

pMath

Generated by Doxygen 1.5.9

Sun Aug 21 19:54:18 2011

## Contents

<b>1</b>	<b>The pMath Computer Algebra System Library</b>	<b>1</b>
<b>2</b>	<b>Todo List</b>	<b>1</b>
<b>3</b>	<b>Deprecated List</b>	<b>2</b>
<b>4</b>	<b>Module Index</b>	<b>2</b>
4.1	Modules . . . . .	2
<b>5</b>	<b>Namespace Index</b>	<b>3</b>
5.1	Namespace List . . . . .	3
<b>6</b>	<b>Data Structure Index</b>	<b>3</b>
6.1	Class Hierarchy . . . . .	3
<b>7</b>	<b>Data Structure Index</b>	<b>5</b>
7.1	Data Structures . . . . .	5
<b>8</b>	<b>File Index</b>	<b>6</b>
8.1	File List . . . . .	6
<b>9</b>	<b>Module Documentation</b>	<b>8</b>
9.1	Custom Objects . . . . .	8
9.1.1	Detailed Description . . . . .	9
9.1.2	Typedef Documentation . . . . .	9
9.1.3	Function Documentation . . . . .	9
9.2	Expressions . . . . .	10
9.2.1	Detailed Description . . . . .	12
9.2.2	Typedef Documentation . . . . .	12
9.2.3	Function Documentation . . . . .	12
9.3	Numbers . . . . .	16
9.3.1	Detailed Description . . . . .	20
9.3.2	Define Documentation . . . . .	21
9.3.3	Typedef Documentation . . . . .	26
9.3.4	Enumeration Type Documentation . . . . .	26
9.3.5	Function Documentation . . . . .	27

9.4	Objects - the Base of pMath	36
9.4.1	Detailed Description	39
9.4.2	Define Documentation	40
9.4.3	Typedef Documentation	42
9.4.4	Enumeration Type Documentation	44
9.4.5	Function Documentation	45
9.4.6	Variable Documentation	47
9.5	Strings	48
9.5.1	Detailed Description	50
9.5.2	Define Documentation	50
9.5.3	Typedef Documentation	51
9.5.4	Function Documentation	51
9.6	Symbols	57
9.6.1	Detailed Description	59
9.6.2	Typedef Documentation	59
9.6.3	Enumeration Type Documentation	61
9.6.4	Function Documentation	62
9.7	C++ Binding	66
9.7.1	Detailed Description	67
9.8	Parsing Code	68
9.8.1	Detailed Description	72
9.8.2	Example	72
9.8.3	Define Documentation	73
9.8.4	Typedef Documentation	75
9.8.5	Enumeration Type Documentation	76
9.8.6	Function Documentation	78
9.9	General Purpose Types	86
9.9.1	Detailed Description	86
9.9.2	Define Documentation	87
9.9.3	Typedef Documentation	87
9.10	Atomic Operations	87
9.10.1	Detailed Description	89
9.10.2	Nice to read:	89
9.10.3	Define Documentation	89

9.10.4	Function Documentation . . . . .	90
9.11	Thread Messaging . . . . .	93
9.11.1	Detailed Description . . . . .	95
9.11.2	Typedef Documentation . . . . .	95
9.11.3	Function Documentation . . . . .	95
9.12	Multithreading with pMath . . . . .	99
9.12.1	Detailed Description . . . . .	100
9.12.2	Synchronization . . . . .	100
9.12.3	Typedef Documentation . . . . .	101
9.12.4	Function Documentation . . . . .	101
9.13	Debugging . . . . .	106
9.13.1	Detailed Description . . . . .	106
9.13.2	Function Documentation . . . . .	106
9.14	File API . . . . .	107
9.14.1	Detailed Description . . . . .	109
9.14.2	Enumeration Type Documentation . . . . .	109
9.14.3	Function Documentation . . . . .	110
9.15	Hashtables . . . . .	117
9.15.1	Detailed Description . . . . .	119
9.15.2	Typedef Documentation . . . . .	119
9.15.3	Function Documentation . . . . .	121
9.16	Object Utility Functions . . . . .	124
9.16.1	Detailed Description . . . . .	125
9.16.2	Typedef Documentation . . . . .	125
9.16.3	Function Documentation . . . . .	125
9.17	Memory Management . . . . .	130
9.17.1	Detailed Description . . . . .	131
9.17.2	Function Documentation . . . . .	131
9.18	Messages . . . . .	133
9.18.1	Detailed Description . . . . .	133
9.18.2	Function Documentation . . . . .	134
9.19	Threadsafe Stacks . . . . .	135
9.19.1	Detailed Description . . . . .	136
9.19.2	Typedef Documentation . . . . .	136

9.19.3	Function Documentation	137
9.20	Version Information	138
9.20.1	Function Documentation	138
9.21	Front-ends	138
9.21.1	Detailed Description	139
9.21.2	Function Documentation	139
<b>10</b>	<b>Namespace Documentation</b>	<b>141</b>
10.1	pmath Namespace Reference	141
10.1.1	Detailed Description	144
10.1.2	Function Documentation	144
<b>11</b>	<b>Data Structure Documentation</b>	<b>151</b>
11.1	BinaryFile Class Reference	151
11.1.1	Detailed Description	152
11.1.2	Constructor & Destructor Documentation	152
11.1.3	Member Function Documentation	153
11.2	BinaryUserStream Class Reference	153
11.2.1	Detailed Description	154
11.2.2	Member Function Documentation	154
11.3	Expr Class Reference	155
11.3.1	Detailed Description	158
11.3.2	Constructor & Destructor Documentation	158
11.3.3	Member Function Documentation	159
11.3.4	Field Documentation	165
11.4	File Class Reference	165
11.4.1	Detailed Description	165
11.4.2	Constructor & Destructor Documentation	166
11.4.3	Member Function Documentation	166
11.5	Gather Class Reference	167
11.5.1	Detailed Description	168
11.5.2	Constructor & Destructor Documentation	168
11.5.3	Member Function Documentation	168
11.5.4	Field Documentation	169
11.6	pmath_atomic2_t Struct Reference	169

11.6.1 Detailed Description . . . . .	169
11.7 <code>pmath_atomic_t</code> Struct Reference . . . . .	169
11.7.1 Detailed Description . . . . .	169
11.8 <code>pmath_binary_file_api_t</code> Class Reference . . . . .	170
11.8.1 Detailed Description . . . . .	170
11.8.2 Field Documentation . . . . .	170
11.9 <code>pmath_cstr_writer_info_t</code> Struct Reference . . . . .	171
11.9.1 Detailed Description . . . . .	171
11.9.2 Field Documentation . . . . .	171
11.10 <code>pmath_custom_t</code> Class Reference . . . . .	172
11.10.1 Detailed Description . . . . .	172
11.11 <code>pmath_expr_t</code> Class Reference . . . . .	172
11.11.1 Detailed Description . . . . .	174
11.11.2 Friends And Related Function Documentation . . . . .	174
11.12 <code>pmath_float_t</code> Class Reference . . . . .	175
11.12.1 Detailed Description . . . . .	175
11.13 <code>pmath_hashtable_t</code> Class Reference . . . . .	175
11.13.1 Detailed Description . . . . .	176
11.14 <code>pmath_ht_class_t</code> Class Reference . . . . .	176
11.14.1 Detailed Description . . . . .	177
11.14.2 Field Documentation . . . . .	177
11.15 <code>pmath_integer_t</code> Class Reference . . . . .	177
11.15.1 Detailed Description . . . . .	179
11.16 <code>pmath_messages_t</code> Class Reference . . . . .	179
11.16.1 Detailed Description . . . . .	180
11.17 <code>pmath_number_t</code> Class Reference . . . . .	180
11.17.1 Detailed Description . . . . .	181
11.18 <code>pmath_quotient_t</code> Class Reference . . . . .	181
11.18.1 Detailed Description . . . . .	181
11.19 <code>pmath_rational_t</code> Class Reference . . . . .	181
11.19.1 Detailed Description . . . . .	182
11.20 <code>pmath_span_array_t</code> Class Reference . . . . .	182
11.20.1 Detailed Description . . . . .	183
11.21 <code>pmath_span_t</code> Class Reference . . . . .	183

11.21.1 Detailed Description . . . . .	184
11.22pmath_stack_t Class Reference . . . . .	184
11.22.1 Detailed Description . . . . .	184
11.23pmath_string_t Class Reference . . . . .	184
11.23.1 Detailed Description . . . . .	186
11.24pmath_symbol_t Class Reference . . . . .	186
11.24.1 Detailed Description . . . . .	187
11.24.2 Friends And Related Function Documentation . . . . .	187
11.25pmath_t Class Reference . . . . .	188
11.25.1 Detailed Description . . . . .	189
11.25.2 Member Function Documentation . . . . .	190
11.25.3 Friends And Related Function Documentation . . . . .	191
11.25.4 Field Documentation . . . . .	191
11.26pmath_text_file_api_t Class Reference . . . . .	192
11.26.1 Detailed Description . . . . .	192
11.26.2 Field Documentation . . . . .	192
11.27pmath_thread_t Class Reference . . . . .	193
11.27.1 Detailed Description . . . . .	194
11.27.2 Friends And Related Function Documentation . . . . .	194
11.28pmath_threadlock_t Class Reference . . . . .	195
11.28.1 Detailed Description . . . . .	195
11.29pmath_write_ex_t Struct Reference . . . . .	195
11.29.1 Detailed Description . . . . .	196
11.29.2 Field Documentation . . . . .	196
11.30ReadableBinaryFile Class Reference . . . . .	197
11.30.1 Detailed Description . . . . .	197
11.30.2 Constructor & Destructor Documentation . . . . .	198
11.30.3 Member Function Documentation . . . . .	198
11.31ReadableTextFile Class Reference . . . . .	198
11.31.1 Detailed Description . . . . .	199
11.31.2 Constructor & Destructor Documentation . . . . .	199
11.31.3 Member Function Documentation . . . . .	200
11.32String Class Reference . . . . .	200
11.32.1 Detailed Description . . . . .	202

11.32.2 Constructor & Destructor Documentation . . . . .	202
11.32.3 Member Function Documentation . . . . .	203
11.33 TextFile Class Reference . . . . .	206
11.33.1 Detailed Description . . . . .	207
11.33.2 Constructor & Destructor Documentation . . . . .	207
11.33.3 Member Function Documentation . . . . .	207
11.34 TextUserStream Class Reference . . . . .	208
11.34.1 Detailed Description . . . . .	208
11.34.2 Member Function Documentation . . . . .	209
11.35 UserStream Class Reference . . . . .	210
11.35.1 Detailed Description . . . . .	210
11.35.2 Constructor & Destructor Documentation . . . . .	211
11.35.3 Member Function Documentation . . . . .	211
11.36 WriteableBinaryFile Class Reference . . . . .	212
11.36.1 Detailed Description . . . . .	213
11.36.2 Constructor & Destructor Documentation . . . . .	213
11.36.3 Member Function Documentation . . . . .	213
11.37 WriteableTextFile Class Reference . . . . .	214
11.37.1 Detailed Description . . . . .	214
11.37.2 Constructor & Destructor Documentation . . . . .	214
11.37.3 Member Function Documentation . . . . .	215
<b>12 File Documentation</b>	<b>215</b>
12.1 pmath-config.h File Reference . . . . .	215
12.1.1 Define Documentation . . . . .	216
12.2 pmath-core/custom.h File Reference . . . . .	218
12.3 pmath-core/expressions.h File Reference . . . . .	218
12.4 pmath-core/numbers.h File Reference . . . . .	218
12.5 pmath-core/objects-inline.h File Reference . . . . .	219
12.5.1 Define Documentation . . . . .	220
12.5.2 Function Documentation . . . . .	223
12.6 pmath-core/objects.h File Reference . . . . .	223
12.7 pmath-core/strings.h File Reference . . . . .	226
12.8 pmath-core/symbols.h File Reference . . . . .	226



12.9 pmath-cpp.h File Reference . . . . .	227
12.10 pmath-language/charnames.h File Reference . . . . .	230
12.11 pmath-language/scanner.h File Reference . . . . .	231
12.12 pmath-language/tokens.h File Reference . . . . .	231
12.13 pmath-types.h File Reference . . . . .	235
12.14 pmath-util/approximate.h File Reference . . . . .	235
12.15 pmath-util/compression.h File Reference . . . . .	236
12.16 pmath-util/concurrency/atomic.h File Reference . . . . .	236
12.16.1 Define Documentation . . . . .	236
12.17 pmath-util/concurrency/atomic/non-atomic.h File Reference . . . . .	237
12.18 pmath-util/concurrency/threadlocks.h File Reference . . . . .	238
12.19 pmath-util/concurrency/threadmsg.h File Reference . . . . .	238
12.20 pmath-util/concurrency/threadpool.h File Reference . . . . .	238
12.20.1 Typedef Documentation . . . . .	239
12.20.2 Function Documentation . . . . .	239
12.21 pmath-util/concurrency/threads.h File Reference . . . . .	240
12.22 pmath-util/debug.h File Reference . . . . .	240
12.23 pmath-util/dlmalloc.h File Reference . . . . .	241
12.24 pmath-util/emit-and-gather.h File Reference . . . . .	241
12.25 pmath-util/evaluation.h File Reference . . . . .	241
12.26 pmath-util/files.h File Reference . . . . .	241
12.27 pmath-util/hashtables.h File Reference . . . . .	243
12.28 pmath-util/helpers.h File Reference . . . . .	243
12.29 pmath-util/line-writer.h File Reference . . . . .	244
12.30 pmath-util/memory.h File Reference . . . . .	244
12.31 pmath-util/messages.h File Reference . . . . .	245
12.32 pmath-util/mixed-file.h File Reference . . . . .	245
12.33 pmath-util/serialize.h File Reference . . . . .	245
12.33.1 Enumeration Type Documentation . . . . .	246
12.33.2 Function Documentation . . . . .	246
12.34 pmath-util/stacks.h File Reference . . . . .	247
12.35 pmath-util/strtod.h File Reference . . . . .	247
12.35.1 Function Documentation . . . . .	247
12.36 pmath-util/version.h File Reference . . . . .	248

<a href="#">12.37 version.h File Reference</a> . . . . .	248
<a href="#">12.37.1 Define Documentation</a> . . . . .	248
<a href="#">12.37.2 Variable Documentation</a> . . . . .	249
<a href="#">12.38 pmath.h File Reference</a> . . . . .	250

## 1 The pMath Computer Algebra System Library

### Author:

Peter Frentrup

### Date:

2011

### Introduction

pMath is a free CAS for Windows and Unix like systems. The whole CAS consists of three projects:

- The pMath library documented here, which implements the parser, interpreter, mathematical functionality and OS binding.
- The RichMath graphical front-end.
- Addon libraries/modules for the pMath library and Language (e.g. a Java binding).

You as a user (front-end or module programmer) of the pMath library just have to `#include <pmath.h>` and link with the appropriate library file.

This document does not cover the pMath *Language* itself.

### Links/Dependencies

pMath is build on top of several open source libraries:

- GMP (<http://gmplib.org>)
- MPFR (<http://www.mpfr.org>)
- PCRE (<http://www.pcre.org>)

## 2 Todo List

**Class [pmath\\_thread\\_t](#)** Implement `pmath_run_parallel(number_of_parallel_threads, callback)`.

**Global [pmath\\_task\\_t](#)** document [pmath-util/concurrency/threadpool.h](#)

**Global [pmath\\_thread\\_send\\_wait](#)** Check, what happens if `mq` belongs to a parent thread.

## 3 Deprecated List

Global [pmath\\_symbol\\_t::pmath\\_symbol\\_get\\_value\(pmath\\_symbol\\_t symbol\)](#)

Global [pmath\\_symbol\\_t::pmath\\_symbol\\_set\\_value\(pmath\\_symbol\\_t symbol, pmath\\_t value\)](#)

Global [pmath\\_symbol\\_t::pmath\\_symbol\\_synchronized\(pmath\\_symbol\\_t symbol, pmath\\_callback\\_t callback, v](#)

## 4 Module Index

### 4.1 Modules

Here is a list of all modules:

<b>Custom Objects</b>	<b><a href="#">8</a></b>
<b>Expressions</b>	<b><a href="#">10</a></b>
<b>Numbers</b>	<b><a href="#">16</a></b>
<b>Objects - the Base of pMath</b>	<b><a href="#">36</a></b>
<b>Strings</b>	<b><a href="#">48</a></b>
<b>Symbols</b>	<b><a href="#">57</a></b>
<b>C++ Binding</b>	<b><a href="#">66</a></b>
<b>Parsing Code</b>	<b><a href="#">68</a></b>
<b>General Purpose Types</b>	<b><a href="#">86</a></b>
<b>Atomic Operations</b>	<b><a href="#">87</a></b>
<b>Thread Messaging</b>	<b><a href="#">93</a></b>
<b>Multithreading with pMath</b>	<b><a href="#">99</a></b>
<b>Debugging</b>	<b><a href="#">106</a></b>
<b>File API</b>	<b><a href="#">107</a></b>
<b>Hashtables</b>	<b><a href="#">117</a></b>
<b>Object Utility Functions</b>	<b><a href="#">124</a></b>

Memory Management	130
Messages	133
Threadsafe Stacks	135
Version Information	138
Front-ends	138

## 5 Namespace Index

### 5.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">pmath</a> (Provides the C++ binding )	141
---	-----

## 6 Data Structure Index

### 6.1 Class Hierarchy

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

Expr	155
File	165
BinaryFile	151
ReadableBinaryFile	197
WriteableBinaryFile	212
TextFile	206
ReadableTextFile	198
WriteableTextFile	214
String	200
Gather	167
pmath_atomic2_t	169
pmath_atomic_t	169
pmath_binary_file_api_t	170

<b>pmath_cstr_writer_info_t</b>	<b>171</b>
<b>pmath_hashtable_t</b>	<b>175</b>
<b>pmath_ht_class_t</b>	<b>176</b>
<b>pmath_span_array_t</b>	<b>182</b>
<b>pmath_span_t</b>	<b>183</b>
<b>pmath_stack_t</b>	<b>184</b>
<b>pmath_t</b>	<b>188</b>
<b>pmath_custom_t</b>	<b>172</b>
<b>pmath_messages_t</b>	<b>179</b>
<b>pmath_expr_t</b>	<b>172</b>
<b>pmath_number_t</b>	<b>180</b>
<b>pmath_float_t</b>	<b>175</b>
<b>pmath_rational_t</b>	<b>181</b>
<b>pmath_integer_t</b>	<b>177</b>
<b>pmath_quotient_t</b>	<b>181</b>
<b>pmath_string_t</b>	<b>184</b>
<b>pmath_symbol_t</b>	<b>186</b>
<b>pmath_text_file_api_t</b>	<b>192</b>
<b>pmath_thread_t</b>	<b>193</b>
<b>pmath_threadlock_t</b>	<b>195</b>
<b>pmath_write_ex_t</b>	<b>195</b>
<b>UserStream</b>	<b>210</b>
<b>BinaryUserStream</b>	<b>153</b>
<b>TextUserStream</b>	<b>208</b>

## 7 Data Structure Index

### 7.1 Data Structures

Here are the data structures with brief descriptions:

<b>BinaryFile</b> (A wrapper for pMath binary file objects (byte data streams) )	151
<b>BinaryUserStream</b> (Abstract base class for C++ callbacks used as pMath binary files )	153
<b>Expr</b> (A wrapper for <b>pmath_t</b> and drived types )	155
<b>File</b> (A wrapper for pMath file objects (data streams) )	165
<b>Gather</b> (Utility class for emitting and gathering expressions/building lists )	167
<b>pmath_atomic2_t</b> (A 2-pointer-sized atomic variable type )	169
<b>pmath_atomic_t</b> (A pointer-sized atomic variable type )	169
<b>pmath_binary_file_api_t</b> (Access functions for binary files )	170
<b>pmath_cstr_writer_info_t</b> (Additional information for <b>pmath_utf8_writer()</b> or <b>pmath_native_writer()</b> )	171
<b>pmath_custom_t</b> (The Custom Object class )	172
<b>pmath_expr_t</b> (The Expression class )	172
<b>pmath_float_t</b> (The Floating Point Number class )	175
<b>pmath_hashtable_t</b> (The Hashtable class )	175
<b>pmath_ht_class_t</b> (A hashtable interface )	176
<b>pmath_integer_t</b> (The Integer class )	177
<b>pmath_messages_t</b> (A message queue for interthread communication )	179
<b>pmath_number_t</b> (The abstract Number class )	180
<b>pmath_quotient_t</b> (The Quotient class )	181
<b>pmath_rational_t</b> (The abstract Rational Number class )	181
<b>pmath_span_array_t</b> (Internal flat representaion of spans )	182
<b>pmath_span_t</b> (Represents a span in a <b>span-array</b> )	183
<b>pmath_stack_t</b> (The type of pMath's threadsafe stacks )	184
<b>pmath_string_t</b> (The string class )	184

<a href="#">pmath_symbol_t</a> (The Symbol class )	186
<a href="#">pmath_t</a> (The basic type of all pMath objects )	188
<a href="#">pmath_text_file_api_t</a> (Access functions for text files )	192
<a href="#">pmath_thread_t</a> (The Representation of a thread )	193
<a href="#">pmath_threadlock_t</a> (A reentrant lock for threads )	195
<a href="#">pmath_write_ex_t</a> (Command structure for <a href="#">pmath_write_ex()</a> . This should be initalized with <code>memset(&amp;ex, 0, sizeof(ex)); ex.size = sizeof(ex); ...</code> )	195
<a href="#">ReadableBinaryFile</a> (A wrapper for readable pMath binary file objects (byte data streams) )	197
<a href="#">ReadableTextFile</a> (A wrapper for pMath readable text file objects (byte data streams) )	198
<a href="#">String</a> (A wrapper for <a href="#">pmath_string_t</a> )	200
<a href="#">TextFile</a> (A wrapper for pMath text file objects (byte data streams) )	206
<a href="#">TextUserStream</a> (Abstract base class for C++ callbacks used as pMath text files )	208
<a href="#">UserStream</a> (Abstract base class for C++ callbacks used as pMath files )	210
<a href="#">WriteableBinaryFile</a> (A wrapper for writeable pMath binary file objects (byte data streams) )	212
<a href="#">WriteableTextFile</a> (A wrapper for pMath writeable text file objects (byte data streams) )	214

## 8 File Index

### 8.1 File List

Here is a list of all files with brief descriptions:

<a href="#">pmath-config.h</a>	215
<a href="#">pmath-cpp.h</a>	227
<a href="#">pmath-types.h</a>	235
<a href="#">pmath.h</a>	250
<a href="#">version.h</a>	248

---

<a href="#">pmath-core/custom.h</a>	218
<a href="#">pmath-core/expressions.h</a>	218
<a href="#">pmath-core/numbers.h</a>	218
<a href="#">pmath-core/objects-inline.h</a>	219
<a href="#">pmath-core/objects.h</a>	223
<a href="#">pmath-core/strings.h</a>	226
<a href="#">pmath-core/symbols.h</a>	226
<a href="#">pmath-language/charnames.h</a>	230
<a href="#">pmath-language/scanner.h</a>	231
<a href="#">pmath-language/tokens.h</a>	231
<a href="#">pmath-util/approximate.h</a>	235
<a href="#">pmath-util/compression.h</a>	236
<a href="#">pmath-util/debug.h</a>	240
<a href="#">pmath-util/dlmalloc.h</a>	241
<a href="#">pmath-util/emit-and-gather.h</a>	241
<a href="#">pmath-util/evaluation.h</a>	241
<a href="#">pmath-util/files.h</a>	241
<a href="#">pmath-util/hashtables.h</a>	243
<a href="#">pmath-util/helpers.h</a>	243
<a href="#">pmath-util/line-writer.h</a>	244
<a href="#">pmath-util/memory.h</a>	244
<a href="#">pmath-util/messages.h</a>	245
<a href="#">pmath-util/mixed-file.h</a>	245
<a href="#">pmath-util/serialize.h</a>	245
<a href="#">pmath-util/stacks.h</a>	247
<a href="#">pmath-util/strtod.h</a>	247
<a href="#">pmath-util/version.h</a>	248



<a href="#">pmath-util/concurrency/atomic.h</a>	236
<a href="#">pmath-util/concurrency/threadlocks.h</a>	238
<a href="#">pmath-util/concurrency/threadmsg.h</a>	238
<a href="#">pmath-util/concurrency/threadpool.h</a>	238
<a href="#">pmath-util/concurrency/threads.h</a>	240
<a href="#">pmath-util/concurrency/atomic/non-atomic.h</a>	237

## 9 Module Documentation

### 9.1 Custom Objects

Encapsulate arbitrary data in pMath objects.

#### Data Structures

- class [pmath\\_custom\\_t](#)  
*The Custom Object class.*

#### Typedefs

- typedef [pmath\\_t](#) [pmath\\_custom\\_t](#)

#### Functions

- [pmath\\_custom\\_t](#) [pmath\\_custom\\_new](#) (void \*data, [pmath\\_callback\\_t](#) destructor)  
*Create a custom object.*
- void \* [pmath\\_custom\\_get\\_data](#) ([pmath\\_custom\\_t](#) custom)  
*Get a custom object's data member.*
- [pmath\\_bool\\_t](#) [pmath\\_custom\\_has\\_destructor](#) ([pmath\\_custom\\_t](#) custom, [pmath\\_callback\\_t](#) dtor)  
*Check for a custom object's data type.*

### 9.1.1 Detailed Description

Encapsulate arbitrary data in pMath objects.

Custom Objects consist of a pointer and a destructor. The destructor is called (with the pointer as its argument) when the custom object's reference pointer yields zero.

Custom Objects are not evaluateable. This means evaluation of such an object returns PMATH\_NULL. But you can store custom objects in symbols (directly with [pmath\\_symbol\\_set\\_value\(\)](#)).

A symbol that holds a custom object remains unevaluated. It can also contain function definitions. But those must be set *after* setting the value with `pmath_symbol_set_value(my_symbol, my_custom_object)`:

Example: You want to store a custom object and a function definition in a symbol (`my_symbol/:` `answer(my_symbol):= 42`).

```
pmath_custom_t my_custom_object = pmath_custom_new(my_data, my_destructor);
pmath_symbol_set_value(my_symbol, my_custom_object);

pmath_unref(pmath_evaluate(
    pmath_parse_string("'1'/: answer('1'):= 42", 1, pmath_ref(my_symbol))));
```

### 9.1.2 Typedef Documentation

#### 9.1.2.1 typedef pmath\_t pmath\_custom\_t

### 9.1.3 Function Documentation

#### 9.1.3.1 void \* pmath\_custom\_get\_data (pmath\_custom\_t custom) [inherited]

Get a custom object's data member.

##### Parameters:

*custom* A custom object.

##### Returns:

The objects data member or PMATH\_NULL if *custom* is PMATH\_NULL.

Note that you cannot assume anything about the content of this pointer unless you know its destructor (check [pmath\\_custom\\_has\\_destructor](#)).

All access to `*data` must be threadsafe/synchronized. By convention, you are the only one who moves custom objects with your destructor around (other modules should not handle custom objects whose destructor they do not know). And normally, each of your

custom objects is stored in one symbol. So synchronization can be done with [pmath\\_symbol\\_synchronized\(\)](#). If one of these conditions is not met and a custom object could be accessed from multiple threads (See [Multithreading with pMath](#)), you must also store a synchronization object (e.g. symbol or threadlock) in the *data* member and use this.

### 9.1.3.2 `pmath_bool_t pmath_custom_has_destructor (pmath_custom_t custom, pmath_callback_t dtor)` [inherited]

Check for a custom object's data type.

#### Parameters:

*custom* A custom object.

*dtor* A callback function.

#### Returns:

TRUE if the object's destructor is *dtor*.

### 9.1.3.3 `pmath_custom_t pmath_custom_new (void * data, pmath_callback_t destructor)` [inherited]

Create a custom object.

#### Parameters:

*data* An arbitrary pointer.

*destructor* A function that will be called on object destruction to enable freeing of *data*.

#### Returns:

A custom object or PMATH\_NULL on failure (in that case, *destructor(data)* is called immediately).

## 9.2 Expressions

Expression objects in pMath.

#### Data Structures

- class [pmath\\_expr\\_t](#)  
The Expression class.

### Typedefs

- typedef [pmath\\_t](#) [pmath\\_expr\\_t](#)

### Functions

- [pmath\\_expr\\_t](#) [pmath\\_expr\\_new](#) ([pmath\\_t](#) head, [size\\_t](#) length)  
*Create a new expression.*
- [pmath\\_expr\\_t](#) [pmath\\_expr\\_new\\_extended](#) ([pmath\\_t](#) head, [size\\_t](#) length,...)  
*Create a new expression with all items given.*
- [pmath\\_expr\\_t](#) [pmath\\_expr\\_resize](#) ([pmath\\_expr\\_t](#) expr, [size\\_t](#) new\_length)  
*Resize an expression.*
- [pmath\\_expr\\_t](#) [pmath\\_expr\\_append](#) ([pmath\\_expr\\_t](#) expr, [size\\_t](#) count,...)  
*Append some items to an expression.*
- [size\\_t](#) [pmath\\_expr\\_length](#) ([pmath\\_expr\\_t](#) expr)  
*Get an expression's length.*
- [pmath\\_t](#) [pmath\\_expr\\_get\\_item](#) ([pmath\\_expr\\_t](#) expr, [size\\_t](#) index)  
*Get an item from an expression.*
- [pmath\\_t](#) [pmath\\_expr\\_extract\\_item](#) ([pmath\\_expr\\_t](#) expr, [size\\_t](#) index)  
*Extract an item from an expression.*
- [pmath\\_expr\\_t](#) [pmath\\_expr\\_get\\_item\\_range](#) ([pmath\\_expr\\_t](#) expr, [size\\_t](#) start, [size\\_t](#) length)  
*Get multiple items from an expression.*
- [pmath\\_expr\\_t](#) [pmath\\_expr\\_set\\_item](#) ([pmath\\_expr\\_t](#) expr, [size\\_t](#) index, [pmath\\_t](#) item)  
*Set an item in an expression.*
- [pmath\\_expr\\_t](#) [pmath\\_expr\\_remove\\_all](#) ([pmath\\_expr\\_t](#) expr, [pmath\\_t](#) rem)  
*Remove all occurrences of an object from an expression.*
- [pmath\\_expr\\_t](#) [pmath\\_expr\\_sort](#) ([pmath\\_expr\\_t](#) expr)  
*Sort an expression.*
- [pmath\\_expr\\_t](#) [pmath\\_expr\\_flatten](#) ([pmath\\_expr\\_t](#) expr, [pmath\\_t](#) head, [size\\_t](#) depth)  
*Flatten an expression.*

### 9.2.1 Detailed Description

Expression objects in pMath.

Any pMath language-level expression (lists, terms, function calls, ...) is stored in a [pmath\\_expr\\_t](#) – an array of pMath objects.

See also:

[Object Utility Functions](#)

### 9.2.2 Typedef Documentation

#### 9.2.2.1 typedef pmath\_t pmath\_expr\_t

### 9.2.3 Function Documentation

#### 9.2.3.1 pmath\_expr\_t pmath\_expr\_append (pmath\_expr\_t *expr*, size\_t *count*, ...) [inherited]

Append some items to an expression.

**Parameters:**

*expr* The old expression. It will be freed/invalid after the call.

*count* The number of items to append.

... exactly count pmath\_ts. Do not use them after the call.

**Returns:**

PMATH\_NULL or a new expression that contains all items of *expr* followed by the items in '...'. You must destroy it with [pmath\\_unref\(\)](#).

If *expr* == PMATH\_NULL, the returns value is pmath\_expr\_new\_extended(PMATH\_NULL, count, ...).

#### 9.2.3.2 pmath\_t pmath\_expr\_extract\_item (pmath\_expr\_t *expr*, size\_t *index*) [inherited]

Extract an item from an expression.

**Parameters:**

*expr* A pMath expression.

*index* The index of the item.

**Returns:**

Same as [pmath\\_expr\\_get\\_item\(\)](#) but if *expr* has `refcount==1`, the item might be removed from *expr* to ensure that the item's `refcount` is 1.

Normally, you should use [pmath\\_expr\\_get\\_item\(\)](#). This function is for use in loops which modify items of an expression.

**9.2.3.3 `pmath_expr_t` `pmath_expr_flatten` (`pmath_expr_t` *expr*, `pmath_t` *head*, `size_t` *depth*)** [inherited]

Flatten an expression.

**Parameters:**

*expr* A pMath expression. It will be destroyed, do not use it after the call.

*head* The head of items, that should be flattened out. It will be destroyed, so you can use `'pmath_expr_get_item(expr)'` directly to use *expr*'s head.

*depth* The depth to which level flattening should be done. A value of 0 means 'no flattening'.

**Returns:**

A new expression where all items, that have the same head as *expr*, will be flattened.

**9.2.3.4 `pmath_t` `pmath_expr_get_item` (`pmath_expr_t` *expr*, `size_t` *index*)** [inherited]

Get an item from an expression.

**Parameters:**

*expr* A pMath expression.

*index* The index of the item.

**Returns:**

A copy of the requested item, if *index* is not greater than the length of *expr* and `PMATH_NULL` otherwise. You must destroy it with [pmath\\_unref\(\)](#).

### 9.2.3.5 `pmath_expr_t pmath_expr_get_item_range (pmath_expr_t expr, size_t start, size_t length)` [inherited]

Get multiple items from an expression.

#### Parameters:

*expr* A pMath expression. It will *not* be destroyed.

*start* The start index of the items.

*length* The number of the items.

#### Returns:

A new expression with the same head as *expr*. Its length is  $\max(0, \min(\text{start} + \text{length}, 1 + \text{pmath\_expr\_length}(\text{expr})) - \text{start})$  and it contains the items from *expr* beginning at index *start*.

### 9.2.3.6 `size_t pmath_expr_length (pmath_expr_t expr)` [inherited]

Get an expression's length.

#### Parameters:

*expr* A pMath expression.

#### Returns:

The number of items in *expr* (not counting the head).

### 9.2.3.7 `pmath_expr_t pmath_expr_new (pmath_t head, size_t length)` [inherited]

Create a new expression.

#### Parameters:

*head* The expression's head (index 0). Do not use them after the call.

*length* The number of additional items in the expression.

#### Returns:

PMATH\_NULL or a new expression with head at index 0 and all other items initialized to PMATH\_NULL. You must destroy it with [pmath\\_unref\(\)](#).

### 9.2.3.8 `pmath_expr_t` `pmath_expr_new_extended` (`pmath_t` *head*, `size_t` *length*, ...) [inherited]

Create a new expression with all items given.

#### Parameters:

*head* The expression's head (index 0). Do not use them after the call.

*length* The number of additional items in the expression.

... exactly *length* `pmath_ts`. Do not use them after the call.

#### Returns:

PMATH\_NULL or a new expression with head at index 0 all items at index *i* = 1..*length* initialized to the *i*'th argument in '...'. You must destroy it with `pmath_unref()`.

### 9.2.3.9 `pmath_expr_t` `pmath_expr_remove_all` (`pmath_expr_t` *expr*, `pmath_t` *rem*) [inherited]

Remove all occurrences of an object from an expression.

#### Parameters:

*expr* A pMath expression. It will be destroyed, do not use it after the call.

*rem* The object to be removed. It will *not* be destroyed.

#### Returns:

PMATH\_NULL or a new expression that contains no occurrences of *rem* (except maybe the head). It is a shrunk version of *expr*.

### 9.2.3.10 `pmath_expr_t` `pmath_expr_resize` (`pmath_expr_t` *expr*, `size_t` *new\_length*) [inherited]

Resize an expression.

#### Parameters:

*expr* The old expression. It will be freed/invalid after the call.

*new\_length* The new length of the expression.



**Returns:**

PMATH\_NULL or a new expression of length `new_length`. You must destroy it with `pmath_unref()`.

If `expr`'s length is less than or equals to `new_length`, all items at 1..`expr`'s length) are be copied and the rest is initialized with PMATH\_NULL. Otherwise, all items at 1..`new_length` are copied, those at (`new_length`+1)..`expr`'s length) are be freed and the rest is initialized with PMATH\_NULL.

### 9.2.3.11 `pmath_expr_t pmath_expr_set_item (pmath_expr_t expr, size_t index, pmath_t item)` [inherited]

Set an item in an expression.

**Parameters:**

*expr* A pMath expression. It will be destroyed, do not use it after the call.

*index* The index of the to-be-changed item.

*item* The new value of the item. It will be destroyed, do not use it after the call.

**Returns:**

PMATH\_NULL or a new expression with item at `index`. You must destroy it with `pmath_unref()`.

If `index` is greater than `expr`'s length, item will be destroyed and the return value is `expr`.

### 9.2.3.12 `pmath_expr_t pmath_expr_sort (pmath_expr_t expr)` [inherited]

Sort an expression.

**Parameters:**

*expr* A pMath expression. It will be destroyed, do not use it after the call.

**Returns:**

A new expression where all items from `expr` are sorted (except the head, which remains unchanged).

## 9.3 Numbers

Number objects in pMath.

### Data Structures

- class `pmath_number_t`  
*The abstract Number class.*
- class `pmath_rational_t`  
*The abstract Rational Number class.*
- class `pmath_integer_t`  
*The Integer class.*
- class `pmath_quotient_t`  
*The Quotient class.*
- class `pmath_float_t`  
*The Floating Point Number class.*

### Defines

- `#define PMATH_MACHINE_PRECISION 0`
- `#define PMATH_AUTO_PRECISION 1`
- `#define pmath_integer_new_si32(si) PMATH_FROM_INT32(si)`  
*Create an integer object from an int32\_t.*
- `#define pmath_integer_new_siptr(si)`  
*Create an integer object from an intptr\_t.*
- `#define pmath_integer_new_uiptr(ui)`  
*Create an integer object from an uintptr\_t.*
- `#define pmath_integer_fits_si32(integer) pmath_is_int32(integer)`  
*Check whether a pMath integer is in range  $-2^{31} .. 2^{31}-1$ .*
- `#define pmath_integer_fits_siptr(integer)`  
*Check whether a pMath integer fits into an intptr\_t.*
- `#define pmath_integer_fits_uiptr(integer)`  
*Check whether a pMath integer fits into an uintptr\_t.*
- `#define pmath_integer_get_siptr`  
*Convert a pMath integer to a intptr\_t.*
- `#define pmath_integer_get_uiptr`  
*Convert a pMath integer to a uintptr\_t.*

- `pmath_integer_new_si32(si)`  
*Create an integer object from an `int32_t`.*
- `pmath_integer_new_siptr(si)`  
*Create an integer object from an `intptr_t`.*
- `pmath_integer_new_uiptr(ui)`  
*Create an integer object from an `uintptr_t`.*
- `pmath_integer_fits_si32(integer)`  
*Check whether a `pMath` integer is in range  $-2^{31} .. 2^{31}-1$ .*
- `pmath_integer_fits_siptr(integer)`  
*Check whether a `pMath` integer fits into an `intptr_t`.*
- `pmath_integer_fits_uiptr(integer)`  
*Check whether a `pMath` integer fits into an `uintptr_t`.*
- `pmath_integer_get_siptr`  
*Convert a `pMath` integer to a `intptr_t`.*
- `pmath_integer_get_uiptr`  
*Convert a `pMath` integer to a `uintptr_t`.*

### Typedefs

- `typedef pmath_t pmath_number_t`
- `typedef pmath_number_t pmath_rational_t`
- `typedef pmath_rational_t pmath_integer_t`
- `typedef pmath_rational_t pmath_mpint_t`
- `typedef pmath_rational_t pmath_quotient_t`
- `typedef pmath_number_t pmath_float_t`
- `typedef pmath_float_t pmath_mpfloating_t`

### Enumerations

- `enum pmath_precision_control_t {`  
`PMATH_PREC_CTRL_AUTO = 0, PMATH_PREC_CTRL_MACHINE-`  
`PREC = 1,`  
`PMATH_PREC_CTRL_GIVEN_PREC = 2, PMATH_PREC_CTRL_GIVEN-`  
`ACC = 3 }`

## Functions

- double `pmath_accuracy` (`pmath_t` obj)  
*Get the accuracy (in bits) of an object.*
- double `pmath_precision` (`pmath_t` obj)  
*Get the precision (in bits) of an object.*
- `pmath_t` `pmath_set_accuracy` (`pmath_t` obj, double acc)  
*Set an object's accuracy in bits.*
- `pmath_t` `pmath_set_precision` (`pmath_t` obj, double prec)  
*Set an object's accuracy in bits.*
- `pmath_t` `pmath_approximate` (`pmath_t` obj, double precision\_goal, double accuracy\_goal, `pmath_bool_t` \*aborted)  
*Approximate an object.*
- `pmath_integer_t` `pmath_integer_new_slong` (signed long int si)  
*Create an integer object from a signed long.*
- `pmath_integer_t` `pmath_integer_new_ulong` (unsigned long int ui)  
*Create an integer object from an unsigned long.*
- `pmath_integer_t` `pmath_integer_new_ui32` (`uint32_t` ui)  
*Create an integer object from an `uint32_t`.*
- `pmath_integer_t` `pmath_integer_new_si64` (`int64_t` si)  
*Create an integer object from an `int64_t`.*
- `pmath_integer_t` `pmath_integer_new_ui64` (`uint64_t` ui)  
*Create an integer object from an `uint64_t`.*
- `pmath_integer_t` `pmath_integer_new_data` (`size_t` count, int order, int size, int endian, `size_t` nails, const void \*data)  
*Create an integer object from a data buffer.*
- `pmath_integer_t` `pmath_integer_new_str` (const char \*str, int base)  
*Create an integer object from a C String.*
- `pmath_rational_t` `pmath_rational_new` (`pmath_integer_t` numerator, `pmath_integer_t` denominator)  
*Create a rational number.*
- `pmath_integer_t` `pmath_rational_numerator` (`pmath_rational_t` rational)  
*Get the numerator of a rational number.*

- [pmath\\_integer\\_t pmath\\_rational\\_denominator](#) ([pmath\\_rational\\_t](#) rational)  
*Get the denominator of a rational number.*
- [pmath\\_number\\_t pmath\\_float\\_new\\_str](#) (const char \*str, int base, [pmath\\_precision\\_control\\_t](#) precision\_control, double base\_precision\_accuracy)  
*Create a floating point number from a string.*
- [pmath\\_bool\\_t pmath\\_integer\\_fits\\_ui32](#) ([pmath\\_integer\\_t](#) integer)  
*Check whether a pMath integer is in range  $0 \dots 2^{32}-1$ .*
- [pmath\\_bool\\_t pmath\\_integer\\_fits\\_si64](#) ([pmath\\_integer\\_t](#) integer)  
*Check whether a pMath integer is in range  $-2^{63} \dots 2^{63}-1$ .*
- [pmath\\_bool\\_t pmath\\_integer\\_fits\\_ui64](#) ([pmath\\_integer\\_t](#) integer)  
*Check whether a pMath integer is in range  $0 \dots 2^{64}-1$ .*
- [int32\\_t pmath\\_integer\\_get\\_si32](#) ([pmath\\_integer\\_t](#) integer)  
*Convert a pMath integer to a signed long int.*
- [uint32\\_t pmath\\_integer\\_get\\_ui32](#) ([pmath\\_integer\\_t](#) integer)  
*Convert a pMath integer to a unsigned long int.*
- [int64\\_t pmath\\_integer\\_get\\_si64](#) ([pmath\\_integer\\_t](#) integer)  
*Convert a pMath integer to an int64\_t.*
- [uint64\\_t pmath\\_integer\\_get\\_ui64](#) ([pmath\\_integer\\_t](#) integer)  
*Convert a pMath integer to a uint64\_t.*
- [double pmath\\_number\\_get\\_d](#) ([pmath\\_number\\_t](#) number)  
*Convert a pMath number to a double.*
- [int pmath\\_number\\_sign](#) ([pmath\\_number\\_t](#) num)  
*Get a number's sign.*
- [pmath\\_number\\_t pmath\\_number\\_neg](#) ([pmath\\_number\\_t](#) num)  
*Get a number's negative.*

### 9.3.1 Detailed Description

Number objects in pMath.

pMath supports arbitrary big integers and rational values, floating point numbers in machine precision or with automatic precision tracking and complex numbers (the latter are represented by ordinary [pmath\\_expr\\_t](#), all other number types have their own internal representation).

