Raspberry Pi - webové rozhranie

Generated by Doxygen 1.8.6

Sun Dec 7 2014 19:00:27

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

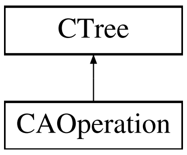
Class Documentation

4.1 CAOperation Class Reference

Arithmetic operations node class.

`#include <tree.hpp>`

Inheritance diagram for CAOperation:



Public Types

- enum **Operation** {
  TIMES, **DIVIDE**, **MODULO**, **MINUS**,
  PLUS }

Public Member Functions

- CAOperation (Operation o, CTree ∗op1, CTree ∗op2)

  Constructor.
- virtual CArray ∗ getArray (CArray ∗data)

  Return reference to the array with one dereference.
- virtual int isTrue (CArray ∗data)

  Method to determinate is element true/false, used in If/For condition.

Protected Attributes

- Operation **m_o**
- CArray **m_val**
- CTree ∗ **m_op1**
- CTree ∗ **m_op2**

4.1.1 Detailed Description

Arithmetic operations node class.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 CAOperation::CAOperation (Operation *o*, CTree ∗ *op1*, CTree ∗ *op2*)

Constructor.

Parameters

`in`	*o*	Operation type
`in`	*op1*	First operand
`in`	*op2*	Second operand

4.1.3 Member Function Documentation

4.1.3.1 CArray ∗ CAOperation::getArray (CArray ∗ *data*) `[virtual]`

Return reference to the array with one dereference.

Parameters

`in`	*data*	Data from View

Return values

CArray∗	Array data

Reimplemented from CTree.

4.1.3.2 int CAOperation::isTrue (CArray ∗ *data*) `[virtual]`

Method to determinate is element true/false, used in If/For condition.

Parameters

`in`	*data*	Data from View

Reimplemented from CTree.

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

- lib/tree.hpp
- lib/tree.cpp

4.2 CArray Class Reference

Classes

- struct Tnode

Public Member Functions

- **CArray** (const CArray &x)
- **CArray** (const int x)

- **CArray** (const float x)
- **CArray** (const char ∗x)
- void **Print** ()
- CArray ∗ getNext ()

    Method to crossing item array.
- CArray & **operator[ ]** (const char ∗x)
- CArray & **operator[ ]** (const int x)
- CArray & **operator=** (const CArray &x)
- CArray operator∗ (const CArray &x)

    Times two item if they are interger.
- CArray operator/ (const CArray &x)

    Divide two item if they are interger.
- CArray operator% (const CArray &x)

    Modulo two item if they are interger.
- CArray operator+ (const CArray &x)

    Plus two item if they are interger.
- CArray operator- (const CArray &x)

    Minus two item if they are interger.
- CArray operator& (const CArray &x)

    String concat two items.
- int operator== (const CArray &x)

    Compare two items.
- int operator!= (const CArray &x)

    Compare two items.
- int operator< (const CArray &x)

    Compare length two items.
- int operator> (const CArray &x)

    Compare length two items.
- int operator<= (const CArray &x)

    Compare length two items.
- int operator>= (const CArray &x)

    Compare length two items.
- char ∗ getstr () const

    Return c string (char∗)
- int empty ()

    Is empty?
- int length ()

    Return string length.

Protected Member Functions

- int **isNumber** () const

Protected Attributes

- Tnode ∗ **root**
- Tnode ∗ **getnext**
- char ∗ **str**

Friends

- ostream & **operator**<< (ostream &os, const CArray &x)

4.2.1 Member Function Documentation

4.2.1.1 int CArray::empty ( )

Is empty?

If items are numbers compare by value, otherwise compare by string length

Return values

1	Item is empty string or 0
0	Else

4.2.1.2 CArray ∗ CArray::getNext ( )

Method to crossing item array.

This function return item by item, at the end return NULL

Return values

NULL	End items.
CArray	Item

4.2.1.3 char ∗ CArray::getstr ( ) const

Return c string (char∗)

Return values

char∗	C string

4.2.1.4 int CArray::length ( )

Return string length.

Return values

int	String length

4.2.1.5 int CArray::operator!= (const CArray & *x*)

Compare two items.

Return values

0	when equal
1	when not equal

4.2.1.6 CArray CArray::operator% (const CArray & *x*)

Modulo two item if they are interger.

Return values

CArray	Return in CArray

4.2.1.7 CArray CArray::operator& (const CArray & *x*)

String concat two items.

Return values

CArray	Return in CArray

4.2.1.8 CArray CArray::operator∗ (const CArray & *x*)

Times two item if they are interger.

Return values

CArray	Return in CArray

4.2.1.9 CArray CArray::operator+ (const CArray & *x*)

Plus two item if they are interger.

Return values

CArray	Return in CArray

4.2.1.10 CArray CArray::operator- (const CArray & *x*)

Minus two item if they are interger.

Return values

CArray	Return in CArray

4.2.1.11 CArray CArray::operator/ (const CArray & *x*)

Divide two item if they are interger.

Return values

CArray	Return in CArray

4.2.1.12 int CArray::operator< (const CArray & *x*)

Compare length two items.

If items are numbers compare by value, otherwise compare by string length

Return values

1	Item lower, or smaller length
0	Item greater, or greater length

4.2.1.13 int CArray::operator<= (const CArray & *x*)

Compare length two items.

If items are numbers compare by value, otherwise compare by string length

Return values

1	Item lower or equal, or smaller or equal length
0	Item greater, or greater length

4.2.1.14 int CArray::operator== (const CArray & *x*)

Compare two items.

