fcal

  Namespaces.
- fcal::scanner

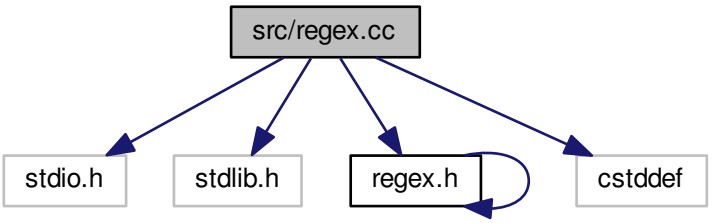
Functions

- char * fcal::scanner::ReadInputFromFile (const char *filename)
- char * fcal::scanner::ReadInput (int argc, char **argv)

8.15 src/regex.cc File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <regex.h>
#include <cstddef>
```

Include dependency graph for regex.cc:



Namespaces

- fcal

  Namespaces.
- fcal::scanner

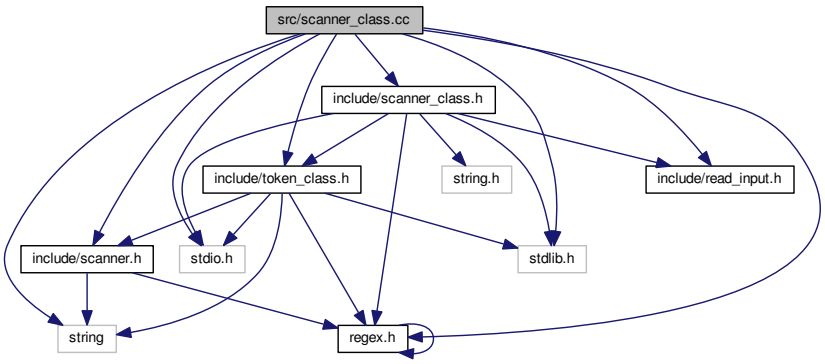
Functions

- regex_t * fcal::scanner::make_regex (const char *pattern)
- int fcal::scanner::match_regex (regex_t *re, const char *text)

8.16 src/scanner_class.cc File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include "include/scanner.h"
#include "include/scanner_class.h"
#include "include/token_class.h"
#include "include/regex.h"
#include "include/read_input.h"
```

Include dependency graph for scanner_class.cc:



Namespaces

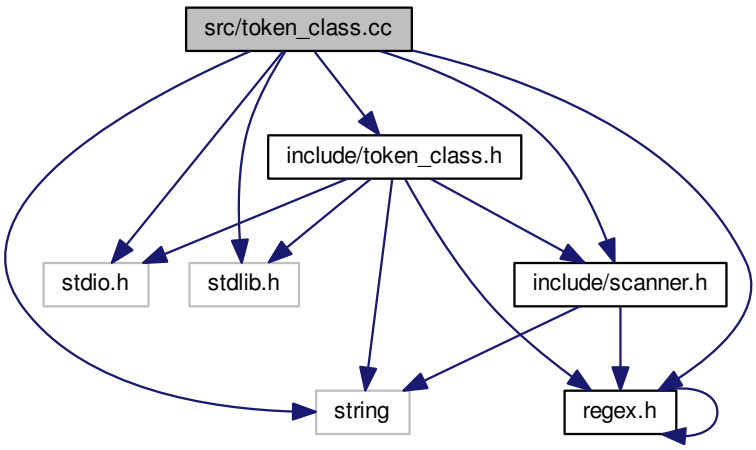
- fcal

  Namespaces.
- fcal::scanner

8.17 src/token_class.cc File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <regex.h>
#include <string>
#include "include/token_class.h"
#include "include/scanner.h"
```

Include dependency graph for token_class.cc:



Namespaces

- fcal

  Namespaces.
- fcal::scanner

Index