Note that it might be more convenient to use `pmath_build_value()` than the specialized constructors represented here, because the former supports Infinity and Undefined (NaN) values for C doubles.

The GNU Multiple Precision Library (<http://gmplib.org/>) is used for integer and rational arithmetic and the MPFR library (<http://www.mpfr.org/>) for floating point arithmetic.

### 9.3.2 Define Documentation

#### 9.3.2.1 `#define PMATH_AUTO_PRECISION 1`

#### 9.3.2.2 `pmath_integer_fits_si32(integer)` [inherited]

Check whether a pMath integer is in range  $-2^{31} .. 2^{31}-1$ .

##### Parameters:

*integer* A pMath integer. It won't be freed.

##### Returns:

TRUE iff the value is small enough for an `int32_t`.

#### 9.3.2.3 `#define pmath_integer_fits_si32(integer) pmath_is_int32(integer)`

Check whether a pMath integer is in range  $-2^{31} .. 2^{31}-1$ .

##### Parameters:

*integer* A pMath integer. It won't be freed.

##### Returns:

TRUE iff the value is small enough for an `int32_t`.

#### 9.3.2.4 `pmath_integer_fits_siptr(integer)` [inherited]

Check whether a pMath integer fits into an `intptr_t`.

##### Parameters:

*integer* A pMath integer. It won't be freed.

**Returns:**

TRUE iff the value is small enough.

**9.3.2.5 #define pmath\_integer\_fits\_siptr(integer)**

Check whether a pMath integer fits into an intptr\_t.

**Parameters:**

*integer* A pMath integer. It wont be freed.

**Returns:**

TRUE iff the value is small enough.

**9.3.2.6 pmath\_integer\_fits\_uiptr(integer) [inherited]**

Check whether a pMath integer fits into an uintptr\_t.

**Parameters:**

*integer* A pMath integer. It wont be freed.

**Returns:**

TRUE iff the value is small enough.

**9.3.2.7 #define pmath\_integer\_fits\_uiptr(integer)**

Check whether a pMath integer fits into an uintptr\_t.

**Parameters:**

*integer* A pMath integer. It wont be freed.

**Returns:**

TRUE iff the value is small enough.

#### 9.3.2.8 `pmath_integer_get_siptr` [inherited]

Convert a pMath integer to a `intptr_t`.

**Parameters:**

*integer* A pMath integer. It wont be freed.

**Returns:**

The integer's value if it fits.

**See also:**

[pmath\\_integer\\_fits\\_siptr](#)

#### 9.3.2.9 `#define pmath_integer_get_siptr`

Convert a pMath integer to a `intptr_t`.

**Parameters:**

*integer* A pMath integer. It wont be freed.

**Returns:**

The integer's value if it fits.

**See also:**

[pmath\\_integer\\_fits\\_siptr](#)

#### 9.3.2.10 `pmath_integer_get_uiptr` [inherited]

Convert a pMath integer to a `uintptr_t`.

**Parameters:**

*integer* A pMath integer. It wont be freed.

**Returns:**

The integer's value if it fits.

**See also:**

[pmath\\_integer\\_fits\\_uiptr](#)



#### 9.3.2.11 `#define pmath_integer_get_uiptr`

Convert a pMath integer to a `uintptr_t`.

**Parameters:**

*integer* A pMath integer. It wont be freed.

**Returns:**

The integer's value if it fits.

**See also:**

[pmath\\_integer\\_fits\\_uiptr](#)

#### 9.3.2.12 `pmath_integer_new_si32(si)` [inherited]

Create an integer object from an `int32_t`.

**Parameters:**

*si* An `int32_t`.

**Returns:**

A pMath integer with the specified value.

#### 9.3.2.13 `#define pmath_integer_new_si32(si) PMATH_FROM_INT32(si)`

Create an integer object from an `int32_t`.

**Parameters:**

*si* An `int32_t`.

**Returns:**

A pMath integer with the specified value.

#### 9.3.2.14 `pmath_integer_new_siptr(si)` [inherited]

Create an integer object from an `intptr_t`.

**Parameters:**

*si* An `intptr_t` value.

**Returns:**

A pMath integer with the specified value or `PMATH_NULL`.

**9.3.2.15 #define pmath\_integer\_new\_siptr(si)**

Create an integer object from an `intptr_t`.

**Parameters:**

*si* An `intptr_t` value.

**Returns:**

A pMath integer with the specified value or `PMATH_NULL`.

**9.3.2.16 pmath\_integer\_new\_uiptr(ui) [inherited]**

Create an integer object from an `uintptr_t`.

**Parameters:**

*ui* A `uintptr_t` value.

**Returns:**

A pMath integer with the specified value or `PMATH_NULL`.

**9.3.2.17 #define pmath\_integer\_new\_uiptr(ui)**

Create an integer object from an `uintptr_t`.

**Parameters:**

*ui* A `uintptr_t` value.

**Returns:**

A pMath integer with the specified value or `PMATH_NULL`.

9.3.2.18 `#define PMATH_MACHINE_PRECISION 0`

### 9.3.3 Typedef Documentation

9.3.3.1 `typedef pmath_number_t pmath_float_t`

9.3.3.2 `typedef pmath_rational_t pmath_integer_t`

9.3.3.3 `typedef pmath_float_t pmath_mpfloor_t`

9.3.3.4 `typedef pmath_rational_t pmath_mpint_t`

9.3.3.5 `typedef pmath_t pmath_number_t`

9.3.3.6 `typedef pmath_rational_t pmath_quotient_t`

9.3.3.7 `typedef pmath_number_t pmath_rational_t`

### 9.3.4 Enumeration Type Documentation

9.3.4.1 `enum pmath_precision_control_t`

Enumerator:

*PMATH\_PREC\_CTRL\_AUTO*

*PMATH\_PREC\_CTRL\_MACHINE\_PREC*

*PMATH\_PREC\_CTRL\_GIVEN\_PREC*

*PMATH\_PREC\_CTRL\_GIVEN\_ACC*

### 9.3.5 Function Documentation

#### 9.3.5.1 double pmath\_accuracy (pmath\_t *obj*)

Get the accuracy (in bits) of an object.

**Parameters:**

*obj* An object. It will be freed.

**Returns:**

The number of known bits after the decimal point.

HUGE\_VAL is given for exact quantities. If *obj* is an expression, the minimum of its items' accuracies is returned.

Note that the builtin function Accuracy() uses base 10, but this function operates on base 2.

#### 9.3.5.2 pmath\_t pmath\_approximate (pmath\_t *obj*, double *precision\_goal*, double *accuracy\_goal*, pmath\_bool\_t \**aborted*)

Approximate an object.

**Parameters:**

*obj* An object. It will be freed.

*precision\_goal* The requested precision in bits.

*accuracy\_goal* The requested accuracy in bits.

*aborted* [out] Whether the approximation was aborted and an N::meprec should be generated. When this is NULL, N::meprec will be generated automatically if necessary.

**Returns:**

The approximated object.

Use `prec == -HUGE_VAL` or `acc == -HUGE_VAL` for machine precision. Use `acc == HUGE_VAL` if the accuracy is not important and use `prec == HUGE_VAL` if the precision is not important.

### 9.3.5.3 `pmath_number_t` `pmath_float_new_str` (`const char * str`, `int base`, `pmath_precision_control_t precision_control`, `double base_precision_accuracy`) [inherited]

Create a floating point number from a string.

#### Parameters:

**str** A C-string representing the value in a given *base*. It should have the form "ddd.ddd" or simply "ddd". An exponent can be appended with "ennn" or if *base* != 10 "@nnn".

**base** The base between 2 and 36.

**precision\_control** flag for controlling the precision.

**base\_precision\_accuracy** given precision or accuracy. depending on the value of the above flag.

#### Returns:

a new pMath floating point number or PMATH\_NULL on error or the integer 0 (see below when this happens).

#### Remarks:

*precision\_control* may have one of the following values:

- `PMATH_PREC_CTRL_AUTO`:  
The precision is specified by the number of digits given in *str*. It may result in a pMath machine float, mulit-precision float or integer.  
The value of *base\_precision\_accuracy* will be ignored.
- `PMATH_PREC_CTRL_MACHINE_PREC`:  
The result is a pMath machine float.  
The value of *base\_precision\_accuracy* will be ignored.
- `PMATH_PREC_CTRL_GIVEN_PREC`:  
If the number's value is 0, the *integer* 0 will be returned.  
The precision is given by *base\_precision\_accuracy* (interpreted in the given base).
- `PMATH_PREC_CTRL_GIVEN_ACC`:  
*base\_precision\_accuracy* specifies the accuracy (the number of known *base*-digits after the point). The precision is calculated appropriately.

For a multiprecision float  $x \neq 0$  with absolute error *dx*, *accuracy* and *precision* are:

```
accuracy = -Log(base, dx)
precision = -Log(base, dx / Abs(x))
```

So  $\text{precision} = \text{accuracy} + \text{Log}(\text{base}, \text{Abs}(x))$ .

#### 9.3.5.4 `pmath_bool_t pmath_integer_fits_si64 (pmath_integer_t integer)` [inherited]

Check whether a pMath integer is in range  $-2^{63} .. 2^{63}-1$ .

##### Parameters:

*integer* A pMath integer. It wont be freed.

##### Returns:

TRUE iff the value is small enough for an `int64_t`.

#### 9.3.5.5 `pmath_bool_t pmath_integer_fits_ui32 (pmath_integer_t integer)` [inherited]

Check whether a pMath integer is in range  $0 .. 2^{32}-1$ .

##### Parameters:

*integer* A pMath integer. It wont be freed.

##### Returns:

TRUE iff the value is small enough for an `uint32_t`.

#### 9.3.5.6 `pmath_bool_t pmath_integer_fits_ui64 (pmath_integer_t integer)` [inherited]

Check whether a pMath integer is in range  $0 .. 2^{64}-1$ .

##### Parameters:

*integer* A pMath integer. It wont be freed.

##### Returns:

TRUE iff the value is small enough for an `uint64_t`.

#### 9.3.5.7 `int32_t pmath_integer_get_si32 (pmath_integer_t integer)` [inherited]

Convert a pMath integer to a signed long int.

**Parameters:**

*integer* A pMath integer. It wont be freed.

**Returns:**

The integer's value if it fits.

**See also:**

[pmath\\_integer\\_fits\\_si32](#)

#### 9.3.5.8 `int64_t pmath_integer_get_si64 (pmath_integer_t integer)` [inherited]

Convert a pMath integer to an int64\_t.

**Parameters:**

*integer* A pMath integer. It wont be freed.

**Returns:**

The integer's value if it fits.

**See also:**

[pmath\\_integer\\_fits\\_si32](#)

#### 9.3.5.9 `uint32_t pmath_integer_get_ui32 (pmath_integer_t integer)` [inherited]

Convert a pMath integer to a unsigned long int.

**Parameters:**

*integer* A pMath integer. It wont be freed.

**Returns:**

The integer's value if it fits.

**See also:**

[pmath\\_integer\\_fits\\_ui32](#)

#### 9.3.5.10 `uint64_t pmath_integer_get_ui64 (pmath_integer_t integer)` [inherited]

Convert a pMath integer to a `uint64_t`.

**Parameters:**

*integer* A pMath integer. It wont be freed.

**Returns:**

The integer's value if it fits.

**See also:**

[pmath\\_integer\\_fits\\_ui32](#)

#### 9.3.5.11 `pmath_integer_t pmath_integer_new_data (size_t count, int order, int size, int endian, size_t nails, const void * data)` [inherited]

Create an integer object from a data buffer.

**Parameters:**

*count* The number of words to be read.

*order* The order of the words: 1 for most significant word first or -1 for least significant first.

*size* The size (in bytes) of a word.

*endian* The byte order within each word: 1 for most significant byte first, -1 for least significant first, or 0 for the native endianness of the CPU.

*nails* The most significant *nails* bits of each word are skipped. This can be 0 to use the full words.

*data* The buffer to read from.

**Returns:**

A non-negative integer.

**See also:**

GMP's `mpz_import()`



**9.3.5.12 `pmath_integer_t pmath_integer_new_si64(int64_t si)`** [inherited]

Create an integer object from an `int64_t`.

**Parameters:**

*si* An `int64_t` value.

**Returns:**

A pMath integer with the specified value or `PMATH_NULL`.

**9.3.5.13 `pmath_integer_t pmath_integer_new_slong(signed long int si)`**  
[inherited]

Create an integer object from a signed long.

**Parameters:**

*si* A signed long int.

**Returns:**

A pMath integer with the specified value or `PMATH_NULL`.

**9.3.5.14 `pmath_integer_t pmath_integer_new_str(const char * str, int base)`**  
[inherited]

Create an integer object from a C String.

**Parameters:**

*str* A string representing the value in base *base*.

*base* The base.

**Returns:**

A pMath integer with the specified value or `PMATH_NULL`.

See GMP's `mpz_set_str` for mor information about the parameters.

**9.3.5.15** `pmath_integer_t pmath_integer_new_ui32 (uint32_t ui)`  
[inherited]

Create an integer object from an uint32\_t.

**Parameters:**

*ui* An uint32\_t

**Returns:**

A pMath integer with the specified value or PMATH\_NULL.

**9.3.5.16** `pmath_integer_t pmath_integer_new_ui64 (uint64_t ui)`  
[inherited]

Create an integer object from an uint64\_t.

**Parameters:**

*ui* A uint64\_t value.

**Returns:**

A pMath integer with the specified value or PMATH\_NULL.

**9.3.5.17** `pmath_integer_t pmath_integer_new_ulong (unsigned long int ui)`  
[inherited]

Create an integer object from an unsigned long.

**Parameters:**

*ui* An unsigned long int.

**Returns:**

A pMath integer with the specified value or PMATH\_NULL.

**9.3.5.18** `double pmath_number_get_d (pmath_number_t number)`  
[inherited]

Convert a pMath number to a double.

**Parameters:**

*number* A pMath number. It wont be freed.

**Returns:**

The number's value if it fits.

**9.3.5.19** `pmath_number_t pmath_number_neg (pmath_number_t num)`  
[inherited]

Get a number's negative.

**Parameters:**

*num* A pMath number. It will be freed, do not use it afterwards.

**Returns:**

-num

**9.3.5.20** `int pmath_number_sign (pmath_number_t num)` [inherited]

Get a number's sign.

**Parameters:**

*num* A pMath number. It wont be freed.

**Returns:**

The number's sign (-1, 0 or 1)

**9.3.5.21** `double pmath_precision (pmath_t obj)`

Get the precision (in bits) of an object.

**Parameters:**

*obj* An object. It will be freed.

**Returns:**

The number of known bits.

HUGE\_VAL is given for exact quantities. -HUGE\_VAL means "machine precision". If *obj* is an expression, the minimum of its items' accuracies is returned.

Note that the builtin function Precision() uses base 10, but this function operates on base 2.

#### 9.3.5.22 pmath\_integer\_t pmath\_rational\_denominator (pmath\_rational\_t *rational*) [inherited]

Get the denominator of a rational number.

##### Parameters:

*rational* A rational number (integer or quotient). It wont be freed.

##### Returns:

A reference to the denominator of *rational* if it is a quotient or 1 if it is an integer. You have to destroy the result e.g. with [pmath\\_unref\(\)](#).

#### 9.3.5.23 pmath\_rational\_t pmath\_rational\_new (pmath\_integer\_t *numerator*, pmath\_integer\_t *denominator*) [inherited]

Create a rational number.

##### Parameters:

*numerator* The quotient's numerator. It will be freed.

*denominator* The quotient's denominator. It will be freed.

##### Returns:

An integer, if *denominator* divides *numerator* or a quotient in canonical form otherwise. If denominator is zero, PMATH\_NULL will be returned.

#### 9.3.5.24 pmath\_integer\_t pmath\_rational\_numerator (pmath\_rational\_t *rational*) [inherited]

Get the numerator of a rational number.

##### Parameters:

*rational* A rational number (integer or quotient). It wont be freed.

**Returns:**

A reference to the numerator of *rational* if it is a quotient or *rational* itself if it is an integer. You have to destroy the result e.g. with `pmath_unref()`.

**9.3.5.25 `pmath_t pmath_set_accuracy (pmath_t obj, double acc)`**

Set an object's accuracy in bits.

**Parameters:**

*obj* An object. It will be freed.

*acc* The new number of known bits after the decimal point.

**Returns:**

The new object.

Use `acc == -HUGE_VAL` for machine precision and `prec == -HUGE_VAL` if you want to convert all floating point numbers to exact rational numbers.

Note that the builtin function `SetAccuracy()` uses base 10, but this function operates on base 2.

**9.3.5.26 `pmath_t pmath_set_precision (pmath_t obj, double prec)`**

Set an object's accuracy in bits.

**Parameters:**

*obj* An object. It will be freed.

*prec* The new number of known bits.

**Returns:**

The new object.

Use `prec == -HUGE_VAL` for machine precision and `prec == -HUGE_VAL` if you want to convert all floating point numbers to exact rational numbers.

Note that the builtin function `SetPrecision()` uses base 10, but this function operates on base 2.

**9.4 Objects - the Base of pMath**

The basic class for all pMath objects.

## Data Structures

- class `pmath_t`

*The basic type of all pMath objects.*

- struct `pmath_write_ex_t`

*Command structure for `pmath_write_ex()`. This should be initialized with `memset(&ex, 0, sizeof(ex)); ex.size = sizeof(ex); ...`*

## Defines

- `#define PMATH_TAGMASK_BITCOUNT 12`
- `#define PMATH_TAGMASK_NONDOUBLE 0x7FF00000U`
- `#define PMATH_TAGMASK_POINTER 0xFFF00000U`
- `#define PMATH_TAG_INVALID (PMATH_TAGMASK_NONDOUBLE | 0xFFFFF)`
- `#define PMATH_TAG_MAGIC (PMATH_TAGMASK_NONDOUBLE | 0x10000)`
- `#define PMATH_TAG_INT32 (PMATH_TAGMASK_NONDOUBLE | 0x20000)`
- `#define PMATH_TAG_STR0 (PMATH_TAGMASK_NONDOUBLE | 0x30000)`
- `#define PMATH_TAG_STR1 (PMATH_TAGMASK_NONDOUBLE | 0x40000)`
- `#define PMATH_TAG_STR2 (PMATH_TAGMASK_NONDOUBLE | 0x50000)`
- `#define PMATH_FROM_INT32(i) (PMATH_FROM_TAG(PMATH_TAG_INT32, (i)))`
- `#define PMATH_THREAD_KEY_PARSE SYMBOLS PMATH_FROM_TAG(PMATH_TAG_MAGIC, 252)`
- `#define PMATH_THREAD_KEY_PARSER ARGUMENTS PMATH_FROM_TAG(PMATH_TAG_MAGIC, 253)`
- `#define PMATH_ABORT_EXCEPTION PMATH_FROM_TAG(PMATH_TAG_MAGIC, 254)`
- `#define PMATH_STATIC_UNDEFINED { (((uint64_t)PMATH_TAG_MAGIC) << 32) | 255 }`
- `#define PMATH_STATIC_NULL { ((uint64_t)PMATH_TAGMASK_POINTER) << 32 }`

## Typedefs

- `typedef int pmath_type_t`

*The type or class of a pMath object.*

- `typedef int pmath_write_options_t`

*Options for `pmath_write()`.*

- typedef void(\* [pmath\\_proc\\_t](#))([pmath\\_t](#))  
*A simple procedure operating on an object.*
- typedef void(\* [pmath\\_param\\_proc\\_t](#))(void \*, [pmath\\_t](#))  
*A parameterized procedure operating on an object.*
- typedef [pmath\\_t](#)(\* [pmath\\_func\\_t](#))([pmath\\_t](#))  
*A simple function operating on an object and returning one.*
- typedef unsigned int(\* [pmath\\_hash\\_func\\_t](#))([pmath\\_t](#))  
*A hash function for an object.*
- typedef [pmath\\_bool\\_t](#)(\* [pmath\\_equal\\_func\\_t](#))([pmath\\_t](#), [pmath\\_t](#))  
*A comparison function for two objects.*
- typedef int(\* [pmath\\_compare\\_func\\_t](#))([pmath\\_t](#), [pmath\\_t](#))  
*A comparison function to determine the order of two objects.*

### Enumerations

- enum {  
[PMATH\\_TYPE\\_SHIFT\\_MP\\_FLOAT](#) = 0, [PMATH\\_TYPE\\_SHIFT\\_MP\\_INT](#),  
[PMATH\\_TYPE\\_SHIFT\\_QUOTIENT](#), [PMATH\\_TYPE\\_SHIFT\\_BIGSTRING](#),  
[PMATH\\_TYPE\\_SHIFT\\_SYMBOL](#), [PMATH\\_TYPE\\_SHIFT\\_EXPRESSION\\_](#)-  
[GENERAL](#),  
[PMATH\\_TYPE\\_SHIFT\\_EXPRESSION\\_GENERAL\\_PART](#), [PMATH\\_TYPE\\_](#)-  
[SHIFT\\_RESERVED\\_1](#),  
[PMATH\\_TYPE\\_SHIFT\\_CUSTOM](#), [PMATH\\_TYPE\\_SHIFT\\_COUNT](#) }
- enum {  
[PMATH\\_TYPE\\_MP\\_INT](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_MP\\_INT](#),  
[PMATH\\_TYPE\\_QUOTIENT](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_QUOTIENT](#),  
[PMATH\\_TYPE\\_MP\\_FLOAT](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_MP\\_FLOAT](#),  
[PMATH\\_TYPE\\_BIGSTRING](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_BIGSTRING](#),  
[PMATH\\_TYPE\\_SYMBOL](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_SYMBOL](#),  
[PMATH\\_TYPE\\_EXPRESSION\\_GENERAL](#) = 1 << [PMATH\\_TYPE\\_](#)-  
[SHIFT\\_EXPRESSION\\_GENERAL](#),  
[PMATH\\_TYPE\\_EXPRESSION\\_GENERAL\\_PART](#) = 1 << [PMATH\\_](#)-  
[TYPE\\_SHIFT\\_EXPRESSION\\_GENERAL\\_PART](#), [PMATH\\_TYPE\\_](#)-  
[EXPRESSION](#) = [PMATH\\_TYPE\\_EXPRESSION\\_GENERAL](#) | [PMATH\\_](#)-  
[TYPE\\_EXPRESSION\\_GENERAL\\_PART](#),  
[PMATH\\_TYPE\\_CUSTOM](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_CUSTOM](#) }

- enum {  
`PMATH_WRITE_OPTIONS_FULLEXPR` = 1 << 0, `PMATH_WRITE_OPTIONS_FULLSTR` = 1 << 1,  
`PMATH_WRITE_OPTIONS_FULLNAME` = 1 << 2, `PMATH_WRITE_OPTIONS_INPUTEXPR` = 1 << 3 }

### Functions

- `pmath_t PMATH_FROM_TAG` (uint32\_t tag, int32\_t value)
- `pmath_t PMATH_FROM_PTR` (void \*p)
- unsigned int `pmath_hash` (`pmath_t` obj)  
*Calculates an object's hash value.*
- int `pmath_compare` (`pmath_t` objA, `pmath_t` objB)  
*Compares two objects syntactically.*
- void `pmath_write` (`pmath_t` obj, `pmath_write_options_t` options, void(\*write)(void \*user, const uint16\_t \*data, int len), void \*user)  
*Write an object to a stream.*
- void `pmath_write_ex` (struct `pmath_write_ex_t` \*info, `pmath_t` obj)  
*Advanced function to write an object to a stream.*
- `pmath_bool_t pmath_is_evaluated` (`pmath_t` obj)  
*Test whether an object is already evaluated.*
- void `pmath_write_with_pagewidth` (`pmath_t` obj, `pmath_write_options_t` options, void(\*write)(void \*user, const uint16\_t \*data, int len), void \*user, int page\_width, int indentation\_width)  
*Write an object to a stream with a maximum line width.*

### Variables

- static PMATH\_UNUSED const `pmath_t PMATH_UNDEFINED`  
*Magic value to indicate unset variable values/...*
- static PMATH\_UNUSED const `pmath_t PMATH_NULL`  
*The NULL pointer. /\ in pMath.*

#### 9.4.1 Detailed Description

The basic class for all pMath objects.



pMath works on objects. They can be expressions (trees of pMath objects), symbols, numbers, strings or ‘magic objects’ (special integer values).

For efficiency reasons, 32 bit integers, double precision floating point values, short strings up to 2 characters and magic values are stored inline in the `pmath_t` struct. The struct size is only 8 bytes (= sizeof(double)) thanks to a technique called NaN-boxing.

This implementation could change in future version and/or on different architectures, so do not rely on it.

See also:

[Object Utility Functions](#)

#### 9.4.2 Define Documentation

**9.4.2.1** `#define PMATH_ABORT_EXCEPTION PMATH_FROM_TAG(PMATH_TAG_MAGIC, 254)`

**9.4.2.2** `#define PMATH_FROM_INT32(i) (PMATH_FROM_TAG(PMATH_TAG_INT32, (i)))`

**9.4.2.3** `#define PMATH_STATIC_NULL { ((uint64_t)PMATH_TAGMASK_POINTER) << 32 }`

**9.4.2.4** `#define PMATH_STATIC_UNDEFINED { (((uint64_t)PMATH_TAG_MAGIC) << 32) | 255 }`

**9.4.2.5** `#define PMATH_TAG_INT32 (PMATH_TAGMASK_NONDOUBLE | 0x20000)`

**9.4.2.6** `#define PMATH_TAG_INVALID (PMATH_TAGMASK_NONDOUBLE | 0xFFFFF)`

**9.4.2.7** `#define PMATH_TAG_MAGIC (PMATH_TAGMASK_NONDOUBLE | 0x10000)`

**9.4.2.8** `#define PMATH_TAG_STR0 (PMATH_TAGMASK_NONDOUBLE | 0x30000)`

**9.4.2.9** `#define PMATH_TAG_STR1 (PMATH_TAGMASK_NONDOUBLE | 0x40000)`

**9.4.2.10** `#define PMATH_TAG_STR2 (PMATH_TAGMASK_NONDOUBLE | 0x50000)`

**9.4.2.11** `#define PMATH_TAGMASK_BITCOUNT 12`

**9.4.2.12** `#define PMATH_TAGMASK_NONDOUBLE 0x7FF00000U`

**9.4.2.13** `#define PMATH_TAGMASK_POINTER 0xFFF00000U`

**9.4.2.14** `#define PMATH_THREAD_KEY_PARSERARGUMENTS PMATH_FROM_TAG(PMATH_TAG_MAGIC, 253)`

**9.4.2.15** `#define PMATH_THREAD_KEY_PARSESYMBOLS PMATH_FROM_TAG(PMATH_TAG_MAGIC, 252)`

### 9.4.3 Typedef Documentation

#### 9.4.3.1 `typedef int(* pmath_compare_func_t)(pmath_t, pmath_t)`

A comparison function to determine the order of two objects.

The return value is  $<0$ ,  $=0$  or  $>0$ , if the first argument is less, equal to or greater than the second respectively. Both arguments won't be destroyed by the function.

#### 9.4.3.2 `typedef pmath_bool_t(* pmath_equal_func_t)(pmath_t, pmath_t)`

A comparison function for two objects.

The return value is nonzero, if both objects equal and zero otherwise. Note that a `pmath_compare_func_t` cannot be cast to `pmath_equal_func_t`, because their return values have opposite meanings.

#### 9.4.3.3 `typedef pmath_t(* pmath_func_t)(pmath_t)`

A simple function operating on an object and returning one.

It depends on the context whether the argument is destroyed by the function or not.

#### 9.4.3.4 `typedef unsigned int(* pmath_hash_func_t)(pmath_t)`

A hash function for an object.

If two objects equal, their hash values equal.

#### 9.4.3.5 `typedef void(* pmath_param_proc_t)(void *, pmath_t)`

A parameterized procedure operating on an object.

It depends on the context whether the (second) argument is destroyed by the procedure or not.

#### 9.4.3.6 `typedef void(* pmath_proc_t)(pmath_t)`

A simple procedure operating on an object.

It depends on the context whether the argument is destroyed by the procedure or not.

#### 9.4.3.7 typedef int pmath\_type\_t

The type or class of a pMath object.

This is a bitset of the `PMATH_TYPE_XXX` constants:

- `PMATH_TYPE_MP_FLOAT`: The object is a floating point number with arbitrary precision. You can cast it to [pmath\\_float\\_t](#) and [pmath\\_number\\_t](#).
- `PMATH_TYPE_INTEGER`: The object is an integer value. You can cast it to [pmath\\_integer\\_t](#), [pmath\\_rational\\_t](#) and [pmath\\_number\\_t](#).
- `PMATH_TYPE_QUOTIENT`: The object is a reduced quotient of two integer values, where the denominator is never 0 or 1. You can cast quotient objects to [pmath\\_rational\\_t](#) and thus to [pmath\\_number\\_t](#) too.
- `PMATH_TYPE_BIGSTRING`: The object is a string. You can cast it to [pmath\\_string\\_t](#).
- `PMATH_TYPE_SYMBOL`: The object is a symbol. You can cast it to [pmath\\_symbol\\_t](#).
- `PMATH_TYPE_CUSTOM`: The object is a custom object. You can cast it to [pmath\\_custom\\_t](#).

#### 9.4.3.8 typedef int pmath\_write\_options\_t

Options for [pmath\\_write\(\)](#).

These options can be one or more of the following:

- `PMATH_WRITE_OPTIONS_FULLEXPR` All expressions are written in the form `f(a, b, ...)` without any syntactic sugar.

Supersedes `PMATH_WRITE_OPTIONS_INPUTEXPR`.

- `PMATH_WRITE_OPTIONS_FULLSTR` Strings are written with quotes and escape sequences.
- `PMATH_WRITE_OPTIONS_FULLNAME` Names are written with their full namespace path.

- `PMATH_WRITE_OPTIONS_INPUTEXPR` Expressions are written in a form that is valid pMath input.

Note that this does not automatically imply `PMATH_WRITE_OPTIONS_FULLSTR`.

#### 9.4.4 Enumeration Type Documentation

##### 9.4.4.1 anonymous enum

Enumerator:

*PMATH\_TYPE\_SHIFT\_MP\_FLOAT*  
*PMATH\_TYPE\_SHIFT\_MP\_INT*  
*PMATH\_TYPE\_SHIFT\_QUOTIENT*  
*PMATH\_TYPE\_SHIFT\_BIGSTRING*  
*PMATH\_TYPE\_SHIFT\_SYMBOL*  
*PMATH\_TYPE\_SHIFT\_EXPRESSION\_GENERAL*  
*PMATH\_TYPE\_SHIFT\_EXPRESSION\_GENERAL\_PART*  
*PMATH\_TYPE\_SHIFT\_RESERVED\_1*  
*PMATH\_TYPE\_SHIFT\_CUSTOM*  
*PMATH\_TYPE\_SHIFT\_COUNT*

##### 9.4.4.2 anonymous enum

Enumerator:

*PMATH\_TYPE\_MP\_INT*  
*PMATH\_TYPE\_QUOTIENT*  
*PMATH\_TYPE\_MP\_FLOAT*  
*PMATH\_TYPE\_BIGSTRING*  
*PMATH\_TYPE\_SYMBOL*  
*PMATH\_TYPE\_EXPRESSION\_GENERAL*  
*PMATH\_TYPE\_EXPRESSION\_GENERAL\_PART*  
*PMATH\_TYPE\_EXPRESSION*  
*PMATH\_TYPE\_CUSTOM*

## 9.4.4.3 anonymous enum

**Enumerator:***PMATH\_WRITE\_OPTIONS\_FULLEXPR**PMATH\_WRITE\_OPTIONS\_FULLSTR**PMATH\_WRITE\_OPTIONS\_FULLNAME**PMATH\_WRITE\_OPTIONS\_INPUTEXPR*

## 9.4.5 Function Documentation

9.4.5.1 `int pmath_compare (pmath_t objA, pmath_t objB)` [inherited]

Compares two objects syntactically.

**Parameters:**

*objA* The first object.

*objB* The second one.

**Returns:**

< 0 if *objA* is less than *objB*, == 0 if both are equal and > 0 if *objA* is greater than *objB*.

‘syntactically’ means that for two symbols X and Y, `pmath_compare(X, Y) < 0` even if X:=2 and Y:=1, because X appears before Y in the alphabet.

**Note:**

`pmath_equals(A, B)` might return FALSE although `pmath_compare(A, B) == 0` e.g. for an integer A and q floating point value B.

9.4.5.2 `pmath_t PMATH_FROM_PTR (void * p)`9.4.5.3 `pmath_t PMATH_FROM_TAG (uint32_t tag, int32_t value)`

**9.4.5.4 unsigned int pmath\_hash (pmath\_t *obj*)** [inherited]

Calculates an object's hash value.

**Parameters:**

*obj* The object.

**Returns:**

A hash value.

`pmath_equals(A, B)` implies `pmath_hash(A) == pmath_hash(B)`.

**9.4.5.5 pmath\_bool\_t pmath\_is\_evaluated (pmath\_t *obj*)** [inherited]

Test whether an object is already evaluated.

**Parameters:**

*obj* Any pMath object. It will *\*not\** be freed.

**Returns:**

TRUE if a call to `pmath_evaluate` would not change the object.

**9.4.5.6 void pmath\_write (pmath\_t *obj*, pmath\_write\_options\_t *options*, void (\*)(void \**user*, const uint16\_t \**data*, int *len*) *write*, void \**user*)** [inherited]

Write an object to a stream.

**Parameters:**

*obj* The object to be written.

*options* Some options defining the format.

*write* The stream's output function.

*user* The user-argument of `write` (e.g. the stream itself).

**See also:**

[pmath\\_utf8\\_writer](#)

**9.4.5.7** `void pmath_write_ex (struct pmath_write_ex_t * info, pmath_t obj)`  
 [inherited]

Advanced function to write an object to a stream.

**Parameters:**

*info* All the actual parameters.

*obj* The object to be written.

**See also:**

[pmath\\_write](#)

**9.4.5.8** `void pmath_write_with_pagewidth (pmath_t obj,  
 pmath_write_options_t options, void(*) (void *user, const uint16_t  
 *data, int len) write, void * user, int page_width, int indentation_width)`  
 [inherited]

Write an object to a stream with a maximum line width.

**Parameters:**

*obj* The object to be written.

*options* Some options defining the format.

*write* The stream's output function.

*user* The user-argument of write (e.g. the stream itself).

*page\_width* The page width. This should be at least 6.

*indentation\_width* The number of spaces to insert after every implicit line break.

If *page\_width* < 0, the global variable \$PageWidth is used.

**See also:**

[pmath\\_write](#)

## 9.4.6 Variable Documentation

**9.4.6.1** `PMATH_UNUSED const pmath_t PMATH_NULL` [static]

The NULL pointer. `/\` in pMath.



### 9.4.6.2 PMATH\_UNUSED const pmath\_t PMATH\_UNDEFINED [static]

Magic value to indicate unset variable values/...

## 9.5 Strings

String objects in pMath.

### Data Structures

- struct [pmath\\_cstr\\_writer\\_info\\_t](#)  
*Additional information for [pmath\\_utf8\\_writer\(\)](#) or [pmath\\_native\\_writer\(\)](#).*
- class [pmath\\_string\\_t](#)  
*The string class.*

### Defines

- `#define PMATH\_C\_STRING(cstr) pmath\_string\_insert\_latin1(PMATH_-  
NULL, 0, (cstr), -1)`  
*Short form to convert a C String to a pMath String.*
- `PMATH\_C\_STRING(cstr)`  
*Short form to convert a C String to a pMath String.*

### Typedefs

- typedef [pmath\\_t](#) [pmath\\_string\\_t](#)

### Functions

- void [pmath\\_utf8\\_writer](#) (void \*user, const uint16\_t \*data, int len)  
*A write function for [pmath\\_write\(\)](#) that converts to utf8.*
- void [pmath\\_native\\_writer](#) (void \*user, const uint16\_t \*data, int len)  
*A write function for [pmath\\_write\(\)](#) that converts to the current console encoding.*
- uint32\_t [pmath\\_char\\_from\\_name](#) (const char \*name)  
*Get a named character.*
- const char \* [pmath\\_char\\_to\\_name](#) (uint32\_t unichar)

*Get a character's name.*

- `pmath_string_t pmath_string_new` (int capacity)  
*Create an empty pMath String.*
- `pmath_string_t pmath_string_insert_latin1` (`pmath_string_t` str, int inspos, const char \*ins, int inslen)  
*Insert an Latin-1 encoded buffer into a pMath String.*
- `pmath_string_t pmath_string_from_utf8` (const char \*str, int len)  
*Convert an UTF-8 encoded buffer to a pMath String.*
- char \* `pmath_string_to_utf8` (`pmath_string_t` str, int \*result\_len)  
*Convert a pMath string to a zero-terminated UTF-8 string.*
- `pmath_string_t pmath_string_from_native` (const char \*str, int len)  
*Convert a string buffer in the current console character encoding to a pMath String.*
- char \* `pmath_string_to_native` (`pmath_string_t` str, int \*result\_len)  
*Convert a pMath string to a string in the current console character encoding.*
- `pmath_string_t pmath_string_insert_codepage` (`pmath_string_t` str, int inspos, const char \*ins, int inslen, const uint16\_t \*cp)  
*Insert a byte string into a pMath string using a translation array.*
- `pmath_string_t pmath_string_insert_ucs2` (`pmath_string_t` str, int inspos, const uint16\_t \*ins, int inslen)  
*Insert a UCS-2 buffer into a pMath String.*
- `pmath_string_t pmath_string_insert` (`pmath_string_t` str, int inspos, `pmath_string_t` ins)  
*Insert one pMath String into another pMath String.*
- `pmath_string_t pmath_string_concat` (`pmath_string_t` prefix, `pmath_string_t` postfix)  
*Concatenate two pMath Strings.*
- `pmath_string_t pmath_string_part` (`pmath_string_t` string, int start, int length)  
*Extract a substring of a pMath String.*
- const uint16\_t \* `pmath_string_buffer` (`pmath_string_t` \*string)  
*Get a string's buffer for reading.*
- int `pmath_string_length` (`pmath_string_t` string)  
*Get a string's length.*

- `pmath_bool_t pmath_string_equals_latin1 (pmath_string_t string, const char *latin1)`

*Compare a pMath string with a C string.*

### 9.5.1 Detailed Description

String objects in pMath.

pMath stores strings in UCS-2 format (like Java and Windows NT). But pMath strings are not zero terminated.

Do not confuse pMath String characters (`uint16_t`) with `wchar_t`: `sizeof(wchar_t)` differs on different Systems (Linux: 4 bytes, Windows: 2 bytes). So you cannot simply convert `wchar_t*` strings to pMath strings.

### 9.5.2 Define Documentation

#### 9.5.2.1 `PMATH_C_STRING(cstr)` [inherited]

Short form to convert a C String to a pMath String.

##### Parameters:

*cstr* A C String (zero-terminated char buffer).

##### Returns:

A pMath String representing the Latin-1 C string *cstr*.

This is a wrapper macro around `pmath_string_insert_latin1()`.

#### 9.5.2.2 `#define PMATH_C_STRING(cstr) pmath_string_insert_latin1(PMATH_NULL, 0, (cstr), -1)`

Short form to convert a C String to a pMath String.

##### Parameters:

*cstr* A C String (zero-terminated char buffer).

##### Returns:

A pMath String representing the Latin-1 C string *cstr*.

This is a wrapper macro around `pmath_string_insert_latin1()`.

### 9.5.3 Typedef Documentation

#### 9.5.3.1 typedef pmath\_t pmath\_string\_t

### 9.5.4 Function Documentation

#### 9.5.4.1 uint32\_t pmath\_char\_from\_name (const char \* *name*)

Get a named character.

**Parameters:**

*name* The ASCII-name of the character. e.g. "Sum"

**Returns:**

The character code or 0xFFFFFFFFU on error

#### 9.5.4.2 const char\* pmath\_char\_to\_name (uint32\_t *unichar*)

Get a character's name.

**Parameters:**

*unichar* A unicode character

**Returns:**

The ASCII-name or NULL if it is unnamed

#### 9.5.4.3 void pmath\_native\_writer (void \* *user*, const uint16\_t \* *data*, int *len*)

A *write* function for [pmath\\_write\(\)](#) that converts to the current console encoding.

This callback function is used like [pmath\\_utf8\\_writer\(\)](#).

#### 9.5.4.4 const uint16\_t \* pmath\_string\_buffer (pmath\_string\_t \* *string*) [inherited]

Get a string's buffer for reading.

**Parameters:**

*string* A pointer to a string.

**Returns:**

A pointer to the string's buffer. This buffer is guaranteed to be `pmath_string_length(str) * sizeof(uint16_t)` bytes long.

Do not forget that pMath strings are not zero-terminated.

#### 9.5.4.5 `pmath_string_t pmath_string_concat (pmath_string_t prefix, pmath_string_t postfix)` [inherited]

Concatenate two pMath Strings.

**Parameters:**

*prefix* A pMath String. It will be freed.

*postfix* A pMath String. It will be freed.

**Returns:**

PMATH\_NULL on failure or a pMath String that consists of *prefix* followed by *postfix*. You must destroy it.

If one of the two strings is PMATH\_NULL, the other string will be returned.

#### 9.5.4.6 `pmath_bool_t pmath_string_equals_latin1 (pmath_string_t string, const char * latin1)` [inherited]

Compare a pMath string with a C string.

**Parameters:**

*string* A string. It wont be freed.

*latin1* A C string (zero terminated).

**Returns:**

Whether the two string are equals.

This function is a short form for

```
tmp = PMATH_C_STRING(latin1);
result = pmath_equals(string, tmp);
pmath_unref(tmp);
```

#### 9.5.4.7 `pmath_string_t pmath_string_from_native (const char * str, int len)` [inherited]

Convert a string buffer in the current console character encoding to a pMath String.

**Parameters:**

*str* A byte string. It wont be freed.

*len* The byte-length of *ins* or -1 if it is zero-terminated.

**Returns:**

PMATH\_NULL on failure or a pMath String. You must destroy it.

#### 9.5.4.8 `pmath_string_t pmath_string_from_utf8 (const char * str, int len)` [inherited]

Convert an UTF-8 encoded buffer to a pMath String.

**Parameters:**

*str* A byte string. It wont be freed.

*len* The byte-length of *ins* or -1 if it is zero-terminated.

**Returns:**

PMATH\_NULL on failure or a pMath String. You must destroy it.

#### 9.5.4.9 `pmath_string_t pmath_string_insert (pmath_string_t str, int inspos, pmath_string_t ins)` [inherited]

Insert one pMath String into another pMath String.

**Parameters:**

*str* A pMath String or PMATH\_NULL. It will be freed.

*inspos* The position, at which *ins* should be inserted in *str*.

*ins* A pMath String or PMATH\_NULL. It will be freed.

**Returns:**

PMATH\_NULL on failure or a pMath String. You must destroy it.

**9.5.4.10** `pmath_string_t pmath_string_insert_codepage (pmath_string_t str, int inspos, const char * ins, int inslen, const uint16_t * cp)` [inherited]

Insert a byte string into a pMath string using a translation array.