Return values

1	when equal
0	when not equal

4.2.1.15 int CArray::operator> (const CArray & *x*)

Compare length two items.

If items are numbers compare by value, otherwise compare by string length

Return values

0	Item lower, or smaller length
1	Item greater, or greater length

4.2.1.16 int CArray::operator>= (const CArray & *x*)

Compare length two items.

If items are numbers compare by value, otherwise compare by string length

Return values

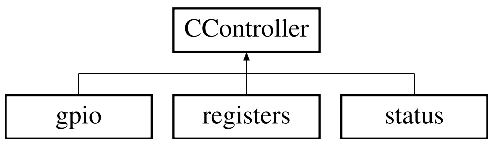
0	Item lower, or smaller length
1	Item greater or equal, or greater or equal length

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

- lib/array.hpp
- lib/array.cpp

4.3 CController Class Reference

Inheritance diagram for CController:



Public Member Functions

- virtual CView ∗ **Run** ()

Public Attributes

- CHeader ∗ **header**
- CMenu ∗ **menu**

Protected Attributes

- CArray **_SERVER**
- CArray **_GET**
- CArray **_POST**

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

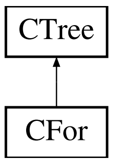
- lib/controller.hpp
- lib/controller.cpp

4.4 CFor Class Reference

For node class, iterated loop.

`#include <tree.hpp>`

Inheritance diagram for CFor:



Public Member Functions

- CFor (char ∗id, int first, int last, CTree ∗block, CTree ∗next)

    Constructor.
- virtual void Print (ostream &os, CArray ∗data)

    Method to print element to stream.

Protected Attributes

- char ∗ **m_id**
- int **m_first**
- int **m_last**
- CTree ∗ **m_block**

4.4.1 Detailed Description

For node class, iterated loop.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 CFor::CFor (char ∗ *id,* int *first,* int *last,* CTree ∗ *block,* CTree ∗ *next*)

Constructor.

Parameters

in	*id*	Iteration variables identifier
in	*first*	Iterate from
in	*last*	Iterate to
in	*block*	Iterated block
in	*next*	Next template node

4.4.3 Member Function Documentation

4.4.3.1 void CFor::Print (ostream & *os,* CArray ∗ *data*) [virtual]

Method to print element to stream.

This function parse query and send answer

Parameters

in	*os*	Output stream
in	*data*	Data from View class, used in template

Reimplemented from CTree.

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

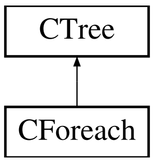
- lib/tree.hpp
- lib/tree.cpp

4.5 CForeach Class Reference

Foreach node class, passes all elements in array (CArray)

`#include <tree.hpp>`

Inheritance diagram for CForeach:



Public Member Functions

- CForeach (char ∗id, CVar ∗var, CTree ∗block, CTree ∗next)

    Constructor.
- virtual void Print (ostream &os, CArray ∗data)

    Method to print element to stream.

Protected Attributes

- char ∗ **m_id**
- CVar ∗ **m_var**
- CTree ∗ **m_block**

4.5.1 Detailed Description

Foreach node class, passes all elements in array (CArray)

4.5.2 Constructor & Destructor Documentation

4.5.2.1 CForeach::CForeach (char ∗ *id,* CVar ∗ *var,* CTree ∗ *block,* CTree ∗ *next*)

Constructor.

Parameters

in	*id*	Iteration variables identifier
in	*var*	Iterated variable
in	*block*	Iterated block
in	*next*	Next template node

4.5.3 Member Function Documentation

4.5.3.1 void CForeach::Print (ostream & *os,* CArray ∗ *data*) [virtual]

Method to print element to stream.

This function parse query and send answer

Parameters

in	*os*	Output stream
in	*data*	Data from View class, used in template

Reimplemented from CTree.

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

- lib/tree.hpp
- lib/tree.cpp

4.6 CHeader Class Reference

Public Member Functions

- void AddItem (const char ∗x)

  Add new html header.

Protected Member Functions

- char ∗∗ isSimilar (const char ∗x)

  Find similar header.
- int compare (const char ∗x, const char ∗y)

Case insensitive header compare.
- int itemSize ()

  Return header count.
- void Print (ostream &os) const

  Print header to stream.

Protected Attributes

- char $**$ **m_headers**

Friends

- ostream & operator<< (ostream &os, const CHeader &x)

  Operator <<.

4.6.1 Member Function Documentation

4.6.1.1 void CHeader::AddItem (const char $*$ *x*)

Add new html header.

Parameters

in	*x*	Header

4.6.1.2 int CHeader::compare (const char $*$ *x*, const char $*$ *y*) [protected]

Case insensitive header compare.

Parameters

in	*x*	First header
in	*y*	Second header

4.6.1.3 char $**$ CHeader::isSimilar (const char $*$ *x*) [protected]

Find similar header.

Check if that head exist

Parameters

in	*x*	Header

Return values

NULL	Not found.
char$**$	Pointer to header row.

4.6.1.4 int CHeader::itemSize ( ) [protected]

Return header count.

Return values

int	Size.

4.6.1.5 void CHeader::Print (ostream & *os*) const [protected]

Print header to stream.

Parameters

in	*os*	Output stream

4.6.2 Friends And Related Function Documentation

4.6.2.1 ostream& operator<< (ostream & *os*, const CHeader & *x*) [friend]

Operator <<.

Print headers to output stream

Parameters

in	*os*	Output stream
in	*x*	Cheader

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

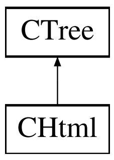
- lib/header.hpp
- lib/header.cpp

4.7 CHtml Class Reference

Html node class.