**Parameters:**

- str* A pMath String or PMATH\_NULL. It will be freed.
- inspos* The position, at which *ins* should be inserted in *str*.
- ins* A byte string.
- inslen* The length of *ins* or -1 if it is zero-terminated.
- cp* An array of 256 uint16\_t values that are used to convert bytes to UCS-2 characters.

**Returns:**

PMATH\_NULL on failure or a pMath String. You must destroy it.

If *str* is PMATH\_NULL, it is assumed to be the empty string.

**9.5.4.11** `pmath_string_t pmath_string_insert_latin1 (pmath_string_t str, int inspos, const char * ins, int inslen)` [inherited]

Insert an Latin-1 encoded buffer into a pMath String.

**Parameters:**

- str* A pMath String or PMATH\_NULL. It will be freed.
- inspos* The position, at which *ins* should be inserted in *str*.
- ins* A byte string.
- inslen* The length of *ins* or -1 if it is zero-terminated.

**Returns:**

PMATH\_NULL on failure or a pMath String. You must destroy it.

If *str* is PMATH\_NULL, it is assumed to be the empty string. The result is equivalent to a call to [pmath\\_string\\_insert\\_codepage\(\)](#) with a codepage that translates every byte *b* to (uint16\_t)(unsigned char)b.

**9.5.4.12** `pmath_string_t pmath_string_insert_ucs2 (pmath_string_t str, int inspos, const uint16_t * ins, int inslen)` [inherited]

Insert a UCS-2 buffer into a pMath String.

**Parameters:**

- str* A pMath String or PMATH\_NULL. It will be freed.
- inspos* The position, at which *ins* should be inserted in *str*.
- ins* A uint16\_t string. This is *not* a wchar\_t string.
- inslen* The length of *ins* or -1 if it is zero-terminated.

**Returns:**

PMATH\_NULL on failure or a pMath String. You must destroy it.

If *str* is PMATH\_NULL, it is assumed to be the empty string.

**9.5.4.13 int pmath\_string\_length (pmath\_string\_t *string*)** [inherited]

Get a string's length.

**Parameters:**

- string* A string. It remains valid after the function call, so you have to destroy it manually.

**Returns:**

The length (in uint16\_t characters) of the string. It is never negative.

**9.5.4.14 pmath\_string\_t pmath\_string\_new (int *capacity*)** [inherited]

Create an empty pMath String.

**Parameters:**

- capacity* The initial capacity of the string. Must not be negative.

**Returns:**

A new pMath String or PMATH\_NULL on failure. You must destroy it.

**9.5.4.15 pmath\_string\_t pmath\_string\_part (pmath\_string\_t *string*, int *start*, int *length*)** [inherited]

Extract a substring of a pMath String.



**Parameters:**

*string* A pMath String. It will be freed.

*start* the substring's start index.

*length* the substring's length or -1 for the whole substring beginning at start.

**Returns:**

PMATH\_NULL on failure or a pMath String.

If start or start+length are out of bounds, they will be truncated, the the resulting string's length is not necessarily length.

#### 9.5.4.16 `char * pmath_string_to_native (pmath_string_t str, int * result_len)` [inherited]

Convert a pMath string to a string in the current console character encoding.

**Parameters:**

*str* A pMath string. It won't be freed.

*result\_len* Position, where the string length of the returned buffer may be stored.

**Returns:**

A zero terminated string or PMATH\_NULL on error. You have to free the memory with `pmath_mem_free(result, *size_ptr)`.

**Note:**

pMath strings may contain embedded `'\0'`, but C strings may not. However, the conversion is done to the whole string even though your C functions will only *see* the content up to the first `'\0'`.

#### 9.5.4.17 `char * pmath_string_to_utf8 (pmath_string_t str, int * result_len)` [inherited]

Convert a pMath string to a zero-terminated UTF-8 string.

**Parameters:**

*str* A pMath string. It won't be freed.

*result\_len* Position, where the string length of the returned buffer may be stored.

**Returns:**

A zero-terminated UTF-8 string or PMATH\_NULL on error. You have to free the memory with `pmath_mem_free(result, *size_ptr)`.

**Note:**

pMath strings may contain embedded `'\0'`, but C strings may not. However, the conversion is done to the whole string even though your C functions will only *see* the content up to the first `'\0'`.

**9.5.4.18 void pmath\_utf8\_writer (void \*user, const uint16\_t \*data, int len)**

A *write* function for [pmath\\_write\(\)](#) that converts to utf8.

[pmath\\_write\(\)](#) writes output as utf16/ucs2. This function can be used to convert to utf8 on the fly. The *user* parameter to `pmath_write` must point to a [pmath\\_cstr\\_writer\\_info\\_t](#).

Here is an example on how to use it:

```
void my_utf8_output(FILE *f, const char *str){
    fprintf(f, "%s", str);
}

...

pmath_cstr_writer_info_t info;
info.write_cstr = (void(*) (void*, const char*))my_utf8_output;
info.user = stdout; // will be first argument of my_utf8_output

pmath_print(some_object, some_options, pmath_utf8_writer, &info);
```

**9.6 Symbols**

Symbol objects in pMath.

**Data Structures**

- class [pmath\\_symbol\\_t](#)  
*The Symbol class.*

**Typedefs**

- typedef [pmath\\_t](#) [pmath\\_symbol\\_t](#)
- typedef int [pmath\\_symbol\\_attributes\\_t](#)  
*The (bitset) type of symbol attributes.*

## Enumerations

- enum {
  - PMATH\_SYMBOL\_ATTRIBUTE\_PROTECTED = 1 << 0, PMATH\_SYMBOL\_ATTRIBUTE\_HOLDFIRST = 1 << 1,
  - PMATH\_SYMBOL\_ATTRIBUTE\_HOLDREST = 1 << 2, PMATH\_SYMBOL\_ATTRIBUTE\_HOLDALL = PMATH\_SYMBOL\_ATTRIBUTE\_HOLDFIRST | PMATH\_SYMBOL\_ATTRIBUTE\_HOLDREST,
  - PMATH\_SYMBOL\_ATTRIBUTE\_SYMMETRIC = 1 << 3, PMATH\_SYMBOL\_ATTRIBUTE\_ASSOCIATIVE = 1 << 4,
  - PMATH\_SYMBOL\_ATTRIBUTE\_NHOLDFIRST = 1 << 5, PMATH\_SYMBOL\_ATTRIBUTE\_NHOLDREST = 1 << 6,
  - PMATH\_SYMBOL\_ATTRIBUTE\_NHOLDALL = PMATH\_SYMBOL\_ATTRIBUTE\_NHOLDFIRST | PMATH\_SYMBOL\_ATTRIBUTE\_NHOLDREST, PMATH\_SYMBOL\_ATTRIBUTE\_TEMPORARY = 1 << 7,
  - PMATH\_SYMBOL\_ATTRIBUTE\_LISTABLE = 1 << 8, PMATH\_SYMBOL\_ATTRIBUTE\_DEEPHOLDALL = 1 << 9,
  - PMATH\_SYMBOL\_ATTRIBUTE\_HOLDALLCOMPLETE = 1 << 10, PMATH\_SYMBOL\_ATTRIBUTE\_ONEIDENTITY = 1 << 11,
  - PMATH\_SYMBOL\_ATTRIBUTE\_THREADLOCAL = 1 << 12, PMATH\_SYMBOL\_ATTRIBUTE\_NUMERICFUNCTION = 1 << 13,
  - PMATH\_SYMBOL\_ATTRIBUTE\_READPROTECTED = 1 << 14, PMATH\_SYMBOL\_ATTRIBUTE\_SEQUENCEHOLD = 1 << 15,
  - PMATH\_SYMBOL\_ATTRIBUTE\_REMOVED = 1 << 16, PMATH\_SYMBOL\_ATTRIBUTE\_DEFINITEFUNCTION = 1 << 17 }

## Functions

- void `pmath_symbol_remove` (`pmath_symbol_t` symbol)
 

*Remove a symbol completely from the system.*
- `pmath_symbol_t` `pmath_symbol_iter_next` (`pmath_symbol_t` old)
 

*Iterate through the global symbol table.*
- `pmath_symbol_t` `pmath_symbol_get` (`pmath_string_t` name, `pmath_bool_t` create)
 

*Get a symbol by its fully qualified name.*
- `pmath_symbol_t` `pmath_symbol_create_temporary` (`pmath_string_t` name, `pmath_bool_t` unique)
 

*Create a new temporary symbol.*
- `pmath_symbol_t` `pmath_symbol_find` (`pmath_string_t` name, `pmath_bool_t` create)
 

*Find a symbol by its fully qualified name.*

*Find a symbol in the current namespace search path.*

- `pmath_string_t pmath_symbol_name (pmath_symbol_t symbol)`  
*Get a symbol's name.*
- `pmath_symbol_attributes_t pmath_symbol_get_attributes (pmath_symbol_t symbol)`  
*Get a symbol's attributes.*
- `void pmath_symbol_set_attributes (pmath_symbol_t symbol, pmath_symbol_attributes_t attr)`  
*Set a symbol's attributes.*
- `pmath_t pmath_symbol_get_value (pmath_symbol_t symbol)`  
*Get a symbol's value.*
- `void pmath_symbol_set_value (pmath_symbol_t symbol, pmath_t value)`  
*Set a symbol's value.*
- `void pmath_symbol_synchronized (pmath_symbol_t symbol, pmath_callback_t callback, void *data)`  
*Execute a function synchronized to a symbol.*
- `void pmath_symbol_update (pmath_symbol_t symbol)`  
*Update a symbol manually.*

### 9.6.1 Detailed Description

Symbol objects in pMath.

### 9.6.2 Typedef Documentation

#### 9.6.2.1 `typedef int pmath_symbol_attributes_t`

The (bitset) type of symbol attributes.

A pMath symbol (here called 'sym') can have one or more of the following values (concatenated with "|"):

- `PMATH_SYMBOL_ATTRIBUTE_PROTECTED`  
Any assignment to sym will fail.
- `PMATH_SYMBOL_ATTRIBUTE_HOLDFIRST`  
When evaluating 'sym(a,b,...)', the first argument (a) will not be evaluated automatically.

- `PMATH_SYMBOL_ATTRIBUTE_HOLDREST`  
When evaluating `'sym(a,b,...)'` all the arguments `b,...` will not be evaluated automatically.
- `PMATH_SYMBOL_ATTRIBUTE_HOLDALL`  
combines `HOLDFIRST` and `HOLDREST`.
- `PMATH_SYMBOL_ATTRIBUTE_SYMMETRIC`  
An expression `'sym(a,b,...)'` will be sorted automatically and thus `sym(a,b) = sym(b,a)`.
- `PMATH_SYMBOL_ATTRIBUTE_ASSOCIATIVE`  
An expression `'sym(...,sym(a,...,z),...)'` will be flattened automatically to `sym(...,a,...,z,...)`.
- `PMATH_SYMBOL_ATTRIBUTE_NHOLDFIRST`  
The first argument (`a`) of `'sym(a,b,...)'` will not be affected by `Approximate(sym(a,b,...))`.
- `PMATH_SYMBOL_ATTRIBUTE_NHOLDREST`  
All the argument `'b,...'` in `'sym(a,b,...)'` will not be affected by `Approximate(sym(a,b,...))`.
- `PMATH_SYMBOL_ATTRIBUTE_NHOLDALL`  
combines `NHOLDFIRST` and `NHOLDREST`.
- `PMATH_SYMBOL_ATTRIBUTE_TEMPORARY`  
The symbol `sym` will be deleted immediately when it is no longer referenced. It will be freed automatically (but not immediately), when there is no external reference to the symbol (just the symbol's own function definitions/...).
- `PMATH_SYMBOL_ATTRIBUTE_LISTABLE`  
Any expression `sym(...)` will be threaded automatically over lists. (e.g. `{a,b} + {c,d}` becomes `{a+c, b+d}`).
- `PMATH_SYMBOL_ATTRIBUTE_DEEPHOLDALL` The arguments `'a,...'` in an expression `'sym(...)(a,...)'` will not be evaluated automatically.
- `PMATH_SYMBOL_ATTRIBUTE_HOLDALLCOMPLETE`  
Like `HOLDALL`, but in an expression `'sym(a,b,...)'` all arguments (`a,b,...`) wont be touched even if they have the form `'eval(...)'`. Additionally, rules for `sym` defined in one of its arguments (e.g. `'a: sym(a):= "hi"'`) wont be used.
- `PMATH_SYMBOL_ATTRIBUTE_ONEIDENTITY`  
Used for pattern matching (in combination with `ASSOCIATIVE`) to say that `'sym(x)'` matches `x`. Note that it does not automatically evaluate `'sym(x)'` to `x`.
- `PMATH_SYMBOL_ATTRIBUTE_THREADLOCAL`  
The symbol's value is local to the current thread. That means, an assignment to `sym` in one thread wont affect it in another thread.

- `PMATH_SYMBOL_ATTRIBUTE_NUMERICFUNCTION`  
‘`sym(x,...)`’ is numeric if all the arguments are numeric.
- `PMATH_SYMBOL_ATTRIBUTE_READPROTECTED` ‘`??sym`’ wont print out the value/function definitions for `sym`.
- `PMATH_SYMBOL_ATTRIBUTE_SEQUENCEHOLD` `Sequence(...)` wont be sliced when it appears as an argument to ‘`sym(...)`’
- `PMATH_SYMBOL_ATTRIBUTE_REMOVED` The symbol was removed, but there are pending references to it.
- `PMATH_SYMBOL_ATTRIBUTE_DEFINITEFUNCTION`  
‘`sym(ConditionalExpression(arg, cond))`’ becomes ‘`ConditionalExpression(sym(arg), cond)`’

### 9.6.2.2 `typedef pmath_t pmath_symbol_t`

## 9.6.3 Enumeration Type Documentation

### 9.6.3.1 anonymous enum

#### Enumerator:

`PMATH_SYMBOL_ATTRIBUTE_PROTECTED`  
`PMATH_SYMBOL_ATTRIBUTE_HOLDFIRST`  
`PMATH_SYMBOL_ATTRIBUTE_HOLDREST`  
`PMATH_SYMBOL_ATTRIBUTE_HOLDALL`  
`PMATH_SYMBOL_ATTRIBUTE_SYMMETRIC`  
`PMATH_SYMBOL_ATTRIBUTE_ASSOCIATIVE`  
`PMATH_SYMBOL_ATTRIBUTE_NHOLDFIRST`  
`PMATH_SYMBOL_ATTRIBUTE_NHOLDREST`  
`PMATH_SYMBOL_ATTRIBUTE_NHOLDALL`  
`PMATH_SYMBOL_ATTRIBUTE_TEMPORARY`  
`PMATH_SYMBOL_ATTRIBUTE_LISTABLE`  
`PMATH_SYMBOL_ATTRIBUTE_DEEPHOLDALL`  
`PMATH_SYMBOL_ATTRIBUTE_HOLDALLCOMPLETE`  
`PMATH_SYMBOL_ATTRIBUTE_ONEIDENTITY`  
`PMATH_SYMBOL_ATTRIBUTE_THREADLOCAL`  
`PMATH_SYMBOL_ATTRIBUTE_NUMERICFUNCTION`

*PMATH\_SYMBOL\_ATTRIBUTE\_READPROTECTED*  
*PMATH\_SYMBOL\_ATTRIBUTE\_SEQUENCEHOLD*  
*PMATH\_SYMBOL\_ATTRIBUTE\_REMOVED*  
*PMATH\_SYMBOL\_ATTRIBUTE\_DEFINITEFUNCTION*

#### 9.6.4 Function Documentation

##### 9.6.4.1 `pmath_symbol_t pmath_symbol_create_temporary (pmath_string_t name, pmath_bool_t unique)` [inherited]

Create a new temporary symbol.

###### Parameters:

*name* The base name of the temporary symbol. It will be freed.  
*unique* Whether to add a unique number to the symbol name.

###### Returns:

A new pMath Symbol. You must destroy it with [pmath\\_unref\(\)](#). It has the Temporary attribute.

the name of the returned symbol is of the form `name$nnn` (or `name$` if `unique` is false)

If `name` already has the form `"sym$nnn"` or `"sym$"`, the function acts as if `name` would be simply `"sym"`.

If there already exists a symbol with the generated name, that symbol will be returned and its attributes will be set to Temporary before.

##### 9.6.4.2 `pmath_symbol_t pmath_symbol_find (pmath_string_t name, pmath_bool_t create)` [inherited]

Find a symbol in the current namespace search path.

###### Parameters:

*name* The symbol's name. It will be freed.  
*create* Whether to create a new symbol, if none was found.

###### Returns:

`PMATH_NULL` or a symbol called `name` that must be destroyed with [pmath\\_unref\(\)](#).

#### 9.6.4.3 `pmath_symbol_t` `pmath_symbol_get` (`pmath_string_t` *name*, `pmath_bool_t` *create*) [inherited]

Get a symbol by its fully qualified name.

##### Parameters:

*name* The symbol's name including its namespace. It will be freed.

*create* Whether to create a new symbol, if none was found.

##### Returns:

PMATH\_NULL or a symbol called *name* that must be destroyed with [pmath\\_unref\(\)](#).

#### 9.6.4.4 `pmath_symbol_attributes_t` `pmath_symbol_get_attributes` (`pmath_symbol_t` *symbol*) [inherited]

Get a symbol's attributes.

##### Parameters:

*symbol* A pMath symbol. It wont be freed.

##### Returns:

The symbol's attributes.

#### 9.6.4.5 `pmath_t` `pmath_symbol_get_value` (`pmath_symbol_t` *symbol*) [inherited]

Get a symbol's value.

### Deprecated

##### Parameters:

*symbol* A pMath symbol.

##### Returns:

The symbol's value. You must free it with [pmath\\_unref\(\)](#). Note that not every object is evaluatable (e.g. [Custom Objects](#) ).



#### 9.6.4.6 `pmath_symbol_t pmath_symbol_iter_next (pmath_symbol_t old)`

Iterate through the global symbol table.

**Parameters:**

*old* The previous symbol. It will be freed.

**Returns:**

The next symbol.

To actually iterate through the whole list, use the following pattern:

```
pmath_symbol_t iter = pmath_ref(PMATH_SYMBOL_LIST);
do{
    ... loop body here ...

    iter = pmath_symbol_iter_next(iter);
}while(iter && !pmath_same(iter, PMATH_SYMBOL_LIST));
pmath_unref(iter);
```

#### 9.6.4.7 `pmath_string_t pmath_symbol_name (pmath_symbol_t symbol)` [inherited]

Get a symbol's name.

**Parameters:**

*symbol* A pMath symbol.

**Returns:**

The name of the symbol. You must destroy it with [pmath\\_unref\(\)](#).

#### 9.6.4.8 `void pmath_symbol_remove (pmath_symbol_t symbol)`

Remove a symbol completely from the system.

**Parameters:**

*symbol* a pMath symbol. It will be freed.

Symbols with attribute protected wont be removed.

This function walks through the internal list of all known symbols and replaces any occurencies with ‘Symbol("name")’.

There might be more references (e.g. on the stack or in other thread’s local variable tables), so it is possible that the symbol still exists in the system.

Note that all builtin symbols (the PMATH\_SYMBOL\_XXX) are also referenced in a seperate list and so cannot be removed completely from the system. However, their appearances in all other places will be removed. So this is a very dangerous function.

**9.6.4.9 void pmath\_symbol\_set\_attributes (pmath\_symbol\_t *symbol*,  
pmath\_symbol\_attributes\_t *attr*)** [inherited]

Set a symbol’s attributes.

**Parameters:**

*symbol* A pMath symbol. It wont be freed.

*attr* The new attributes.

**9.6.4.10 void pmath\_symbol\_set\_value (pmath\_symbol\_t *symbol*, pmath\_t  
*value*)** [inherited]

Set a symbol’s value.

**Deprecated**

**Parameters:**

*symbol* A pMath symbol. It wont be freed

*value* The new value. It will be freed.

This function ignores the Protected-attribute. You should only use it during symbol initialization and/or when you want to store non-evaluatable values in a symbol. In all other cases, evaluate an expression with the head PMATH\_SYMBOL\_ASSIGN or PMATH\_SYMBOL\_ASSIGNDELAYED.

**9.6.4.11 void pmath\_symbol\_synchronized (pmath\_symbol\_t *symbol*,  
pmath\_callback\_t *callback*, void \* *data*)** [inherited]

Execute a function synchronized to a symbol.

## Deprecated

### Parameters:

- symbol* The symbol to lock. It wont be freed.
- callback* The function to be executed when the symbol is locked.
- data* A pointer that will be passed to callback.

### See also:

[Multithreading with pMath](#)

#### 9.6.4.12 void pmath\_symbol\_update (pmath\_symbol\_t *symbol*) [inherited]

Update a symbol manually.

### Parameters:

- symbol* A pMath symbol. It wont be freed.

You normally do not have to call this, since every change in a symbol yields an update. But there are some situations where you might to update it manually. The update mechanism is an optimization. Any expresseion or symbol, that is up to date while evaluation, wont be evaluated again. After an evaluation, expressions are updated automatically.

## 9.7 C++ Binding

### Data Structures

- class [Expr](#)  
*A wrapper for [pmath\\_t](#) and drived types.*
- class [String](#)  
*A wrapper for [pmath\\_string\\_t](#).*
- class [Gather](#)  
*Utility class for emitting and gathering expressions/building lists.*
- class [File](#)  
*A wrapper for pMath file objects (data streams).*
- class [BinaryFile](#)

*A wrapper for pMath binary file objects (byte data streams).*

- class [ReadableBinaryFile](#)

*A wrapper for readable pMath binary file objects (byte data streams).*

- class [WritableBinaryFile](#)

*A wrapper for writeable pMath binary file objects (byte data streams).*

- class [TextFile](#)

*A wrapper for pMath text file objects (byte data streams).*

- class [ReadableTextFile](#)

*A wrapper for pMath readable text file objects (byte data streams).*

- class [WritableTextFile](#)

*A wrapper for pMath writeable text file objects (byte data streams).*

- class [UserStream](#)

*Abstract base class for C++ callbacks used as pMath files.*

- class [BinaryUserStream](#)

*Abstract base class for C++ callbacks used as pMath binary files.*

- class [TextUserStream](#)

*Abstract base class for C++ callbacks used as pMath text files.*

## Namespaces

- namespace [pmath](#)

*Provides the C++ binding.*

### 9.7.1 Detailed Description

There exists a thin layer to easily use pMath with C++. This is usable preferable over the C API because it handles reference counting/type checking automatically and leads to less "boilerplate code".

To use it, simply `#include <pmath-cpp.h>`. The classes are in the [pmath](#) namespace.

This namespace also contains numerous helper functions to easily construct expression trees. None of the classes and functions generate C++ exceptions, they are all fault tolerant (in contrast to most of the plain C API).

## 9.8 Parsing Code

Translating pMath code or boxes to pMath objects.

### Data Structures

- class `pmath_span_array_t`  
*Internal flat representaion of spans.*
- class `pmath_span_t`  
*Represents a span in a `span-array`.*

### Defines

- `#define PMATH_RUN(code)`  
*Execute some pMath code.*
- `#define PMATH_RUN_ARGS(code, format,...)`  
*Execute some pMath code with arguments.*
- `#define PMATH_CHAR_INVISIBLECALL 0x2061`  
*The Function application character.*
- `#define PMATH_CHAR_VECTOR 0x21C0`  
*The arrow above names to indicate a vector.*
- `#define PMATH_CHAR_RULE 0x2192`  
*The "  $\rightarrow$  " operator.*
- `#define PMATH_CHAR_RULEDELAYED 0x29F4`  
*The "  $\cdot>$  " operator.*
- `#define PMATH_CHAR_ASSIGN 0x2254`  
*The "  $:=$  " operator.*
- `#define PMATH_CHAR_ASSIGNDELAYED 0x2A74`  
*The "  $::=$  " operator.*
- `#define PMATH_CHAR_INTEGRAL_D 0x2146`  
*The integral "d".*
- `#define PMATH_CHAR_PIECEWISE 0xF361`  
*The left curly bracket for cases.*
- `#define PMATH_CHAR_ALIASDELIMITER 0xF764`

*The character inserted by Richmath with ESCAPE or CAPSLOCK.*

- #define `PMATH_CHAR_ALIASINDICATOR` 0xF768  
*A character that looks like PMATH\_CHAR\_ALIASDELIMITER but has no effect.*
- #define `PMATH_CHAR_LEFT_BOX` 0xFFFF9  
*Start of box code inside a string.*
- #define `PMATH_CHAR_RIGHT_BOX` 0xFFFFB  
*End of box code inside a string.*
- #define `PMATH_CHAR_BOX` 0xFDD0  
*Represents a box.*
- #define `PMATH_CHAR_PLACEHOLDER` 0xFFFFD  
*The Placeholder character. In richmath, type CAPSLOCK pl CAPSLOCK to insert it.*

### Typedefs

- typedef struct `_pmath_span_array_t` `pmath_span_array_t`
- typedef struct `_pmath_span_t` `pmath_span_t`

### Enumerations

- enum `pmath_token_t` {  
`PMATH_TOK_NONE`, `PMATH_TOK_SPACE`,  
`PMATH_TOK_DIGIT`, `PMATH_TOK_STRING`,  
`PMATH_TOK_NAME`, `PMATH_TOK_NAME2`,  
`PMATH_TOK_BINARY_LEFT`, `PMATH_TOK_BINARY_RIGHT`,  
`PMATH_TOK_BINARY_LEFT_AUTOARG`, `PMATH_TOK_BINARY_LEFT_OR_PREFIX`,  
`PMATH_TOK_NARY`, `PMATH_TOK_NARY_AUTOARG`,  
`PMATH_TOK_NARY_OR_PREFIX`, `PMATH_TOK_POSTFIX_OR_PREFIX`,  
`PMATH_TOK_PREFIX`, `PMATH_TOK_POSTFIX`,  
`PMATH_TOK_CALL`, `PMATH_TOK_LEFTCALL`,  
`PMATH_TOK_LEFT`, `PMATH_TOK_RIGHT`,  
`PMATH_TOK_PRETEXT`, `PMATH_TOK_ASSIGNTAG`,  
`PMATH_TOK_PLUSPLUS`, `PMATH_TOK_COLON`,  
`PMATH_TOK_TILDES`, `PMATH_TOK_SLOT`,  
`PMATH_TOK_QUESTION`, `PMATH_TOK_INTEGRAL`,  
`PMATH_TOK_COMMENTEND` }

*Token classes known in the pMath language.*

- enum {  
PMATH\_PREC\_ANY = 0, PMATH\_PREC\_SEQ = 10,  
PMATH\_PREC\_EVAL = 20, PMATH\_PREC\_ASS = 30,  
PMATH\_PREC\_MODY = 40, PMATH\_PREC\_LAZY = 50,  
PMATH\_PREC\_FUNC = 60, PMATH\_PREC\_REPL = 80,  
PMATH\_PREC\_RULE = 90, PMATH\_PREC\_MAP = 100,  
PMATH\_PREC\_STR = 110, PMATH\_PREC\_COND = 120,  
PMATH\_PREC\_ALT = 130, PMATH\_PREC\_OR = 150,  
PMATH\_PREC\_XOR = 155, PMATH\_PREC\_AND = 160,  
PMATH\_PREC\_ARROW = 170, PMATH\_PREC\_REL = 180,  
PMATH\_PREC\_UNION = 190, PMATH\_PREC\_ISECT = 200,  
PMATH\_PREC\_RANGE = 210, PMATH\_PREC\_ADD = 220,  
PMATH\_PREC\_CIRCADD = 230, PMATH\_PREC\_PLUMI = 240,  
PMATH\_PREC\_CIRCMUL = 250, PMATH\_PREC\_MUL = 260,  
PMATH\_PREC\_DIV = 270, PMATH\_PREC\_MIDDOT = 280,  
PMATH\_PREC\_CROSS = 290, PMATH\_PREC\_MUL2 = 300,  
PMATH\_PREC\_POW = 310, PMATH\_PREC\_FAC = 320,  
PMATH\_PREC\_APL = 330, PMATH\_PREC\_REPEAT = 340,  
PMATH\_PREC\_TEST = 350, PMATH\_PREC\_INC = 360,  
PMATH\_PREC\_CALL = 400, PMATH\_PREC\_DIFF = 410,  
PMATH\_PREC\_PRIM = 1000 }

## Functions

- `pmath_span_array_t * pmath_spans_from_string (pmath_string_t *code, pmath_string_t(*line_reader)(void *), pmath_bool_t(*subsuperscriptbox_at_index)(int, void *), pmath_string_t(*underoverscriptbox_at_index)(int, void *), void(*error)(pmath_string_t, int, void *, pmath_bool_t), void *data)`  
*Parses pMath code to a span array.*
- `pmath_t pmath_boxes_from_spans (pmath_span_array_t *spans, pmath_string_t string, pmath_bool_t parseable, pmath_t(*box_at_index)(int, void *), void *data)`  
*Convert a span-array with the according code to boxed form.*
- `pmath_span_array_t * pmath_spans_from_boxes (pmath_t boxes, pmath_string_t *result_string, void(*make_box)(int, pmath_t, void *), void *data)`  
*Convert boxed form back to span-array and code.*
- `pmath_token_t pmath_token_analyse (const uint16_t *str, int len, int *prec)`

*Analyse a token.*

- `int pmath_token_prefix_precedence (const uint16_t *str, int len, int defprec)`  
*Give the prefix operator precedence for a token.*
- `static PMATH_INLINE pmath_bool_t pmath_token_maybe_first (pmath_token_t tok)`  
*Test whether a token may be the first token in a subexpression.*
- `static PMATH_INLINE pmath_bool_t pmath_token_maybe_rest (pmath_token_t tok)`  
*Test whether a token need not be the first token in a subexpression.*
- `static PMATH_INLINE pmath_bool_t pmath_char_is_left (uint16_t ch)`  
*Test if a unicode character is a left bracket.*
- `static PMATH_INLINE uint16_t pmath_right_fence (uint16_t left)`  
*Get the corresponding right bracket to a given left bracket or 0.*
- `static PMATH_INLINE pmath_bool_t pmath_char_is_right (uint16_t ch)`  
*Test if a unicode character is a right bracket.*
- `static PMATH_INLINE pmath_bool_t pmath_char_is_name (uint16_t ch)`  
*Test if a unicode character can be the start of an identifier/name.*
- `static PMATH_INLINE pmath_bool_t pmath_char_is_integral (uint16_t ch)`  
*Test if a unicode character is an integral.*
- `static PMATH_INLINE pmath_bool_t pmath_token_maybe_bigop (pmath_token_t tok)`  
*Test if a token may be a big operator.*
- `static PMATH_INLINE pmath_bool_t pmath_char_maybe_bigop (uint16_t ch)`  
*Test if a unicode character may be a big operation, e.g. Union, Sum.*
- `static PMATH_INLINE pmath_bool_t pmath_char_is_digit (uint16_t ch)`  
*Test if a unicode character is a digit '0' - '9'.*
- `static PMATH_INLINE pmath_bool_t pmath_char_is_36digit (uint16_t ch)`  
*Test if a unicode character is a base-36 digit '0' - '9', 'a' - 'z', 'A' - 'Z'.*
- `static PMATH_INLINE pmath_bool_t pmath_char_is_basedigit (int base, uint16_t ch)`  
*Test if in a given base, a unicode character is a digit.*
- `static PMATH_INLINE pmath_bool_t pmath_char_is_hexdigit (uint16_t ch)`



*Test if a unicode character is a hexadecimal digit.*

- void `pmath_span_array_free` (`pmath_span_array_t` \*spans)  
*Destroy a span-array and all its spans.*
- int `pmath_span_array_length` (`pmath_span_array_t` \*spans)  
*Get a span-array's length.*
- `pmath_bool_t` `pmath_span_array_is_token_end` (`pmath_span_array_t` \*spans, int pos)  
*Test the token-end-flag at an index.*
- `pmath_bool_t` `pmath_span_array_is_operand_start` (`pmath_span_array_t` \*spans, int pos)  
*Test the operator-start-flag at an index.*
- `pmath_span_t` \* `pmath_span_at` (`pmath_span_array_t` \*spans, int pos)  
*Get a span starting at an index.*
- `pmath_span_t` \* `pmath_span_next` (`pmath_span_t` \*span)  
*Get the next-shorter span starting at the same position.*
- int `pmath_span_end` (`pmath_span_t` \*span)  
*Get end of a span.*
- `pmath_t` `pmath_string_expand_boxes` (`pmath_string_t` s)  
*Expand a string that contains boxes to a list of Strings and Boxes.*
- `pmath_t` `pmath_parse_string` (`pmath_string_t` code)  
*Parse a string to an expression.*
- `pmath_t` `pmath_parse_string_args` (const char \*code, const char \*format,...)  
*Parse a string with additional arguments to an expression.*

### 9.8.1 Detailed Description

Translating pMath code or boxes to pMath objects.

The pMath language supports standard mathematical notation (such as sums, ...). Therefore, code can be given as Boxes.

### 9.8.2 Example

The boxed form of  $\sum_{i=1}^n f(i)$  is

```
{UnderoverscriptBox("[Sum]", {"i", "=", "1"}, "n"),
{"f", "(", "i", ")"}]}
```

or

```
{{"[Sum]", SubsuperscriptBox({"i", "=", "1"}, "n")},
{"f", "(", "i", ")"}]}
```

(\[Sum] is the Unicode character U+2211: "N-ARY SUMMATION").

It will be translated to `HoldComplete(Sum(f(i), i->1..n))` by `System`MakeExpression`.

Front-ends have to convert their own representation of the code to the boxed form before parsing.

### 9.8.3 Define Documentation

#### 9.8.3.1 `#define PMATH_CHAR_ALIASDELIMITER 0xF764`

The character inserted by Richmath with ESCAPE or CAPSLOCK.

#### 9.8.3.2 `#define PMATH_CHAR_ALIASINDICATOR 0xF768`

A character that looks like `PMATH_CHAR_ALIASDELIMITER` but has no effect.

#### 9.8.3.3 `#define PMATH_CHAR_ASSIGN 0x2254`

The `":="` operator.

#### 9.8.3.4 `#define PMATH_CHAR_ASSIGNDELAYED 0x2A74`

The `"::="` operator.

#### 9.8.3.5 `#define PMATH_CHAR_BOX 0xFDD0`

Represents a box.

#### 9.8.3.6 `#define PMATH_CHAR_INTEGRAL_D 0x2146`

The integral "d".

**9.8.3.7 #define PMATH\_CHAR\_INVISIBLECALL 0x2061**

The Function application character.

**9.8.3.8 #define PMATH\_CHAR\_LEFT\_BOX 0xFFFF9**

Start of box code inside a string.

**9.8.3.9 #define PMATH\_CHAR\_PIECEWISE 0xF361**

The left curly bracket for cases.

**9.8.3.10 #define PMATH\_CHAR\_PLACEHOLDER 0xFFFFD**

The Placeholder character. In richmath, type CAPSLOCK pl CAPSLOCK to insert it.

**9.8.3.11 #define PMATH\_CHAR\_RIGHT\_BOX 0xFFFFB**

End of box code inside a string.

**9.8.3.12 #define PMATH\_CHAR\_RULE 0x2192**

The "  $\rightarrow$  " operator.

**9.8.3.13 #define PMATH\_CHAR\_RULEDELAYED 0x29F4**

The "  $:$ > " operator.

**9.8.3.14 #define PMATH\_CHAR\_VECTOR 0x21C0**

The arrow above names to indicate a vector.

#### 9.8.3.15 #define PMATH\_RUN(code)

##### Value:

```
pmath_unref( \
    pmath_evaluate( \
        pmath_parse_string( \
            PMATH_C_STRING(code))) )
```

Execute some pMath code.

##### Parameters:

*code* The code as a C string (zero terminated).

#### 9.8.3.16 #define PMATH\_RUN\_ARGS(code, format, ...)

##### Value:

```
pmath_unref( \
    pmath_evaluate( \
        pmath_parse_string_args( \
            (code), (format), __VA_ARGS__) ) )
```

Execute some pMath code with arguments.

##### Parameters:

*code* The code as a C string.

*format* The argument's type format string.

... The arguments.

See [pmath\\_build\\_value\(\)](#) for the meaning of *format* and ...

### 9.8.4 Typedef Documentation

#### 9.8.4.1 typedef struct \_pmath\_span\_array\_t pmath\_span\_array\_t

#### 9.8.4.2 typedef struct \_pmath\_span\_t pmath\_span\_t

### 9.8.5 Enumeration Type Documentation

#### 9.8.5.1 anonymous enum

Enumerator:

*PMATH\_PREC\_ANY*  
*PMATH\_PREC\_SEQ*  
*PMATH\_PREC\_EVAL*  
*PMATH\_PREC\_ASS*  
*PMATH\_PREC\_MODY*  
*PMATH\_PREC\_LAZY*  
*PMATH\_PREC\_FUNC*  
*PMATH\_PREC\_REPL*  
*PMATH\_PREC\_RULE*  
*PMATH\_PREC\_MAP*  
*PMATH\_PREC\_STR*  
*PMATH\_PREC\_COND*  
*PMATH\_PREC\_ALT*  
*PMATH\_PREC\_OR*  
*PMATH\_PREC\_XOR*  
*PMATH\_PREC\_AND*  
*PMATH\_PREC\_ARROW*  
*PMATH\_PREC\_REL*  
*PMATH\_PREC\_UNION*  
*PMATH\_PREC\_ISECT*  
*PMATH\_PREC\_RANGE*  
*PMATH\_PREC\_ADD*  
*PMATH\_PREC\_CIRCADD*  
*PMATH\_PREC\_PLUMI*  
*PMATH\_PREC\_CIRCMUL*  
*PMATH\_PREC\_MUL*  
*PMATH\_PREC\_DIV*  
*PMATH\_PREC\_MIDDOT*  
*PMATH\_PREC\_CROSS*  
*PMATH\_PREC\_MUL2*  
*PMATH\_PREC\_POW*  
*PMATH\_PREC\_FAC*

*PMATH\_PREC\_APL*  
*PMATH\_PREC\_REPEAT*  
*PMATH\_PREC\_TEST*  
*PMATH\_PREC\_INC*  
*PMATH\_PREC\_CALL*  
*PMATH\_PREC\_DIFF*  
*PMATH\_PREC\_PRIM*

#### 9.8.5.2 enum pmath\_token\_t

Token classes known in the pMath language.

**Enumerator:**

*PMATH\_TOK\_NONE*  
*PMATH\_TOK\_SPACE*  
*PMATH\_TOK\_DIGIT*  
*PMATH\_TOK\_STRING*  
*PMATH\_TOK\_NAME*  
*PMATH\_TOK\_NAME2*  
*PMATH\_TOK\_BINARY\_LEFT*  
*PMATH\_TOK\_BINARY\_RIGHT*  
*PMATH\_TOK\_BINARY\_LEFT\_AUTOARG*  
*PMATH\_TOK\_BINARY\_LEFT\_OR\_PREFIX*  
*PMATH\_TOK\_NARY*  
*PMATH\_TOK\_NARY\_AUTOARG*  
*PMATH\_TOK\_NARY\_OR\_PREFIX*  
*PMATH\_TOK\_POSTFIX\_OR\_PREFIX*  
*PMATH\_TOK\_PREFIX*  
*PMATH\_TOK\_POSTFIX*  
*PMATH\_TOK\_CALL*  
*PMATH\_TOK\_LEFTCALL*  
*PMATH\_TOK\_LEFT*  
*PMATH\_TOK\_RIGHT*  
*PMATH\_TOK\_PRETEXT*  
*PMATH\_TOK\_ASSIGNTAG*  
*PMATH\_TOK\_PLUSPLUS*  
*PMATH\_TOK\_COLON*

***PMATH\_TOK\_TILDES***  
***PMATH\_TOK\_SLOT***  
***PMATH\_TOK\_QUESTION***  
***PMATH\_TOK\_INTEGRAL***  
***PMATH\_TOK\_COMMENTEND***

### 9.8.6 Function Documentation

**9.8.6.1** `pmath_t pmath_boxes_from_spans (pmath_span_array_t * spans,  
pmath_string_t string, pmath_bool_t parseable, pmath_t(*) (int, void *)  
box_at_index, void * data)`

Convert a span-array with the according code to boxed form.

#### Parameters:

***spans*** A span-array. It can be obtained by [pmath\\_spans\\_from\\_string\(\)](#) or [pmath\\_spans\\_from\\_boxes\(\)](#).  
***string*** The corresponding code to *span*. It wont be freed.  
***parseable*** Whether whitespace and comments should be skipped or not.  
***box\_at\_index*** An optional function that returns the box at a given position (indicated by the PMATH\_CHAR\_BOX character). This function will be called (at most) one time for every box and in their order of apperance.  
***data*** A pointer that will be provided as the last argument to *box\_at\_index*.

#### Returns:

A pMath object representing the boxed form. It must be freed.

**9.8.6.2** `static PMATH_INLINE pmath_bool_t pmath_char_is_36digit (uint16_t  
ch) [static]`

Test if a unicode character is a base-36 digit '0' - '9', 'a' - 'z', 'A' - 'Z'.

**9.8.6.3** `static PMATH_INLINE pmath_bool_t pmath_char_is_basedigit (int  
base, uint16_t ch) [static]`

Test if in a given base, a unicode character is a digit.

**9.8.6.4** `static PMATH_INLINE pmath_bool_t pmath_char_is_digit (uint16_t  
ch) [static]`

Test if a unicode character is a digit '0' - '9'.

**9.8.6.5** `static PMATH_INLINE pmath_bool_t pmath_char_is_hexdigit  
(uint16_t ch) [static]`

Test if a unicode character is a hexadecimal digit.

**9.8.6.6** `static PMATH_INLINE pmath_bool_t pmath_char_is_integral  
(uint16_t ch) [static]`

Test if a unicode character is an integral.

**9.8.6.7** `static PMATH_INLINE pmath_bool_t pmath_char_is_left (uint16_t ch)  
[static]`

Test if a unicode character is a left bracket.

**9.8.6.8** `static PMATH_INLINE pmath_bool_t pmath_char_is_name (uint16_t  
ch) [static]`

Test if a unicode character can be the start of an identifier/name.

**9.8.6.9** `static PMATH_INLINE pmath_bool_t pmath_char_is_right (uint16_t  
ch) [static]`

Test if a unicode character is a right bracket.

**9.8.6.10** `static PMATH_INLINE pmath_bool_t pmath_char_maybe_bigop  
(uint16_t ch) [static]`

Test if a unicode character may be a big operation, e.g. Union, Sum.



#### 9.8.6.11 `pmath_t pmath_parse_string (pmath_string_t code)` [related, inherited]

Parse a string to an expression.

##### Parameters:

*code* A pMath String representing the code. It will be freed.

##### Returns:

A pMath object.

This function returns `ToExpression("code")`, but does not evaluate this released result.

#### 9.8.6.12 `pmath_t pmath_parse_string_args (const char * code, const char * format, ...)` [related, inherited]

Parse a string with additional arguments to an expression.

##### Parameters:

*code* A pMath String representing the code. It will be freed.

*format* A format string for the arguments.

... The additional arguments.

##### Returns:

A pMath object.

This function is a short hand for

1. assigning `pmath_build_value(format, ...)` to the symbol `$ParserArguments`
2. then calling `pmath_parse_string(PMATH_C_STRING(code))` and
3. restoring the old value of `$ParserArguments`
4. returning the result of the `pmath_parse_string`-call.

##### See also:

[pmath\\_build\\_value](#)  
[pmath\\_parse\\_string](#)

**9.8.6.13** `static PMATH_INLINE uint16_t pmath_right_fence (uint16_t left)`  
[static]

Get the corresponding right bracket to a given left bracket or 0.

**9.8.6.14** `void pmath_span_array_free (pmath_span_array_t * spans)`  
[inherited]

Destroy a span-array and all its spans.

**Parameters:**

*spans* The span-array.

**9.8.6.15** `pmath_bool_t pmath_span_array_is_operand_start`  
`(pmath_span_array_t * spans, int pos)` [inherited]

Test the operator-start-flag at an index.

**Parameters:**

*spans* The span-array.

*pos* The position. Must be between 0 and `pmath_span_array_length(spans)-1`.

**Returns:**

Whether an operator starts at the specified position.

**9.8.6.16** `pmath_bool_t pmath_span_array_is_token_end (pmath_span_array_t`  
`* spans, int pos)` [inherited]

Test the token-end-flag at an index.

**Parameters:**

*spans* The span-array.

*pos* The position. Must be between 0 and `pmath_span_array_length(spans)-1`.

**Returns:**

Whether a token ends at the specified position.

**9.8.6.17** `int pmath_span_array_length (pmath_span_array_t * spans)`  
[inherited]

Get a span-array's length.

**Parameters:**

*spans* The span-array.

**Returns:**

Its length or 0 if it's PMATH\_NULL.

**9.8.6.18** `pmath_span_t * pmath_span_at (pmath_span_array_t * spans, int pos)` [inherited]

Get a span starting at an index.

**Parameters:**

*spans* The span-array.

*pos* The position. Must be between 0 and `pmath_span_array_length(spans)-1`.

**Returns:**

The largest span stating at *pos* or PMATH\_NULL if there is no span.

**9.8.6.19** `int pmath_span_end (pmath_span_t * span)` [inherited]

Get end of a span.

**Parameters:**

*span* A span.

**Returns:**

The last index which is covered by the span.

**9.8.6.20** `pmath_span_t * pmath_span_next (pmath_span_t * span)`  
`[inherited]`

Get the next-shorter span starting at the same position.

**Parameters:**

*span* A span.

**Returns:**

The next-shorter span starting at *pos* or PMATH\_NULL if there is no span.

**9.8.6.21** `pmath_span_array_t* pmath_spans_from_boxes (pmath_t boxes,  
pmath_string_t * result_string, void (*)(int, pmath_t, void *) make_box,  
void * data)`

Convert boxed form back to span-array and code.

**Parameters:**

*boxes* A pMath object representing the boxed form.

*result\_string* A pointer where the resulting code will go to. Its previous value is ignored.

*make\_box* A function that converts a boxed form (pMath object) to an actual box. It must free this object (the second argument).

*data* A pointer that will be provided to *make\_box* as the last argument.

**Returns:**

A span-array. It must be freed with [pmath\\_span\\_array\\_free\(\)](#) when it is no longer needed.

**9.8.6.22** `pmath_span_array_t* pmath_spans_from_string (pmath_string_t *  
code, pmath_string_t (*)(void *) line_reader, pmath_bool_t (*)(int,  
void *) subsuperscriptbox_at_index, pmath_string_t (*)(int, void *)  
underoverscriptbox_at_index, void (*)(pmath_string_t, int, void *,  
pmath_bool_t) error, void * data)`

Parses pMath code to a span array.

**Parameters:**

*code* A pointer to a pMath string.

**line\_reader** An optional function to be called, when there is more input needed. Its result will be appended to *\*code*.

**subsuperscriptbox\_at\_index** An optional function that returns TRUE iff at a given position in the code (indicated by the PMATH\_CHAR\_BOX character) is a SubscriptBox, SuperscriptBox or SubsuperscriptBox.

**underoverscriptbox\_at\_index** [optional] If there is an UnderscriptBox, OverscriptBox or UnderoverscriptBox at a given position in the code (indicated by the PMATH\_CHAR\_BOX character) its base (e.g. middle part of UnderoverscriptBox) should be returned by this function, otherwise PMATH\_NULL should be returned.

**error** A function that will be called on syntax errors. The first argument is *\*code*. It must not be freed. The second argument is the position in the code. The third argument is *data*. The fourth argument is TRUE if the error is critical and FALSE if it is just a warning (Syntax::new!) This function is optional, if it is PMATH\_NULL, no messages will be generated during the scanning.

**data** An arbitrary pointer, that will be provided as the last argument to the callback functions.

#### Returns:

A span-array that can be used by `pmath_boxes_from_spans` to convert the code to boxed form, which, in turn, is used by `System'MakeExpression()`. The span-array must be freed with `pmath_span_array_free()` when it is no longer needed.

#### 9.8.6.23 `pmath_t pmath_string_expand_boxes (pmath_string_t s)` [related, inherited]

Expand a string that contains boxes to a list of Strings and Boxes.

#### Parameters:

*s* The string to be expanded. It will be freed.

#### Returns:

A string if there is nothing to expand or an expression representing *s* as boxes.

#### 9.8.6.24 `pmath_token_t pmath_token_analyse (const uint16_t * str, int len, int * prec)`

Analyse a token.

#### Parameters:

*str* A UTF16-string.

*len* The length (in uint16\_t-s) of the token *str*.

*prec* Optional address, where to store the default operator precedence for the token.

**Returns:**

The associated token class.

**9.8.6.25** `static PMATH_INLINE pmath_bool_t pmath_token_maybe_bigop  
(pmath_token_t tok) [static]`

Test if a token may be a big operator.

**9.8.6.26** `static PMATH_INLINE pmath_bool_t pmath_token_maybe_first  
(pmath_token_t tok) [static]`

Test whether a token may be the first token in a subexpression.

**Parameters:**

*tok* A token class.

**Returns:**

Whether tok may start a new subexpression

**9.8.6.27** `static PMATH_INLINE pmath_bool_t pmath_token_maybe_rest  
(pmath_token_t tok) [static]`

Test whether a token need not be the first token in a subexpression.

**Parameters:**

*tok* A token class.

**Returns:**

Whether tok may reside inside a subexpression.

### 9.8.6.28 `int pmath_token_prefix_precedence (const uint16_t * str, int len, int defprec)`

Give the prefix operator precedence for a token.

#### Parameters:

*str* A UTF16-string.

*len* The length (in uint16\_t-s) of the token *str*.

*defprec* The default operator precedence as given by [pmath\\_token\\_analyse\(\)](#)

#### Returns:

The prefix operator precedence.

## 9.9 General Purpose Types

Useful type definitions that do not fit into any other category.

#### Defines

- `#define FALSE ((pmath_bool_t)0)`  
*The FALSE value for pmath\_bool\_t.*
- `#define TRUE (!FALSE)`  
*The TRUE value for pmath\_bool\_t.*
- `#define PMATH_INVALID_PTR ((void*)UINTPTR_MAX)`

#### Typedefs

- `typedef char pmath_bool_t`  
*A boolean type.*
- `typedef void(* pmath_callback_t)(void *)`  
*A general callback function.*

### 9.9.1 Detailed Description

Useful type definitions that do not fit into any other category.

### 9.9.2 Define Documentation

#### 9.9.2.1 `#define FALSE ((pmath_bool_t)0)`

The FALSE value for `pmath_bool_t`.

#### 9.9.2.2 `#define PMATH_INVALID_PTR ((void*)UINTPTR_MAX)`

#### 9.9.2.3 `#define TRUE (!FALSE)`

The TRUE value for `pmath_bool_t`.

### 9.9.3 Typedef Documentation

#### 9.9.3.1 `typedef char pmath_bool_t`

A boolean type.

C does not define a boolean type, so we do it here for code clarity. The constants TRUE and FALSE can be used as return values for `pmath_bool_t`, but do not test on these. E.g. use `if (test) ...` instead of `if (test == TRUE) ...` but `return TRUE;` instead of `return 1;`

#### 9.9.3.2 `typedef void(* pmath_callback_t)(void *)`

A general callback function.

This is used in various places where a callback function is needed, that does not only work with pMath objects.

## 9.10 Atomic Operations

Using atomic operations (independent of the rest of the library).

### Defines

- `#define PMATH_ATOMIC_FASTLOOP_COUNT (0)`  
*Loop iterations in spinlocks before yielding control.*



- `#define PMATH_DECLARE_ALIGNED(TYPE, NAME, ALIGNMENT) TYPE NAME`

*Declares a variable with specified alignment.*

## Functions

- `intptr_t pmath_atomic_fetch_add (pmath_atomic_t *atom, intptr_t delta)`  
*Add a value to another.*
- `intptr_t pmath_atomic_fetch_set (pmath_atomic_t *atom, intptr_t new_value)`  
*Exchange a value.*
- `intptr_t pmath_atomic_fetch_compare_and_set (pmath_atomic_t *atom, intptr_t old_value, intptr_t new_value)`  
*Exchange a value if it equals another value.*
- `pmath_bool_t pmath_atomic_compare_and_set (pmath_atomic_t *atom, intptr_t old_value, intptr_t new_value)`  
*Exchange a value if it equals another value.*
- `pmath_bool_t pmath_atomic_compare_and_set_2 (pmath_atomic2_t *atom, intptr_t old_value_fst, intptr_t old_value_snd, intptr_t new_value_fst, intptr_t new_value_snd)`  
*Exchange two values value if they equal another two values.*
- `pmath_bool_t pmath_atomic_have_cas2 (void)`  
*Check, whether the CPU supports `pmath_atomic_compare_and_set_2()`.*
- `void pmath_atomic_barrier (void)`  
*Insert an explicit memory barrier.*
- `void pmath_atomic_lock (pmath_atomic_t *atom)`  
*Try to aquire a lock.*
- `void pmath_atomic_unlock (pmath_atomic_t *atom)`  
*Release a previously aquired lock.*
- `void pmath_atomic_loop_yield (void)`  
*Yield control to another thread (used in spinlocks).*
- `void pmath_atomic_loop_nop (void)`  
*A no-operation or short wait for use in spin locks.*

### 9.10.1 Detailed Description

Using atomic operations (independent of the rest of the library).

pMath provides a collection of functions/macros to do atomic operations. This part of the library is completely independent of the rest of pMath. To use atomic operations, `#include <pmath-util/concurrency/atomic.h>`. You do not have to link to an additional library.

At the moment, supported compilers are GCC and Microsoft Visual C++.

On some platforms, the atomic operations are implemented as inline functions with inline assembler (currently GCC older than 4.x). On other platforms, macros and compiler intrinsic functions (GCC 4.x, MSVC) are used.

### 9.10.2 Nice to read:

- [http://developers.sun.com/solaris/articles/atomic\\_sparc/](http://developers.sun.com/solaris/articles/atomic_sparc/)
- <http://lists.canonical.org/pipermail/kragen-tol/1999-August/000457.html>
- [http://www.angstrom-distribution.org/unstable/sources/libc\\_sources.redhat.com\\_\\_20061019.tar.gz](http://www.angstrom-distribution.org/unstable/sources/libc_sources.redhat.com__20061019.tar.gz)  
[libc/sysdeps/i386/i486/bits/atomic.h](http://www.angstrom-distribution.org/unstable/sources/libc_sysdeps/i386/i486/bits/atomic.h)
- Intel's `cmpxchg8b` and `cmpxchg16b` instructions
- <http://www.cse.msu.edu/~sdf/private/szumoframe-0.3.tar.gz>  
[szumoframe-0.3/src/szumoframe/szumo\\_preamble.h](http://www.cse.msu.edu/~sdf/private/szumoframe-0.3/src/szumoframe/szumo_preamble.h)
- [http://www.tml.tkk.fi/~rakajast/uvsr\\_renderer.tar.gz](http://www.tml.tkk.fi/~rakajast/uvsr_renderer.tar.gz)  
[uvsr\\_renderer/needed\\_externals/threadlib/src/fifo.c](http://www.tml.tkk.fi/~rakajast/uvsr_renderer/needed_externals/threadlib/src/fifo.c)
- `qprof -> atomic_ops` library
- <http://code.google.com/p/google-perftools/>

### 9.10.3 Define Documentation

#### 9.10.3.1 `#define PMATH_ATOMIC_FASTLOOP_COUNT (0)`

Loop iterations in spinlocks before yielding control.

If the thread holding the lock sits on another CPU, spinning around a bit before `pmath_atomic_loop_yield()` reduces idle time. But if the thread holding the lock lives on the

same CPU as the current thread (and thus is interrupted by the current thread), spinning elongates the wait time.

In summary, this should not be a compile time constant as it is now!

#### 9.10.3.2 `#define PMATH_DECLARE_ALIGNED(TYPE, NAME, ALIGNMENT) TYPE NAME`

Declares a variable with specified alignment.

##### Parameters:

*TYPE* The variable type, possibly including the `volatile` modifier.

*NAME* The variable name.

*ALIGNMENT* The alignment in bytes

#### 9.10.4 Function Documentation

##### 9.10.4.1 `void pmath_atomic_barrier (void)`

Insert an explicit memory barrier.

##### 9.10.4.2 `pmath_bool_t pmath_atomic_compare_and_set (pmath_atomic_t * atom, intptr_t old_value, intptr_t new_value)`

Exchange a value if it equals another value.

##### Parameters:

*atom* An atomic variable.

*old\_value* The comparator.

*new\_value* The possible new value of *\*atom*.

##### Returns:

Whether the exchange was performed.

This function compares *\*atom* with *old\_value* and iff both equal sets *\*atom* to *new\_value*, everything atomically and with acquire barrier semantics.

**9.10.4.3** `pmath_bool_t pmath_atomic_compare_and_set_2 (pmath_atomic2_t *atom, intptr_t old_value_fst, intptr_t old_value_snd, intptr_t new_value_fst, intptr_t new_value_snd)`

Exchange two values value if they equal another two values.

**Parameters:**

*\*atom* An atomic variable of size `2 * sizeof(void*)`.

*old\_value\_fst* The first old value.

*old\_value\_snd* The second old value.

*new\_value\_fst* The possible new value of `atom[0]`.

*new\_value\_snd* The possible new value of `atom[1]`.

**Returns:**

Whether the exchange was performed or not.

This function compares `old_value_fst` with `atom[0]` and `old_value_snd` with `atom[1]`. If they equal, `atom[0]` is set to `new_value_fst` and `atom[1]` is set to `new_value_snd` and `TRUE` is returned. Otherwise, `FALSE` will be returned.

This function has acquire barrier semantics.

**Note:**

This function is not available on all Platforms. You must not call it if `pmath_atomic_have_cas2()` returns `FALSE`.

**9.10.4.4** `intptr_t pmath_atomic_fetch_add (pmath_atomic_t *atom, intptr_t delta)`

Add a value to another.

**Parameters:**

*atom* An atomic variable.

*delta* The difference between the new and the old value.

**Returns:**

The old value of `*atom`.

This function increments `*atom` atomically by `delta`. It has full memory barrier semantics.

#### 9.10.4.5 `intptr_t pmath_atomic_fetch_compare_and_set (pmath_atomic_t * atom, intptr_t old_value, intptr_t new_value)`

Exchange a value if it equals another value.

##### Parameters:

*atom* An atomic variable.  
*old\_value* The comparisor.  
*new\_value* The possible new value of *\*atom*.

##### Returns:

The old value of *\*atom*.

You should use [pmath\\_atomic\\_compare\\_and\\_set\(\)](#) if you don't need the exact old value of *\*atom*, because this function might be non-existent on some systems. This function has acquire barrier semantics.

#### 9.10.4.6 `intptr_t pmath_atomic_fetch_set (pmath_atomic_t * atom, intptr_t new_value)`

Exchange a value.

##### Parameters:

*atom* An atomic variable.  
*new\_value* The new value of *\*atom*.

##### Returns:

The old value of *\*atom*.

This function sets *\*atom* to *new\_value* and returns the old value atomically. It has full memory barrier semantics.

#### 9.10.4.7 `pmath_bool_t pmath_atomic_have_cas2 (void)`

Check, whether the CPU supports [pmath\\_atomic\\_compare\\_and\\_set\\_2\(\)](#).

##### Returns:

whether [pmath\\_atomic\\_compare\\_and\\_set\\_2\(\)](#) is supported.

Note, that a call to [pmath\\_atomic\\_compare\\_and\\_set\\_2\(\)](#) will crash your application on any platform that does not support the operation (e.g. pre-Pentiums, early AMD64).

**9.10.4.8 void pmath\_atomic\_lock (pmath\_atomic\_t \* *atom*)**

Try to acquire a lock.

**Parameters:**

*atom* The lock. An atomic variable.

This function implements a spin lock. It has acquire barrier semantics. Use it with [pmath\\_atomic\\_unlock\(\)](#):

```
pmath_atomic_t spin = PMATH_ATOMIC_STATIC_INIT;
...
pmath_atomic_lock(&spin)
... critical section ...
pmath_atomic_unlock(&spin);
```

**9.10.4.9 void pmath\_atomic\_loop\_nop (void)**

A no-operation or short wait for use in spin locks.

**9.10.4.10 void pmath\_atomic\_loop\_yield (void)**

Yield control to another thread (used in spinlocks).

**9.10.4.11 void pmath\_atomic\_unlock (pmath\_atomic\_t \* *atom*)**

Release a previously acquired lock.

**Parameters:**

*atom* The lock. An atomic variable.

**See also:**

[pmath\\_atomic\\_lock](#)

**9.11 Thread Messaging**

Sending messages to other threads.

### Data Structures

- class `pmath_messages_t`  
*A message queue for interthread communication.*

### Typedefs

- typedef `pmath_custom_t` `pmath_messages_t`

### Functions

- double `pmath_tickcount` (void)  
*Gives the seconds since January 1, 1970 (UTC).*
- `pmath_bool_t` `pmath_is_message_queue` (`pmath_t` obj)  
*Test if an object is a message queue.*
- `pmath_messages_t` `pmath_thread_get_queue` (void)  
*Get the current thread's message queue.*
- void `pmath_thread_sleep` (void)  
*Send the current thread to sleep.*
- void `pmath_thread_sleep_timeout` (double abs\_timeout)  
*Send the current thread to sleep.*
- void `pmath_thread_wakeup` (`pmath_messages_t` mq)  
*Wake up another thread.*
- void `pmath_thread_send` (`pmath_messages_t` mq, `pmath_t` msg)  
*Asynchronously send a message to another thread.*
- `pmath_t` `pmath_thread_send_wait` (`pmath_messages_t` mq, `pmath_t` msg, double timeout\_seconds, void(\*idle\_function)(void \*), void \*idle\_data)  
*Send a message to another thread and wait for the answer.*
- void `pmath_thread_send_delayed` (`pmath_messages_t` mq, `pmath_t` msg, double seconds)  
*Asynchronously send a message to a thread sometime in the future.*

### 9.11.1 Detailed Description

Sending messages to other threads.

Every pMath thread has its own message queue. Other threads can send messages to such a queue and optionally wait for a result. Messages to any queue can also be registered for delivery at a later point in time.

Threads can go to sleep when they have no work to do. They will be awoken any time a message arrives to handle it.

Technical Note: Pending messages are handled as soon as time [pmath\\_aborting\(\)](#) is called, which happens periodically. For the pMath code, it looks like asynchronous signals, because messages can occur any time during the evaluation. But from the native code's point of view, messages are synchronous, because they can only occur during [pmath\\_aborting\(\)](#).

**Note:**

Message passing is not signal-safe. You must not send any messages from within a UNIX signal handler.

### 9.11.2 Typedef Documentation

#### 9.11.2.1 typedef `pmath_custom_t` `pmath_messages_t`

### 9.11.3 Function Documentation

#### 9.11.3.1 `pmath_bool_t` `pmath_is_message_queue` (`pmath_t obj`) [*related, inherited*]

Test if an object is a message queue.

**Parameters:**

*obj* Any pMath object. It wont be freed.

**Returns:**

TRUE if the object is a valid message queue object ([pmath\\_messages\\_t](#)).

#### 9.11.3.2 `pmath_messages_t` `pmath_thread_get_queue` (`void`) [*related, inherited*]

Get the current thread's message queue.



**Returns:**

A reference to the message queue or PMATH\_NULL on error. You must destroy it with `pmath_unref()` when its no longer needed.

### 9.11.3.3 void pmath\_thread\_send (pmath\_messages\_t mq, pmath\_t msg) [related, inherited]

Asynchronously send a message to another thread.

**Parameters:**

*mq* The receivers message queue. It wont be freed.

*msg* The message. It will be freed.

The message will be evaluated by the receiver. This function returns immediately. If the receiver cannot handle the message (since it is dead or there is not enough memory), the message will be deleted.

Note that messages might not be handled in the order they were send.

### 9.11.3.4 void pmath\_thread\_send\_delayed (pmath\_messages\_t mq, pmath\_t msg, double seconds) [related, inherited]

Asynchronously send a message to a thread sometime in the future.

**Parameters:**

*mq* The receivers message queue. It wont be freed.

*msg* The message. It will be freed.

*seconds* The delay in seconds before the message will be delivered.

The message will be evaluated by the receiver. This function returns immediately. If the receiver cannot handle the message (since it is dead or there is not enough memory), the message will be deleted.

### 9.11.3.5 pmath\_t pmath\_thread\_send\_wait (pmath\_messages\_t mq, pmath\_t msg, double timeout\_seconds, void (\*)(void \*) idle\_function, void \* idle\_data) [related, inherited]

Send a message to another thread and wait for the answer.

**Parameters:**

*mq* The receivers message queue. It wont be freed.

**msg** The message. It will be freed.

**timeout\_seconds** The maximum number of seconds to wait for the answer. Use HUGE\_VAL if you do not want a timeout.

**idle\_function** An optional function that will be called any time the waiting thread wakes up but there is no answer yet.

**idle\_data** Argument for

- idle\_function.

#### Returns:

The result of `pmath_evaluate(message)` called by the receiver or `PMATH_UNDEFINED` in case of an error.

The message will be evaluated by the receiver. If the receiver cannot handle it (since it is dead or there is not enough memory), the message will be deleted.

The calling thread will fall asleep until

- it receives an answer to return or
- the message is deleted or
- the timeout is reached or
- another abort situation occurs in the calling thread (e.g. `pmath_abort_please()` is called anywhere in the system)

In the last two cases (timeout or abort), the remote evaluation will be aborted.

#### Todo

Check, what happens if mq belongs to a parent thread.

#### 9.11.3.6 void `pmath_thread_sleep (void)` [related, inherited]

Send the current thread to sleep.

The thread will fall asleep until

- it receives a message or
- it is waken up with `pmath_thread_wakeup()` or
- an abort-condition (`pmath_abort_please()` or `pmath_throw()`) is met *anywhere* in the system.

Because of the last point, this function is normally called in a loop:

```
while (!pmath_aborting() && some_wait_condition) {
    pmath_thread_sleep();
}
```

### 9.11.3.7 void `pmath_thread_sleep_timeout` (double *abs\_timeout*) [related, inherited]

Send the current thread to sleep.

#### Parameters:

*abs\_timeout* Timeout in seconds since January 1, 1970 (UTC).

The thread will fall asleep until

- it receives a message or
- it is waken up with [pmath\\_thread\\_wakeup\(\)](#) or
- an abort-condition ([pmath\\_abort\\_please\(\)](#) or [pmath\\_throw\(\)](#)) is met *anywhere* in the system or
- – *abs\_timeout* is passed.

#### See also:

[pmath\\_thread\\_sleep](#), [pmath\\_tickcount](#)

### 9.11.3.8 void `pmath_thread_wakeup` (`pmath_messages_t mq`) [related, inherited]

Wake up another thread.

#### Parameters:

*mq* The message queue associated with the sleeping thread. It wont be freed.

This function wakes up the thread that is associated with the message queue. It is safe to try to wake up threads, that are not sleeping.

To follow the loop-style waiting idiom described in [pmath\\_thread\\_sleep\(\)](#), you must modify `some_wait_condition` *before* calling this function to successfully awake the other thread.

### 9.11.3.9 double `pmath_tickcount` (void)

Gives the seconds since January 1, 1970 (UTC).

#### Returns:

The number of seconds since January 1, 1970 (UTC)

## 9.12 Multithreading with pMath

The Thread abstraction in pMath.

### Data Structures

- class `pmath_threadlock_t`  
*A reentrant lock for threads.*
- class `pmath_thread_t`  
*The Representation of a thread.*

### Typedefs

- typedef struct `_pmath_threadlock_t` \* `pmath_threadlock_t`
- typedef struct `_pmath_thread_t` \* `pmath_thread_t`

### Functions

- `pmath_t pmath_thread_local_save` (`pmath_t` key, `pmath_t` value)  
*Store a thread/thread-local value.*
- `pmath_t pmath_thread_local_load` (`pmath_t` key)  
*Load a thread/thread-local value.*
- void `pmath_throw` (`pmath_t` exception)  
*Throw an exception.*
- `pmath_t pmath_catch` (void)  
*Catch any exception.*
- `pmath_bool_t pmath_aborting` (void)  
*Queries whether pMath was requested to abort the evaluation of the current thread.*
- void `pmath_abort_please` (void)  
*Requests pMath to abort the current evaluation.*
- void `pmath_suspend_all_please` (void)  
*Suspend all other threads. This function does not really suspend threads immediately. Any other thread, that calls `pmath_aborting()` (or `pmath_thread_aborting()`), will block until we call `pmath_resume_all()`.*
- void `pmath_resume_all` (void)  
*Resume all other threads.*

- void `pmath_thread_call_locked` (`pmath_threadlock_t` \*threadlock\_ptr, `pmath_callback_t` callback, void \*data)  
*Execute a function synchronized with a threadlock.*
- `pmath_bool_t` `pmath_thread_queue_is_blocked_by` (`pmath_messages_t` waiter\_mq, `pmath_messages_t` waitee\_mq)  
*Queries whether a thread is blocked by another thread.*
- void `pmath_thread_run_with_interrupt_notifier` (`pmath_callback_t` callback, `pmath_callback_t` notify, void \*callback\_closure, void \*notify\_closure)  
*Execute a function with an interrupt notifier installed.*
- `pmath_thread_t` `pmath_thread_get_current` (void)  
*Get the current pMath thread.*
- `pmath_thread_t` `pmath_thread_get_parent` (`pmath_thread_t` thread)  
*Get a thread's direct parent.*
- `pmath_bool_t` `pmath_thread_is_parent` (`pmath_thread_t` parent, `pmath_thread_t` child)  
*Queries whether a thread is one of the parents of another.*
- `pmath_bool_t` `pmath_thread_aborting` (`pmath_thread_t` thread)  
*Queries whether pMath was requested to abort the evaluation of a specific thread or its parents.*

### 9.12.1 Detailed Description

The Thread abstraction in pMath.

pMath stores several data local to a thread. Therefore, it maintains a `pmath_thread_t` in every operating system thread it runs on. Those `pmath_thread_t` variables are created and freed via `pmath_init()` and `pmath_done()` respectively. Thus, you have to call those two functions once in every thread that uses pMath functions (and abort the thread if `pmath_init()` fails).

pMath Threads can have parents. While one thread is running, its parent thread waits (for all its children) and is effectively immutable. This way, child threads can read their parent thread's local variables.

### 9.12.2 Synchronization

In other environments, you normally do synchronization with mutexes and the like. But if we did so, a deadlock could occur when a mutex is already locked by the parent thread, which in turn is waiting for its children to finish.

The solution is to use pMath threadlocks: You simply synchronize with a [pmath\\_symbol\\_t](#) through [pmath\\_symbol\\_synchronized\(\)](#) or directly with a [pmath\\_threadlock\\_t](#) and [pmath\\_thread\\_call\\_locked\(\)](#). This is reentrant and locks execution to a given thread *and* its child threads. pMath cares about avoiding deadlocks behind the scenes.

Note that threadlocks are needed only if the synchronized code might create child threads or calls other code that utilizes thread locks. Threads might be created by [pmath\\_evaluate\(\)](#) & co.

In other situations, you should use mutexes/semaphores from your operating system library or spinlocks (see [pmath\\_atomic\\_lock\(\)](#) and [pmath\\_atomic\\_unlock\(\)](#) ), because they are much faster.

For some simple changes on global integer/pointer variables, you can use [Atomic Operations](#).

### 9.12.3 Typedef Documentation

#### 9.12.3.1 `typedef struct _pmath_thread_t* pmath_thread_t`

#### 9.12.3.2 `typedef struct _pmath_threadlock_t* pmath_threadlock_t`

### 9.12.4 Function Documentation

#### 9.12.4.1 `void pmath_abort_please (void)`

Requests pMath to abort the current evaluation.

This function is signal-safe.

**See also:**

[pmath\\_continue\\_after\\_abort\(\)](#)

#### 9.12.4.2 `pmath_bool_t pmath_aborting (void)`

Queries whether pMath was requested to abort the evaluation of the current thread.

**Returns:**

Whether the user called [pmath\\_abort\\_please\(\)](#) or an exception was thrown or a time-out is passed.

#### 9.12.4.3 `pmath_t pmath_catch (void)`

Catch any exception.

**Returns:**

exception The exception to be thrown. If there is no exception available, `PMATH_UNDEFINED` will be returned.

If you cannot handle the exception, you can re-throw it with [pmath\\_throw\(\)](#).

#### 9.12.4.4 `void pmath_resume_all (void)`

Resume all other threads.

**See also:**

[pmath\\_suspend\\_all\\_please](#)

#### 9.12.4.5 `void pmath_suspend_all_please (void)`

Suspend all other threads. This function does not really suspend threads immediately. Any other thread, that calls [pmath\\_aborting\(\)](#) (or [pmath\\_thread\\_aborting\(\)](#)), will block until we call [pmath\\_resume\\_all\(\)](#).

#### 9.12.4.6 `pmath_bool_t pmath_thread_aborting (pmath_thread_t thread)` [related, inherited]

Queries whether pMath was requested to abort the evaluation of a specific thread or its parents.

**Parameters:**

*thread* A thread that should be tested.

**Returns:**

Whether the given thread should abort evaluation.

**See also:**

[pmath\\_aborting](#)

**9.12.4.7** `void pmath_thread_call_locked (pmath_threadlock_t * threadlock_ptr,  
pmath_callback_t callback, void * data)` [related, inherited]

Execute a function synchronized with a threadlock.

**Parameters:**

*threadlock\_ptr* A pointer to the threadlock.

*callback* The function to be executed when the symbol is locked.

*data* A pointer that will be passed to callback.

All you have to do is initialize the threadlock `threadlock_ptr` points to with `PMATH_NULL` before you call this function for the first time:

```
static pmath_threadlock_t lock = PMATH_NULL;  
...  
pmath_thread_call_locked(&lock, my_callback, my_data);
```

To synchronize with a symbol, use [pmath\\_symbol\\_synchronized\(\)](#).

**9.12.4.8** `pmath_thread_t pmath_thread_get_current (void)` [related, inherited]

Get the current pMath thread.

**Returns:**

A [pmath\\_thread\\_t](#). This is `PMATH_NULL`, if you did not register the current thread to pMath via [pmath\\_init\(\)](#).

**9.12.4.9** `pmath_thread_t pmath_thread_get_parent (pmath_thread_t thread)`  
[related, inherited]

Get a thread's direct parent.

**Parameters:**

*thread* A thread.

**Returns:**

The direct parent of thread. Usually `PMATH_NULL`.



#### 9.12.4.10 `pmath_bool_t pmath_thread_is_parent (pmath_thread_t parent, pmath_thread_t child)` [related, inherited]

Queries whether a thread is one of the parents of another.

##### Parameters:

*parent* A thread.

*child* A thread.

##### Returns:

TRUE, if *parent* is a parent thread of *child* or if *parent* == *child*. FALSE otherwise.

It is important to know that a parent thread is never executed in parallel with its children. However, to check for threads that depend on *child* (e.g. to evaluate a function on a specific thread or any thread that it waits on), use [pmath\\_thread\\_queue\\_is\\_blocked\\_by\(\)](#).

#### 9.12.4.11 `pmath_t pmath_thread_local_load (pmath_t key)`

Load a thread/thread-local value.

##### Parameters:

*key* A key that was used to save the value with `_pmath_thread_local_save()` before. It won't be freed.

##### Returns:

PMATH\_UNDEFINED or the stored value. You must destroy the it.

If there is nothing stored for *key* in the current thread, its parent threads are processed. If none of them stores something under '*key*' and *key* is a symbol, The global value is used.

#### 9.12.4.12 `pmath_t pmath_thread_local_save (pmath_t key, pmath_t value)`

Store a thread/thread-local value.

##### Parameters:

*key* The key that can be used to obtain the value with `_pmath_thread_local_load()`. It won't be freed.

*value* The thread/thread-local value. It will be freed.

**Returns:**

[PMATH\\_UNDEFINED](#) or the previous value that was stored with the same key.  
You must destroy it.

Note that keys of the form 'symboltag' are used to store whether a message should be suppressed (value [PMATH\\_SYMBOL\\_OFF](#)) or not (value [PMATH\\_NULL](#)).

All keys that are [magic numbers](#), have special meanings for [pmath\\_thread\\_local\\_save\(\)](#). You should not use them as the a key.

Keys which are only symbols are used for thread-local symbols (

**See also:**

[pmath\\_symbol\\_attributes\\_t](#)).

**9.12.4.13** [pmath\\_bool\\_t](#) [pmath\\_thread\\_queue\\_is\\_blocked\\_by](#)  
([pmath\\_messages\\_t](#) *waiter\_mq*, [pmath\\_messages\\_t](#) *waitee\_mq*)  
[related, inherited]

Queries whether a thread is blocked by another thread.

**Parameters:**

*waiter\_mq* A message queue. It will be freed

*waitee\_mq* A message queue. It will be freed.

**Returns:**

TRUE, if thread which owns *waiter\_mq* is a blocked by the thread which owns *waitee\_mq* or if *waiter\_mq* == *waitee\_mq*. FALSE otherwise.

A use-case for this function is a function that wants to be evaluated on a specific thread A or any thread that A waits on. See [builtin\\_interrupt\(\)](#) in the reference front-end implementation `test.exe`

**9.12.4.14** [void](#) [pmath\\_thread\\_run\\_with\\_interrupt\\_notifier](#) ([pmath\\_callback\\_t](#) *callback*, [pmath\\_callback\\_t](#) *notify*, [void \\*](#) *callback\_closure*, [void \\*](#) *notify\_closure*) [related, inherited]

Execute a function with an interrupt notifier installed.

**Parameters:**

*callback* The function to be called.

**notify** Is called when a message is delivered to the current thread. This function is called by the thread that sends the message, but concurrent sending threads wait on each other when calling the function. *notify* must not call any function that could send messages, because that would lead to a deadlock.

**callback\_closure** The argument for *callback*.

**notify\_closure** The argument for *notify*.

#### 9.12.4.15 void pmath\_throw (pmath\_t exception)

Throw an exception.

##### Parameters:

**exception** The exception to be thrown. It will be freed. You cannot throw the magic number `PMATH_UNDEFINED`.

If there is already an uncought exception, this new exception is lost.

## 9.13 Debugging

### Functions

- void `pmath_debug_print` (const char \*fmt,...)  
*Print out a simple debug message.*
- void `pmath_debug_print_object` (const char \*pre, `pmath_t` obj, const char \*post)  
*Print a pMath object to the debug log.*
- void `pmath_debug_print_stack` (void)  
*Print the current pMath stack trace to the debug log.*

#### 9.13.1 Detailed Description

These functions are for logging purposes. They default to `((void)0)` unless `PMATH_DEBUG_LOG` is defined.

#### 9.13.2 Function Documentation

##### 9.13.2.1 void pmath\_debug\_print (const char \*fmt, ...)

Print out a simple debug message.

**Parameters:**

- fmt* A printf-compatible format string
- ... The variables to be printed as specified by format.

The format string and arguments are as in printf.

### 9.13.2.2 void pmath\_debug\_print\_object (const char \* *pre*, pmath\_t *obj*, const char \* *post*)

Print a pMath object to the debug log.

**Parameters:**

- pre* A string that should be printed before the object.
- obj* A pMath object. It wont be freed.
- post* A string that should be printed after the object.

### 9.13.2.3 void pmath\_debug\_print\_stack (void)

Print the current pMath stack trace to the debug log.

## 9.14 File API

Unified API to access file or other memory content.

**Data Structures**

- class [pmath\\_binary\\_file\\_api\\_t](#)  
*Access functions for binary files.*
- class [pmath\\_text\\_file\\_api\\_t](#)  
*Access functions for text files.*

**Enumerations**

- enum [pmath\\_files\\_status\\_t](#) {  
[PMATH\\_FILE\\_OK](#) = 0, [PMATH\\_FILE\\_INVALID](#) = 1,  
[PMATH\\_FILE\\_ENDOFFILE](#) = 2, [PMATH\\_FILE\\_OTHERERROR](#) = 3,  
[PMATH\\_FILE\\_RECURSIVE](#) = 4 }  
*The status of a file.*

## Functions

- `pmath_symbol_t pmath_file_create_compressor (pmath_t dstfile)`  
*Create a binary file object that compresses its input.*
- `pmath_symbol_t pmath_file_create_uncompressor (pmath_t srcfile)`  
*Create a binary file object that uncompresses its input.*
- `pmath_bool_t pmath_file_test (pmath_t file, int properties)`  
*Check whether a file supports a set of properties.*
- `pmath_files_status_t pmath_file_status (pmath_t file)`  
*Get the current status of a readable file.*
- `size_t pmath_file_read (pmath_t file, void *buffer, size_t buffer_size, pmath_bool_t preserve_internal_buffer)`  
*Read some bytes from a binary file.*
- `pmath_string_t pmath_file_readline (pmath_t file)`  
*Read one line from a text file.*
- `void pmath_file_set_textbuffer (pmath_t file, pmath_string_t buffer)`  
*Set a file's internal text buffer.*
- `size_t pmath_file_write (pmath_t file, const void *buffer, size_t buffer_size)`  
*Write some bytes to a binary file.*
- `pmath_bool_t pmath_file_writetext (pmath_t file, const uint16_t *str, int len)`  
*Write to a text file.*
- `void pmath_file_flush (pmath_t file)`  
*Flush the output buffer of a writeable file.*
- `pmath_bool_t pmath_file_write_object (pmath_t file, pmath_t obj, pmath_write_options_t options)`  
*Write an object to a text file.*
- `pmath_bool_t pmath_file_set_binbuffer (pmath_t file, size_t size)`  
*Set a binary file's buffer size.*
- `void pmath_file_manipulate (pmath_t file, void(*type)(void *), void(*callback)(void *, void *), void *data)`  
*Manipulate a file's internal representation.*
- `pmath_bool_t pmath_file_close (pmath_t file)`  
*Closes a file.*

- [pmath\\_symbol\\_t](#) [pmath\\_file\\_create\\_binary](#) (void \*extra, void(\*extra\_destructor)(void \*), [pmath\\_binary\\_file\\_api\\_t](#) \*api)  
*Create a binary file object.*
- [pmath\\_symbol\\_t](#) [pmath\\_file\\_create\\_text](#) (void \*extra, void(\*extra\_destructor)(void \*), [pmath\\_text\\_file\\_api\\_t](#) \*api)  
*Create a text file object.*
- [pmath\\_symbol\\_t](#) [pmath\\_file\\_create\\_text\\_from\\_binary](#) ([pmath\\_t](#) binfile, const char \*encoding)  
*Create a text file object operating on a binary file.*
- [pmath\\_symbol\\_t](#) [pmath\\_file\\_create\\_binary\\_buffer](#) (size\_t mincapacity)  
*Create a byte-stream file object.*
- size\_t [pmath\\_file\\_binary\\_buffer\\_size](#) ([pmath\\_t](#) binfile)  
*Get The number of readable bytes in a binary buffer.*
- void [pmath\\_file\\_binary\\_buffer\\_manipulate](#) ([pmath\\_t](#) binfile, void(\*callback)(uint8\_t \*readable, uint8\_t \*\*writable, const uint8\_t \*end, void \*closure), void \*closure)  
*Manipulate the content of a binary buffer.*
- void [pmath\\_file\\_create\\_mixed\\_buffer](#) (const char \*encoding, [pmath\\_symbol\\_t](#) \*out\_textfile, [pmath\\_symbol\\_t](#) \*out\_binfile)  
*Creates a mixed binary/text file double ended queue.*

### 9.14.1 Detailed Description

Unified API to access file or other memory content.

A file is a [Symbol](#) (normally with attribute [PMATH\\_SYMBOL\\_ATTRIBUTE\\_TEMPORARY](#)) whose value is a special [Custom Object](#).

The output functions can operate on lists of files.

### 9.14.2 Enumeration Type Documentation

#### 9.14.2.1 enum [pmath\\_files\\_status\\_t](#)

The status of a file.

See also:

[pmath\\_file\\_status\(\)](#)

**Enumerator:**

***PMATH\_FILE\_OK*** No error.

***PMATH\_FILE\_INVALID*** The object is no readable file.

***PMATH\_FILE\_ENDOFFILE*** The (readable) file position is at the end.

***PMATH\_FILE\_OTHERERROR*** There is another problem with the file.

***PMATH\_FILE\_RECURSIVE*** The file is already locked by the current thread.

**9.14.3 Function Documentation**

**9.14.3.1** `void pmath_file_binary_buffer_manipulate (pmath_t binfile,  
void (*)(uint8_t *readable, uint8_t **writable, const uint8_t *end, void  
*closure) callback, void * closure)`

Manipulate the content of a binary buffer.

**Parameters:**

***binfile*** A binary file created with `pmath_file_create_binary_buffer()`. It wont be freed.

***callback*** The callback function that does the manipulation.

***closure*** The fourth parameter for *callback*.

The *callback* function must not write before *readable* or after *end*. *\*writable* gives the current write-position, which is always between *readable* and *end*. It can be changed inside the callback, but must remain between *readable* and *end*.

**9.14.3.2** `size_t pmath_file_binary_buffer_size (pmath_t binfile)`

Get The number of readable bytes in a binary buffer.

**Parameters:**

***binfile*** A binary file created with `pmath_file_create_binary_buffer()`. It wont be freed.

**Returns:**

The number of readable bytes in the binary buffer or 0 on error.

**9.14.3.3** `pmath_bool_t pmath_file_close (pmath_t file)`

Closes a file.

**Parameters:**

*file* A file object. It will be freed.

**Returns:**

Whether file was a valid file object.

Files are closed automatically by the garbage collector when the reference counter becomes zero, which could be at an unpredictable point time in the future. This function closes a file immediatly (by clearing the value of the symbol representing the file).

**9.14.3.4 `pmath_symbol_t pmath_file_create_binary (void * extra, void(*) (void *) extra_destructor, pmath_binary_file_api_t * api)`**

Create a binary file object.

**Parameters:**

*extra* Arbitrary extra data.

*extra\_destructor* A function to destroy the extra data.

*api* The file access functions.

**Returns:**

A newly created binary file object. You can destroy and close it with [pmath\\_file\\_close\(\)](#) or [pmath\\_unref\(\)](#).

The *api* functions are never called by more than one thread at once. This is assured with a non-reentrant spinlock.

**See also:**

[pmath\\_binary\\_file\\_api\\_t](#)

**9.14.3.5 `pmath_symbol_t pmath_file_create_binary_buffer (size_t mincapacity)`**

Create a byte-stream file object.

**Parameters:**

*mincapacity* The initial size of the buffer.

**Returns:**

A newly created binary file object. You can destroy and close it with [pmath\\_file\\_close\(\)](#) or [pmath\\_unref\(\)](#).

You can write to a byte-buffer and read previously written data from it. Note that this does not support random access to the data.



#### 9.14.3.6 `pmath_symbol_t pmath_file_create_compressor (pmath_t dstfile)`

Create a binary file object that compresses its input.

**Parameters:**

*dstfile* A writable binary file object to write the compressed data to. It will be freed.

**Returns:**

A binary file object or PMATH\_NULL on error.

The compression uses zlib.

#### 9.14.3.7 `void pmath_file_create_mixed_buffer (const char * encoding, pmath_symbol_t * out_textfile, pmath_symbol_t * out_binfile)`

Creates a mixed binary/text file double ended queue.

**Parameters:**

*encoding* The encoding name. Possible values are "latin1", and "base85"

*out\_textfile* Will be set to the readable/writable text end.

*out\_binfile* Will be set to the readable/writable binary end.