`#include <tree.hpp>`

Inheritance diagram for CHtml:



Public Member Functions

- **CHtml** (char ∗html, CTree ∗next)
- virtual void Print (ostream &os, CArray ∗data)

  Method to print element to stream.

Protected Attributes

- char ∗ **m_html**

4.7.1 Detailed Description

Html node class.

4.7.2 Member Function Documentation

4.7.2.1 void CHtml::Print (ostream & *os*, CArray ∗ *data*) [virtual]

Method to print element to stream.

This function parse query and send answer

Parameters

in	*os*	Output stream
in	*data*	Data from View class, used in template

Reimplemented from CTree.

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

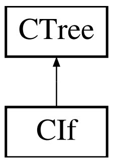
- lib/tree.hpp
- lib/tree.cpp

4.8 CIf Class Reference

If/else condition node class.

`#include <tree.hpp>`

Inheritance diagram for CIf:



Public Member Functions

- CIf (CTree ∗cond, CTree ∗tree, CTree ∗else_tree, CTree ∗next)

  Constructor.
- virtual void Print (ostream &os, CArray ∗data)

  Method to print element to stream.

Protected Attributes

- CTree ∗ **m_condition**
- CTree ∗ **m_tree**
- CTree ∗ **m_else**

4.8.1 Detailed Description

If/else condition node class.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 CIf::CIf (CTree ∗ *cond*, CTree ∗ *tree*, CTree ∗ *else_tree*, CTree ∗ *next*)

Constructor.

Parameters

in	*cond*	Condition expression block
in	*tree*	True statement block
in	*else_tree*	False statement block
in	*next*	Next template node

4.8.3 Member Function Documentation

4.8.3.1 void CIf::Print (ostream & *os*, CArray ∗ *data*) [virtual]

Method to print element to stream.

This function parse query and send answer

Parameters

in	*os*	Output stream
in	*data*	Data from View class, used in template

Reimplemented from CTree.

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

- lib/tree.hpp
- lib/tree.cpp

4.9 CLexer Class Reference

Lexer class.

`#include <lexer.hpp>`

Classes

- struct LexSymbol

Public Types

- enum **SymbolType** {
  HTML, **LPRINT**, **RPRINT**, **LSTAT**,
  RSTAT, **STRING**, **IDENT**, **NUMB**,
  PLUS, **MINUS**, **TIMES**, **DIVIDE**,
  MOD, **AND**, **OR**, **EQ**,
  NEQ, **LT**, **GT**, **LTE**,
  GTE, **LPAR**, **RPAR**, **ASSIGN**,
  LBRA, **RBRA**, **LSBRA**, **RSBRA**,
  COMMA, **SEMICOLON**, **RANGE**, **kwVAR**,
  kwINCLUDE, **kwIF**, **kwENDIF**, **kwELSE**,
  kwFOR, **kwENDFOR**, **kwIN**, **kwBLOCKPRINT**,
  kwMENUPRINT, **kwBLOCK**, **kwENDBLOCK**, **EOI**,
  ERR }

- typedef struct CLexer::LexSymbol **LexSymbol**

Public Member Functions

- CLexer (const char ∗fileName)

  Constructor.
- CLexer::LexSymbol getLexem ()

  Method to get next lexem.
- int getLineNum ()

  Get processing line number.
- char ∗ getError ()

  Get error report.
- char ∗ getFileName ()

  Get template file name.

Static Public Attributes

- static const char ∗ **SymbolTable** [45]

4.9.1 Detailed Description

Lexer class.

Read input file and return lexems.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 CLexer::CLexer (const char ∗ *fileName*)

Constructor.

Parameters

in	*fileName*	Template file path.

4.9.3 Member Function Documentation

4.9.3.1 char ∗ CLexer::getError ( )

Get error report.

Return values

char∗	Error report.

4.9.3.2 char ∗ CLexer::getFileName ( )

Get template file name.

Return values

*char**	Template name

4.9.3.3 CLexer::LexSymbol CLexer::getLexem ( )

Method to get next lexem.

Return values

LexSymbol	Lex Symbol

4.9.3.4 int CLexer::getLineNum ( )

Get processing line number.

Return values

int	Line number.

4.9.4 Member Data Documentation

4.9.4.1 const char ∗ CLexer::SymbolTable [static]

Initial value:

```
= {
    "text html", "LPRINT", "RPRINT", "LSTAT", "RSTAT", "string",
    "identifier", "number", "plus", "minus", "times", "divide", "modulo", "and", "or",
    "equal", "not equal", "<", ">", "<=", ">=", "(", ")", "=", "LBRA", "RBRA", "LSBRA", "RSBRA",
    ",", ";", "..",
    "key word 'variable'", "key word 'include'", "key word 'if'", "key word 'endif'", "key word 'else'",
    "key word 'for'", "key word 'endfor'", "key word 'in'",
    "key word 'blockprint'", "key word 'menuprint'", "key word 'block'", "key word 'endblock'",
    "end of file", "error"
}
```

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

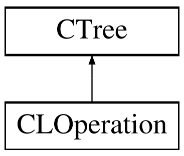
- lib/lexer.hpp
- lib/lexer.cpp

4.10 CLOperation Class Reference

Logic operations node class.

`#include <tree.hpp>`

Inheritance diagram for CLOperation:



Public Types

- enum **Operation** {
  EQ, **NEQ**, **LT**, **LTE**,
  GT, **GTE**, **AND**, **OR** }

Public Member Functions

- CLOperation (Operation o, CTree ∗op1, CTree ∗op2)

  Constructor.
- virtual CArray ∗ getArray (CArray ∗data)

  Return reference to the array with one dereference.
- virtual int isTrue (CArray ∗data)

  Method to determinate is element true/false, used in If/For condition.