#### 9.14.3.8 `pmath_symbol_t pmath_file_create_text (void * extra, void(*)(void *) extra_destructor, pmath_text_file_api_t * api)`

Create a text file object.

**Parameters:**

*extra* Arbitrary extra data.

*extra\_destructor* A function to destroy the extra data.

*api* The file access functions.

**Returns:**

A newly created text file object. You can destroy and close it with [pmath\\_file\\_close\(\)](#) or [pmath\\_unref\(\)](#).

The *api* functions are never called by more than one thread at once. This is assured with a non-reentrant spinlock.

See also:

[pmath\\_text\\_file\\_api\\_t](#)

#### 9.14.3.9 `pmath_symbol_t pmath_file_create_text_from_binary (pmath_t binfile, const char * encoding)`

Create a text file object operating on a binary file.

**Parameters:**

*binfile* A binary file. It will be freed.

*encoding* A text encoding that the iconv library knows.

**Returns:**

A newly created text file object. You can destroy and close it with [pmath\\_file\\_close\(\)](#) or [pmath\\_unref\(\)](#).

#### 9.14.3.10 `pmath_symbol_t pmath_file_create_uncompressor (pmath_t srcfile)`

Create a binary file object that uncompresses its input.

**Parameters:**

*srcfile* A readable binary file object to read the compressed data from. It will be freed.

**Returns:**

A binary file object or PMATH\_NULL on error.

The uncompression uses zlib.

#### 9.14.3.11 `void pmath_file_flush (pmath_t file)`

Flush the output buffer of a writeable file.

**Parameters:**

*file* A writeable binary file object. It wont be freed.

**9.14.3.12 void pmath\_file\_manipulate (pmath\_t *file*, void(\*)(void \*) *type*, void(\*)(void \*, void \*) *callback*, void \* *data*)**

Manipulate a file's internal representation.

**Parameters:**

*file* A file object. It wont be freed.

*type* The *extra\_destructor* that was provided to [pmath\\_file\\_create\\_binary\(\)](#) or [pmath\\_file\\_create\\_text\(\)](#).

*callback* A callback function. The first argument will be the *extra* parameter that was provided to [pmath\\_file\\_create\\_binary\(\)](#) or [pmath\\_file\\_create\\_text\(\)](#).

*data* The second parameter for *callback*.

If *file* is a valid file object and if *type* is the *extra\_destructor* which *file* was created with, then and only then *callback* will be called.

This function does not support lists of writeable files.

**9.14.3.13 size\_t pmath\_file\_read (pmath\_t *file*, void \* *buffer*, size\_t *buffer\_size*, pmath\_bool\_t *preserve\_internal\_buffer*)**

Read some bytes from a binary file.

**Parameters:**

*file* A readable binary file object. It wont be freed.

*buffer* The read bytes will go here.

*buffer\_size* The number of bytes you would like to read/size of *buffer*.

*preserve\_internal\_buffer* If TRUE, a subsequent call will get the same buffer content. If FALSE, the file pointer will be moved.

**Returns:**

Number of read bytes. This is never more than *buffer\_size*. If *preserve\_internal\_buffer* is TRUE or in case of an error, this can be less than *buffer\_size*.

**9.14.3.14 pmath\_string\_t pmath\_file\_readline (pmath\_t *file*)**

Read one line from a text file.

**Parameters:**

*file* A readable text file object. It wont be freed.

**Returns:**

The next line in the file.

**9.14.3.15 pmath\_bool\_t pmath\_file\_set\_binbuffer (pmath\_t *file*, size\_t *size*)**

Set a binary file's buffer size.

**Parameters:**

*file* A binary file. It wont be freed.

*size* The new size of the buffer in bytes.

**Returns:**

TRUE if the operation succeeded.

This function might clear the old buffer. So it should be called before any file read operation is done.

**9.14.3.16 void pmath\_file\_set\_textbuffer (pmath\_t *file*, pmath\_string\_t *buffer*)**

Set a file's internal text buffer.

**Parameters:**

*file* A readable text file. It wont be freed.

*buffer* A string. It should not contain any new line characters. It will be freed.

**9.14.3.17 pmath\_files\_status\_t pmath\_file\_status (pmath\_t *file*)**

Get the current status of a readable file.

**Parameters:**

*file* A readable file object. It wont be freed.

**Returns:**

pmath\_files\_status\_t The current file status.

### 9.14.3.18 `pmath_bool_t pmath_file_test (pmath_t file, int properties)`

Check whether a file supports a set of properties.

**Parameters:**

*file* A file object. It wont be freed.

*properties* File properties. See remarks section.

**Returns:**

TRUE iff the file supports all of the requested properties.

**Remarks:**

*properties* can be zero or more of the following values:

- `PMATH_FILE_PROP_READ` : The file is readable.
- `PMATH_FILE_PROP_WRITE` : The file is writeable.
- `PMATH_FILE_PROP_BINARY` : It is a binary file.
- `PMATH_FILE_PROP_TEXT` : It is a text file.

Lists of writeable files are writeable, too. Only Symbols can be readable files.

### 9.14.3.19 `size_t pmath_file_write (pmath_t file, const void * buffer, size_t buffer_size)`

Write some bytes to a binary file.

**Parameters:**

*file* A writeable binary file object. It wont be freed.

*buffer* The data to be written.

*buffer\_size* The number of bytes to write/size of *buffer*.

**Returns:**

The number of written bytes. This is less than *buffer\_size* in case of an error. If file is a list of files this is the smallest number of written bytes to the single files.

### 9.14.3.20 `pmath_bool_t pmath_file_write_object (pmath_t file, pmath_t obj, pmath_write_options_t options)`

Write an object to a text file.

**Parameters:**

- file* A writeable text file object. It wont be freed.
- obj* An object. It wont be freed.
- options* Some options defining the format.

**Returns:**

Whether the operation succeeded.

#### 9.14.3.21 `pmath_bool_t pmath_file_writetext (pmath_t file, const uint16_t * str, int len)`

Write to a text file.

**Parameters:**

- file* A writeable text file object. It wont be freed.
- str* A UTF-16 string buffer. e.g. `pmath_string_buffer(&some_text)`
- len* The number of `uint16_t` in the buffer. e.g. `pmath_string_length(some_text)`.  
This can be -1 if *str* is zero terminated.

**Returns:**

Whether the operation succeeded.

## 9.15 Hashtables

A general hashtable implementation.

**Data Structures**

- class `pmath_ht_class_t`  
*A hashtable interface.*
- class `pmath_hashtable_t`  
*The Hashtable class.*

**Typedefs**

- typedef struct `_pmath_hashtable_t` \* `pmath_hashtable_t`
- typedef void(\* `pmath_ht_entry_callback_t`)(void \*entry, void \*data)  
*A callback function for hashtable entries.*

- typedef void [>\(\\* pmath\\_ht\\_entry\\_copy\\_t\)](#)(void \*entry)  
*An entry copy function.*
- typedef unsigned int [\(\\* pmath\\_ht\\_entry\\_hash\\_func\\_t\)](#)(void \*entry)  
*An entry hash function.*
- typedef unsigned int [\(\\* pmath\\_ht\\_key\\_hash\\_func\\_t\)](#)(void \*key)  
*A key hash function.*
- typedef [pmath\\_bool\\_t](#) [\(\\* pmath\\_ht\\_entry\\_equal\\_func\\_t\)](#)(void \*entry1, void \*entry2)  
*An entry comparision function.*
- typedef [pmath\\_bool\\_t](#) [\(\\* pmath\\_ht\\_entry\\_equals\\_key\\_func\\_t\)](#)(void \*entry, void \*key)  
*An entry to key comparision function.*

### Functions

- [pmath\\_hashtable\\_t](#) [pmath\\_ht\\_create](#) (const [pmath\\_ht\\_class\\_t](#) \*klass, unsigned int minsize)  
*Create a new hashtable.*
- [pmath\\_hashtable\\_t](#) [pmath\\_ht\\_copy](#) ([pmath\\_hashtable\\_t](#) ht, [pmath\\_ht\\_entry\\_copy\\_t](#) entry\_copy)  
*Copy a given hashtable.*
- void [pmath\\_ht\\_destroy](#) ([pmath\\_hashtable\\_t](#) ht)  
*Destroy a given hashtable.*
- void [pmath\\_ht\\_clear](#) ([pmath\\_hashtable\\_t](#) ht)  
*Clear a given hashtable.*
- unsigned int [pmath\\_ht\\_capacity](#) ([pmath\\_hashtable\\_t](#) ht)  
*Get the capacity of a given hashtable.*
- unsigned int [pmath\\_ht\\_count](#) ([pmath\\_hashtable\\_t](#) ht)  
*Get the number of valid entries in a given hashtable.*
- void \* [pmath\\_ht\\_entry](#) ([pmath\\_hashtable\\_t](#) ht, unsigned int i)  
*Get any entry of a given hashtable.*
- void \* [pmath\\_ht\\_search](#) ([pmath\\_hashtable\\_t](#) ht, void \*key)  
*Search for an entry in a given hashtable.*

- void \* `pmath_ht_insert` (`pmath_hashtable_t` ht, void \*entry)  
*Insert an entry into a given hashtable.*
- void \* `pmath_ht_remove` (`pmath_hashtable_t` ht, void \*key)  
*Remove an entry from a given hashtable.*

### 9.15.1 Detailed Description

A general hashtable implementation.

The user of this API is responsible for the memory layout of the entries.

### 9.15.2 Typedef Documentation

#### 9.15.2.1 typedef struct \_pmath\_hashtable\_t\* pmath\_hashtable\_t

#### 9.15.2.2 typedef void(\* pmath\_ht\_entry\_callback\_t)(void \*entry, void \*data)

A callback function for hashtable entries.

##### Parameters:

- entry* An entry. Never PMATH\_NULL.  
*data* Additional data for the callback.

#### 9.15.2.3 typedef void\*(\* pmath\_ht\_entry\_copy\_t)(void \*entry)

An entry copy function.

##### Parameters:

- entry* An entry. Never PMATH\_NULL.

##### Returns:

- A new entry that is a copy.



#### 9.15.2.4 `typedef pmath_bool_t(* pmath_ht_entry_equal_func_t)(void *entry1, void *entry2)`

An entry comparison function.

**Parameters:**

*entry1* The first entry.

*entry2* The second entry.

**Returns:**

TRUE if both entries' keys are equal, FALSE otherwise.

#### 9.15.2.5 `typedef pmath_bool_t(* pmath_ht_entry_equals_key_func_t)(void *entry, void *key)`

An entry to key comparison function.

**Parameters:**

*entry* An entry.

*key* The key of another entry.

**Returns:**

TRUE if both entry's key equals the given key, FALSE otherwise.

#### 9.15.2.6 `typedef unsigned int(* pmath_ht_entry_hash_func_t)(void *entry)`

An entry hash function.

**Parameters:**

*entry* An entry. Never PMATH\_NULL.

**Returns:**

A hash value for the entry's key.

### 9.15.2.7 typedef unsigned int(\* pmath\_ht\_key\_hash\_func\_t)(void \*key)

A key hash function.

**Parameters:**

*key* A key.

**Returns:**

A hash value for the key.

### 9.15.3 Function Documentation

#### 9.15.3.1 unsigned int pmath\_ht\_capacity (pmath\_hashtable\_t *ht*) [inherited]

Get the capacity of a given hashtable.

**Parameters:**

*ht* The hashtable.

**Returns:**

The current maximum possible index of an entry in the table.

#### 9.15.3.2 void pmath\_ht\_clear (pmath\_hashtable\_t *ht*) [inherited]

Clear a given hashtable.

**Parameters:**

*ht* The hashtable.

#### 9.15.3.3 pmath\_hashtable\_t pmath\_ht\_copy (pmath\_hashtable\_t *ht*, pmath\_ht\_entry\_copy\_t *entry\_copy*) [inherited]

Copy a given hashtable.

**Parameters:**

*ht* The old hashtable

*entry\_copy* A function for copying entires.

**Returns:**

The new hashtable or NULL on error.

**9.15.3.4 unsigned int pmath\_ht\_count (pmath\_hashtable\_t *ht*)** [inherited]

Get the number of valid entries in a given hashtable.

**Parameters:**

*ht* The hashtable.

**Returns:**

The number of non-NULL entires.

**9.15.3.5 pmath\_hashtable\_t pmath\_ht\_create (const pmath\_ht\_class\_t \* *klass*, unsigned int *minsize*)** [inherited]

Create a new hashtable.

**Parameters:**

*klass* An interface pointer.

*minsize* Initial minimal size.

**Returns:**

The new hashtable or NULL on error.

**9.15.3.6 void pmath\_ht\_destroy (pmath\_hashtable\_t *ht*)** [inherited]

Destroy a given hashtable.

**Parameters:**

*ht* The hashtable.

### 9.15.3.7 `void * pmath_ht_entry (pmath_hashtable_t ht, unsigned int i)` [inherited]

Get any entry of a given hashtable.

#### Parameters:

*ht* The hashtable.

*i* The index. from range 0..pmath\_ht\_capacity(ht) - 1

#### Returns:

The entry or NULL. It is owned by the table.

### 9.15.3.8 `void * pmath_ht_insert (pmath_hashtable_t ht, void * entry)` [inherited]

Insert an entry into a given hashtable.

#### Parameters:

*ht* The hashtable.

*entry* The entry. It must be compatible with all functions of the table's interface.

#### Returns:

NULL or a possible old entry or the entry itself in case of an error. You must destroy it.

### 9.15.3.9 `void * pmath_ht_remove (pmath_hashtable_t ht, void * key)` [inherited]

Remove an entry from a given hashtable.

#### Parameters:

*ht* The hashtable.

*key* The entry's key. It will be send to the kes\_hash and entry\_keys\_equal functions of the table's interface pointer.

#### Returns:

NULL or the old entry. You must destroy it.

**9.15.3.10** `void * pmath_ht_search (pmath_hashtable_t ht, void * key)`  
 [inherited]

Search for an entry in a given hashtable.

**Parameters:**

*ht* The hashtable.

*key* The key. It will be send to the `kes_hash` and `entry_keys_equal` functions of the table's interface pointer.

**Returns:**

The entry or NULL. It is owned by the table.

## 9.16 Object Utility Functions

Utility funtuions for pMath Objects and Expressions.

**Typedefs**

- typedef `pmath_bool_t(* pmath_stack_walker_t)(pmath_t head, void *closure)`  
*A stack walker function.*

**Functions**

- `pmath_bool_t pmath_is_expr_of (pmath_t obj, pmath_symbol_t head)`  
*Check if an object is an expression with a specified head.*
- `pmath_bool_t pmath_is_expr_of_len (pmath_t obj, pmath_symbol_t head, size_t length)`  
*Check if an object is an expression with a specified head and length.*
- `pmath_t pmath_current_head (void)`  
*Get the currently evaluated function.*
- `void pmath_walk_stack (pmath_stack_walker_t walker, void *closure)`  
*Walk up the current thread's and its parents' stack.*
- `void pmath_gather_begin (pmath_t pattern)`  
*Start gathering emitted objects.*
- `pmath_expr_t pmath_gather_end (void)`  
*Finish gathering emitted objects.*

- void `pmath_emit` (`pmath_t` object, `pmath_t` tag)  
*Emit an object to be gathered by the appropriate surrounding `pmath_gather_begin()` ... `pmath_gather_end()` function pair.*
- `pmath_t` `pmath_build_value_v` (const char \*format, va\_list args)  
*Generate a List of objects with a format string.*
- `pmath_t` `pmath_build_value` (const char \*format,...)  
*Generate a List of objects with a format string.*
- `pmath_expr_t` `pmath_options_extract` (`pmath_expr_t` expr, size\_t last\_nonoption)  
*Extract option values from an expression.*
- `pmath_t` `pmath_option_value` (`pmath_t` fn, `pmath_t` name, `pmath_t` extra)  
*Retrieve a option value of a given function.*

### 9.16.1 Detailed Description

Utility functions for pMath Objects and Expressions.

Here are some utility functions that simplify access to Expressions (or pMath Objects in general), but do not really fit one of these topics.

### 9.16.2 Typedef Documentation

#### 9.16.2.1 typedef `pmath_bool_t`(\* `pmath_stack_walker_t`)(`pmath_t` head, void \*closure)

A stack walker function.

The return value specifies, whether the walk on the stack go on.

### 9.16.3 Function Documentation

#### 9.16.3.1 `pmath_t` `pmath_build_value` (const char \**format*, ...) [related, inherited]

Generate a List of objects with a format string.

#### Parameters:

*format* A string that specifies the tuple's item's type. See below.

... The tuple/list items

The format string characters specify the item's type:

- `b [int]` Converts a C int to True or False.
- `i [int]`
- `l [long int]`
- `k [long long int]`
- `n [ssize_t]`
- `I [unsigned int]`
- `L [unsigned long int]`
- `K [unsigned long long int]`
- `N [size_t]`
- `f [double]` NaN's and Infinity are converted to the symbols Undefined and +/- Infinity respectively.
- `o [pmath_t]` A pMath object, the reference is stolen.
- `c [int]` Convert a C int representing a (unicode) character to a string of length 1.
- `s [char*]` Converts a zero-terminated C string to a pMath string using Latin-1 encoding.
- `s# [char*,int]` Takes a char buffer and a length to build a pMath string of that length using Latin-1 encoding.
- `z [char*]` Takes a zero-terminated C string and converts it to a symbol using [pmath\\_symbol\\_find\(\)](#).
- `u [char*]` Converts a zero-terminated C string to a pMath string using UTF-8 encoding.
- `u# [char*,int]` Takes a char buffer and a length to build a pMath string of that length using UTF-8 encoding.
- `U [uint16_t*]` Converts a zero-terminated double-byte C string to a pMath string using UTF-16 encoding. This is generally useful only where `sizeof(uint16_t) == sizeof(wchar_t)`, e.g. on Windows but not on Linux.

- `U# [uint16_t*,int]` Takes a character buffer and a length to build a pMath string of that length using UTF-16 encoding. This is generally useful only on platforms with `sizeof(uint16_t) == sizeof(wchar_t)`, e.g. on Windows but not on Linux.
- `Ctt [matching the 2 t's]` Build a complex value.
- `Qtt [matching the 2 t's]` Build a rational value.
- `(items) [matching items]` constructs a sublist of items.

**Note:**

When the format string denotes only one object, this object will be returned alone. So for a [pmath\\_t](#) `x`, `pmath_build_value("o", x) == x`. If you want to return a list in any case, use "...": "i" gives an integer, "ii" and "(ii)" a list of two integers and "(i)" a list of one integer.

### 9.16.3.2 `pmath_t pmath_build_value_v (const char *format, va_list args)` [related, inherited]

Generate a List of objects with a format string.

**Parameters:**

*format* A string that specifies the tuple's item's type.  
*args* A `va_list` - variable argument list.

**See also:**

[pmath\\_build\\_value](#)

### 9.16.3.3 `pmath_t pmath_current_head (void)`

Get the currently evaluated function.

**Returns:**

The head of the expression that is currently evaluated (in the calling thread). You have to destroy it.



#### 9.16.3.4 void `pmath_emit` (`pmath_t object`, `pmath_t tag`) [related, inherited]

Emit an object to be gathered by the appropriate surrounding `pmath_gather_begin()` ... `pmath_gather_end()` function pair.

##### Parameters:

*object* The object to be emitted, it will be freed.

*tag* A tag object. The surrounding Gather() with a pattern, that matches `tag` will collect the `object`. `tag` will be freed.

##### See also:

[pmath\\_gather\\_begin](#)  
[pmath\\_gather\\_end](#)

#### 9.16.3.5 void `pmath_gather_begin` (`pmath_t pattern`) [related, inherited]

Start gathering emitted objects.

##### Parameters:

*pattern* A pattern that is used to determine which emitted objects should be gathered (testing the emit-tag, not the object itself). It will be freed.

Use `pmath_gather_end()` to finish gathering. Calls to `pmath_gather_begin()` ... `pmath_gather_end()` can be nested.

The emit-and-gather mechanism is useful when you want to create a list but do not know its final length in advance.

##### See also:

[pmath\\_emit](#)

#### 9.16.3.6 `pmath_expr_t` `pmath_gather_end` (`void`) [related, inherited]

Finish gathering emitted objects.

##### Returns:

A list of all emitted objects since the last `pmath_gather_begin()` whose emit-tag matched the `pattern` parameter given to that `pmath_gather_begin()`. You must free it.

See also:

[pmath\\_emit](#)

#### 9.16.3.7 `pmath_bool_t pmath_is_expr_of (pmath_t obj, pmath_symbol_t head)`

Check if an object is an expression with a specified head.

**Parameters:**

*obj* A pMath object. It wont be freed.

*head* A pMath symbol. It wont be freed.

**Returns:**

TRUE iff *obj* is an expression with the given *head* symbol.

#### 9.16.3.8 `pmath_bool_t pmath_is_expr_of_len (pmath_t obj, pmath_symbol_t head, size_t length)`

Check if an object is an expression with a specified head and length.

**Parameters:**

*obj* A pMath object. It wont be freed.

*head* A pMath symbol. It wont be freed.

*length* The requested expression length.

**Returns:**

TRUE iff *obj* is an expression with the given *length* and *head* symbol.

#### 9.16.3.9 `pmath_t pmath_option_value (pmath_t fn, pmath_t name, pmath_t extra)` [*related*, *inherited*]

Retrieve a option value of a given function.

**Parameters:**

*fn* The function for which the requested option value is defined. It wont be freed.  
If it is PMATH\_NULL, the current head (see [pmath\\_current\\_head](#) ) will be used.

**name** The name of the option value (in general, a symbol). It wont be freed.

**extra** A list of extra option rules or PMATH\_UNDEFINED. It wont be freed. If it is not PMATH\_UNDEFINED, it must be a rule ('a->b', 'a:>b') or a list of rules.

**Returns:**

The requested option value.

**9.16.3.10** `pmath_expr_t pmath_options_extract (pmath_expr_t expr, size_t last_nonoption)` [related, inherited]

Extract option values from an expression.

**Parameters:**

**expr** The expression containing option values. It wont be freed.

**last\_nonoption** The index of the last argument that is not an option rule.

**Returns:**

A list of all given option values or PMATH\_NULL on error. You must destroy it.

Imagine, `expr = 'f(a,b,A->1,B->2)'` and `last_nonoption = 2`, then the return value is a list `{A->1, B->2}`. You can now use this return value as the `extra` parameter in `pmath_option_value()`.

If `last_nonoption` was 1, a message would be generated (b is no rule ...) and the return value is PMATH\_NULL. In that case, the calling function should have no further effects and return.

**9.16.3.11** `void pmath_walk_stack (pmath_stack_walker_t walker, void * closure)`

Walk up the current thread's and its parents' stack.

**Parameters:**

**walker** A callback function.

**closure** A pointer that will be provided to `walker` as the second argument.

## 9.17 Memory Management

Memory management for pMath.

## Functions

- void \* [pmath\\_mem\\_alloc](#) (size\_t size)  
*Allocate some amount of memory.*
- void \* [pmath\\_mem\\_realloc](#) (void \*p, size\_t new\_size)  
*Change the size of a memory-chunk.*
- void \* [pmath\\_mem\\_realloc\\_no\\_failfree](#) (void \*p, size\_t new\_size)  
*Change the size of a memory-chunk.*
- void [pmath\\_mem\\_free](#) (void \*p)  
*Free a memory-chunk.*
- void [pmath\\_mem\\_usage](#) (size\_t \*current, size\_t \*max)  
*Get memory usage information.*

### 9.17.1 Detailed Description

Memory management for pMath.

These functions may return NULL. In this case, the current evaluation will abort and used cache will be freed to safe memory (the garbage collector is invoked and a pMath exception is thrown so [pmath\\_aborting\(\)](#) yields TRUE).

### 9.17.2 Function Documentation

#### 9.17.2.1 void\* [pmath\\_mem\\_alloc](#) (size\_t size)

Allocate some amount of memory.

##### Parameters:

*size* The number of bytes to be allocated.

##### Returns:

A pointer to a block of mamory of at least size bytes or NULL.

You must free the result with [pmath\\_mem\\_free\(\)](#) or indirectly via [pmath\\_mem\\_realloc\(\)](#).

#### 9.17.2.2 void [pmath\\_mem\\_free](#) (void \*p)

Free a memory-chunk.

**Parameters:**

*p* NULL or a pointer to a block of *old\_size* bytes allocated with [pmath\\_mem\\_alloc\(\)](#) or [pmath\\_mem\\_realloc\(\)](#).

**9.17.2.3 void\* pmath\_mem\_realloc (void \**p*, size\_t *new\_size*)**

Change the size of a memory-chunk.

**Parameters:**

*p* NULL or a pointer to a block of *old\_size* bytes allocated with [pmath\\_mem\\_alloc\(\)](#) or [pmath\\_mem\\_realloc\(\)](#).

*new\_size* The requested new size.

**Returns:**

A pointer to a block of at least *new\_size* bytes or NULL.

If there is not enough memory available or if *new\_size* == 0, NULL is returned. Otherwise, the result points to a block of *new\_size* bytes, whose first min(*old\_size*,*new\_size*) bytes are copied from the old *p*. The rest is initialized with 0.

The old pointer *p* will *\_always\_* be freed. even, if the resizing failed with NULL.

You must later free the result with [pmath\\_mem\\_free\(\)](#) or indirectly via [pmath\\_mem\\_realloc\(\)](#).

**9.17.2.4 void\* pmath\_mem\_realloc\_no\_failfree (void \**p*, size\_t *new\_size*)**

Change the size of a memory-chunk.

**Parameters:**

*p* NULL or a pointer to a block of *old\_size* bytes allocated with [pmath\\_mem\\_alloc\(\)](#) or [pmath\\_mem\\_realloc\(\)](#).

*new\_size* The requested new size.

**Returns:**

A pointer to a block of at least *new\_size* bytes or NULL.

If there is enough memory, this acts like [pmath\\_mem\\_realloc\(\)](#). Otherwise, *p* is *not* freed and NULL is returned.

### 9.17.2.5 void pmath\_mem\_usage (size\_t \* *current*, size\_t \* *max*)

Get memory usage information.

#### Parameters:

*current* Here goes the number of currently allocated bytes.

*max* here goes the maximum number of allocated bytes so far.

## 9.18 Messages

Error handling and informing the user.

### Functions

- void [pmath\\_message](#) ([pmath\\_symbol\\_t](#) symbol, const char \*tag, size\_t argcount,...)  
*Print a message with pMath object arguments.*
- void [pmath\\_message\\_argxxx](#) (size\_t given, size\_t min, size\_t max)  
*Generate a General::arg\* message (invalid argument count).*
- [pmath\\_string\\_t](#) [pmath\\_message\\_find\\_text](#) ([pmath\\_t](#) name)  
*Find a message's text.*
- void [pmath\\_message\\_syntax\\_error](#) ([pmath\\_string\\_t](#) code, int position, [pmath\\_string\\_t](#) filename, int lines\_before\_code)  
*Print a syntax warning or error message.*

### 9.18.1 Detailed Description

Error handling and informing the user.

When you encounter an error such as wrong argument usage in pMath functions, you just print out a message and return the handled expression unevaluated. You do *not* call [pmath\\_abort\\_please\(\)](#).

Example: The built-in Sin function invokes Sin::arg1 and returns unevaluated when it is called with a wrong number of arguments.

If you notice that a memory allocation failed, do nothing at all (even do not try to print a message). pMath automatically calls [pmath\\_abort\\_please\(\)](#) on out-of-memory and thus no message would be shown.

Messages are similar to Mathematicas approach. On the language level, you enter 'Message(symbol::tag, ...)' to print a message with optional inserted values '...'. You can use Backquotes to refer to given values.

You can use all messages of the symbol General with every other symbol.

### 9.18.2 Function Documentation

#### 9.18.2.1 void pmath\_message (pmath\_symbol\_t *symbol*, const char \* *tag*, size\_t *argcount*, ...)

Print a message with pMath object arguments.

##### Parameters:

*symbol* The symbol that defines the message. It wont be freed. PMATH\_NULL is be treated as [pmath\\_current\\_head\(\)](#).

*tag* The message's tag as a zero-terminated C string.

*argcount* The number of following arguments.

... Exactly *argcount* pMath objects. They will all be freed.

##### Note:

If `symbol==PMATH_NULL` and [pmath\\_current\\_head\(\)](#) is an expression `f()`, the function acts as if [pmath\\_current\\_head\(\)](#) was `f`.

All the symbols and expressions in ... will be embedded in `HoldForm(...)`, because `Message()` would evaluate them. If you want one of the values to be evaluated, do it manually.

#### 9.18.2.2 void pmath\_message\_argxxx (size\_t *given*, size\_t *min*, size\_t *max*)

Generate a General::arg\* message (invalid argument count).

##### Parameters:

*given* The given number of arguments.

*min* The minimum expected number of arguments.

*max* The maximum expected number of arguments.

One of the following messages may be generated: General::arg1

General::argxu

General::argx

General::argtu

General::argt

General::argru

General::argr

General::argmu

General::argm

### 9.18.2.3 `pmath_string_t pmath_message_find_text (pmath_t name)`

Find a message's text.

#### Parameters:

*name* An expression of the form `symbol::name` (that is `MessageName(symbol, "name")`). It will be freed.

#### Returns:

The message's content or `PMATH_NULL` if nothing was found or the [magic value `PMATH\_UNDEFINED`](#), if *name* does not have the expected form.

This function can be used in front-ends that overwrite the built-in `Message` function.

### 9.18.2.4 `void pmath_message_syntax_error (pmath_string_t code, int position, pmath_string_t filename, int lines_before_code)`

Print a syntax warning or error message.

#### Parameters:

*code* The code. A pMath string. It wont be freed.

*position* The position of the syntax error in the code.

*filename* The file from where the input came or `PMATH_NULL` to omit it. It will be freed.

*lines\_before\_code* The number of lines in the input file before *code* was read. This is ignored if *filename* is `PMATH_NULL`.

This function produces a `Syntax::bgn`, `Syntax::bgnf`, `Syntax::nxt`, `Syntax::nxtf`, `Syntax::more`, `Syntax::moref`, `Syntax::newl` or `Syntax::newlf` message, depending on where the syntax error is in the *code* and whether *filename* is `PMATH_NULL` or not. It can be used to report errors/warnings from [`pmath\_spans\_from\_string\(\)`](#).

## 9.19 Threadsafe Stacks

Fast threadsafe stacks in pMath.



## Data Structures

- class `pmath_stack_t`  
*The type of pMath's threadsafe stacks.*

## Typedefs

- typedef struct `_pmath_stack_t` \* `pmath_stack_t`

## Functions

- `pmath_stack_t` `pmath_stack_new` (void)  
*Create an empty stack.*
- void `pmath_stack_free` (`pmath_stack_t` stack)  
*Destroy a stack.*
- void `pmath_stack_push` (`pmath_stack_t` stack, void \*item)  
*Push an item onto a stack.*
- void \* `pmath_stack_pop` (`pmath_stack_t` stack)  
*Pop an item from a stack.*

### 9.19.1 Detailed Description

Fast threadsafe stacks in pMath.

`pmath_stack_t` is a stack abstraction that provides threadsafe push and pop operations. You can push and pop any structure whose first element is a pointer (which you must not touch).

If your CPU supports it, very fast lockfree operations are used for push and pop.

#### References

- Fober, Orlarey, Letz: "Lock-Free Techniques for Concurrent Access to Shared Objects" ([http://jim.afim-asso.org/jim2002/articles/L17\\_Fober.pdf](http://jim.afim-asso.org/jim2002/articles/L17_Fober.pdf))

### 9.19.2 Typedef Documentation

#### 9.19.2.1 typedef struct `_pmath_stack_t`\* `pmath_stack_t`

### 9.19.3 Function Documentation

#### 9.19.3.1 void pmath\_stack\_free (pmath\_stack\_t *stack*)

Destroy a stack.

**Parameters:**

*stack* A stack previously created with [pmath\\_stack\\_new\(\)](#). May be PMATH\_NULL.

You must manually pop and free all items on the stack before calling this function, because those items would not be freed automatically.

#### 9.19.3.2 pmath\_stack\_t pmath\_stack\_new (void)

Create an empty stack.

**Returns:**

A new stack or PMATH\_NULL.

Note that you cannot push anything onto a PMATH\_NULL stack, so check the result. Free the result with [pmath\\_stack\\_free\(\)](#).

#### 9.19.3.3 void\* pmath\_stack\_pop (pmath\_stack\_t *stack*)

Pop an item from a stack.

**Parameters:**

*stack* The stack to pop an item from. Must not be PMATH\_NULL.

**Returns:**

The top of stack or PMATH\_NULL if it is empty.

#### 9.19.3.4 void pmath\_stack\_push (pmath\_stack\_t *stack*, void \* *item*)

Push an item onto a stack.

**Parameters:**

*stack* The stack to where the item should be pushed. Must not be PMATH\_NULL.

*item* The item to be pushed. It must point to a structure whose first element is a pointer. Must not be PMATH\_NULL.

## 9.20 Version Information

### Functions

- void [pmath\\_version\\_datetime](#) (int \*year, int \*month, int \*day, int \*hour, int \*minute, int \*second)

*Get the date and time when pMath was compiled.*

- double [pmath\\_version\\_number](#) (void)

*Get version number (major + minor/100).*

- long [pmath\\_version\\_number\\_part](#) (int index)

*Get version number part.*

### 9.20.1 Function Documentation

#### 9.20.1.1 void [pmath\\_version\\_datetime](#) (int \* *year*, int \* *month*, int \* *day*, int \* *hour*, int \* *minute*, int \* *second*)

Get the date and time when pMath was compiled.

#### 9.20.1.2 double [pmath\\_version\\_number](#) (void)

Get version number (major + minor/100).

#### 9.20.1.3 long [pmath\\_version\\_number\\_part](#) (int *index*)

Get version number part.

### Parameters:

*index* The number index. Major=1, Minor=2, Revision=3, Subversion=4

## 9.21 Front-ends

Functions for use front-ends.

## Functions

- [pmath\\_bool\\_t pmath\\_continue\\_after\\_abort](#) (void)  
*Requests pMath to stop aborting the current evaluation.*
- [pmath\\_t pmath\\_session\\_execute](#) (pmath\_t input, pmath\_bool\_t \*aborted)  
*Execute an expression and change \$History and \$Line appropriately.*
- [pmath\\_t pmath\\_session\\_start](#) (void)  
*Saves some global state when an interactive dialog session starts.*
- void [pmath\\_session\\_end](#) (pmath\_t old\_state)  
*Restore some global state when an interactive dialog session ends.*
- [pmath\\_bool\\_t pmath\\_init](#) (void)  
*Initialize the pMath CAS library.*
- void [pmath\\_done](#) (void)  
*Free all resources used by the pMath CAS library and unload all modules.*

### 9.21.1 Detailed Description

Functions for use front-ends.

A front-end to pMath is an executable that initializes and finalizes the library and handles User input and output or otherwise invokes expression evaluation with the library. That is, what pMath does: parse and evaluate pMath code.

### 9.21.2 Function Documentation

#### 9.21.2.1 [pmath\\_bool\\_t pmath\\_continue\\_after\\_abort](#) (void)

Requests pMath to stop aborting the current evaluation.

#### Returns:

Whether the global aborting-flag was set (by [pmath\\_abort\\_please\(\)](#))

This is for use in front-ends to allow the user to continue working after he aborted an evaluation.

Any uncought exception will be deleted silently.

This function also clears the \$MessageCount cache.

#### See also:

[pmath\\_abort\\_please\(\)](#)

### 9.21.2.2 void pmath\_done (void)

Free all resources used by the pMath CAS library and unload all modules.

See also:

[pmath\\_init](#)

### 9.21.2.3 pmath\_bool\_t pmath\_init (void)

Initialize the pMath CAS library.

Returns:

TRUE, if the initialization was successfull, FALSE otherwise.

Any thread must call [pmath\\_init\(\)](#) before using any other pMath function (exception: [Atomic Operations](#)). If the initialization fails, the thread should stop. Otherwise, [pmath\\_done\(\)](#) must be called before the thread ends.

When this function is called for the first time, the `maininit.pmath` file is executed. This file is searched in the following directories in order:

1. The application directory
2. The directory specified in the `PMATH_BASEDIRECTORY` environment variable
3. Some operating system specific directories:
  - `/etc/pmath`, `/etc/local/pmath`, `/usr/share/pmath`, `/usr/share/local/pmath` on Unix like systems
  - The "pmath" subfolder in the "common program files" and "program files" directories on Microsoft Windows, if existent.

### 9.21.2.4 void pmath\_session\_end (pmath\_t *old\_state*)

Restore some global state when an interactive dialog session ends.

Parameters:

*old\_state* The object returned by [pmath\\_session\\_start\(\)](#). It will be freed.

This function should be used by a frontend when implementing the `Dialog()` function.

### 9.21.2.5 `pmath_t pmath_session_execute (pmath_t input, pmath_bool_t * aborted)`

Execute an expression and change \$History and \$Line appropriately.

#### Parameters:

*input* The input expression. It will be freed.

*aborted* Optional address of a flag that will be set iff the evaluation was aborted  
by [pmath\\_abort\\_please\(\)](#)

#### Returns:

The return value is the same as `pmath_evaluate(input)`

This function also calls `pmath_collect_temporary_symbols` before evaluation.

### 9.21.2.6 `pmath_t pmath_session_start (void)`

Saves some global state when an interactive dialog session starts.

#### Returns:

A pMath object to be given to [pmath\\_session\\_end\(\)](#)

This function should be used by a frontend when implementing the `Dialog()` function.

## 10 Namespace Documentation

### 10.1 pmath Namespace Reference

Provides the C++ binding.

#### Data Structures

- class [Expr](#)  
*A wrapper for [pmath\\_t](#) and derived types.*
- class [String](#)  
*A wrapper for [pmath\\_string\\_t](#).*
- class [Gather](#)  
*Utility class for emitting and gathering expressions/building lists.*

- class [File](#)  
*A wrapper for pMath file objects (data streams).*
- class [BinaryFile](#)  
*A wrapper for pMath binary file objects (byte data streams).*
- class [ReadableBinaryFile](#)  
*A wrapper for readable pMath binary file objects (byte data streams).*
- class [WritableBinaryFile](#)  
*A wrapper for writable pMath binary file objects (byte data streams).*
- class [TextFile](#)  
*A wrapper for pMath text file objects (byte data streams).*
- class [ReadableTextFile](#)  
*A wrapper for pMath readable text file objects (byte data streams).*
- class [WritableTextFile](#)  
*A wrapper for pMath writable text file objects (byte data streams).*
- class [UserStream](#)  
*Abstract base class for C++ callbacks used as pMath files.*
- class [BinaryUserStream](#)  
*Abstract base class for C++ callbacks used as pMath binary files.*
- class [TextUserStream](#)  
*Abstract base class for C++ callbacks used as pMath text files.*

## Functions

- [Expr Number](#) (double d)
- [Expr Complex](#) (Expr re, Expr im)
- [Expr Imaginary](#) (Expr im)
- [Expr Rational](#) (Expr num, Expr den)
- [Expr Ref](#) (pmath\_t o)
- [Expr Symbol](#) (pmath\_symbol\_t h)
- [Expr SymbolPi](#) ()
- [Expr MakeList](#) (size\_t len)
- [Expr Call](#) (Expr h)
- [Expr Call](#) (Expr h, Expr x1)
- [Expr Call](#) (Expr h, Expr x1, Expr x2)
- [Expr Call](#) (Expr h, Expr x1, Expr x2, Expr x3)

- [Expr Call \(Expr h, Expr x1, Expr x2, Expr x3, Expr x4\)](#)
- [Expr Call \(Expr h, Expr x1, Expr x2, Expr x3, Expr x4, Expr x5\)](#)
- [Expr Call \(Expr h, Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6\)](#)
- [Expr Call \(Expr h, Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6, Expr x7\)](#)
- [Expr Call \(Expr h, Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6, Expr x7, Expr x8\)](#)
- [Expr Call \(Expr h, Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6, Expr x7, Expr x8, Expr x9\)](#)
- [Expr List \(\)](#)
- [Expr List \(Expr x1\)](#)
- [Expr List \(Expr x1, Expr x2\)](#)
- [Expr List \(Expr x1, Expr x2, Expr x3\)](#)
- [Expr List \(Expr x1, Expr x2, Expr x3, Expr x4\)](#)
- [Expr List \(Expr x1, Expr x2, Expr x3, Expr x4, Expr x5\)](#)
- [Expr List \(Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6\)](#)
- [Expr List \(Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6, Expr x7\)](#)
- [Expr List \(Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6, Expr x7, Expr x8\)](#)
- [Expr List \(Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6, Expr x7, Expr x8, Expr x9\)](#)
- [Expr Rule \(Expr l, Expr r\)](#)
- [Expr RuleDelayed \(Expr l, Expr r\)](#)
- [Expr Power \(Expr x, Expr y\)](#)
- [Expr Sqrt \(Expr x\)](#)
- [Expr Inv \(Expr x\)](#)
- [Expr Exp \(Expr x\)](#)
- [Expr Log \(Expr x\)](#)
- [Expr Log \(Expr b, Expr x\)](#)
- [Expr Sin \(Expr x\)](#)
- [Expr Cos \(Expr x\)](#)
- [Expr Tan \(Expr x\)](#)
- [Expr ArcSin \(Expr x\)](#)
- [Expr ArcCos \(Expr x\)](#)
- [Expr ArcTan \(Expr x\)](#)
- [Expr Times \(Expr x1, Expr x2\)](#)
- [Expr Times \(Expr x1, Expr x2, Expr x3\)](#)
- [Expr Times \(Expr x1, Expr x2, Expr x3, Expr x4\)](#)
- [Expr Divide \(Expr x, Expr y\)](#)
- [Expr Plus \(Expr x1, Expr x2\)](#)
- [Expr Plus \(Expr x1, Expr x2, Expr x3\)](#)
- [Expr Plus \(Expr x1, Expr x2, Expr x3, Expr x4\)](#)
- [Expr Minus \(Expr x\)](#)
- [Expr Minus \(Expr x, Expr y\)](#)
- [Expr Abs \(Expr x\)](#)
- [Expr Arg \(Expr x\)](#)
- [Expr Sign \(Expr x\)](#)



- [Expr Re \(Expr x\)](#)
- [Expr Im \(Expr x\)](#)
- [Expr Ceiling \(Expr x\)](#)
- [Expr Ceiling \(Expr x, Expr a\)](#)
- [Expr Floor \(Expr x\)](#)
- [Expr Floor \(Expr x, Expr a\)](#)
- [Expr Round \(Expr x\)](#)
- [Expr Round \(Expr x, Expr a\)](#)
- [Expr Quotient \(Expr m, Expr n\)](#)
- [Expr Quotient \(Expr m, Expr n, Expr d\)](#)
- [Expr Mod \(Expr m, Expr n\)](#)
- [Expr Mod \(Expr m, Expr n, Expr d\)](#)
- [Expr Evaluate \(Expr x\)](#)
- [Expr ParseArgs \(const char \\*code, Expr arglist\)](#)
- [Expr Parse \(String code\)](#)
- [Expr Parse \(const char \\*code\)](#)
- [Expr Parse \(const char \\*code, Expr x1\)](#)
- [Expr Parse \(const char \\*code, Expr x1, Expr x2\)](#)
- [Expr Parse \(const char \\*code, Expr x1, Expr x2, Expr x3\)](#)
- [Expr Parse \(const char \\*code, Expr x1, Expr x2, Expr x3, Expr x4\)](#)

### 10.1.1 Detailed Description

Provides the C++ binding.