Protected Attributes

- Operation **m_o**
- CArray **m_val**
- CTree ∗ **m_op1**
- CTree ∗ **m_op2**

4.10.1 Detailed Description

Logic operations node class.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 CLOperation::CLOperation (Operation *o*, CTree ∗ *op1*, CTree ∗ *op2*)

Constructor.

Parameters

in	*o*	Operation type
in	*op1*	First operand
in	*op2*	Second operand

4.10.3 Member Function Documentation

4.10.3.1 CArray ∗ CLOperation::getArray (CArray ∗ *data*) [virtual]

Return reference to the array with one dereference.

Parameters

in	*data*	Data from View

Return values

CArray∗	Array data

Reimplemented from CTree.

4.10.3.2 int CLOperation::isTrue (CArray * *data*) [virtual]

Method to determinate is element true/false, used in If/For condition.

Parameters

in	*data*	Data from View

Reimplemented from CTree.

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

- lib/tree.hpp
- lib/tree.cpp

4.11 CMenu Class Reference

Menu class.

`#include <menu.hpp>`

Classes

- struct TSubMenu

Public Member Functions

- void AddSubmenu (const char ∗name, const char ∗url, const bool isSet)

  Add submenu entry.

Protected Member Functions

- int is_so (const char ∗s) const

  Is shared object?
- void Print (ostream &os) const

  Print menu item to stream.

Protected Attributes

- TSubMenu ∗ **m_root**

Friends

- ostream & operator<< (ostream &os, const CMenu &x)

  Operator <<.

4.11.1 Detailed Description

Menu class.

Generate html menu from availavble modules in dir ./modules/∗.so

4.11.2 Member Function Documentation

4.11.2.1 void CMenu::AddSubmenu (const char ∗ *name,* const char ∗ *url,* const bool *isSet*)

Add submenu entry.

Show under active module

Parameters

`in`	*name*	Submenu name
`in`	*url*	Submenu url
`in`	*isSet*	Is this item active now?

4.11.2.2 int CMenu::is_so (const char ∗ *s*) const [protected]

Is shared object?

This method say if file is ∗.so (shared object file)

Parameters

`in`	*s*	File name

Return values

0	Successfully.
1	With error.

4.11.2.3 void CMenu::Print (ostream & *os*) const [protected]

Print menu item to stream.

This function parse query and send answer

Parameters

`in`	*os*	Output stream

4.11.3 Friends And Related Function Documentation

4.11.3.1 ostream& operator<< (ostream & *os*, const CMenu & *x*) [friend]

Operator <<.

Print menu to stream

Parameters

`in`	*ostream*	Output stream
`in`	*x*	Menu

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

- lib/menu.hpp
- lib/menu.cpp

4.12 CParser Class Reference

Public Member Functions

- CParser (const char ∗file, const CArray ∗data)

  Constructor.
- CTree ∗ execParser ()

  Method start parsing.

- char ∗ getError ()

  Method to get error, that occurred during parsing.

4.12.1 Constructor & Destructor Documentation

4.12.1.1 CParser::CParser (const char ∗ *file*, const CArray ∗ *data*)

Constructor.

Parameters

in	*file*	Template file path
in	*data*	Data array from View

4.12.2 Member Function Documentation

4.12.2.1 CTree ∗ CParser::execParser ( )

Method start parsing.

Return values

CTree	Syntax tree

4.12.2.2 char ∗ CParser::getError ( )

Method to get error, that occurred during parsing.

Return values

char∗	Error text

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

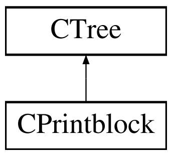
- lib/parser.hpp
- lib/parser.cpp

4.13 CPrintblock Class Reference

Class PrintBlock to render separated block in this node.

`#include <tree.hpp>`

Inheritance diagram for CPrintblock:



Public Member Functions

- CPrintblock (char ∗block, CTree ∗next)

  Constructor.

- virtual void Print (ostream &os, CArray ∗data)

  Method to print element to stream.

Protected Attributes

- char ∗ **m_block**

4.13.1 Detailed Description

Class PrintBlock to render separated block in this node.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 CPrintblock::CPrintblock (char ∗ *block*, CTree ∗ *next*)

Constructor.

Parameters

in	*block*	Block identifier.
in	*next*	Next template node.

4.13.3 Member Function Documentation

4.13.3.1 void CPrintblock::Print (ostream & *os*, CArray ∗ *data*) [virtual]

Method to print element to stream.

This function parse query and send answer

Parameters

in	*os*	Output stream
in	*data*	Data from View class, used in template

Reimplemented from CTree.

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

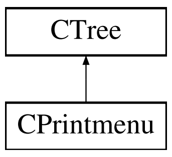
- lib/tree.hpp
- lib/tree.cpp

4.14 CPrintmenu Class Reference

Class PrintMenu to print menu (Cmenu) in this node.

`#include <tree.hpp>`

Inheritance diagram for CPrintmenu:



Public Member Functions

- CPrintmenu (CTree ∗next)

  Constructor.
- virtual void Print (ostream &os, CArray ∗data)

  Method to print element to stream.

Additional Inherited Members

4.14.1 Detailed Description

Class PrintMenu to print menu (Cmenu) in this node.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 CPrintmenu::CPrintmenu (CTree ∗ *next*)

Constructor.

Parameters

`in`	*next*	Next template identifier.

4.14.3 Member Function Documentation

4.14.3.1 void CPrintmenu::Print (ostream & *os*, CArray ∗ *data*) `[virtual]`

Method to print element to stream.

This function parse query and send answer

Parameters

`in`	*os*	Output stream
`in`	*data*	Data from View class, used in template

Reimplemented from CTree.

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

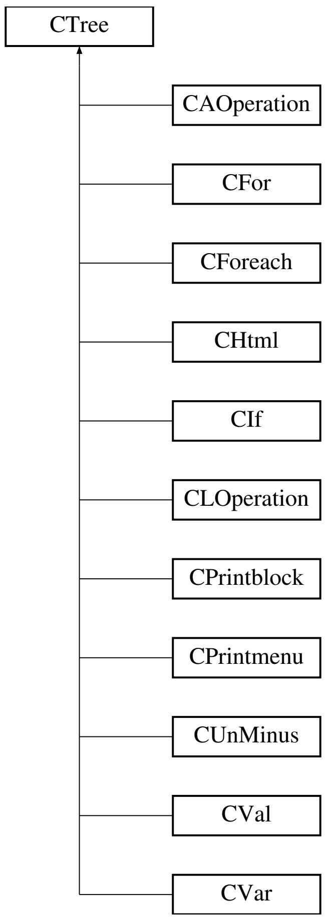
- lib/tree.hpp
- lib/tree.cpp

4.15 CTree Class Reference

Main virtual class.

`#include <tree.hpp>`

Inheritance diagram for CTree:



Public Member Functions

- virtual void Print (ostream &os, CArray ∗data)

  Method to print element to stream.
- virtual ∼CTree ()

  Destructor.
- virtual int isTrue (CArray ∗data)

  Method to determinate is element true/false, used in If/For condition.
- virtual CArray ∗ getArray (CArray ∗data)

  Return reference to the array with one dereference.
- virtual CTree ∗ append (CTree ∗x)

  Append new node to end.

Protected Attributes

- CTree ∗ **m_next**

4.15.1 Detailed Description

Main virtual class.

4.15.2 Member Function Documentation

4.15.2.1 CTree ∗ CTree::append (CTree ∗ *x*) `[virtual]`

Append new node to end.

Parameters

`in`	*x*	New node

Return values

CTree∗	Return pointer to ifself

4.15.2.2 virtual **CArray**∗ CTree::getArray (**CArray** ∗ *data*) `[inline]`, `[virtual]`

Return reference to the array with one dereference.

Parameters

`in`	*data*	Data from View

Return values

CArray∗	Array data

Reimplemented in CUnMinus, CLOperation, CAOperation, CVal, and CVar.

4.15.2.3 virtual int CTree::isTrue (**CArray** ∗ *data*) `[inline]`, `[virtual]`

Method to determinate is element true/false, used in If/For condition.

Parameters

`in`	*data*	Data from View

Reimplemented in CUnMinus, CLOperation, CAOperation, CVal, and CVar.

4.15.2.4 virtual void CTree::Print (ostream & *os*, **CArray** ∗ *data*) `[inline]`, `[virtual]`

Method to print element to stream.

This function parse query and send answer

Parameters

`in`	*os*	Output stream
`in`	*data*	Data from View class, used in template

Reimplemented in CIf, CFor, CForeach, CPrintmenu, CPrintblock, CVar, and CHtml.

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

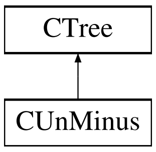
- lib/tree.hpp
- lib/tree.cpp

4.16 CUnMinus Class Reference

Unary minus node class.

`#include <tree.hpp>`

Inheritance diagram for CUnMinus:



Public Member Functions

- CUnMinus (CTree ∗tree)

  Constructor.
- virtual CArray ∗ getArray (CArray ∗data)

  Return reference to the array with one dereference.
- virtual int isTrue (CArray ∗data)

  Method to determinate is element true/false, used in If/For condition.

Protected Attributes

- CArray **m_val**
- CTree ∗ **m_tree**

4.16.1 Detailed Description

Unary minus node class.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 CUnMinus::CUnMinus (CTree ∗ *tree*)

Constructor.

Parameters

in	*tree*	Expression.

4.16.3 Member Function Documentation

4.16.3.1 CArray ∗ CUnMinus::getArray (CArray ∗ *data*) [virtual]

Return reference to the array with one dereference.

Parameters

in	*data*	Data from View

Return values

CArray∗	Array data

Reimplemented from CTree.

4.16.3.2 int CUnMinus::isTrue (CArray ∗ *data*) [virtual]

Method to determinate is element true/false, used in If/For condition.

Parameters

`in`	*data*	Data from View

Reimplemented from CTree.

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

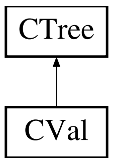
- lib/tree.hpp
- lib/tree.cpp

4.17 CVal Class Reference

Value node class, integer or string.

```
#include <tree.hpp>
```

Inheritance diagram for CVal:



Public Member Functions

- CVal (const int x)

    Constructor.
- **CVal** (const char ∗x)
- virtual CArray ∗ getArray (CArray ∗data)

    Return reference to the array with one dereference.
- virtual int isTrue (CArray ∗data)

    Method to determinate is element true/false, used in If/For condition.

Protected Attributes

- CArray ∗ **m_val**

4.17.1 Detailed Description

Value node class, integer or string.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 CVal::CVal (const int *x*)

Constructor.

Parameters

`in`	*x*	int Numeric value
`in`	*x*	char∗ String value

4.17.3 Member Function Documentation

4.17.3.1 CArray ∗ CVal::getArray (CArray ∗ *data*) `[virtual]`

Return reference to the array with one dereference.