### 10.1.2 Function Documentation

**10.1.2.1 Expr pmath::Abs (Expr x)** `[inline]`

**10.1.2.2 Expr pmath::ArcCos (Expr x)** `[inline]`

**10.1.2.3 Expr pmath::ArcSin (Expr x)** `[inline]`

**10.1.2.4 Expr pmath::ArcTan (Expr x)** `[inline]`

**10.1.2.5** Expr pmath::Arg (Expr *x*) [inline]

**10.1.2.6** Expr pmath::Call (Expr *h*, Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*, Expr *x5*, Expr *x6*, Expr *x7*, Expr *x8*, Expr *x9*) [inline]

**10.1.2.7** Expr pmath::Call (Expr *h*, Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*, Expr *x5*, Expr *x6*, Expr *x7*, Expr *x8*) [inline]

**10.1.2.8** Expr pmath::Call (Expr *h*, Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*, Expr *x5*, Expr *x6*, Expr *x7*) [inline]

**10.1.2.9** Expr pmath::Call (Expr *h*, Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*, Expr *x5*, Expr *x6*) [inline]

**10.1.2.10** Expr pmath::Call (Expr *h*, Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*, Expr *x5*) [inline]

**10.1.2.11** Expr pmath::Call (Expr *h*, Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*) [inline]

**10.1.2.12** Expr pmath::Call (Expr *h*, Expr *x1*, Expr *x2*, Expr *x3*) [inline]

**10.1.2.13** Expr pmath::Call (Expr *h*, Expr *x1*, Expr *x2*) [inline]

**10.1.2.14** Expr pmath::Call (Expr *h*, Expr *xI*) [inline]

**10.1.2.15** Expr pmath::Call (Expr *h*) [inline]

**10.1.2.16** Expr pmath::Ceiling (Expr *x*, Expr *a*) [inline]

**10.1.2.17** Expr pmath::Ceiling (Expr *x*) [inline]

**10.1.2.18** Expr pmath::Complex (Expr *re*, Expr *im*) [inline]

**10.1.2.19** Expr pmath::Cos (Expr *x*) [inline]

**10.1.2.20** Expr pmath::Divide (Expr *x*, Expr *y*) [inline]

**10.1.2.21** Expr pmath::Evaluate (Expr *x*) [inline]

**10.1.2.22** Expr pmath::Exp (Expr *x*) [inline]

**10.1.2.23** Expr pmath::Floor (Expr *x*, Expr *a*) [inline]

**10.1.2.24** Expr pmath::Floor (Expr *x*) [inline]

**10.1.2.25** Expr pmath::Im (Expr *x*) [inline]

**10.1.2.26** Expr pmath::Imaginary (Expr *im*) [inline]

**10.1.2.27** Expr pmath::Inv (Expr *x*) [inline]

**10.1.2.28** Expr pmath::List (Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*, Expr *x5*,  
Expr *x6*, Expr *x7*, Expr *x8*, Expr *x9*) [inline]

**10.1.2.29** Expr pmath::List (Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*, Expr *x5*,  
Expr *x6*, Expr *x7*, Expr *x8*) [inline]

**10.1.2.30** Expr pmath::List (Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*, Expr *x5*,  
Expr *x6*, Expr *x7*) [inline]

**10.1.2.31** Expr pmath::List (Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*, Expr *x5*,  
Expr *x6*) [inline]

**10.1.2.32** Expr pmath::List (Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*, Expr *x5*)  
[inline]

**10.1.2.33** Expr pmath::List (Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*) [inline]

**10.1.2.34** Expr pmath::List (Expr *x1*, Expr *x2*, Expr *x3*) [inline]

**10.1.2.35** Expr pmath::List (Expr *x1*, Expr *x2*) [inline]

**10.1.2.36** Expr pmath::List (Expr *x1*) [inline]

**10.1.2.37** Expr pmath::List () [inline]

**10.1.2.38** Expr pmath::Log (Expr *b*, Expr *x*) [inline]

**10.1.2.39** Expr pmath::Log (Expr *x*) [inline]

**10.1.2.40** Expr pmath::MakeList (size\_t *len*) [inline]

**10.1.2.41** Expr pmath::Minus (Expr *x*, Expr *y*) [inline]

**10.1.2.42** Expr pmath::Minus (Expr *x*) [inline]

**10.1.2.43** Expr pmath::Mod (Expr *m*, Expr *n*, Expr *d*) [inline]

**10.1.2.44** Expr pmath::Mod (Expr *m*, Expr *n*) [inline]

**10.1.2.45** Expr pmath::Number (double *d*) [inline]

**10.1.2.46** Expr pmath::Parse (const char \* *code*, Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*) [inline]

**10.1.2.47** Expr pmath::Parse (const char \* *code*, Expr *x1*, Expr *x2*, Expr *x3*) [inline]

**10.1.2.48** Expr pmath::Parse (const char \* *code*, Expr *x1*, Expr *x2*) [inline]

**10.1.2.49** Expr pmath::Parse (const char \* *code*, Expr *x1*) [inline]

**10.1.2.50** Expr pmath::Parse (const char \* *code*) [inline]

**10.1.2.51** Expr pmath::Parse (String *code*) [inline]

**10.1.2.52** Expr pmath::ParseArgs (const char \* *code*, Expr *arglist*) [inline]

**10.1.2.53** Expr pmath::Plus (Expr *x1*, Expr *x2*, Expr *x3*, Expr *x4*) [inline]

**10.1.2.54** Expr pmath::Plus (Expr *x1*, Expr *x2*, Expr *x3*) [inline]

**10.1.2.55** Expr pmath::Plus (Expr *x1*, Expr *x2*) [inline]

**10.1.2.56** Expr pmath::Power (Expr *x*, Expr *y*) [inline]

**10.1.2.57** Expr pmath::Quotient (Expr *m*, Expr *n*, Expr *d*) [inline]

**10.1.2.58** Expr pmath::Quotient (Expr *m*, Expr *n*) [inline]

**10.1.2.59** Expr pmath::Rational (Expr *num*, Expr *den*) [inline]

**10.1.2.60** Expr pmath::Re (Expr *x*) [inline]

**10.1.2.61** Expr pmath::Ref (pmath\_t *o*) [inline]

**10.1.2.62** Expr pmath::Round (Expr *x*, Expr *a*) [inline]

**10.1.2.63** Expr pmath::Round (Expr *x*) [inline]

**10.1.2.64** Expr pmath::Rule (Expr *l*, Expr *r*) [inline]

**10.1.2.65** Expr pmath::RuleDelayed (Expr *l*, Expr *r*) [inline]

**10.1.2.66** Expr pmath::Sign (Expr *x*) [inline]

**10.1.2.67** `Expr pmath::Sin (Expr x)` `[inline]`

**10.1.2.68** `Expr pmath::Sqrt (Expr x)` `[inline]`

**10.1.2.69** `Expr pmath::Symbol (pmath_symbol_t h)` `[inline]`

**10.1.2.70** `Expr pmath::SymbolPi ()` `[inline]`

**10.1.2.71** `Expr pmath::Tan (Expr x)` `[inline]`

**10.1.2.72** `Expr pmath::Times (Expr x1, Expr x2, Expr x3, Expr x4)`  
`[inline]`

**10.1.2.73** `Expr pmath::Times (Expr x1, Expr x2, Expr x3)` `[inline]`

**10.1.2.74** `Expr pmath::Times (Expr x1, Expr x2)` `[inline]`

## 11 Data Structure Documentation

### 11.1 BinaryFile Class Reference

A wrapper for pMath binary file objects (byte data streams).

Inherits [pmath::File](#).

Inherited by [ReadableBinaryFile](#), and [WriteableBinaryFile](#).



### Public Member Functions

- [BinaryFile](#) ()
- [BinaryFile](#) ([pmath\\_t](#) file\_object) throw ()
- [BinaryFile](#) (const [Expr](#) &file\_object) throw ()
- [BinaryFile](#) (const [BinaryFile](#) &src) throw ()
- [size\\_t](#) [get\\_buffer\\_size](#) () const throw ()

*Get the current buffer size in bytes.*

- [bool](#) [set\\_buffer\\_size](#) ([size\\_t](#) size) throw ()  
*See [file\\_set\\_binbuffer\(\)](#).*

### Static Public Member Functions

- static [BinaryFile](#) [create\\_buffer](#) ([size\\_t](#) mincapacity) throw ()  
*Create a memory buffer for as a double ended queue.*

#### 11.1.1 Detailed Description

A wrapper for pMath binary file objects (byte data streams).

#### 11.1.2 Constructor & Destructor Documentation

##### 11.1.2.1 [BinaryFile](#) () `[inline]`

##### 11.1.2.2 [BinaryFile](#) ([pmath\\_t](#) *file\_object*) throw () `[inline, explicit]`

##### 11.1.2.3 [BinaryFile](#) (const [Expr](#) & *file\_object*) throw () `[inline, explicit]`

##### 11.1.2.4 [BinaryFile](#) (const [BinaryFile](#) & *src*) throw () `[inline]`

### 11.1.3 Member Function Documentation

#### 11.1.3.1 static BinaryFile create\_buffer (size\_t mincapacity) throw () [inline, static]

Create a memory buffer for as a double ended queue.

**Returns:**

The binary file. Can be used as [ReadableBinaryFile](#) and as [WriteableBinaryFile](#)

#### 11.1.3.2 size\_t get\_buffer\_size () const throw () [inline]

Get the current buffer size in bytes.

#### 11.1.3.3 bool set\_buffer\_size (size\_t size) throw () [inline]

See file\_set\_binbuffer().

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

- [pmath-cpp.h](#)

## 11.2 BinaryUserStream Class Reference

Abstract base class for C++ callbacks used as pMath binary files.

Inherits [pmath::UserStream](#).

### Protected Member Functions

- virtual [pmath\\_files\\_status\\_t](#) status ()=0  
*Called by pMath to check for end-of-file and other errors.*
- virtual void [flush](#) ()  
*Called by pMath to flush data to disk.*
- virtual size\_t [read](#) (void \*buffer, size\_t buffer\_size)=0  
*Called by pMath to read data.*
- virtual size\_t [write](#) (const void \*buffer, size\_t buffer\_size)=0  
*Called by pMath to write data.*

- [BinaryFile convert\\_to\\_file](#) (bool readable, bool writeable)  
*Convert to a binary file. pMath will take ownership of the C++ object.*
- [ReadableBinaryFile convert\\_to\\_file\\_readonly](#) ()
- [WritableBinaryFile convert\\_to\\_file\\_writeonly](#) ()

### 11.2.1 Detailed Description

Abstract base class for C++ callbacks used as pMath binary files.

### 11.2.2 Member Function Documentation

#### 11.2.2.1 [BinaryFile convert\\_to\\_file](#) (bool *readable*, bool *writeable*) [inline, protected]

Convert to a binary file. pMath will take ownership of the C++ object.

Because pMath now owns the C++ object, you must not touch it directly after this call

#### 11.2.2.2 [ReadableBinaryFile convert\\_to\\_file\\_readonly](#) () [inline, protected]

#### 11.2.2.3 [WritableBinaryFile convert\\_to\\_file\\_writeonly](#) () [inline, protected]

#### 11.2.2.4 [virtual void flush](#) () [inline, protected, virtual]

Called by pMath to flush data to disk.

#### 11.2.2.5 [virtual size\\_t read](#) (void \* *buffer*, size\_t *buffer\_size*) [protected, pure virtual]

Called by pMath to read data.

**11.2.2.6** `virtual pmath_files_status_t status ()` [protected, pure virtual]

Called by pMath to check for end-of-file and other errors.

**11.2.2.7** `virtual size_t write (const void * buffer, size_t buffer_size)`  
[protected, pure virtual]

Called by pMath to write data.

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

- [pmath-cpp.h](#)

## 11.3 Expr Class Reference

A wrapper for [pmath\\_t](#) and derived types.

Inherited by [File](#), and [String](#).

### Public Member Functions

- [Expr](#) () throw ()  
*Initialize with PMATH\_NULL.*
- [Expr](#) ([pmath\\_t](#) obj) throw ()  
*Construct from a [pmath\\_t](#), that will be freed automatically with the [Expr](#).*
- [Expr](#) (const [Expr](#) &src) throw ()  
*Copy an [Expr](#), incrementing the reference counter.*
- [Expr](#) (int i) throw ()  
*Construct from an int.*
- [Expr](#) (double f) throw ()  
*Construct from a double. May yield Infinity or Undefined (NaN) values.*
- [~Expr](#) () throw ()  
*Destructor. Frees the wrapped object.*
- [Expr](#) & [operator=](#) (const [Expr](#) &src) throw ()  
*Copy an [Expr](#). Increments the new value's reference counter and frees the old one.*
- bool [is\\_custom](#) () const throw ()

- bool [is\\_double](#) () const throw ()
- bool [is\\_expr](#) () const throw ()
- bool [is\\_float](#) () const throw ()
- bool [is\\_integer](#) () const throw ()
- bool [is\\_int32](#) () const throw ()
- bool [is\\_magic](#) () const throw ()
- bool [is\\_mpffloat](#) () const throw ()
- bool [is\\_null](#) () const throw ()
- bool [is\\_number](#) () const throw ()
- bool [is\\_pointer](#) () const throw ()
- bool [is\\_quotient](#) () const throw ()
- bool [is\\_rational](#) () const throw ()
- bool [is\\_string](#) () const throw ()
- bool [is\\_symbol](#) () const throw ()
- bool [is\\_pointer\\_of](#) ([pmath\\_type\\_t](#) type) const throw ()
- bool [is\\_evaluated](#) () const throw ()
- unsigned int [hash](#) () const throw ()
 

*Get a hash value.*
- bool [operator==](#) (const [Expr](#) &other) const throw ()
 

*Check for identity. The pMath == operator.*
- bool [operator!=](#) (const [Expr](#) &other) const throw ()
 

*Check for non-identity. The pMath != operator.*
- int [compare](#) (const [Expr](#) &other) const throw ()
 

*Compare with another Expr. See [pmath\\_compare\(\)](#).*
- bool [operator<](#) (const [Expr](#) &other) const throw ()
 

*Compare with another Expr. See [pmath\\_compare\(\)](#).*
- bool [operator<=](#) (const [Expr](#) &other) const throw ()
 

*Compare with another Expr. See [pmath\\_compare\(\)](#).*
- bool [operator>](#) (const [Expr](#) &other) const throw ()
 

*Compare with another Expr. See [pmath\\_compare\(\)](#).*
- bool [operator>=](#) (const [Expr](#) &other) const throw ()
 

*Compare with another Expr. See [pmath\\_compare\(\)](#).*
- [pmath\\_t](#) [release](#) () throw ()
 

*Return the [pmath\\_t](#) and discard it. Caller must [pmath\\_unref\(\)](#) it.*
- const [pmath\\_t](#) [get](#) () const throw ()
 

*Get the [pmath\\_t](#). Reference is held by the Expr object.*

- `bool is_valid () const throw ()`  
*Check for not holding the null pointer.*
- `size_t expr_length () const throw ()`  
*Length of the expression or 0 on error.*
- `Expr operator[] (size_t i) const throw ()`  
*Get the i-th argument of the expression.*
- `void set (size_t i, Expr e) throw ()`  
*Change the i-th argument of an expression.*
- `void set (size_t i, size_t j, Expr e) throw ()`  
*Change the (i,j)-th argument of a matrix.*
- `double to_double (double def=0.0) const throw ()`  
*Convert to a double.*
- `String to_string (pmath_write_options_t options=0) const throw ()`  
*Convert to a string. The pMath ToString function.*
- `void write_to_file (WriteableTextFile file, pmath_write_options_t options=0) const throw ()`  
*Write to a file/text stream.*
- `pmath_serialize_error_t serialize (WriteableBinaryFile file) const throw ()`  
*Serialize to a binary file/stream.*
- `bool operator== (pmath_t o1, const Expr &o2) throw()`  
*check for identity. The pMath === operator.*
- `bool operator!= (pmath_t o1, const Expr &o2) throw()`  
*check for non-identity. The pMath != operator.*
- `bool operator== (const Expr &o1, pmath_t o2) throw()`
- `bool operator!= (const Expr &o1, pmath_t o2) throw()`

#### Static Public Member Functions

- `static Expr deserialize (ReadableBinaryFile file, pmath_serialize_error_t *error) throw ()`  
*Deserialize an Expr from a binary file/stream.*

#### Static Protected Member Functions

- `static void write_to_string (void *user, const uint16_t *data, int len) throw ()`

### Protected Attributes

- [pmath\\_t\\_obj](#)

#### 11.3.1 Detailed Description

A wrapper for [pmath\\_t](#) and derived types.

This class wraps a single [pmath\\_t](#), so its `sizeof(Expr) == sizeof(pmath_t)`.

#### 11.3.2 Constructor & Destructor Documentation

##### 11.3.2.1 `Expr () throw ()` [inline]

Initialize with `PMATH_NULL`.

##### 11.3.2.2 `Expr (pmath_t obj) throw ()` [inline, explicit]

Construct from a [pmath\\_t](#), that will be freed automatically with the [Expr](#).

##### 11.3.2.3 `Expr (const Expr & src) throw ()` [inline]

Copy an [Expr](#), incrementing the reference counter.

##### 11.3.2.4 `Expr (int i) throw ()` [inline]

Construct from an int.

##### 11.3.2.5 `Expr (double f) throw ()` [inline]

Construct from a double. May yield Infinity or Undefined (NaN) values.

##### 11.3.2.6 `~Expr () throw ()` [inline]

Destructor. Frees the wrapped object.

### 11.3.3 Member Function Documentation

#### 11.3.3.1 `int compare (const Expr & other) const throw ()` [inline]

Compare with another [Expr](#). See [pmath\\_compare\(\)](#).

#### 11.3.3.2 `Expr deserialize (ReadableBinaryFile file, pmath_serialize_error_t * error) throw ()` [inline, static]

Deserialize an [Expr](#) from a binary file/stream.

**Parameters:**

*file* The binary file/stream. It must be writeable.

*error* An error number is stored here. May be NULL if not needed.

**Returns:**

The deserialized expression.

#### 11.3.3.3 `size_t expr_length () const throw ()` [inline]

Length of the expression or 0 on error.

#### 11.3.3.4 `const pmath_t get () const throw ()` [inline]

Get the [pmath\\_t](#). Reference is held by the [Expr](#) object.

#### 11.3.3.5 `unsigned int hash () const throw ()` [inline]

Get a hash value.

#### 11.3.3.6 `bool is_custom () const throw ()` [inline]

#### 11.3.3.7 `bool is_double () const throw ()` [inline]



**11.3.3.8** `bool is_evaluated () const throw ()` [inline]

**11.3.3.9** `bool is_expr () const throw ()` [inline]

**11.3.3.10** `bool is_float () const throw ()` [inline]

**11.3.3.11** `bool is_int32 () const throw ()` [inline]

**11.3.3.12** `bool is_integer () const throw ()` [inline]

**11.3.3.13** `bool is_magic () const throw ()` [inline]

**11.3.3.14** `bool is_mpfloor () const throw ()` [inline]

**11.3.3.15** `bool is_null () const throw ()` [inline]

**11.3.3.16** `bool is_number () const throw ()` [inline]

**11.3.3.17** `bool is_pointer () const throw ()` [inline]

**11.3.3.18** `bool is_pointer_of (pmath_type_t type) const throw ()` [inline]

**11.3.3.19** `bool is_quotient () const throw () [inline]`

**11.3.3.20** `bool is_rational () const throw () [inline]`

**11.3.3.21** `bool is_string () const throw () [inline]`

**11.3.3.22** `bool is_symbol () const throw () [inline]`

**11.3.3.23** `bool is_valid () const throw () [inline]`

Check for not holding the null pointer.

**11.3.3.24** `bool operator!= (const Expr & o1, pmath_t o2) throw() [inline]`

**11.3.3.25** `bool operator!= (pmath_t o1, const Expr & o2) throw() [inline]`

check for non-identity. The pMath != operator.

**11.3.3.26** `bool operator!= (const Expr & other) const throw () [inline]`

Check for non-identity. The pMath != operator.

**11.3.3.27** `bool operator< (const Expr & other) const throw () [inline]`

Compare with another [Expr](#). See [pmath\\_compare\(\)](#).

**11.3.3.28** `bool operator<= (const Expr & other) const throw ()` [inline]

Compare with another [Expr](#). See [pmath\\_compare\(\)](#).

**11.3.3.29** `Expr& operator= (const Expr & src) throw ()` [inline]

Copy an [Expr](#). Increments the new value's reference counter and frees the old one.

**11.3.3.30** `bool operator== (const Expr & o1, pmath_t o2) throw()` [inline]

**11.3.3.31** `bool operator== (pmath_t o1, const Expr & o2) throw()` [inline]

check for identity. The pMath === operator.

**11.3.3.32** `bool operator== (const Expr & other) const throw ()` [inline]

Check for identity. The pMath === operator.

**11.3.3.33** `bool operator> (const Expr & other) const throw ()` [inline]

Compare with another [Expr](#). See [pmath\\_compare\(\)](#).

**11.3.3.34** `bool operator>= (const Expr & other) const throw ()` [inline]

Compare with another [Expr](#). See [pmath\\_compare\(\)](#).

**11.3.3.35** `Expr operator[] (size_t i) const throw ()` [inline]

Get the *i*-th argument of the expression.

**Parameters:**

*i* Index. May be > [expr\\_length\(\)](#).

**Returns:**

The  $i$ -th argument if the object is a [pmath\\_expr\\_t](#). `expr[0]` is the head, `expr[1]` the first argument and `expr[length()]` the last argument.

**11.3.3.36 `pmath_t release () throw ()` [inline]**

Return the [pmath\\_t](#) and discard it. Caller must [pmath\\_unref\(\)](#) it.

**11.3.3.37 `pmath_serialize_error_t serialize (WriteableBinaryFile file) const throw ()` [inline]**

Serialize to a binary file/stream.

**Parameters:**

*file* The binary file/stream. It must be writeable.

**Returns:**

An error number.

**11.3.3.38 `void set (size_t i, size_t j, Expr e) throw ()` [inline]**

Change the  $(i,j)$ -th argument of a matrix.

**Parameters:**

*i* The matrix row.

*j* The matrix column.

*e* The new element.

**11.3.3.39 `void set (size_t i, Expr e) throw ()` [inline]**

Change the  $i$ -th argument of an expression.

**Parameters:**

*i* Index. May be  $> \text{expr\_length}()$ .

*e* The new element.

**11.3.3.40 double to\_double (double *def* = 0.0) const throw () [inline]**

Convert to a double.

**Parameters:**

*def* Optional default value.

**Returns:**

the double value if the object is a numeric object and *def* otherwise.

**11.3.3.41 String to\_string (pmath\_write\_options\_t *options* = 0) const throw () [inline]**

Convert to a string. The pMath ToString function.

**Parameters:**

*options* Optional formatting options.

**Returns:**

The [String](#) representation.

**11.3.3.42 void write\_to\_file (WritableTextFile *file*, pmath\_write\_options\_t *options* = 0) const throw () [inline]**

Write to a file/text stream.

**Parameters:**

*file* The text file object. It must be writeable.

*options* Optional formatting options.

**11.3.3.43 static void write\_to\_string (void \* *user*, const uint16\_t \* *data*, int *len*) throw () [inline, static, protected]**

### 11.3.4 Field Documentation

#### 11.3.4.1 `pmath_t_obj` [protected]

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

- [pmath-cpp.h](#)

## 11.4 File Class Reference

A wrapper for pMath file objects (data streams).

Inherits [pmath::Expr](#).

Inherited by [BinaryFile](#), and [TextFile](#).

### Public Member Functions

- [File](#) () throw ()
- [File](#) ([pmath\\_t](#) file\_object) throw ()
- [File](#) (const [Expr](#) &file\_object) throw ()
- [File](#) (const [File](#) &src) throw ()
- bool [has\\_capabilities](#) (int properties) const throw ()  
*Test for file properties/capabilities.*
- bool [is\\_file](#) () const throw ()
- bool [is\\_readable](#) () const throw ()
- bool [is\\_writeable](#) () const throw ()
- bool [is\\_binary](#) () const throw ()
- bool [is\\_text](#) () const throw ()
- [pmath\\_files\\_status\\_t](#) [status](#) () const throw ()
- void [flush](#) () throw ()  
*Flush the output buffer of a writeable file.*
- void [close](#) () throw ()  
*Closes a file immediatly instead of letting the garbage collector close it later.*

#### 11.4.1 Detailed Description

A wrapper for pMath file objects (data streams).

This class provides some stream utility functions in addition to [Expr](#). Note that a pMath file does not have to correspond to any operating system file object.

### 11.4.2 Constructor & Destructor Documentation

**11.4.2.1** `File () throw ()` `[inline]`

**11.4.2.2** `File (pmath_t file_object) throw ()` `[inline, explicit]`

**11.4.2.3** `File (const Expr & file_object) throw ()` `[inline, explicit]`

**11.4.2.4** `File (const File & src) throw ()` `[inline]`

### 11.4.3 Member Function Documentation

**11.4.3.1** `void close () throw ()` `[inline]`

Closes a file immediatly instead of letting the garbage collector close it later.

**11.4.3.2** `void flush () throw ()` `[inline]`

Flush the output buffer of a writeable file.

**11.4.3.3** `bool has_capabilities (int properties) const throw ()` `[inline]`

Test for file properties/capabilities.

**Parameters:**

*properties* 0 or one or more of the PMATH\_FILE\_PROP\_XXX constants.

**Returns:**

Whether the file has all the specified capabilities.

**11.4.3.4** `bool is_binary () const throw ()` `[inline]`

**11.4.3.5** `bool is_file () const throw ()` `[inline]`

**11.4.3.6** `bool is_readable () const throw ()` `[inline]`

**11.4.3.7** `bool is_text () const throw ()` `[inline]`

**11.4.3.8** `bool is_writable () const throw ()` `[inline]`

**11.4.3.9** `pmath_files_status_t status () const throw ()` `[inline]`

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

- [pmath-cpp.h](#)

## 11.5 Gather Class Reference

Utility class for emitting and gathering expressions/building lists.

### Public Member Functions

- [Gather](#) () throw ()  
*The constructor. Starts gathering.*
- [Gather](#) (Expr pattern) throw ()
- [~Gather](#) () throw ()
- [Expr end](#) () throw ()  
*end gathering. Calling [end\(\)](#) multiple times returns `PMATH_NULL`.*



### Static Public Member Functions

- static void [emit](#) ([Expr](#) e) throw ()  
*Emit a value to be gathered.*
- static void [emit](#) ([Expr](#) e, [Expr](#) tag) throw ()  
*Emit a value to be gathered.*

### Protected Attributes

- bool [ended](#)

#### 11.5.1 Detailed Description

Utility class for emitting and gathering expressions/building lists.

Gathering begins with the construction of the object and ends with a call to [end\(\)](#) or the object destruction. This removes the burden of calling [pmath\\_gather\\_end\(\)](#) for every [pmath\\_gather\\_begin\(\)](#).

#### 11.5.2 Constructor & Destructor Documentation

##### 11.5.2.1 [Gather \(\)](#) throw () [inline]

The constructor. Starts gathering.

##### 11.5.2.2 [Gather \(Expr pattern\)](#) throw () [inline, explicit]

##### 11.5.2.3 [~Gather \(\)](#) throw () [inline]

#### 11.5.3 Member Function Documentation

##### 11.5.3.1 static void [emit \(Expr e, Expr tag\)](#) throw () [inline, static]

Emit a value to be gathered.

### 11.5.3.2 `static void emit (Expr e) throw ()` `[inline, static]`

Emmit a value to be gathered.

### 11.5.3.3 `Expr end () throw ()` `[inline]`

end gathering. Calling `end()` multiple times returns `PMATH_NULL`.

## 11.5.4 Field Documentation

### 11.5.4.1 `bool ended` `[protected]`

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

- [pmath-cpp.h](#)

## 11.6 `pmath_atomic2_t` Struct Reference

A 2-pointer-sized atomic variable type.

### 11.6.1 Detailed Description

A 2-pointer-sized atomic variable type.

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

- `pmath-util/concurrency/atomic.h`

## 11.7 `pmath_atomic_t` Struct Reference

A pointer-sized atomic variable type.

### 11.7.1 Detailed Description

A pointer-sized atomic variable type.

Initialize it with `PMATH_ATOMIC_STATIC_INIT`.

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

- `pmath-util/concurrency/atomic.h`

## 11.8 pmath\_binary\_file\_api\_t Class Reference

Access functions for binary files.

### Data Fields

- [size\\_t struct\\_size](#)  
*The structure size. Always initialize this with `sizeof(pmath_binary_file_api_t)`.*
- [pmath\\_files\\_status\\_t\(\\* status\\_function\)](#)(void \*extra)  
*A function to get the current file status This can be `PMATH_NULL` if `read_function` is also `PMATH_NULL`.*
- [size\\_t\(\\* read\\_function\)](#)(void \*extra, void \*buffer, size\_t buffer\_size)  
*An optional callback function for reading bytes.*
- [size\\_t\(\\* write\\_function\)](#)(void \*extra, const void \*buffer, size\_t buffer\_size)  
*An optional callback function for writing bytes.*
- [void\(\\* flush\\_function\)](#)(void \*extra)  
*An optional callback function for flushing an output buffer.*

### 11.8.1 Detailed Description

Access functions for binary files.

See also:

[pmath\\_file\\_create\\_binary](#)

### 11.8.2 Field Documentation

#### 11.8.2.1 void(\* flush\_function)(void \*extra)

An optional callback function for flushing an output buffer.

#### 11.8.2.2 size\_t(\* read\_function)(void \*extra, void \*buffer, size\_t buffer\_size)

An optional callback function for reading bytes.

### 11.8.2.3 pmath\_files\_status\_t(\* status\_function)(void \*extra)

A function to get the current file status This can be PMATH\_NULL if read\_function is also PMATH\_NULL.

### 11.8.2.4 size\_t struct\_size

The structure size. Always initialize this with `sizeof(pmath_binary_file_api_t)`.

### 11.8.2.5 size\_t(\* write\_function)(void \*extra, const void \*buffer, size\_t buffer\_size)

An optional callback function for writing bytes.

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

- [pmath-util/files.h](#)

## 11.9 pmath\_cstr\_writer\_info\_t Struct Reference

Additional information for [pmath\\_utf8\\_writer\(\)](#) or [pmath\\_native\\_writer\(\)](#).

### Data Fields

- void(\* [write\\_cstr](#))(void \*[user](#), const char \*cstr)
- void \* [user](#)

### 11.9.1 Detailed Description

Additional information for [pmath\\_utf8\\_writer\(\)](#) or [pmath\\_native\\_writer\(\)](#).

### 11.9.2 Field Documentation

#### 11.9.2.1 void\* user

### 11.9.2.2 `void(* write_cstr)(void *user, const char *cstr)`

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

- [pmath-core/strings.h](#)

## 11.10 `pmath_custom_t` Class Reference

The Custom Object class.

Inherits [pmath\\_t](#).

Inherited by [pmath\\_messages\\_t](#).

### Public Member Functions

- [pmath\\_custom\\_t pmath\\_custom\\_new](#) (void \*data, [pmath\\_callback\\_t](#) destructor)  
*Create a custom object.*
- void \* [pmath\\_custom\\_get\\_data](#) ([pmath\\_custom\\_t](#) custom)  
*Get a custom object's data member.*
- [pmath\\_bool\\_t pmath\\_custom\\_has\\_destructor](#) ([pmath\\_custom\\_t](#) custom, [pmath\\_callback\\_t](#) dtor)  
*Check for a custom object's data type.*

### 11.10.1 Detailed Description

The Custom Object class.

Since it is derived from [pmath\\_t](#), you can provide it to any function that accepts a [pmath\\_t](#).

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

- [pmath-core/custom.h](#)

## 11.11 `pmath_expr_t` Class Reference

The Expression class.

Inherits [pmath\\_t](#).

### Public Member Functions

- [pmath\\_expr\\_t pmath\\_expr\\_new](#) ([pmath\\_t](#) head, [size\\_t](#) length)  
*Create a new expression.*
- [pmath\\_expr\\_t pmath\\_expr\\_new\\_extended](#) ([pmath\\_t](#) head, [size\\_t](#) length,...)  
*Create a new expression with all items given.*
- [pmath\\_expr\\_t pmath\\_expr\\_resize](#) ([pmath\\_expr\\_t](#) expr, [size\\_t](#) new\_length)  
*Resize an expression.*
- [pmath\\_expr\\_t pmath\\_expr\\_append](#) ([pmath\\_expr\\_t](#) expr, [size\\_t](#) count,...)  
*Append some items to an expression.*
- [size\\_t pmath\\_expr\\_length](#) ([pmath\\_expr\\_t](#) expr)  
*Get an expression's length.*
- [pmath\\_t pmath\\_expr\\_get\\_item](#) ([pmath\\_expr\\_t](#) expr, [size\\_t](#) index)  
*Get an item from an expression.*
- [pmath\\_t pmath\\_expr\\_extract\\_item](#) ([pmath\\_expr\\_t](#) expr, [size\\_t](#) index)  
*Extract an item from an expression.*
- [pmath\\_expr\\_t pmath\\_expr\\_get\\_item\\_range](#) ([pmath\\_expr\\_t](#) expr, [size\\_t](#) start, [size\\_t](#) length)  
*Get multiple items from an expression.*
- [pmath\\_expr\\_t pmath\\_expr\\_set\\_item](#) ([pmath\\_expr\\_t](#) expr, [size\\_t](#) index, [pmath\\_t](#) item)  
*Set an item in an expression.*
- [pmath\\_expr\\_t pmath\\_expr\\_remove\\_all](#) ([pmath\\_expr\\_t](#) expr, [pmath\\_t](#) rem)  
*Remove all occurrences of an object from an expression.*
- [pmath\\_expr\\_t pmath\\_expr\\_sort](#) ([pmath\\_expr\\_t](#) expr)  
*Sort an expression.*
- [pmath\\_expr\\_t pmath\\_expr\\_flatten](#) ([pmath\\_expr\\_t](#) expr, [pmath\\_t](#) head, [size\\_t](#) depth)  
*Flatten an expression.*

### Related Functions

(Note that these are not member functions.)

- void [pmath\\_gather\\_begin](#) ([pmath\\_t](#) pattern)

*Start gathering emitted objects.*

- `pmath_expr_t pmath_gather_end` (void)  
*Finish gathering emitted objects.*
- void `pmath_emit` (`pmath_t` object, `pmath_t` tag)  
*Emit an object to be gathered by the appropriate surrounding `pmath_gather_begin()` ... `pmath_gather_end()` function pair.*
- `pmath_t pmath_evaluate_expression` (`pmath_expr_t` expr, `pmath_bool_t` apply\_rules)  
*Partly evaluate an expression.*
- `pmath_expr_t pmath_options_extract` (`pmath_expr_t` expr, `size_t` last\_nonoption)  
*Extract option values from an expression.*
- `pmath_t pmath_option_value` (`pmath_t` fn, `pmath_t` name, `pmath_t` extra)  
*Retrieve a option value of a given function.*

### 11.11.1 Detailed Description

The Expression class.

Because `pmath_expr_t` is derived from `pmath_t`, you can use expressions wherever a `pmath_t` is accepted. E.g. you calculate an expression's hash value with `pmath_hash()`.

The `pmath_type_t` of strings is `PMATH_TYPE_EXPRESSION`.

Expressions are arranged as array of objects. At index 0 is allways the head (function name). All arguments are at indices 1 to the length of the expression (including).

At the pMath language level, any exprssion can be entered in the form `'f(a,b,c,...)'` or `'a.f(b,c,...)'`. For expressions with only one argument, `'a.f'` can be used instead of `'a.f()'`.

### 11.11.2 Friends And Related Function Documentation

#### 11.11.2.1 `pmath_t pmath_evaluate_expression` (`pmath_expr_t` expr, `pmath_bool_t` apply\_rules) [related]

Partly evaluate an expression.

#### Parameters:

**expr** A pMath expression. It will be freed. Do not use it afterwards.

**apply\_rules** Whether to apply custom rules the the whole expression or not.

**Returns:**

A new object produced by pMath's and user defined evaluation rules.

This function prepares an expression for evaluation (evaluates its items, ...).

Unlike `pmath_evaluate()`, it does not evaluate its result. In fact, `pmath_evaluate()` is basically a loop that calls `pmath_evaluate_expression()` until the result does not change any more.

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

- [pmath-core/expressions.h](#)
- [pmath-util/emit-and-gather.h](#)
- [pmath-util/evaluation.h](#)
- [pmath-util/helpers.h](#)

## 11.12 `pmath_float_t` Class Reference

The Floating Point Number class.

Inherits [pmath\\_number\\_t](#).

**Public Member Functions**

- [pmath\\_number\\_t](#) `pmath_float_new_str` (const char \*str, int base, [pmath\\_precision\\_control\\_t](#) precision\_control, double base\_precision\_accuracy)

*Create a floating point number from a string.*

### 11.12.1 Detailed Description

The Floating Point Number class.

Because `pmath_float_t` is derived from [pmath\\_number\\_t](#), you can use pMath integers wherever a [pmath\\_number\\_t](#) is accepted.

The [pmath\\_type\\_t](#) of floats is `PMATH_TYPE_FLOAT`.

There are two hidden implementations of floating point numbers in pMath. One operates on `double` values. The other uses MPFR for multiple precision numbers and provides automatic precision tracking.

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

- [pmath-core/numbers.h](#)

## 11.13 `pmath_hashtable_t` Class Reference

The Hashtable class.



### Public Member Functions

- `pmath_hashtable_t pmath_ht_create` (const `pmath_ht_class_t` \*class, unsigned int minsize)  
*Create a new hashtable.*
- `pmath_hashtable_t pmath_ht_copy` (`pmath_hashtable_t` ht, `pmath_ht_entry_copy_t` entry\_copy)  
*Copy a given hashtable.*
- void `pmath_ht_destroy` (`pmath_hashtable_t` ht)  
*Destroy a given hashtable.*
- void `pmath_ht_clear` (`pmath_hashtable_t` ht)  
*Clear a given hashtable.*
- unsigned int `pmath_ht_capacity` (`pmath_hashtable_t` ht)  
*Get the capacity of a given hashtable.*
- unsigned int `pmath_ht_count` (`pmath_hashtable_t` ht)  
*Get the number of valid entries in a given hashtable.*
- void \* `pmath_ht_entry` (`pmath_hashtable_t` ht, unsigned int i)  
*Get any entry of a given hashtable.*
- void \* `pmath_ht_search` (`pmath_hashtable_t` ht, void \*key)  
*Search for an entry in a given hashtable.*
- void \* `pmath_ht_insert` (`pmath_hashtable_t` ht, void \*entry)  
*Insert an entry into a given hashtable.*
- void \* `pmath_ht_remove` (`pmath_hashtable_t` ht, void \*key)  
*Remove an entry from a given hashtable.*

#### 11.13.1 Detailed Description

The Hashtable class.

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

- `pmath-util/hashtables.h`

## 11.14 `pmath_ht_class_t` Class Reference

A hashtable interface.

### Data Fields

- [`pmath\_callback\_t entry\_destructor`](#)
- [`pmath\_ht\_entry\_hash\_func\_t entry\_hash`](#)
- [`pmath\_ht\_entry\_equal\_func\_t entry\_keys\_equal`](#)
- [`pmath\_ht\_key\_hash\_func\_t key\_hash`](#)
- [`pmath\_ht\_entry\_equals\_key\_func\_t entry\_equals\_key`](#)

#### 11.14.1 Detailed Description

A hashtable interface.

#### 11.14.2 Field Documentation

##### 11.14.2.1 `pmath_callback_t entry_destructor`

##### 11.14.2.2 `pmath_ht_entry_equals_key_func_t entry_equals_key`

##### 11.14.2.3 `pmath_ht_entry_hash_func_t entry_hash`

##### 11.14.2.4 `pmath_ht_entry_equal_func_t entry_keys_equal`

##### 11.14.2.5 `pmath_ht_key_hash_func_t key_hash`

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

- `pmath-util/hashtables.h`

### 11.15 `pmath_integer_t` Class Reference

The Integer class.

Inherits [`pmath\_rational\_t`](#).

**Public Member Functions**

- [pmath\\_integer\\_t pmath\\_integer\\_new\\_slong](#) (signed long int si)  
*Create an integer object from a signed long.*
- [pmath\\_integer\\_t pmath\\_integer\\_new\\_ulong](#) (unsigned long int ui)  
*Create an integer object from an unsigned long.*
- [pmath\\_integer\\_new\\_si32](#)(si)  
*Create an integer object from an int32\_t.*
- [pmath\\_integer\\_t pmath\\_integer\\_new\\_ui32](#) (uint32\_t ui)  
*Create an integer object from an uint32\_t.*
- [pmath\\_integer\\_t pmath\\_integer\\_new\\_si64](#) (int64\_t si)  
*Create an integer object from an int64\_t.*
- [pmath\\_integer\\_t pmath\\_integer\\_new\\_ui64](#) (uint64\_t ui)  
*Create an integer object from an uint64\_t.*
- [pmath\\_integer\\_new\\_siptr](#)(si)  
*Create an integer object from an intptr\_t.*
- [pmath\\_integer\\_new\\_uiptr](#)(ui)  
*Create an integer object from an uintptr\_t.*
- [pmath\\_integer\\_t pmath\\_integer\\_new\\_data](#) (size\_t count, int order, int size, int endian, size\_t nails, const void \*data)  
*Create an integer object from a data buffer.*
- [pmath\\_integer\\_t pmath\\_integer\\_new\\_str](#) (const char \*str, int base)  
*Create an integer object from a C String.*
- [pmath\\_integer\\_fits\\_si32](#)(integer)  
*Check whether a pMath integer is in range  $-2^{31} .. 2^{31}-1$ .*
- [pmath\\_bool\\_t pmath\\_integer\\_fits\\_ui32](#) (pmath\_integer\_t integer)  
*Check whether a pMath integer is in range  $0 .. 2^{32}-1$ .*
- [pmath\\_bool\\_t pmath\\_integer\\_fits\\_si64](#) (pmath\_integer\_t integer)  
*Check whether a pMath integer is in range  $-2^{63} .. 2^{63}-1$ .*
- [pmath\\_bool\\_t pmath\\_integer\\_fits\\_ui64](#) (pmath\_integer\_t integer)  
*Check whether a pMath integer is in range  $0 .. 2^{64}-1$ .*
- [pmath\\_integer\\_fits\\_siptr](#)(integer)

*Check whether a pMath integer fits into an `intptr_t`.*

- `pmath_integer_fits_uiptr`(integer)  
*Check whether a pMath integer fits into an `uintptr_t`.*
- `int32_t pmath_integer_get_si32` (pmath\_integer\_t integer)  
*Convert a pMath integer to a signed long int.*
- `uint32_t pmath_integer_get_ui32` (pmath\_integer\_t integer)  
*Convert a pMath integer to a unsigned long int.*
- `int64_t pmath_integer_get_si64` (pmath\_integer\_t integer)  
*Convert a pMath integer to an `int64_t`.*
- `uint64_t pmath_integer_get_ui64` (pmath\_integer\_t integer)  
*Convert a pMath integer to a `uint64_t`.*
- `pmath_integer_get_siptr`  
*Convert a pMath integer to a `intptr_t`.*
- `pmath_integer_get_uiptr`  
*Convert a pMath integer to a `uintptr_t`.*

### 11.15.1 Detailed Description

The Integer class.

Because `pmath_integer_t` is derived from `pmath_rational_t`, you can use pMath integers wherever a `pmath_rational_t` is accepted.

The `pmath_type_t` of integers is `PMATH_TYPE_INTEGER`.

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

- `pmath-core/numbers.h`

## 11.16 `pmath_messages_t` Class Reference

A message queue for interthread communication.

Inherits `pmath_custom_t`.

### Related Functions

(Note that these are not member functions.)