Parameters

`in`	*data*	Data from View

Return values

CArray∗	Array data

Reimplemented from CTree.

4.17.3.2 int CVal::isTrue (CArray ∗ *data*) `[virtual]`

Method to determinate is element true/false, used in If/For condition.

Parameters

`in`	*data*	Data from View

Reimplemented from CTree.

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

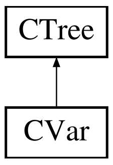
- lib/tree.hpp
- lib/tree.cpp

4.18 CVar Class Reference

Variable node class, print one value from data Array from View, indexed by string or variable.

`#include <tree.hpp>`

Inheritance diagram for CVar:



Public Types

- enum **IdentType** { **IDENT**, **STRING** }

Public Member Functions

- CVar (char ∗var, IdentType type, CVar ∗array, CTree ∗next)

Constructor.
- virtual CVar ∗ **appendVar** (CVar ∗x)
- virtual void Print (ostream &os, CArray ∗data)

  Method to print element to stream.
- virtual int isTrue (CArray ∗data)

  Method to determinate is element true/false, used in If/For condition.
- virtual CArray ∗ getArray (CArray ∗data)

  Return reference to the array with one dereference.

Protected Attributes

- CVar ∗ **m_array**
- char ∗ **m_var**
- IdentType **m_type**

4.18.1 Detailed Description

Variable node class, print one value from data Array from View, indexed by string or variable.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 CVar::CVar (char ∗ *var,* IdentType *type,* CVar ∗ *array,* CTree ∗ *next*)

Constructor.

Parameters

in	*var*	Index identifier
in	*type*	Index identifier is string or varible identifier?
in	*array*	Next index or NULL if array end.
in	*next*	Next template node.

4.18.3 Member Function Documentation

4.18.3.1 CArray ∗ CVar::getArray (CArray ∗ *data*) [virtual]

Return reference to the array with one dereference.

Parameters

in	*data*	Data from View

Return values

CArray∗	Array data

Reimplemented from CTree.

4.18.3.2 int CVar::isTrue (CArray ∗ *data*) [virtual]

Method to determinate is element true/false, used in If/For condition.

Parameters

in	*data*	Data from View

Reimplemented from CTree.

4.18.3.3 void CVar::Print (ostream & *os,* CArray $*$ *data*) [virtual]

Method to print element to stream.

This function parse query and send answer

Parameters

in	*os*	Output stream
in	*data*	Data from View class, used in template

Reimplemented from CTree.

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

- lib/tree.hpp
- lib/tree.cpp

4.19 CView Class Reference

View class.

`#include <view.hpp>`

Public Member Functions

- CView (const char $*$path, CArray $*$data)

    Constructor.
- $\sim$CView ()

    Destructor.

Protected Attributes

- CArray $*$ **m_data**
- char $*$ **m_path**

Friends

- ostream & operator<< (ostream &os, const CView &x)

    Operator <<.

4.19.1 Detailed Description

View class.

Class render html code to client, use lexer, parser and syntax tree to generate html code.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 CView::CView (const char ∗ *path*, CArray ∗ *data*)

Constructor.

Parameters

in	*path*	Path to template file
in	*data*	Generated data used in template

4.19.3 Friends And Related Function Documentation

4.19.3.1 ostream& operator<< (ostream & *os,* const CView & *x*) [friend]

Operator <<.

Parameters

in	*os*	Output stream
in	*x*	View

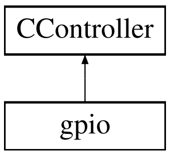
The documentation for this class was generated from the following files:

- lib/view.hpp
- lib/view.cpp

4.20 gpio Class Reference

Gpio module class.

Inheritance diagram for gpio:



Public Member Functions

- virtual CView ∗ **Run** ()

Additional Inherited Members

4.20.1 Detailed Description

Gpio module class.

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

- modules/gpio.cpp

4.21 CLexer::LexSymbol Struct Reference

Public Attributes

- SymbolType **type**
- char ∗ **ident**

- int **number**

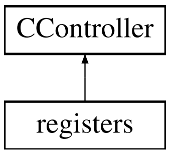
The documentation for this struct was generated from the following file:

- lib/lexer.hpp

4.22 registers Class Reference

Register module class.

Inheritance diagram for registers:



Public Member Functions

- virtual CView ∗ **Run** ()

Additional Inherited Members

4.22.1 Detailed Description

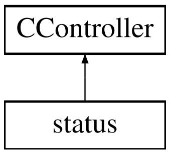
Register module class.

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

- modules/registers.cpp

4.23 status Class Reference

Inheritance diagram for status:



Public Member Functions

- virtual CView ∗ **Run** ()

Additional Inherited Members

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

- modules/status.cpp

4.24 TBlock Struct Reference

Public Member Functions

- **TBlock** (char ∗id, CTree ∗tree)

Public Attributes

- char ∗ **m_id**
- CTree ∗ **m_tree**
- TBlock ∗ **m_next**

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

- lib/tree.cpp

4.25 CArray::Tnode Struct Reference

Public Member Functions

- **Tnode** (const char ∗x)

Public Attributes

- Tnode ∗ **next**
- CArray ∗ **arr**
- char ∗ **id**

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

- lib/array.hpp
- lib/array.cpp

4.26 CMenu::TSubMenu Struct Reference

Public Member Functions

- **TSubMenu** (const char ∗name, const char ∗url, const bool isSet, TSubMenu ∗next)

Public Attributes

- char ∗ **m_name**
- char ∗ **m_url**
- bool **m_isset**
- TSubMenu ∗ **m_next**

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

- lib/menu.hpp
- lib/menu.cpp

4.27 TVar Struct Reference

Public Member Functions

- **TVar** (char ∗id, CArray ∗arr)

Public Attributes

- char ∗ **m_id**
- CArray ∗ **m_arr**
- TVar ∗ **m_next**

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

- lib/tree.cpp

Chapter 5

File Documentation

5.1 lib/array.hpp File Reference

infinite associative array

```
#include <iostream>
```

Classes

- class CArray
- struct CArray::Tnode

5.1.1 Detailed Description

infinite associative array Bohdan Vico (vicobohd@fit.cvut.cz)

Date

November, 2014

5.2 lib/header.hpp File Reference

Html header.

```
#include <iostream>
```

Classes

- class CHeader

5.2.1 Detailed Description

Html header. Bohdan Vico (vicobohd@fit.cvut.cz)

Date

November, 2014

5.3 lib/lexer.cpp File Reference

lexer

```
#include "lexer.hpp"
```

Variables

- struct {

5.3.1 Detailed Description

lexer Bohdan Vico (vicobohd@fit.cvut.cz)

Date

November, 2014

5.3.2 Variable Documentation

5.3.2.1 const { ... } keyWordTable[ ]

Initial value:

```
= {
   {"if", CLexer::kwIF},
   {"endif", CLexer::kwENDIF},
   {"else", CLexer::kwELSE},
   {"for", CLexer::kwFOR},
   {"in", CLexer::kwIN},
   {"endfor", CLexer::kwENDFOR},
   {"include", CLexer::kwINCLUDE},
   {"blockprint", CLexer::kwBLOCKPRINT},
   {"menuprint", CLexer::kwMENUPRINT},
   {"block", CLexer::kwBLOCK},
   {"endblock", CLexer::kwENDBLOCK},
   {NULL, ( CLexer::SymbolType ) 0}
}
```

5.4 lib/lexer.hpp File Reference

lexer

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Classes

- class CLexer

  Lexer class.
- struct CLexer::LexSymbol

5.4.1 Detailed Description

lexer Bohdan Vico (vicobohd@fit.cvut.cz)

Date

November, 2014

5.5 lib/menu.cpp File Reference

menu class

```
#include "menu.hpp"
#include <stdlib.h>
#include <sys/types.h>
#include <dirent.h>
#include <string.h>
```

Functions

- ostream & operator<< (ostream &os, const CMenu &x)

Variables

- char ∗ **g_module_name**

5.5.1 Detailed Description

menu class Bohdan Vico (vicobohd@fit.cvut.cz)

Date

November, 2014

5.5.2 Function Documentation

5.5.2.1 ostream& operator<< (ostream & *os*, const **CMenu** & *x*)

Print menu to stream

Parameters

in	*ostream*	Output stream
in	*x*	Menu

5.6 lib/menu.hpp File Reference

menu class

```
#include <iostream>
```

Classes

- class CMenu

  Menu class.
- struct CMenu::TSubMenu

5.6.1 Detailed Description

menu class Bohdan Vico (`vicobohd@fit.cvut.cz`)

Date

November, 2014

5.7 lib/parser.cpp File Reference

template parser, return syntax tree

```
#include "parser.hpp"
#include <iostream>
#include <stdlib.h>
#include <stdio.h>
```

Variables

- CLexer::LexSymbol **g_s**

5.7.1 Detailed Description

template parser, return syntax tree Bohdan Vico (`vicobohd@fit.cvut.cz`)

Date

November, 2014

5.8 lib/parser.hpp File Reference

template parser, return syntax tree

```
#include "lexer.hpp"
#include "array.hpp"
#include "tree.hpp"
```

Classes

- class CParser

5.8.1 Detailed Description

template parser, return syntax tree Bohdan Vico (`vicobohd@fit.cvut.cz`)

Date

November, 2014

5.9 lib/tree.hpp File Reference

class represents syntactic structure of template file

```
#include "array.hpp"
```

Classes

- class CTree

  Main virtual class.
- class CHtml

  Html node class.
- class CVar

  Variable node class, print one value from data Array from View, indexed by string or variable.
- class CPrintblock

  Class PrintBlock to render separated block in this node.
- class CPrintmenu

  Class PrintMenu to print menu (Cmenu) in this node.
- class CForeach

  Foreach node class, passes all elements in array (CArray)
- class CFor

  For node class, iterated loop.
- class CVal

  Value node class, integer or string.
- class CIf

  If/else condition node class.
- class CAOperation

  Arithmetic operations node class.
- class CLOperation

  Logic operations node class.
- class CUnMinus

  Unary minus node class.

Functions

- void addVar (char ∗id, CArray ∗arr)

  Function to add template variable to variable-table.
- void delVar (char ∗id)

  Delete variable from variable-table.
- CArray ∗ getVar (const char ∗id)

  Get variable content.
- void addBlock (char ∗id, CTree ∗tree)

Add template block.
- CTree ∗ getBlock (const char ∗id)

  Get block root node.
- void delBlocks ()

  Delete blocks.

5.9.1 Detailed Description

class represents syntactic structure of template file Bohdan Vico (vicobohd@fit.cvut.cz)

Date

November, 2014

5.9.2 Function Documentation

5.9.2.1 void addBlock (char ∗ *id*, CTree ∗ *tree*)

Add template block.

Template block is separate syntax tree

Parameters

in	*id*	Block identifier
in	*tree*	Block tree structure

5.9.2.2 void addVar (char ∗ *id*, CArray ∗ *arr*)

Function to add template variable to variable-table.