- `pmath_bool_t pmath_is_message_queue` (pmath\_t obj)

*Test if an object is a message queue.*

- `pmath_messages_t pmath_thread_get_queue` (void)  
*Get the current thread's message queue.*
- void `pmath_thread_sleep` (void)  
*Send the current thread to sleep.*
- void `pmath_thread_sleep_timeout` (double abs\_timeout)  
*Send the current thread to sleep.*
- void `pmath_thread_wakeup` (`pmath_messages_t` mq)  
*Wake up another thread.*
- void `pmath_thread_send` (`pmath_messages_t` mq, `pmath_t` msg)  
*Asynchronously send a message to another thread.*
- `pmath_t pmath_thread_send_wait` (`pmath_messages_t` mq, `pmath_t` msg, double timeout\_seconds, void(\*idle\_function)(void \*), void \*idle\_data)  
*Send a message to another thread and wait for the answer.*
- void `pmath_thread_send_delayed` (`pmath_messages_t` mq, `pmath_t` msg, double seconds)  
*Asynchronously send a message to a thread sometime in the future.*

### 11.16.1 Detailed Description

A message queue for interthread communication.

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

- `pmath-util/concurrency/threadmsg.h`

## 11.17 `pmath_number_t` Class Reference

The abstract Number class.

Inherits `pmath_t`.

Inherited by `pmath_float_t`, and `pmath_rational_t`.

### Public Member Functions

- double `pmath_number_get_d` (`pmath_number_t` number)  
*Convert a pMath number to a double.*

- `int pmath_number_sign (pmath_number_t num)`  
*Get a number's sign.*
- `pmath_number_t pmath_number_neg (pmath_number_t num)`  
*Get a number's negative.*

### 11.17.1 Detailed Description

The abstract Number class.

Because `pmath_integer_t` is derived from `pmath_number_t`, you can use pMath integers wherever a `pmath_number_t` is accepted.

See also:

[Objects - the Base of pMath](#)

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

- `pmath-core/numbers.h`

## 11.18 `pmath_quotient_t` Class Reference

The Quotient class.

Inherits `pmath_rational_t`.

### 11.18.1 Detailed Description

The Quotient class.

Because `pmath_quotient_t` is derived from `pmath_rational_t`, you can use pMath integers wherever a `pmath_rational_t` is accepted.

The `pmath_type_t` of quotients is `PMATH_TYPE_QUOTIENT`.

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

- `pmath-core/numbers.h`

## 11.19 `pmath_rational_t` Class Reference

The abstract Rational Number class.

Inherits `pmath_number_t`.

Inherited by `pmath_integer_t`, and `pmath_quotient_t`.

### Public Member Functions

- [pmath\\_rational\\_t pmath\\_rational\\_new](#) ([pmath\\_integer\\_t](#) numerator, [pmath\\_integer\\_t](#) denominator)  
*Create a rational number.*
- [pmath\\_integer\\_t pmath\\_rational\\_numerator](#) ([pmath\\_rational\\_t](#) rational)  
*Get the numerator of a rational number.*
- [pmath\\_integer\\_t pmath\\_rational\\_denominator](#) ([pmath\\_rational\\_t](#) rational)  
*Get the denominator of a rational number.*

#### 11.19.1 Detailed Description

The abstract Rational Number class.

Because [pmath\\_rational\\_t](#) is derived from [pmath\\_number\\_t](#), you can use pMath integers wherever a [pmath\\_number\\_t](#) is accepted.

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

- [pmath-core/numbers.h](#)

## 11.20 pmath\_span\_array\_t Class Reference

Internal flat representaion of spans.

### Public Member Functions

- void [pmath\\_span\\_array\\_free](#) ([pmath\\_span\\_array\\_t](#) \*spans)  
*Destroy a span-array and all its spans.*
- int [pmath\\_span\\_array\\_length](#) ([pmath\\_span\\_array\\_t](#) \*spans)  
*Get a span-array's length.*
- [pmath\\_bool\\_t](#) [pmath\\_span\\_array\\_is\\_token\\_end](#) ([pmath\\_span\\_array\\_t](#) \*spans, int pos)  
*Test the token-end-flag at an index.*
- [pmath\\_bool\\_t](#) [pmath\\_span\\_array\\_is\\_operand\\_start](#) ([pmath\\_span\\_array\\_t](#) \*spans, int pos)  
*Test the operator-start-flag at an index.*
- [pmath\\_span\\_t](#) \* [pmath\\_span\\_at](#) ([pmath\\_span\\_array\\_t](#) \*spans, int pos)  
*Get a span starting at an index.*

### 11.20.1 Detailed Description

Internal flat representaion of spans.

The box-form of the expression `aa*bbb+cc^~dd*ee` is `{{"aa", "*", "bbb"}, "+", {{ "cc", "^", {"-", "dd"}}, "*", "ee"}}`

But those lists are not practicable for front-ends, so pMath provides a flat representation called *span-array*. It is an array of spans and flags (see [pmath\\_span\\_at\(\)](#), [pmath\\_span\\_array\\_is\\_token\\_end\(\)](#), [pmath\\_span\\_array\\_is\\_operand\\_start\(\)](#)).

The above code would be scanned to the following span array (the text itself is not stored):

```

                                aa*bbb+cc^~dd*ee
operand start:  x  x  x  xx  x
token end:      xx  xx  xxx  xx  x
spans <         /      \      /      \
                /        \      /      \
index:         0        5      10     15
```

So at index 0 is a span which ends with index 15. It has a next span (a shorter span that starts at the same position) which ends with index 5.

Another span is at index 7. It ends with index 15 and has a next span which ends with index 12.

The last span is at index 10 and also ends with index 12.

No two spans may cross-overlap so there is a definite hierachy. You create such a span-array with [pmath\\_spans\\_from\\_string\(\)](#) and destroy it with [pmath\\_span\\_array\\_free\(\)](#). Conversion to and from boxes expressions can be done via [pmath\\_boxes\\_from\\_spans\(\)](#) and [pmath\\_spans\\_from\\_boxes\(\)](#) respectively.

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

- `pmath-language/scanner.h`

## 11.21 `pmath_span_t` Class Reference

Represents a span in a [span-array](#).

### Public Member Functions

- [pmath\\_span\\_t \\* pmath\\_span\\_next](#) ([pmath\\_span\\_t \\*span](#))  
*Get the next-shorter span starting at the same position.*
- [int pmath\\_span\\_end](#) ([pmath\\_span\\_t \\*span](#))  
*Get end of a span.*



### 11.21.1 Detailed Description

Represents a span in a [span-array](#).

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

- [pmath-language/scanner.h](#)

## 11.22 `pmath_stack_t` Class Reference

The type of pMath's threadsafe stacks.

### 11.22.1 Detailed Description

The type of pMath's threadsafe stacks.

You can create it with [pmath\\_stack\\_new\(\)](#) and must destroy it with [pmath\\_stack\\_free\(\)](#).

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

- [pmath-util/stacks.h](#)

## 11.23 `pmath_string_t` Class Reference

The string class.

Inherits [pmath\\_t](#).

### Public Member Functions

- [pmath\\_string\\_t pmath\\_string\\_new](#) (int capacity)  
*Create an empty pMath String.*
- [pmath\\_string\\_t pmath\\_string\\_insert\\_latin1](#) ([pmath\\_string\\_t](#) str, int inspos, const char \*ins, int inslen)  
*Insert an Latin-1 encoded buffer into a pMath String.*
- [pmath\\_string\\_t pmath\\_string\\_from\\_utf8](#) (const char \*str, int len)  
*Convert an UTF-8 encoded buffer to a pMath String.*
- char \* [pmath\\_string\\_to\\_utf8](#) ([pmath\\_string\\_t](#) str, int \*result\_len)  
*Convert a pMath string to a zero-terminated UTF-8 string.*
- [pmath\\_string\\_t pmath\\_string\\_from\\_native](#) (const char \*str, int len)  
*Convert a string buffer in the current console character encoding to a pMath String.*
- char \* [pmath\\_string\\_to\\_native](#) ([pmath\\_string\\_t](#) str, int \*result\_len)

*Convert a pMath string to a string in the current console character encoding.*

- [PMATH\\_C\\_STRING](#)(cstr)  
*Short form to convert a C String to a pMath String.*
- [pmath\\_string\\_t pmath\\_string\\_insert\\_codepage](#) ([pmath\\_string\\_t](#) str, int inspos, const char \*ins, int inslen, const uint16\_t \*cp)  
*Insert a byte string into a pMath string using a translation array.*
- [pmath\\_string\\_t pmath\\_string\\_insert\\_ucs2](#) ([pmath\\_string\\_t](#) str, int inspos, const uint16\_t \*ins, int inslen)  
*Insert a UCS-2 buffer into a pMath String.*
- [pmath\\_string\\_t pmath\\_string\\_insert](#) ([pmath\\_string\\_t](#) str, int inspos, [pmath\\_string\\_t](#) ins)  
*Insert one pMath String into another pMath String.*
- [pmath\\_string\\_t pmath\\_string\\_concat](#) ([pmath\\_string\\_t](#) prefix, [pmath\\_string\\_t](#) postfix)  
*Concatenate two pMath Strings.*
- [pmath\\_string\\_t pmath\\_string\\_part](#) ([pmath\\_string\\_t](#) string, int start, int length)  
*Extract a substring of a pMath String.*
- const uint16\_t \* [pmath\\_string\\_buffer](#) ([pmath\\_string\\_t](#) \*string)  
*Get a string's buffer for reading.*
- int [pmath\\_string\\_length](#) ([pmath\\_string\\_t](#) string)  
*Get a string's length.*
- [pmath\\_bool\\_t pmath\\_string\\_equals\\_latin1](#) ([pmath\\_string\\_t](#) string, const char \*latin1)  
*Compare a pMath string with a C string.*

## Related Functions

(Note that these are not member functions.)

- [pmath\\_t pmath\\_string\\_expand\\_boxes](#) ([pmath\\_string\\_t](#) s)  
*Expand a string that contains boxes to a list of Strings and Boxes.*
- [pmath\\_t pmath\\_parse\\_string](#) ([pmath\\_string\\_t](#) code)  
*Parse a string to an expression.*
- [pmath\\_t pmath\\_parse\\_string\\_args](#) (const char \*code, const char \*format,...)  
*Parse a string with additional arguments to an expression.*

### 11.23.1 Detailed Description

The string class.

Because `pmath_string_t` is derived from `pmath_t`, you can use strings wherever a `pmath_t` is accepted. E.g. you compare two strings with `pmath_compare()` or `pmath_equals()`.

The `pmath_type_t` of strings is `PMATH_TYPE_STRING`.

See also:

[Objects - the Base of pMath](#)

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

- `pmath-core/strings.h`
- `pmath-language/scanner.h`

## 11.24 `pmath_symbol_t` Class Reference

The Symbol class.

Inherits `pmath_t`.

### Public Member Functions

- `pmath_symbol_t pmath_symbol_get` (`pmath_string_t` name, `pmath_bool_t` create)  
*Get a symbol by its fully qualified name.*
- `pmath_symbol_t pmath_symbol_create_temporary` (`pmath_string_t` name, `pmath_bool_t` unique)  
*Create a new temporary symbol.*
- `pmath_symbol_t pmath_symbol_find` (`pmath_string_t` name, `pmath_bool_t` create)  
*Find a symbol in the current namespace search path.*
- `pmath_string_t pmath_symbol_name` (`pmath_symbol_t` symbol)  
*Get a symbol's name.*
- `pmath_symbol_attributes_t pmath_symbol_get_attributes` (`pmath_symbol_t` symbol)  
*Get a symbol's attributes.*
- `void pmath_symbol_set_attributes` (`pmath_symbol_t` symbol, `pmath_symbol_attributes_t` attr)

*Set a symbol's attributes.*

- `pmath_t pmath_symbol_get_value (pmath_symbol_t symbol)`  
*Get a symbol's value.*
- `void pmath_symbol_set_value (pmath_symbol_t symbol, pmath_t value)`  
*Set a symbol's value.*
- `void pmath_symbol_synchronized (pmath_symbol_t symbol, pmath_callback_t callback, void *data)`  
*Execute a function synchronized to a symbol.*
- `void pmath_symbol_update (pmath_symbol_t symbol)`  
*Update a symbol manually.*

### Related Functions

(Note that these are not member functions.)

- `void pmath_collect_temporary_symbols (void)`  
*Collect unused symbols with the Temporary attribute.*

#### 11.24.1 Detailed Description

The Symbol class.

The pMath language knows symbols as 'named entities' that can hold any pMath object as a value and have some attributes. Those symbols are objects themselves. A symbol name consists of alphanumerical characters (a-z,A-Z,0-9) and apostrophes to separate namespaces. All standard symbols are in the "System" namespace (e.g. System'print). All user defined symbols go to "Global". Every other module has its own namespace.

Because `pmath_symbol_t` is derived from `pmath_t`, you can use strings wherever a `pmath_t` is accepted. E.g. you compare two symbols with `pmath_compare()` or `pmath_equals()`.

The `pmath_type_t` of symbols is `PMATH_TYPE_SYMBOL`.

See also:

[Objects - the Base of pMath](#)

#### 11.24.2 Friends And Related Function Documentation

##### 11.24.2.1 `void pmath_collect_temporary_symbols (void)` [\[related\]](#)

Collect unused symbols with the Temporary attribute.

This function is called periodically by the garbage collector.

**See also:**

[`pmath\_symbol\_attributes\_t`](#)

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

- [`pmath-core/symbols.h`](#)
- [`pmath-util/concurrency/threadpool.h`](#)

## 11.25 `pmath_t` Class Reference

The basic type of all pMath objects.

Inherited by [`pmath\_custom\_t`](#), [`pmath\_expr\_t`](#), [`pmath\_number\_t`](#), [`pmath\_string\_t`](#), and [`pmath\_symbol\_t`](#).

### Public Member Functions

- [`pmath\_bool\_t pmath\_equals`](#) ([`pmath\_t`](#) objA, [`pmath\_t`](#) objB)  
*Compares two objects for identity.*
- [`pmath\_t pmath\_ref`](#) ([`pmath\_t`](#) obj)  
*Increments the reference counter of an object and returns it.*
- void [`pmath\_unref`](#) ([`pmath\_t`](#) obj)  
*Decrements the reference counter of an object and frees its memory if the reference counter becomes 0.*
- unsigned int [`pmath\_hash`](#) ([`pmath\_t`](#) obj)  
*Calculates an object's hash value.*
- int [`pmath\_compare`](#) ([`pmath\_t`](#) objA, [`pmath\_t`](#) objB)  
*Compares two objects syntactically.*
- void [`pmath\_write`](#) ([`pmath\_t`](#) obj, [`pmath\_write\_options\_t`](#) options, void(\*write)(void \*user, const uint16\_t \*data, int len), void \*user)  
*Write an object to a stream.*
- void [`pmath\_write\_ex`](#) (struct [`pmath\_write\_ex\_t`](#) \*info, [`pmath\_t`](#) obj)  
*Advanced function to write an object to a stream.*
- [`pmath\_bool\_t pmath\_is\_evaluated`](#) ([`pmath\_t`](#) obj)  
*Test whether an object is already evaluated.*

- void `pmath_write_with_pagewidth` (`pmath_t` obj, `pmath_write_options_t` options, void(\*write)(void \*user, const uint16\_t \*data, int len), void \*user, int page\_width, int indentation\_width)

*Write an object to a stream with a maximum line width.*

### Data Fields

- uint64\_t `as_bits`
- double `as_double`
- struct {
  - uint32\_t `tag`
  - union {
    - int32\_t `as_int32`
    - uint16\_t `as_chars` [2]
  - } `u`
- } `s`

### Related Functions

(Note that these are not member functions.)

- `pmath_t` `pmath_evaluate` (`pmath_t` obj)
 

*Evaluate an object.*
- `pmath_t` `pmath_build_value_v` (const char \*format, va\_list args)
 

*Generate a List of objects with a format string.*
- `pmath_t` `pmath_build_value` (const char \*format,...)
 

*Generate a List of objects with a format string.*

#### 11.25.1 Detailed Description

The basic type of all pMath objects.

Use `pmath_is_XXX()` to determine whether an object is of a specific type. You must free unused objects with `pmath_unref()`, but if `pmath_is_pointer()` gives FALSE, then calling `pmath_ref()` and `pmath_unref()` is not necessary.

machine precision floating point values (aka double) and certain special values are stored directly in the `pmath_t` object. Long strings, expressions, other values are stored as a pointer. The technique to pack all this in only 8 bytes is called NaN-boxing. See

<http://blog.mozilla.com/rob-sayre/2010/08/02/mozillas-new-javascript-value-repr>

### 11.25.2 Member Function Documentation

#### 11.25.2.1 pmath\_bool\_t pmath\_equals (pmath\_t *objA*, pmath\_t *objB*)

Compares two objects for identity.

**Parameters:**

*objA* The first object.

*objB* The second one.

**Returns:**

TRUE iff both objects are identical.

‘identity’ means, that  $X \neq Y$  is possible, even if  $X$  and  $Y$  evaluate to the same value.

If *objA* and *objB* are symbols, the result is identical to testing  $\text{objA} == \text{objB}$ .

**Note:**

$\text{pmath\_equals}(A, B)$  might return FALSE although  $\text{pmath\_compare}(A, B) == 0$   
e.g. for an integer  $A$  and a floating point value  $B$ .

#### 11.25.2.2 pmath\_t pmath\_ref (pmath\_t *obj*)

Increments the reference counter of an object and returns it.

**Parameters:**

*obj* The object to be referenced.

**Returns:**

The referenced object. You must free the result with [pmath\\_unref\(\)](#).

#### 11.25.2.3 void pmath\_unref (pmath\_t *obj*)

Decrements the reference counter of an object and frees its memory if the reference counter becomes 0.

**Parameters:**

*obj* The object to be destroyed.

### 11.25.3 Friends And Related Function Documentation

#### 11.25.3.1 `pmath_t pmath_evaluate (pmath_t obj)` [related]

Evaluate an object.

**Parameters:**

*obj* Any pMath object. It will be freed. Do not use it afterwards.

**Returns:**

A new object produced by pMath's and user defined evaluation rules.

### 11.25.4 Field Documentation

#### 11.25.4.1 `uint64_t as_bits`

#### 11.25.4.2 `uint16_t as_chars[2]`

#### 11.25.4.3 `double as_double`

#### 11.25.4.4 `int32_t as_int32`

#### 11.25.4.5 `struct { ... } s`

#### 11.25.4.6 `uint32_t tag`

#### 11.25.4.7 `union { ... } u`

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



- [pmath-core/objects.h](#)
- [pmath-core/objects-inline.h](#)
- [pmath-util/evaluation.h](#)
- [pmath-util/helpers.h](#)
- [pmath-util/line-writer.h](#)

## 11.26 `pmath_text_file_api_t` Class Reference

Access functions for text files.

### Data Fields

- `size_t struct_size`  
*The structure size. Always initialize this with `sizeof(pmath_binary_file_api_t)`.*
- `pmath_files_status_t(* status_function)(void *extra)`  
*A function to get the current file status This can be `PMATH_NULL` if `read_function` is also `PMATH_NULL`.*
- `pmath_string_t(* readln_function)(void *extra)`  
*An optional callback function for reading one line.*
- `pmath_bool_t(* write_function)(void *extra, const uint16_t *str, int len)`  
*An optional callback function for writing one line.*
- `void(* flush_function)(void *extra)`  
*An optional callback function for flushing an output buffer.*

### 11.26.1 Detailed Description

Access functions for text files.

See also:

[pmath\\_file\\_create\\_text](#)

### 11.26.2 Field Documentation

#### 11.26.2.1 `void(* flush_function)(void *extra)`

An optional callback function for flushing an output buffer.

**11.26.2.2 `pmath_string_t(* readln_function)(void *extra)`**

An optional callback function for reading one line.

**11.26.2.3 `pmath_files_status_t(* status_function)(void *extra)`**

A function to get the current file status This can be `PMATH_NULL` if `read_function` is also `PMATH_NULL`.

**11.26.2.4 `size_t struct_size`**

The structure size. Allways initialize this with `sizeof(pmath_binary_file_api_t)`.

**11.26.2.5 `pmath_bool_t(* write_function)(void *extra, const uint16_t *str, int len)`**

An optional callback function for writing one line.

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

- `pmath-util/files.h`

**11.27 `pmath_thread_t` Class Reference**

The Representation of a thread.

**Related Functions**

(Note that these are not member functions.)

- `pmath_bool_t pmath_thread_queue_is_blocked_by` (`pmath_messages_t` waiter\_mq, `pmath_messages_t` waitee\_mq)  
*Queries whether a thread is blocked by another thread.*
- `void pmath_thread_run_with_interrupt_notifier` (`pmath_callback_t` callback, `pmath_callback_t` notify, `void *callback_closure`, `void *notify_closure`)  
*Execute a function with an interrupt notifier installed.*

- `pmath_messages_t pmath_thread_fork_daemon (pmath_callback_t callback, pmath_callback_t kill, void *data)`  
*Create a new deamon thread.*
- `pmath_thread_t pmath_thread_get_current (void)`  
*Get the current pMath thread.*
- `pmath_thread_t pmath_thread_get_parent (pmath_thread_t thread)`  
*Get a thread's direct parent.*
- `pmath_bool_t pmath_thread_is_parent (pmath_thread_t parent, pmath_thread_t child)`  
*Queries whether a thread is one of the parents of another.*
- `pmath_bool_t pmath_thread_aborting (pmath_thread_t thread)`  
*Queries whether pMath was requested to abort the evaluation of a specific thread or its parents.*

### 11.27.1 Detailed Description

The Representation of a thread.

Every operating system thread that runs pMath functions has its own `pmath_thread_t` after it successfully initialized with `pmath_init()`.

#### Todo

Implement `pmath_run_parallel(number_of_parallel_threads, callback)`.

### 11.27.2 Friends And Related Function Documentation

#### 11.27.2.1 `pmath_messages_t pmath_thread_fork_daemon (pmath_callback_t callback, pmath_callback_t kill, void *data)` [related]

Create a new deamon thread.

#### Parameters:

***callback*** The thread function.

***kill*** An optional function to inform the thread that it will be killed.

***data*** A pointer to be passed to `callback()` and `kill()`

#### Returns:

A reference to the new thread's message queue or `PMATH_NULL` on error. You have to destroy the result.

pMath will automatically kill any daemon thread when there are no other threads remaining (normally, when `pmath_done()` is called from `main()`). Killing daemons works as follows:

```
for each daemon thread t:
    call t->kill() if the method exists
    throw PMATH_ABORT_EXCEPTION in t

for each daemon thread t:
    wait for t to finish (to return from t->callback)
```

`pmath_init()` will be called automatically before `callback()` and `pmath_done()` after `callback` returns. So the pMath thread (`pmath_get_current()`) will be already initialized and you must not call these two functions in the *callback* routine.

You can use the *kill* function to set your own abort-please-flags if necessary.

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

- `pmath-util/concurrency/threads.h`
- `pmath-util/concurrency/threadmsg.h`
- `pmath-util/concurrency/threadpool.h`

## 11.28 `pmath_threadlock_t` Class Reference

A reentrant lock for threads.

### Related Functions

(Note that these are not member functions.)

- void `pmath_thread_call_locked` (`pmath_threadlock_t` \*threadlock\_ptr, `pmath_callback_t` callback, void \*data)  
*Execute a function synchronized with a threadlock.*

### 11.28.1 Detailed Description

A reentrant lock for threads.

A thread lock is like a thread lock, but it does not block child threads of the currently holding thread.

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

- `pmath-util/concurrency/threadlocks.h`

## 11.29 `pmath_write_ex_t` Struct Reference

Command structure for `pmath_write_ex()`. This should be initialized with `memset(&ex, 0, sizeof(ex)); ex.size = sizeof(ex); ...`

### Data Fields

- `size_t size`  
*must be initialized with `sizeof(struct pmath_write_ex_t)`, for version control*
- `pmath_write_options_t options`
- `void(* write)(void *user, const uint16_t *data, int len)`  
*mandatory, write callback*
- `void * user`  
*first parameter of the callbacks*
- `void(* pre_write)(void *user, pmath_t obj)`  
*optional, called before the `pmath_t` is written*
- `void(* post_write)(void *user, pmath_t obj)`  
*optional, called after the `pmath_t` is written*

#### 11.29.1 Detailed Description

Command structure for `pmath_write_ex()`. This should be initalized with `memset(&ex, 0, sizeof(ex)); ex.size = sizeof(ex); ...`.

#### 11.29.2 Field Documentation

##### 11.29.2.1 `pmath_write_options_t options`

##### 11.29.2.2 `void(* post_write)(void *user, pmath_t obj)`

optional, called after the `pmath_t` is written

##### 11.29.2.3 `void(* pre_write)(void *user, pmath_t obj)`

optional, called before the `pmath_t` is written

##### 11.29.2.4 `size_t size`

must be initialized with `sizeof(struct pmath_write_ex_t)`, for version control

**11.29.2.5 void\* user**

first parameter of the callbacks

**11.29.2.6 void(\* write)(void \*user, const uint16\_t \*data, int len)**

mandatory, write callback

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

- [pmath-core/objects.h](#)

**11.30 ReadableBinaryFile Class Reference**

A wrapper for readable pMath binary file objects (byte data streams).

Inherits [pmath::BinaryFile](#).

**Public Member Functions**

- [ReadableBinaryFile](#) ()
- [ReadableBinaryFile](#) ([pmath\\_t](#) file\_object) throw ()
- [ReadableBinaryFile](#) (const [Expr](#) &file\_object) throw ()
- [ReadableBinaryFile](#) (const [ReadableBinaryFile](#) &src) throw ()
- [size\\_t read](#) (void \*buffer, [size\\_t](#) buffer\_size, [bool](#) preserve\_internal\_buffer=false) throw ()

*Read some bytes from the file. See [pmath\\_file\\_read\(\)](#).*

**Static Public Member Functions**

- static [ReadableBinaryFile create\\_uncompressor](#) ([ReadableBinaryFile](#) srcfile) throw ()

*Create binary file object whose content is uncompressed from another binary file.*

**11.30.1 Detailed Description**

A wrapper for readable pMath binary file objects (byte data streams).

### 11.30.2 Constructor & Destructor Documentation

**11.30.2.1 ReadableBinaryFile ()** [inline]

**11.30.2.2 ReadableBinaryFile (pmath\_t *file\_object*) throw ()** [inline, explicit]

**11.30.2.3 ReadableBinaryFile (const Expr & *file\_object*) throw ()** [inline, explicit]

**11.30.2.4 ReadableBinaryFile (const ReadableBinaryFile & *src*) throw ()** [inline]

### 11.30.3 Member Function Documentation

**11.30.3.1 static ReadableBinaryFile create\_uncompressor (ReadableBinaryFile *srcfile*) throw ()** [inline, static]

Create binary file object whose content is uncompressed from another binary file.

**11.30.3.2 size\_t read (void \* *buffer*, size\_t *buffer\_size*, bool *preserve\_internal\_buffer* = false) throw ()** [inline]

Read some bytes from the file. See [pmath\\_file\\_read\(\)](#).

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

- [pmath-cpp.h](#)

## 11.31 ReadableTextFile Class Reference

A wrapper for pMath readable text file objects (byte data streams).

Inherits [pmath::TextFile](#).

### Public Member Functions

- [ReadableTextFile](#) ()
- [ReadableTextFile](#) ([pmath\\_t](#) file\_object) throw ()
- [ReadableTextFile](#) (const [Expr](#) &file\_object) throw ()
- [ReadableTextFile](#) (const [ReadableTextFile](#) &src) throw ()
- [String](#) [readline](#) () throw ()

*Read the next line from the file.*

### Static Public Member Functions

- static [ReadableTextFile](#) [create\\_from\\_binary](#) ([ReadableBinaryFile](#) binfile, const char \*encoding) throw ()  
*Create a text file from a binary file using a known character encoding.*
- static [ReadableTextFile](#) [create\\_from\\_binary](#) ([ReadableBinaryFile](#) binfile)  
*Create a text file from a binary file using UTF-16BE or UTF-16LE, depending on the machine architecture.*

#### 11.31.1 Detailed Description

A wrapper for pMath readable text file objects (byte data streams).

#### 11.31.2 Constructor & Destructor Documentation

##### 11.31.2.1 [ReadableTextFile](#) () [inline]

##### 11.31.2.2 [ReadableTextFile](#) ([pmath\\_t](#) file\_object) throw () [inline, explicit]

##### 11.31.2.3 [ReadableTextFile](#) (const [Expr](#) &file\_object) throw () [inline, explicit]

##### 11.31.2.4 [ReadableTextFile](#) (const [ReadableTextFile](#) &src) throw () [inline]



### 11.31.3 Member Function Documentation

#### 11.31.3.1 static ReadableTextFile create\_from\_binary (ReadableBinaryFile binfile) [inline, static]

Create a text file from a binary file using UTF-16BE or UTF-16LE, depending on the machine architecture.

#### 11.31.3.2 static ReadableTextFile create\_from\_binary (ReadableBinaryFile binfile, const char \* encoding) throw () [inline, static]

Create a text file from a binary file using a known character encoding.

#### 11.31.3.3 String readline () throw () [inline]

Read the next line from the file.

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

- [pmath-cpp.h](#)

## 11.32 String Class Reference

A wrapper for [pmath\\_string\\_t](#).

Inherits [pmath::Expr](#).

### Public Member Functions

- [String](#) () throw ()
- [String](#) ([pmath\\_string\\_t](#) \_str) throw ()  
*Construct from a [pmath\\_string\\_t](#), stealing the reference.*
- [String](#) (const [Expr](#) &src) throw ()
- [String](#) (const [String](#) &src) throw ()
- [String](#) (const char \*latin1, int len=-1) throw ()  
*Construct from Latin-1 encoded C string.*
- [String](#) & operator= (const [String](#) &src) throw ()
- [String](#) & operator+= (const [String](#) &src) throw ()  
*Append a string.*

- [String](#) & [operator+=](#) (const char \*latin1) throw ()  
*Append a C string.*
- [String](#) & [operator+=](#) (const uint16\_t \*ucs2) throw ()  
*Append a UTF-16-string.*
- [String](#) & [operator+=](#) (uint16\_t ch) throw ()  
*Append a single unicode character.*
- [String](#) [operator+](#) (const [String](#) &other) const throw ()  
*Concatenate two strings.*
- [String](#) [operator+](#) (const char \*latin1) const throw ()
- [String](#) [operator+](#) (const uint16\_t \*ucs2) const throw ()
- [String](#) [operator+](#) (uint16\_t ch) const throw ()  
*Concatenate a [String](#) and a single unicode character.*
- [String](#) [part](#) (int start, int length=-1) const throw ()  
*Get string part.*
- bool [equals](#) (const char \*latin1) const throw ()  
*Check for equality with a C string (Latin-1 encoded).*
- bool [starts\\_with](#) (const [String](#) &s) const throw ()  
*Check for prefix equality.*
- bool [starts\\_with](#) (const char \*latin1, int len=-1) const throw ()
- bool [starts\\_with](#) (const uint16\_t \*ucs2, int len=-1) const throw ()
- void [insert](#) (int pos, const [String](#) &other) throw ()  
*Insert a substring. Changes the object itself.*
- void [insert](#) (int pos, const char \*latin1, int len=-1) throw ()
- void [insert](#) (int pos, const uint16\_t \*ucs2, int len=-1) throw ()
- void [remove](#) (int start, int length) throw ()  
*Remove a substring.*
- const [String](#) [trim](#) () const throw ()  
*Trim leading and trailing whitespace.*
- int [length](#) () const throw ()  
*Get the string length.*
- const uint16\_t \* [buffer](#) () const throw ()  
*Get the UCS-2/UTF-16 const string buffer. This is not zero-terminated.*
- uint16\_t [operator\[\]](#) (int i) const throw ()

*Get a single character or U+0000 on error.*

- const [pmath\\_string\\_t](#) [get\\_as\\_string](#) () const throw ()  
*Get the underlying [pmath\\_string\\_t](#). It remains owned by this object.*

### Static Public Member Functions

- static [String FromUcs2](#) (const uint16\_t \*ucs2, int len=-1) throw ()  
*Construct from UCS-2/UTF-16 encoded string.*
- static [String FromChar](#) (unsigned int unicode) throw ()  
*Construct from a single unicode character.*
- static [String FromUtf8](#) (const char \*utf8, int len=-1) throw ()  
*Construct from UTF-8 encoded C string.*

#### 11.32.1 Detailed Description

A wrapper for [pmath\\_string\\_t](#).

This class provides some string utility functions in addition to [Expr](#).

#### 11.32.2 Constructor & Destructor Documentation

##### 11.32.2.1 [String](#) () throw () [inline]

##### 11.32.2.2 [String](#) ([pmath\\_string\\_t](#) \_str) throw () [inline, explicit]

Construct from a [pmath\\_string\\_t](#), stealing the reference.

##### 11.32.2.3 [String](#) (const Expr & src) throw () [inline]

##### 11.32.2.4 [String](#) (const String & src) throw () [inline]

**11.32.2.5 String (const char \* *latin1*, int *len* = -1) throw ()** [inline]

Construct from Latin-1 encoded C string.

**11.32.3 Member Function Documentation****11.32.3.1 const uint16\_t\* buffer () const throw ()** [inline]

Get the UCS-2/UTF-16 const string buffer. This is not zero-terminated.

**11.32.3.2 bool equals (const char \* *latin1*) const throw ()** [inline]

Check for equality with a C string (Latin-1 encoded).

**11.32.3.3 static String FromChar (unsigned int *unicode*) throw ()** [inline, static]

Construct from a single unicode character.

**11.32.3.4 static String FromUcs2 (const uint16\_t \* *ucs2*, int *len* = -1) throw ()** [inline, static]

Construct from UCS-2/UTF-16 encoded string.

**11.32.3.5 static String FromUtf8 (const char \* *utf8*, int *len* = -1) throw ()** [inline, static]

Construct from UTF-8 encoded C string.

**11.32.3.6 const pmath\_string\_t get\_as\_string () const throw ()** [inline]

Get the underlying [pmath\\_string\\_t](#). It remains owned by this object.

**11.32.3.7** `void insert (int pos, const uint16_t * ucs2, int len = -1) throw ()`  
[inline]

**11.32.3.8** `void insert (int pos, const char * latin1, int len = -1) throw ()`  
[inline]

**11.32.3.9** `void insert (int pos, const String & other) throw ()` [inline]

Insert a substring. Changes the object itself.

**11.32.3.10** `int length () const throw ()` [inline]

Get the string length.

**11.32.3.11** `String operator+ (uint16_t ch) const throw ()` [inline]

Concatenate a [String](#) and a single unicode character.

**11.32.3.12** `String operator+ (const uint16_t * ucs2) const throw ()` [inline]

**11.32.3.13** `String operator+ (const char * latin1) const throw ()` [inline]

**11.32.3.14** `String operator+ (const String & other) const throw ()` [inline]

Concatenate two strings.

**11.32.3.15** `String& operator+= (uint16_t ch) throw ()` [inline]

Append a single unicode character.

**11.32.3.16 String& operator+= (const uint16\_t \* *ucs2*) throw ()** [inline]

Append a UTF-16-string.

**11.32.3.17 String& operator+= (const char \* *latin1*) throw ()** [inline]

Append a C string.

**11.32.3.18 String& operator+= (const String & *src*) throw ()** [inline]

Append a string.

**11.32.3.19 String& operator= (const String & *src*) throw ()** [inline]**11.32.3.20 uint16\_t operator[] (int *i*) const throw ()** [inline]

Get a single character or U+0000 on error.

**11.32.3.21 String part (int *start*, int *length* = -1) const throw ()** [inline]

Get string part.

**11.32.3.22 void remove (int *start*, int *length*) throw ()** [inline]

Remove a substring.

**11.32.3.23 bool starts\_with (const uint16\_t \* *ucs2*, int *len* = -1) const throw ()**  
[inline]

**11.32.3.24** `bool starts_with (const char * latin1, int len = -1) const throw ()`  
[inline]

**11.32.3.25** `bool starts_with (const String & s) const throw ()` [inline]

Check for prefix equality.

**11.32.3.26** `const String trim () const throw ()` [inline]

Trim leading and trailing whitespace.

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

- [pmath-cpp.h](#)

## 11.33 TextFile Class Reference

A wrapper for pMath text file objects (byte data streams).

Inherits [pmath::File](#).

Inherited by [ReadableTextFile](#), and [WriteableTextFile](#).

### Public Member Functions

- [TextFile](#) ()
- [TextFile](#) ([pmath\\_t](#) file\_object) throw ()
- [TextFile](#) (const [Expr](#) &file\_object) throw ()
- [TextFile](#) (const [TextFile](#) &src) throw ()
- void [set\\_buffer](#) ([String](#) buffer)

*Set a file's internal text buffer.*

### Static Public Member Functions

- static [TextFile](#) [create\\_from\\_binary](#) ([BinaryFile](#) binfile, const char \*encoding) throw ()  
*Create a text file from a binary file using a known character encoding.*
- static [TextFile](#) [create\\_from\\_binary](#) ([BinaryFile](#) binfile)  
*Create a text file from a binary file using UTF-16BE or UTF-16LE, depending on the machine architecture.*

### 11.33.1 Detailed Description

A wrapper for pMath text file objects (byte data streams).

### 11.33.2 Constructor & Destructor Documentation

**11.33.2.1** `TextFile ()` `[inline]`

**11.33.2.2** `TextFile (pmath_t file_object) throw ()` `[inline, explicit]`

**11.33.2.3** `TextFile (const Expr & file_object) throw ()` `[inline, explicit]`

**11.33.2.4** `TextFile (const TextFile & src) throw ()` `[inline]`

### 11.33.3 Member Function Documentation

**11.33.3.1** `static TextFile create_from_binary (BinaryFile binfile)` `[inline, static]`

Create a text file from a binary file using UTF-16BE or UTF-16LE, depending on the machine architecture.

**11.33.3.2** `static TextFile create_from_binary (BinaryFile binfile, const char * encoding) throw ()` `[inline, static]`

Create a text file from a binary file using a known character encoding.

**11.33.3.3** `void set_buffer (String buffer)` `[inline]`

Set a file's internal text buffer.



**Parameters:**

*buffer* The new line buffer. It should not contain any newline character!

**See also:**

[pmath\\_file\\_set\\_textbuffer\(\)](#).

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

- [pmath-cpp.h](#)

## 11.34 TextUserStream Class Reference

Abstract base class for C++ callbacks used as pMath text files.

Inherits [pmath::UserStream](#).

**Public Member Functions**

- virtual [pmath\\_files\\_status\\_t](#) [status](#) ()=0  
*Called by pMath to check for end-of-file and other errors.*
- virtual void [flush](#) ()  
*Called by pMath to flush data to disk.*
- virtual [String](#) [readline](#) ()=0  
*Called by pMath to read a line of text, excluding any newline characters.*
- virtual bool [write](#) (const uint16\_t \*str, int len)=0  
*Called by pMath to write text.*

**Protected Member Functions**

- [TextFile](#) [convert\\_to\\_file](#) (bool readable, bool writeable)
- [ReadableTextFile](#) [convert\\_to\\_file\\_readonly](#) ()
- [WriteableTextFile](#) [convert\\_to\\_file\\_writeonly](#) ()

### 11.34.1 Detailed Description

Abstract base class for C++ callbacks used as pMath text files.

### 11.34.2 Member Function Documentation

**11.34.2.1** `TextFile convert_to_file (bool readable, bool writeable)` `[inline, protected]`

**11.34.2.2** `ReadableTextFile convert_to_file_readonly ()` `[inline, protected]`

**11.34.2.3** `WritableTextFile convert_to_file_writeonly ()` `[inline, protected]`

**11.34.2.4** `virtual void flush ()` `[inline, virtual]`

Called by pMath to flush data to disk.

**11.34.2.5** `virtual String readline ()` `[pure virtual]`

Called by pMath to read a line of text, excluding any newline characters.

**11.34.2.6** `virtual pmath_files_status_t status ()` `[pure virtual]`

Called by pMath to check for end-of-file and other errors.

**11.34.2.7** `virtual bool write (const uint16_t * str, int len)` `[pure virtual]`

Called by pMath to write text.

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

- [pmath-cpp.h](#)

## 11.35 UserStream Class Reference

Abstract base class for C++ callbacks used as pMath files.

Inherited by [BinaryUserStream](#), and [TextUserStream](#).

### Public Member Functions

- virtual [~UserStream](#) ()

### Static Public Member Functions

- template<class U >  
static bool [file\\_wraps](#) ([File](#) file)  
*Test whether a pMath file wraps a user stream.*

### Protected Member Functions

- virtual void [dereference](#) ()  
*Called by pMath when the object is no longer needed.*

### Static Protected Member Functions

- template<class U , typename A >  
static bool [manipulate](#) ([File](#) file, void(U::\*callback)(const A &), const A &arg)  
*Call a method on the user stream behind a pMath file.*
- template<class U >  
static bool [manipulate](#) ([BinaryFile](#) file, void(U::\*callback)())  
*Call a method on the user stream behind a pMath file.*
- static void [destructor\\_function](#) (void \*extra)  
*Called by pMath.*

#### 11.35.1 Detailed Description

Abstract base class for C++ callbacks used as pMath files.

The object destructor must be thread-safe (e.g. by not using any global data), because it is typically called from another thread than where the object was created. If synchronization is needed, it can be done in the [dereference\(\)](#) callback method.

All other callback methods are synchronized: pMath ensures that no callback is entered twice at the same time.

### 11.35.2 Constructor & Destructor Documentation

**11.35.2.1** `virtual ~UserStream ()` [inline, virtual]

### 11.35.3 Member Function Documentation

**11.35.3.1** `virtual void dereference ()` [inline, protected, virtual]

Called by pMath when the object is no longer needed.

This method destroys the object by default.

**11.35.3.2** `static void destructor_function (void * extra)` [inline, static, protected]

Called by pMath.

**11.35.3.3** `static bool file_wraps (File file)` [inline, static]

Test whether a pMath file wraps a user stream.

#### Parameters:

*file* The pMath file.

#### Returns:

true iff *file* wraps [UserStream](#) subclass U object

**11.35.3.4** `static bool manipulate (BinaryFile file, void(U::*)() callback)`  
[inline, static, protected]

Call a method on the user stream behind a pMath file.

#### Parameters:

*file* A pMath file that was created from a user stream class U.

*callback* A member method of the user stream function U to be called by pMath.

**Returns:**

Whether the callback was called. That is, whether the file is actually a user stream of class U.

**11.35.3.5 static bool manipulate (File *file*, void(U::\*)(const A &) *callback*, const A & *arg*)** [inline, static, protected]

Call a method on the user stream behind a pMath file.

**Parameters:**

*file* A pMath file that was created from a user stream class U.

*callback* A member method of the user stream function U to be called by pMath.

*arg* An argument to the callback.

**Returns:**

Whether the callback was called. That is, whether the file is actually a user stream of class U.

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

- [pmath-cpp.h](#)

## 11.36 WriteableBinaryFile Class Reference

A wrapper for writeable pMath binary file objects (byte data streams).

Inherits [pmath::BinaryFile](#).

**Public Member Functions**

- [WriteableBinaryFile](#) ()
- [WriteableBinaryFile](#) ([pmath\\_t](#) file\_object) throw ()
- [WriteableBinaryFile](#) (const [Expr](#) &file\_object) throw ()
- [WriteableBinaryFile](#) (const [WriteableBinaryFile](#) &src) throw ()
- [size\\_t write](#) (const void \*buffer, [size\\_t](#) buffer\_size) throw ()

*Write some bytes to the file. See [pmath\\_file\\_write\(\)](#).*

**Static Public Member Functions**

- static [WriteableBinaryFile](#) [create\\_compressor](#) ([WriteableBinaryFile](#) dstfile) throw ()

*Create binary file object whose content is compressed into another binary file.*

### 11.36.1 Detailed Description

A wrapper for writeable pMath binary file objects (byte data streams).