Parameters

in	*id*	Variable identifier (variable name)
in	*arr*	Variable data (content)

5.9.2.3 void delBlocks ( )

Delete blocks.

Delete all insert template blocks

5.9.2.4 void delVar (char ∗ *id*)

Delete variable from variable-table.

Parameters

in	*id*	Variable identifier

5.9.2.5 CTree∗ getBlock (const char ∗ *id*)

Get block root node.

Parameters

in	*id*	Block identifier

Return values

CTree∗	Return root node pointer

5.9.2.6 CArray∗ getVar (const char ∗ *id*)

Get variable content.

Parameters

in	*id*	Variable identifier

Return values

CArray∗	Pointer to variable data

5.10 lib/view.cpp File Reference

view class

```
#include "view.hpp"
#include "tree.hpp"
#include "parser.hpp"
```

Functions

- ostream & operator<< (ostream &os, const CView &x)

5.10.1 Detailed Description

view class Bohdan Vico (vicobohd@fit.cvut.cz)

Date

November, 2014

5.10.2 Function Documentation

5.10.2.1 ostream& operator<< (ostream & *os*, const CView & *x*)

Parameters

in	*os*	Output stream
in	*x*	View

5.11 lib/view.hpp File Reference

view class

```
#include "array.hpp"
#include <iostream>
```

Classes

- class CView
  View class.

5.11.1 Detailed Description

view class Bohdan Vico (vicobohd@fit.cvut.cz)

Date

November, 2014

5.12 modules/gpio.c File Reference

gpio library

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <unistd.h>
```

Macros

- #define **BCM2835_GPIO_BASE** 0x20200000
- #define **GPIO_IN** 0
- #define **GPIO_OUT** 1
- #define **GPIO_ALT0** 2
- #define **GPIO_ALT1** 3
- #define **GPIO_ALT2** 4
- #define **GPIO_ALT3** 5
- #define **GPIO_ALT4** 6
- #define **GPIO_PULL_OFF** 0
- #define **GPIO_PULL_DOWN** 1
- #define **GPIO_PULL_UP** 2
- #define **PAGE_SIZE** (4∗1024)
- #define **BLOCK_SIZE** (4∗1024)
- #define **SERIAL_DEV** "/dev/ttyAMA0"

Functions

- int setup_io ()
  Set up a memory regions to access GPIO.
- int gpio_init (int pin, int function)
  Initialize GPIO pins.
- int gpio_func (int pin)

Get GPIO function.
- int gpio_write (int pin, int value)

  GPIO write.
- int gpio_read (int pin)

  GPIO read.
- int gpio_pull (int pin, int pull)

  Set GPIO pull up/down resistor.
- int get_serial_link_fd ()

  Get serial link file descriptor.

Variables

- volatile unsigned ∗ **g_gpio**

5.12.1 Detailed Description

gpio library Bohdan Vico (vicobohd@fit.cvut.cz)

Date

November, 2014

5.12.2 Function Documentation

5.12.2.1 int get_serial_link_fd ( )

Get serial link file descriptor.

Return values

-1	Fail.
int	File descriptor.

5.12.2.2 int gpio_func (int *pin*)

Get GPIO function.

Parameters

in	*pin*	GPIO pin number

Return values

-1	Fail.
int	GPIO_IN | GPIO_OUT | GPIO_ALT0 | GPIO_ALT1 | GPIO_ALT2 | GPIO_ALT3 | GPIO_ALT4 On success.

5.12.2.3 int gpio_init (int *pin,* int *function*)

Initialize GPIO pins.

Parameters

in	*pin*	GPIO pin number
in	*function*	GPIO function, GPIO_IN | GPIO_OUT | GPIO_ALT0 | GPIO_ALT1 | GPIO_A-LT2 | GPIO_ALT3 | GPIO_ALT4

Return values

0	Success.
1	Fail.

5.12.2.4 int gpio_pull (int *pin,* int *pull*)

Set GPIO pull up/down resistor.

Parameters

in	*pin*	GPIO pin number
in	*pull*	Pull up/down, GPIO_PULL_UP | GPIO_PULL_DOWN | GPIO_PULL_OFF

Return values

1	Fail.
0	Success.

5.12.2.5 int gpio_read (int *pin*)

GPIO read.

Parameters

in	*pin*	GPIO pin number
in	*value*	Value 1/0

Return values

-1	Fail.
1/0	Success.

5.12.2.6 int gpio_write (int *pin,* int *value*)

GPIO write.

Parameters

in	*pin*	GPIO pin number
in	*value*	Value 1/0

Return values

0	Success.
1	Fail.

5.12.2.7 int setup_io ( )

Set up a memory regions to access GPIO.

Parameters

in	*first*	Pointer to save first for value
in	*last*	Pointer to save last for value

Return values

0	Success.
1	Fail.

5.13 modules/gpio.cpp File Reference

gpio module

```
#include "../lib/controller.hpp"
#include "gpio.c"
```

Classes

- class gpio

  Gpio module class.

Functions

- CController ∗ **create** ()
- void **destroy** (CController ∗p)

5.13.1 Detailed Description

gpio module Bohdan Vico (vicobohd@fit.cvut.cz)

Date

November, 2014

5.14 modules/registers.cpp File Reference

registers overview module

```
#include "../lib/controller.hpp"
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#include <stdio.h>
#include <math.h>
```

Classes

- class registers

  Register module class.

Macros

- #define **SOCKET_NAME** "@/tmp/regd"

Functions

- CController ∗ **create** ()
- void **destroy** (CController ∗p)

5.14.1 Detailed Description

registers overview module Bohdan Vico (`vicobohd@fit.cvut.cz`)

Date

November, 2014