### 11.36.2 Constructor & Destructor Documentation

#### 11.36.2.1 WriteableBinaryFile () [inline]

#### 11.36.2.2 WriteableBinaryFile (pmath\_t *file\_object*) throw () [inline, explicit]

#### 11.36.2.3 WriteableBinaryFile (const Expr & *file\_object*) throw () [inline, explicit]

#### 11.36.2.4 WriteableBinaryFile (const WriteableBinaryFile & *src*) throw () [inline]

### 11.36.3 Member Function Documentation

#### 11.36.3.1 static WriteableBinaryFile create\_compressor (WriteableBinaryFile *dstfile*) throw () [inline, static]

Create binary file object whose content is compressed into another binary file.

#### 11.36.3.2 size\_t write (const void \* *buffer*, size\_t *buffer\_size*) throw () [inline]

Write some bytes to the file. See [pmath\\_file\\_write\(\)](#).

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

- [pmath-cpp.h](#)

## 11.37 WriteableTextFile Class Reference

A wrapper for pMath writeable text file objects (byte data streams).

Inherits [pmath::TextFile](#).

### Public Member Functions

- [WriteableTextFile](#) () throw ()
- [WriteableTextFile](#) ([pmath\\_t](#) file\_object) throw ()
- [WriteableTextFile](#) (const [Expr](#) &file\_object) throw ()
- [WriteableTextFile](#) (const [WriteableTextFile](#) &src) throw ()
- void [write](#) ([String](#) str) throw ()

*Write some text to the file.*

### Static Public Member Functions

- static [WriteableTextFile](#) [create\\_from\\_binary](#) ([WriteableBinaryFile](#) binfile, const char \*encoding) throw ()  
*Create a text file from a binary file using a known character encoding.*
- static [WriteableTextFile](#) [create\\_from\\_binary](#) ([WriteableBinaryFile](#) binfile) throw ()  
*Create a text file from a binary file using UTF-16BE or UTF-16LE, depending on the machine architecture.*

#### 11.37.1 Detailed Description

A wrapper for pMath writeable text file objects (byte data streams).

#### 11.37.2 Constructor & Destructor Documentation

##### 11.37.2.1 [WriteableTextFile](#) () throw () [inline]

##### 11.37.2.2 [WriteableTextFile](#) ([pmath\\_t](#) file\_object) throw () [inline, explicit]

**11.37.2.3** `WriteableTextFile (const Expr & file_object) throw ()` [`inline`, `explicit`]

**11.37.2.4** `WriteableTextFile (const WriteableTextFile & src) throw ()` [`inline`]

### 11.37.3 Member Function Documentation

**11.37.3.1** `static WriteableTextFile create_from_binary (WriteableBinaryFile binfile) throw ()` [`inline`, `static`]

Create a text file from a binary file using UTF-16BE or UTF-16LE, depending on the machine architecture.

**11.37.3.2** `static WriteableTextFile create_from_binary (WriteableBinaryFile binfile, const char * encoding) throw ()` [`inline`, `static`]

Create a text file from a binary file using a known character encoding.

**11.37.3.3** `void write (String str) throw ()` [`inline`]

Write some text to the file.

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

- [pmath-cpp.h](#)

## 12 File Documentation

### 12.1 pmath-config.h File Reference

#### Defines

- `#define PMATH_CONCAT_(a, b) a##b`
- `#define PMATH_CONCAT(a, b) PMATH_CONCAT_(a, b)`
- `#define PMATH_STATIC_ASSERT(e) typedef char PMATH_CONCAT(pmath_static_assert_line_, __LINE__)[(e)?1:-1]`



- #define [PMATH\\_NEED\\_GNUC](#)(maj, min) 0
- #define [PMATH\\_ATTRIBUTE\\_PURE](#)
- #define [PMATH\\_ATTRIBUTE\\_USE\\_RESULT](#)
- #define [PMATH\\_ATTRIBUTE\\_NONNULL](#)(...)
- #define [PMATH\\_DEPRECATED](#)
- #define [PMATH\\_LIKELY](#)(cond) (cond)
- #define [PMATH\\_UNLIKELY](#)(cond) (cond)
- #define [PMATH\\_UNUSED](#)
- #define [PMATH\\_FORCE\\_INLINE](#) static \_\_inline
- #define [PMATH\\_INLINE](#) \_\_inline
- #define [PMATH\\_EXTERN\\_C](#)
- #define [PMATH\\_MODULE](#) [PMATH\\_EXTERN\\_C](#) \_\_attribute\_\_((\_\_visibility\_\_("default")))
- #define [PMATH\\_API](#) [PMATH\\_EXTERN\\_C](#) \_\_attribute\_\_((\_\_visibility\_\_("default")))
- #define [PMATH\\_BYTE\\_ORDER](#) (1)

### 12.1.1 Define Documentation

**12.1.1.1 #define [PMATH\\_API](#) [PMATH\\_EXTERN\\_C](#) [\\_\\_attribute\\_\\_\(\(\\_\\_visibility\\_\\_\("default"\)\)\)](#)**

**12.1.1.2 #define [PMATH\\_ATTRIBUTE\\_NONNULL](#)( ...)**

**12.1.1.3 #define [PMATH\\_ATTRIBUTE\\_PURE](#)**

**12.1.1.4 #define [PMATH\\_ATTRIBUTE\\_USE\\_RESULT](#)**

**12.1.1.5 #define [PMATH\\_BYTE\\_ORDER](#) (1)**

**12.1.1.6 #define [PMATH\\_CONCAT](#)(a, b) [PMATH\\_CONCAT](#)\_(a, b)**

**12.1.1.7** `#define PMATH_CONCAT_(a, b) a##b`

**12.1.1.8** `#define PMATH_DEPRECATED`

**12.1.1.9** `#define PMATH_EXTERN_C`

**12.1.1.10** `#define PMATH_FORCE_INLINE static __inline`

**12.1.1.11** `#define PMATH_INLINE __inline`

**12.1.1.12** `#define PMATH_LIKELY(cond) (cond)`

**12.1.1.13** `#define PMATH_MODULE PMATH_EXTERN_C  
__attribute__((__visibility__("default")))`

**12.1.1.14** `#define PMATH_NEED_GNUC(maj, min) 0`

**12.1.1.15** `#define PMATH_STATIC_ASSERT(e) typedef char  
PMATH_CONCAT(pmath_static_assert_line_, __LINE__)[(e)?1:-1]`

**12.1.1.16** `#define PMATH_UNLIKELY(cond) (cond)`

### 12.1.1.17 #define PMATH\_UNUSED

## 12.2 pmath-core/custom.h File Reference

### Typedefs

- typedef [pmath\\_t](#) [pmath\\_custom\\_t](#)

## 12.3 pmath-core/expressions.h File Reference

### Typedefs

- typedef [pmath\\_t](#) [pmath\\_expr\\_t](#)

## 12.4 pmath-core/numbers.h File Reference

### Defines

- #define [PMATH\\_MACHINE\\_PRECISION](#) 0
- #define [PMATH\\_AUTO\\_PRECISION](#) 1
- #define [pmath\\_integer\\_new\\_si32](#)(si) [PMATH\\_FROM\\_INT32](#)(si)  
*Create an integer object from an int32\_t.*
- #define [pmath\\_integer\\_new\\_siptr](#)(si)  
*Create an integer object from an intptr\_t.*
- #define [pmath\\_integer\\_new\\_uiptr](#)(ui)  
*Create an integer object from an uintptr\_t.*
- #define [pmath\\_integer\\_fits\\_si32](#)(integer) [pmath\\_is\\_int32](#)(integer)  
*Check whether a pMath integer is in range  $-2^{31} .. 2^{31}-1$ .*
- #define [pmath\\_integer\\_fits\\_siptr](#)(integer)  
*Check whether a pMath integer fits into an intptr\_t.*
- #define [pmath\\_integer\\_fits\\_uiptr](#)(integer)  
*Check whether a pMath integer fits into an uintptr\_t.*
- #define [pmath\\_integer\\_get\\_siptr](#)  
*Convert a pMath integer to a intptr\_t.*
- #define [pmath\\_integer\\_get\\_uiptr](#)  
*Convert a pMath integer to a uintptr\_t.*

### Typedefs

- typedef `pmath_t` `pmath_number_t`
- typedef `pmath_number_t` `pmath_rational_t`
- typedef `pmath_rational_t` `pmath_integer_t`
- typedef `pmath_rational_t` `pmath_mpint_t`
- typedef `pmath_rational_t` `pmath_quotient_t`
- typedef `pmath_number_t` `pmath_float_t`
- typedef `pmath_float_t` `pmath_mpfloa_t`

### Enumerations

- enum `pmath_precision_control_t` {  
`PMATH_PREC_CTRL_AUTO` = 0, `PMATH_PREC_CTRL_MACHINE_PREC` = 1,  
`PMATH_PREC_CTRL_GIVEN_PREC` = 2, `PMATH_PREC_CTRL_GIVEN_ACC` = 3 }

## 12.5 pmath-core/objects-inline.h File Reference

### Defines

- #define `pmath_is_double(obj)` (((obj).s.tag & PMATH\_TAGMASK\_NONDOUBLE) != PMATH\_TAGMASK\_NONDOUBLE)
- #define `pmath_is_pointer(obj)` (((obj).s.tag & PMATH\_TAGMASK\_POINTER) == PMATH\_TAGMASK\_POINTER)
- #define `pmath_is_magic(obj)` ((obj).s.tag == PMATH\_TAG\_MAGIC)
- #define `pmath_is_int32(obj)` ((obj).s.tag == PMATH\_TAG\_INT32)
- #define `pmath_is_str0(obj)` ((obj).s.tag == PMATH\_TAG\_STR0)
- #define `pmath_is_str1(obj)` ((obj).s.tag == PMATH\_TAG\_STR1)
- #define `pmath_is_str2(obj)` ((obj).s.tag == PMATH\_TAG\_STR2)
- #define `pmath_is_ministr(obj)` (pmath\_is\_str0(obj) || pmath\_is\_str1(obj) || pmath\_is\_str2(obj))
- #define `PMATH_AS_TAG(obj)` ((obj).s.tag)
- #define `PMATH_AS_INT32(obj)` ((obj).s.u.as\_int32)
- #define `pmath_same(objA, objB)` ((objA).as\_bits == (objB).as\_bits)
- #define `pmath_is_null(obj)` (pmath\_same(obj, `PMATH_NULL`))
- #define `pmath_is_mpint(obj)` (pmath\_is\_pointer\_of(obj, PMATH\_TYPE\_MP\_INT))
- #define `pmath_is_mpfloa(obj)` (pmath\_is\_pointer\_of(obj, PMATH\_TYPE\_MP\_FLOAT))
- #define `pmath_is_custom(obj)` (pmath\_is\_pointer\_of(obj, PMATH\_TYPE\_CUSTOM))
- #define `pmath_is_expr(obj)` (pmath\_is\_pointer\_of(obj, PMATH\_TYPE\_EXPRESSION))
- #define `pmath_is_float(obj)` (pmath\_is\_double(obj) || pmath\_is\_mpfloa(obj))

- `#define pmath_is_integer(obj) (pmath_is_int32(obj) || pmath_is_mpint(obj))`
- `#define pmath_is_number(obj) (pmath_is_float(obj) || pmath_is_rational(obj))`
- `#define pmath_is_quotient(obj) (pmath_is_pointer_of(obj, PMATH_TYPE_QUOTIENT))`
- `#define pmath_is_rational(obj) (pmath_is_integer(obj) || pmath_is_quotient(obj))`
- `#define pmath_is_bigstr(obj) (pmath_is_pointer_of(obj, PMATH_TYPE_BIGSTRING))`
- `#define pmath_is_string(obj) (pmath_is_ministr(obj) || pmath_is_bigstr(obj))`
- `#define pmath_is_symbol(obj) (pmath_is_pointer_of(obj, PMATH_TYPE_SYMBOL))`

## Functions

- `struct _pmath_t * PMATH_AS_PTR (pmath_t obj)`
- `double PMATH_AS_DOUBLE (pmath_t obj)`
- `pmath_bool_t pmath_is_pointer_of (pmath_t obj, pmath_type_t type)`
- `pmath_bool_t pmath_is_evaluable (pmath_t obj)`
- `pmath_t PMATH_FROM_DOUBLE (double d)`
- `intptr_t pmath_refcount (pmath_t obj)`

### 12.5.1 Define Documentation

**12.5.1.1** `#define PMATH_AS_INT32(obj) ((obj).s.u.as_int32)`

**12.5.1.2** `#define PMATH_AS_TAG(obj) ((obj).s.tag)`

**12.5.1.3** `#define pmath_is_bigstr(obj) (pmath_is_pointer_of(obj, PMATH_TYPE_BIGSTRING))`

**12.5.1.4** `#define pmath_is_custom(obj) (pmath_is_pointer_of(obj, PMATH_TYPE_CUSTOM))`

**12.5.1.5** `#define pmath_is_double(obj) (((obj).s.tag & PMATH_TAGMASK_NONDOUBLE) != PMATH_TAGMASK_NONDOUBLE)`

**12.5.1.6** `#define pmath_is_expr(obj) (pmath_is_pointer_of(obj,  
PMATH_TYPE_EXPRESSION))`

**12.5.1.7** `#define pmath_is_float(obj) (pmath_is_double(obj) ||  
pmath_is_mpfloor(obj))`

**12.5.1.8** `#define pmath_is_int32(obj) ((obj).s.tag == PMATH_TAG_INT32)`

**12.5.1.9** `#define pmath_is_integer(obj) (pmath_is_int32(obj) ||  
pmath_is_mpint(obj))`

**12.5.1.10** `#define pmath_is_magic(obj) ((obj).s.tag == PMATH_TAG_MAGIC)`

**12.5.1.11** `#define pmath_is_ministr(obj) (pmath_is_str0(obj) ||  
pmath_is_str1(obj) || pmath_is_str2(obj))`

**12.5.1.12** `#define pmath_is_mpfloor(obj) (pmath_is_pointer_of(obj,  
PMATH_TYPE_MP_FLOAT))`

**12.5.1.13** `#define pmath_is_mpint(obj) (pmath_is_pointer_of(obj,  
PMATH_TYPE_MP_INT))`

**12.5.1.14** `#define pmath_is_null(obj) (pmath_same(obj, PMATH_NULL))`

- 12.5.1.15 **#define pmath\_is\_number(obj) (pmath\_is\_float(obj) ||  
pmath\_is\_rational(obj))**
  
- 12.5.1.16 **#define pmath\_is\_pointer(obj) (((obj).s.tag & PMATH\_TAGMASK\_-  
POINTER) == PMATH\_TAGMASK\_POINTER)**
  
- 12.5.1.17 **#define pmath\_is\_quotient(obj) (pmath\_is\_pointer\_of(obj,  
PMATH\_TYPE\_QUOTIENT))**
  
- 12.5.1.18 **#define pmath\_is\_rational(obj) (pmath\_is\_integer(obj) ||  
pmath\_is\_quotient(obj))**
  
- 12.5.1.19 **#define pmath\_is\_str0(obj) ((obj).s.tag == PMATH\_TAG\_STR0)**
  
- 12.5.1.20 **#define pmath\_is\_str1(obj) ((obj).s.tag == PMATH\_TAG\_STR1)**
  
- 12.5.1.21 **#define pmath\_is\_str2(obj) ((obj).s.tag == PMATH\_TAG\_STR2)**
  
- 12.5.1.22 **#define pmath\_is\_string(obj) (pmath\_is\_ministr(obj) ||  
pmath\_is\_bigstr(obj))**
  
- 12.5.1.23 **#define pmath\_is\_symbol(obj) (pmath\_is\_pointer\_of(obj,  
PMATH\_TYPE\_SYMBOL))**

12.5.1.24 `#define pmath_same(objA, objB) ((objA).as_bits == (objB).as_bits)`

## 12.5.2 Function Documentation

12.5.2.1 `double PMATH_AS_DOUBLE (pmath_t obj)`

12.5.2.2 `struct _pmath_t* PMATH_AS_PTR (pmath_t obj)` [read]

12.5.2.3 `pmath_t PMATH_FROM_DOUBLE (double d)`

12.5.2.4 `pmath_bool_t pmath_is_evaluable (pmath_t obj)`

12.5.2.5 `pmath_bool_t pmath_is_pointer_of (pmath_t obj, pmath_type_t type)`

12.5.2.6 `intptr_t pmath_refcount (pmath_t obj)`

## 12.6 pmath-core/objects.h File Reference

### Data Structures

- class [pmath\\_t](#)  
*The basic type of all pMath objects.*
- struct [pmath\\_write\\_ex\\_t](#)  
*Command structure for [pmath\\_write\\_ex\(\)](#). This should be initalized with `memset(&ex, 0, sizeof(ex)); ex.size = sizeof(ex); ...`*



**Defines**

- #define [PMATH\\_TAGMASK\\_BITCOUNT](#) 12
- #define [PMATH\\_TAGMASK\\_NONDOUBLE](#) 0x7FF00000U
- #define [PMATH\\_TAGMASK\\_POINTER](#) 0xFFF00000U
- #define [PMATH\\_TAG\\_INVALID](#) (PMATH\_TAGMASK\_NONDOUBLE | 0xFFFFF)
- #define [PMATH\\_TAG\\_MAGIC](#) (PMATH\_TAGMASK\_NONDOUBLE | 0x10000)
- #define [PMATH\\_TAG\\_INT32](#) (PMATH\_TAGMASK\_NONDOUBLE | 0x20000)
- #define [PMATH\\_TAG\\_STR0](#) (PMATH\_TAGMASK\_NONDOUBLE | 0x30000)
- #define [PMATH\\_TAG\\_STR1](#) (PMATH\_TAGMASK\_NONDOUBLE | 0x40000)
- #define [PMATH\\_TAG\\_STR2](#) (PMATH\_TAGMASK\_NONDOUBLE | 0x50000)
- #define [PMATH\\_FROM\\_INT32\(i\)](#) (PMATH\_FROM\_TAG(PMATH\_TAG\_-INT32, (i)))
- #define [PMATH\\_THREAD\\_KEY\\_PARSESYMBOLS](#) PMATH\_FROM\_TAG(PMATH\_TAG\_MAGIC, 252)
- #define [PMATH\\_THREAD\\_KEY\\_PARSERARGUMENTS](#) PMATH\_FROM\_TAG(PMATH\_TAG\_MAGIC, 253)
- #define [PMATH\\_ABORT\\_EXCEPTION](#) PMATH\_FROM\_TAG(PMATH\_TAG\_MAGIC, 254)
- #define [PMATH\\_STATIC\\_UNDEFINED](#) { (((uint64\_t)PMATH\_TAG\_MAGIC) << 32) | 255 }
- #define [PMATH\\_STATIC\\_NULL](#) { ((uint64\_t)PMATH\_TAGMASK\_POINTER) << 32 }

**Typedefs**

- typedef int [pmath\\_type\\_t](#)  
*The type or class of a pMath object.*
- typedef int [pmath\\_write\\_options\\_t](#)  
*Options for [pmath\\_write\(\)](#).*
- typedef void(\* [pmath\\_proc\\_t](#))([pmath\\_t](#))  
*A simple procedure operating on an object.*
- typedef void(\* [pmath\\_param\\_proc\\_t](#))(void \*, [pmath\\_t](#))  
*A parameterized procedure operating on an object.*
- typedef [pmath\\_t](#)(\* [pmath\\_func\\_t](#))([pmath\\_t](#))  
*A simple function operating on an object and returning one.*

- typedef unsigned int(\* [pmath\\_hash\\_func\\_t](#))([pmath\\_t](#))  
*A hash function for an object.*
- typedef [pmath\\_bool\\_t](#)(\* [pmath\\_equal\\_func\\_t](#))([pmath\\_t](#), [pmath\\_t](#))  
*A comparison function for two objects.*
- typedef int(\* [pmath\\_compare\\_func\\_t](#))([pmath\\_t](#), [pmath\\_t](#))  
*A comparison function to determine the order of two objects.*

### Enumerations

- enum {  
[PMATH\\_TYPE\\_SHIFT\\_MP\\_FLOAT](#) = 0, [PMATH\\_TYPE\\_SHIFT\\_MP\\_INT](#),  
[PMATH\\_TYPE\\_SHIFT\\_QUOTIENT](#), [PMATH\\_TYPE\\_SHIFT\\_BIGSTRING](#),  
[PMATH\\_TYPE\\_SHIFT\\_SYMBOL](#), [PMATH\\_TYPE\\_SHIFT\\_EXPRESSION\\_GENERAL](#),  
[PMATH\\_TYPE\\_SHIFT\\_EXPRESSION\\_GENERAL\\_PART](#), [PMATH\\_TYPE\\_SHIFT\\_RESERVED\\_1](#),  
[PMATH\\_TYPE\\_SHIFT\\_CUSTOM](#), [PMATH\\_TYPE\\_SHIFT\\_COUNT](#) }
- enum {  
[PMATH\\_TYPE\\_MP\\_INT](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_MP\\_INT](#),  
[PMATH\\_TYPE\\_QUOTIENT](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_QUOTIENT](#),  
[PMATH\\_TYPE\\_MP\\_FLOAT](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_MP\\_FLOAT](#),  
[PMATH\\_TYPE\\_BIGSTRING](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_BIGSTRING](#),  
[PMATH\\_TYPE\\_SYMBOL](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_SYMBOL](#),  
[PMATH\\_TYPE\\_EXPRESSION\\_GENERAL](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_EXPRESSION\\_GENERAL](#),  
[PMATH\\_TYPE\\_EXPRESSION\\_GENERAL\\_PART](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_EXPRESSION\\_GENERAL\\_PART](#), [PMATH\\_TYPE\\_EXPRESSION](#) = [PMATH\\_TYPE\\_EXPRESSION\\_GENERAL](#) | [PMATH\\_TYPE\\_EXPRESSION\\_GENERAL\\_PART](#),  
[PMATH\\_TYPE\\_CUSTOM](#) = 1 << [PMATH\\_TYPE\\_SHIFT\\_CUSTOM](#) }
- enum {  
[PMATH\\_WRITE\\_OPTIONS\\_FULLEXPR](#) = 1 << 0, [PMATH\\_WRITE\\_OPTIONS\\_FULLSTR](#) = 1 << 1,  
[PMATH\\_WRITE\\_OPTIONS\\_FULLNAME](#) = 1 << 2, [PMATH\\_WRITE\\_OPTIONS\\_INPUTEXPR](#) = 1 << 3 }

### Functions

- [pmath\\_t](#) [PMATH\\_FROM\\_TAG](#) (uint32\_t tag, int32\_t value)
- [pmath\\_t](#) [PMATH\\_FROM\\_PTR](#) (void \*p)

### Variables

- static PMATH\_UNUSED const [pmath\\_t](#) PMATH\_UNDEFINED  
*Magic value to indicate unset variable values/...*
- static PMATH\_UNUSED const [pmath\\_t](#) PMATH\_NULL  
*The NULL pointer.  $\wedge$  in pMath.*

## 12.7 pmath-core/strings.h File Reference

### Data Structures

- struct [pmath\\_cstr\\_writer\\_info\\_t](#)  
*Additional information for [pmath\\_utf8\\_writer\(\)](#) or [pmath\\_native\\_writer\(\)](#).*

### Defines

- #define [PMATH\\_C\\_STRING](#)(cstr) [pmath\\_string\\_insert\\_latin1](#)(PMATH\_NULL, 0, (cstr), -1)  
*Short form to convert a C String to a pMath String.*

### Typedefs

- typedef [pmath\\_t](#) [pmath\\_string\\_t](#)

### Functions

- void [pmath\\_utf8\\_writer](#) (void \*user, const uint16\_t \*data, int len)  
*A write function for [pmath\\_write\(\)](#) that converts to utf8.*
- void [pmath\\_native\\_writer](#) (void \*user, const uint16\_t \*data, int len)  
*A write function for [pmath\\_write\(\)](#) that converts to the current console encoding.*

## 12.8 pmath-core/symbols.h File Reference

### Typedefs

- typedef [pmath\\_t](#) [pmath\\_symbol\\_t](#)
- typedef int [pmath\\_symbol\\_attributes\\_t](#)  
*The (bitset) type of symbol attributes.*

## Enumerations

- enum {
   
`PMATH_SYMBOL_ATTRIBUTE_PROTECTED` = 1 << 0, `PMATH_SYMBOL_ATTRIBUTE_HOLDFIRST` = 1 << 1,
   
`PMATH_SYMBOL_ATTRIBUTE_HOLDREST` = 1 << 2, `PMATH_SYMBOL_ATTRIBUTE_HOLDALL` = `PMATH_SYMBOL_ATTRIBUTE_HOLDFIRST` | `PMATH_SYMBOL_ATTRIBUTE_HOLDREST`,
   
`PMATH_SYMBOL_ATTRIBUTE_SYMMETRIC` = 1 << 3, `PMATH_SYMBOL_ATTRIBUTE_ASSOCIATIVE` = 1 << 4,
   
`PMATH_SYMBOL_ATTRIBUTE_NHOLDFIRST` = 1 << 5, `PMATH_SYMBOL_ATTRIBUTE_NHOLDREST` = 1 << 6,
   
`PMATH_SYMBOL_ATTRIBUTE_NHOLDALL` = `PMATH_SYMBOL_ATTRIBUTE_NHOLDFIRST` | `PMATH_SYMBOL_ATTRIBUTE_NHOLDREST`, `PMATH_SYMBOL_ATTRIBUTE_TEMPORARY` = 1 << 7,
   
`PMATH_SYMBOL_ATTRIBUTE_LISTABLE` = 1 << 8, `PMATH_SYMBOL_ATTRIBUTE_DEEPHOLDALL` = 1 << 9,
   
`PMATH_SYMBOL_ATTRIBUTE_HOLDALLCOMPLETE` = 1 << 10, `PMATH_SYMBOL_ATTRIBUTE_ONEIDENTITY` = 1 << 11,
   
`PMATH_SYMBOL_ATTRIBUTE_THREADLOCAL` = 1 << 12, `PMATH_SYMBOL_ATTRIBUTE_NUMERICFUNCTION` = 1 << 13,
   
`PMATH_SYMBOL_ATTRIBUTE_READPROTECTED` = 1 << 14, `PMATH_SYMBOL_ATTRIBUTE_SEQUENCEHOLD` = 1 << 15,
   
`PMATH_SYMBOL_ATTRIBUTE_REMOVED` = 1 << 16, `PMATH_SYMBOL_ATTRIBUTE_DEFINITEFUNCTION` = 1 << 17 }

## Functions

- void `pmath_symbol_remove` (`pmath_symbol_t` symbol)
   
*Remove a symbol completely from the system.*
- `pmath_symbol_t` `pmath_symbol_iter_next` (`pmath_symbol_t` old)
   
*Iterate through the global symbol table.*

## 12.9 pmath-cpp.h File Reference

### Data Structures

- class `Expr`
  
*A wrapper for `pmath_t` and derived types.*
- class `String`

A wrapper for [pmath\\_string\\_t](#).

- class [Gather](#)

Utility class for emitting and gathering expressions/building lists.

- class [File](#)

A wrapper for pMath file objects (data streams).

- class [BinaryFile](#)

A wrapper for pMath binary file objects (byte data streams).

- class [ReadableBinaryFile](#)

A wrapper for readable pMath binary file objects (byte data streams).

- class [WritableBinaryFile](#)

A wrapper for writable pMath binary file objects (byte data streams).

- class [TextFile](#)

A wrapper for pMath text file objects (byte data streams).

- class [ReadableTextFile](#)

A wrapper for pMath readable text file objects (byte data streams).

- class [WritableTextFile](#)

A wrapper for pMath writable text file objects (byte data streams).

- class [UserStream](#)

Abstract base class for C++ callbacks used as pMath files.

- class [BinaryUserStream](#)

Abstract base class for C++ callbacks used as pMath binary files.

- class [TextUserStream](#)

Abstract base class for C++ callbacks used as pMath text files.

## Namespaces

- namespace [pmath](#)

Provides the C++ binding.

## Functions

- Expr [Number](#) (double d)
- Expr [Complex](#) (Expr re, Expr im)
- Expr [Imaginary](#) (Expr im)
- Expr [Rational](#) (Expr num, Expr den)
- Expr [Ref](#) ([pmath\\_t](#) o)
- Expr [Symbol](#) ([pmath\\_symbol\\_t](#) h)
- Expr [SymbolPi](#) ()
- Expr [MakeList](#) (size\_t len)
- Expr [Call](#) (Expr h)
- Expr [Call](#) (Expr h, Expr x1)
- Expr [Call](#) (Expr h, Expr x1, Expr x2)
- Expr [Call](#) (Expr h, Expr x1, Expr x2, Expr x3)
- Expr [Call](#) (Expr h, Expr x1, Expr x2, Expr x3, Expr x4)
- Expr [Call](#) (Expr h, Expr x1, Expr x2, Expr x3, Expr x4, Expr x5)
- Expr [Call](#) (Expr h, Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6)
- Expr [Call](#) (Expr h, Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6, Expr x7)
- Expr [Call](#) (Expr h, Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6, Expr x7, Expr x8)
- Expr [Call](#) (Expr h, Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6, Expr x7, Expr x8, Expr x9)
- Expr [List](#) ()
- Expr [List](#) (Expr x1)
- Expr [List](#) (Expr x1, Expr x2)
- Expr [List](#) (Expr x1, Expr x2, Expr x3)
- Expr [List](#) (Expr x1, Expr x2, Expr x3, Expr x4)
- Expr [List](#) (Expr x1, Expr x2, Expr x3, Expr x4, Expr x5)
- Expr [List](#) (Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6)
- Expr [List](#) (Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6, Expr x7)
- Expr [List](#) (Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6, Expr x7, Expr x8)
- Expr [List](#) (Expr x1, Expr x2, Expr x3, Expr x4, Expr x5, Expr x6, Expr x7, Expr x8, Expr x9)
- Expr [Rule](#) (Expr l, Expr r)
- Expr [RuleDelayed](#) (Expr l, Expr r)
- Expr [Power](#) (Expr x, Expr y)
- Expr [Sqrt](#) (Expr x)
- Expr [Inv](#) (Expr x)
- Expr [Exp](#) (Expr x)
- Expr [Log](#) (Expr x)
- Expr [Log](#) (Expr b, Expr x)
- Expr [Sin](#) (Expr x)
- Expr [Cos](#) (Expr x)
- Expr [Tan](#) (Expr x)
- Expr [ArcSin](#) (Expr x)

- Expr [ArcCos](#) (Expr x)
- Expr [ArcTan](#) (Expr x)
- Expr [Times](#) (Expr x1, Expr x2)
- Expr [Times](#) (Expr x1, Expr x2, Expr x3)
- Expr [Times](#) (Expr x1, Expr x2, Expr x3, Expr x4)
- Expr [Divide](#) (Expr x, Expr y)
- Expr [Plus](#) (Expr x1, Expr x2)
- Expr [Plus](#) (Expr x1, Expr x2, Expr x3)
- Expr [Plus](#) (Expr x1, Expr x2, Expr x3, Expr x4)
- Expr [Minus](#) (Expr x)
- Expr [Minus](#) (Expr x, Expr y)
- Expr [Abs](#) (Expr x)
- Expr [Arg](#) (Expr x)
- Expr [Sign](#) (Expr x)
- Expr [Re](#) (Expr x)
- Expr [Im](#) (Expr x)
- Expr [Ceiling](#) (Expr x)
- Expr [Ceiling](#) (Expr x, Expr a)
- Expr [Floor](#) (Expr x)
- Expr [Floor](#) (Expr x, Expr a)
- Expr [Round](#) (Expr x)
- Expr [Round](#) (Expr x, Expr a)
- Expr [Quotient](#) (Expr m, Expr n)
- Expr [Quotient](#) (Expr m, Expr n, Expr d)
- Expr [Mod](#) (Expr m, Expr n)
- Expr [Mod](#) (Expr m, Expr n, Expr d)
- Expr [Evaluate](#) (Expr x)
- Expr [ParseArgs](#) (const char \*code, Expr arglist)
- Expr [Parse](#) (String code)
- Expr [Parse](#) (const char \*code)
- Expr [Parse](#) (const char \*code, Expr x1)
- Expr [Parse](#) (const char \*code, Expr x1, Expr x2)
- Expr [Parse](#) (const char \*code, Expr x1, Expr x2, Expr x3)
- Expr [Parse](#) (const char \*code, Expr x1, Expr x2, Expr x3, Expr x4)

## 12.10 pmath-language/charnames.h File Reference

### Functions

- uint32\_t [pmath\\_char\\_from\\_name](#) (const char \*name)  
*Get a named character.*
- const char \* [pmath\\_char\\_to\\_name](#) (uint32\_t unichar)  
*Get a character's name.*

## 12.11 pmath-language/scanner.h File Reference

### Defines

- #define [PMATH\\_RUN](#)(code)  
*Execute some pMath code.*
- #define [PMATH\\_RUN\\_ARGS](#)(code, format,...)  
*Execute some pMath code with arguments.*

### Typedefs

- typedef struct \_pmath\_span\_array\_t [pmath\\_span\\_array\\_t](#)
- typedef struct \_pmath\_span\_t [pmath\\_span\\_t](#)

### Functions

- [pmath\\_span\\_array\\_t](#) \* [pmath\\_spans\\_from\\_string](#) ([pmath\\_string\\_t](#) \*code, [pmath\\_string\\_t](#)(\*line\_reader)(void \*), [pmath\\_bool\\_t](#)(\*subsuperscriptbox\_at\_index)(int, void \*), [pmath\\_string\\_t](#)(\*underoverscriptbox\_at\_index)(int, void \*), void(\*error)([pmath\\_string\\_t](#), int, void \*, [pmath\\_bool\\_t](#)), void \*data)  
*Parses pMath code to a span array.*
- [pmath\\_t](#) [pmath\\_boxes\\_from\\_spans](#) ([pmath\\_span\\_array\\_t](#) \*spans, [pmath\\_string\\_t](#) string, [pmath\\_bool\\_t](#) parseable, [pmath\\_t](#)(\*box\_at\_index)(int, void \*), void \*data)  
*Convert a span-array with the according code to boxed form.*
- [pmath\\_span\\_array\\_t](#) \* [pmath\\_spans\\_from\\_boxes](#) ([pmath\\_t](#) boxes, [pmath\\_string\\_t](#) \*result\_string, void(\*make\_box)(int, [pmath\\_t](#), void \*), void \*data)  
*Convert boxed form back to span-array and code.*

## 12.12 pmath-language/tokens.h File Reference

### Defines

- #define [PMATH\\_CHAR\\_INVISIBLECALL](#) 0x2061  
*The Function application character.*
- #define [PMATH\\_CHAR\\_VECTOR](#) 0x21C0  
*The arrow above names to indicate a vector.*
- #define [PMATH\\_CHAR\\_RULE](#) 0x2192  
*The "  $\rightarrow$  " operator.*



- #define [PMATH\\_CHAR\\_RULEDELAYED](#) 0x29F4  
*The "·>" operator.*
- #define [PMATH\\_CHAR\\_ASSIGN](#) 0x2254  
*The "·=" operator.*
- #define [PMATH\\_CHAR\\_ASSIGNDELAYED](#) 0x2A74  
*The "·:= " operator.*
- #define [PMATH\\_CHAR\\_INTEGRAL\\_D](#) 0x2146  
*The integral "d".*
- #define [PMATH\\_CHAR\\_PIECEWISE](#) 0xF361  
*The left curly bracket for cases.*
- #define [PMATH\\_CHAR\\_ALIASDELIMITER](#) 0xF764  
*The character inserted by Richmath with ESCAPE or CAPSLOCK.*
- #define [PMATH\\_CHAR\\_ALIASINDICATOR](#) 0xF768  
*A character that looks like PMATH\_CHAR\_ALIASDELIMITER but has no effect.*
- #define [PMATH\\_CHAR\\_LEFT\\_BOX](#) 0xFF9  
*Start of box code inside a string.*
- #define [PMATH\\_CHAR\\_RIGHT\\_BOX](#) 0xFFB  
*End of box code inside a string.*
- #define [PMATH\\_CHAR\\_BOX](#) 0xFDD0  
*Represents a box.*
- #define [PMATH\\_CHAR\\_PLACEHOLDER](#) 0xFFFD  
*The Placeholder character. In richmath, type CAPSLOCK pl CAPSLOCK to insert it.*

## Enumerations

- enum [pmath\\_token\\_t](#) {  
    [PMATH\\_TOK\\_NONE](#), [PMATH\\_TOK\\_SPACE](#),  
    [PMATH\\_TOK\\_DIGIT](#), [PMATH\\_TOK\\_STRING](#),  
    [PMATH\\_TOK\\_NAME](#), [PMATH\\_TOK\\_NAME2](#),  
    [PMATH\\_TOK\\_BINARY\\_LEFT](#), [PMATH\\_TOK\\_BINARY\\_RIGHT](#),  
    [PMATH\\_TOK\\_BINARY\\_LEFT\\_AUTOARG](#),      [PMATH\\_TOK\\_BINARY\\_-](#)  
    [LEFT\\_OR\\_PREFIX](#),  
    [PMATH\\_TOK\\_NARY](#), [PMATH\\_TOK\\_NARY\\_AUTOARG](#),  
}

```

PMATH_TOK_NARY_OR_PREFIX, PMATH_TOK_POSTFIX_OR_PREFIX,
PMATH_TOK_PREFIX, PMATH_TOK_POSTFIX,
PMATH_TOK_CALL, PMATH_TOK_LEFTCALL,
PMATH_TOK_LEFT, PMATH_TOK_RIGHT,
PMATH_TOK_PRETEXT, PMATH_TOK_ASSIGNTAG,
PMATH_TOK_PLUSPLUS, PMATH_TOK_COLON,
PMATH_TOK_TILDES, PMATH_TOK_SLOT,
PMATH_TOK_QUESTION, PMATH_TOK_INTEGRAL,
PMATH_TOK_COMMENTEND }

```

*Token classes known in the pMath language.*

- enum {
 

```

PMATH_PREC_ANY = 0, PMATH_PREC_SEQ = 10,
PMATH_PREC_EVAL = 20, PMATH_PREC_ASS = 30,
PMATH_PREC_MODY = 40, PMATH_PREC_LAZY = 50,
PMATH_PREC_FUNC = 60, PMATH_PREC_REPL = 80,
PMATH_PREC_RULE = 90, PMATH_PREC_MAP = 100,
PMATH_PREC_STR = 110, PMATH_PREC_COND = 120,
PMATH_PREC_ALT = 130, PMATH_PREC_OR = 150,
PMATH_PREC_XOR = 155, PMATH_PREC_AND = 160,
PMATH_PREC_ARROW = 170, PMATH_PREC_REL = 180,
PMATH_PREC_UNION = 190, PMATH_PREC_ISECT = 200,
PMATH_PREC_RANGE = 210, PMATH_PREC_ADD = 220,
PMATH_PREC_CIRCADD = 230, PMATH_PREC_PLUMI = 240,
PMATH_PREC_CIRC MUL = 250, PMATH_PREC_MUL = 260,
PMATH_PREC_DIV = 270, PMATH_PREC_MIDDOT = 280,
PMATH_PREC_CROSS = 290, PMATH_PREC_MUL2 = 300,
PMATH_PREC_POW = 310, PMATH_PREC_FAC = 320,
PMATH_PREC_APL = 330, PMATH_PREC_REPEAT = 340,
PMATH_PREC_TEST = 350, PMATH_PREC_INC = 360,
PMATH_PREC_CALL = 400, PMATH_PREC_DIFF = 410,
PMATH_PREC_PRIM = 1000 }

```

## Functions

- `pmath_token_t pmath_token_analyse` (const uint16\_t \*str, int len, int \*prec)
 

*Analyse a token.*
- `int pmath_token_prefix_precedence` (const uint16\_t \*str, int len, int defprec)

*Give the prefix operator precedence for a token.*

- static PMATH\_INLINE [pmath\\_bool\\_t](#) [pmath\\_token\\_maybe\\_first](#) ([pmath\\_token\\_t](#) tok)

*Test whether a token may be the first token in a subexpression.*

- static PMATH\_INLINE [pmath\\_bool\\_t](#) [pmath\\_token\\_maybe\\_rest](#) ([pmath\\_token\\_t](#) tok)

*Test whether a token need not be the first token in a subexpression.*

- static PMATH\_INLINE [pmath\\_bool\\_t](#) [pmath\\_char\\_is\\_left](#) (uint16\_t ch)

*Test if a unicode character is a left bracket.*

- static PMATH\_INLINE uint16\_t [pmath\\_right\\_fence](#) (uint16\_t left)

*Get the corresponding right bracket to a given left bracket or 0.*

- static PMATH\_INLINE [pmath\\_bool\\_t](#) [pmath\\_char\\_is\\_right](#) (uint16\_t ch)

*Test if a unicode character is a right bracket.*

- static PMATH\_INLINE [pmath\\_bool\\_t](#) [pmath\\_char\\_is\\_name](#) (uint16\_t ch)

*Test if a unicode character can be the start of an identifier/name.*

- static PMATH\_INLINE [pmath\\_bool\\_t](#) [pmath\\_char\\_is\\_integral](#) (uint16\_t ch)

*Test if a unicode character is an integral.*

- static PMATH\_INLINE [pmath\\_bool\\_t](#) [pmath\\_token\\_maybe\\_bigop](#) ([pmath\\_token\\_t](#) tok)

*Test if a token may be a big operator.*

- static PMATH\_INLINE [pmath\\_bool\\_t](#) [pmath\\_char\\_maybe\\_bigop](#) (uint16\_t ch)

*Test if a unicode character may be a big operation, e.g. Union, Sum.*

- static PMATH\_INLINE [pmath\\_bool\\_t](#) [pmath\\_char\\_is\\_digit](#) (uint16\_t ch)

*Test if a unicode character is a digit '0' - '9'.*

- static PMATH\_INLINE [pmath\\_bool\\_t](#) [pmath\\_char\\_is\\_36digit](#) (uint16\_t ch)

*Test if a unicode character is a base-36 digit '0' - '9', 'a' - 'z', 'A' - 'Z'.*

- static PMATH\_INLINE [pmath\\_bool\\_t](#) [pmath\\_char\\_is\\_basedigit](#) (int base, uint16\_t ch)

*Test if in a given base, a unicode character is a digit.*

- static PMATH\_INLINE [pmath\\_bool\\_t](#) [pmath\\_char\\_is\\_hexdigit](#) (uint16\_t ch)

*Test if a unicode character is a hexadecimal digit.*

## 12.13 pmath-types.h File Reference

### Defines

- #define `FALSE` ((`pmath_bool_t`)0)  
*The FALSE value for `pmath_bool_t`.*
- #define `TRUE` (!`FALSE`)  
*The TRUE value for `pmath_bool_t`.*
- #define `PMATH_INVALID_PTR` ((void\*)UINTPTR\_MAX)

### Typedefs

- typedef char `pmath_bool_t`  
*A boolean type.*
- typedef void(\* `pmath_callback_t`)(void \*)  
*A general callback function.*

## 12.14 pmath-util/approximate.h File Reference

### Functions

- double `pmath_accuracy` (`pmath_t` obj)  
*Get the accuracy (in bits) of an object.*
- double `pmath_precision` (`pmath_t` obj)  
*Get the precision (in bits) of an object.*
- `pmath_t` `pmath_set_accuracy` (`pmath_t` obj, double acc)  
*Set an object's accuracy in bits.*
- `pmath_t` `pmath_set_precision` (`pmath_t` obj, double prec)  
*Set an object's accuracy in bits.*
- `pmath_t` `pmath_approximate` (`pmath_t` obj, double precision\_goal, double accuracy\_goal, `pmath_bool_t` \*aborted)  
*Approximate an object.*

## 12.15 pmath-util/compression.h File Reference

### Functions

- [pmath\\_symbol\\_t pmath\\_file\\_create\\_compressor](#) ([pmath\\_t](#) dstfile)  
*Create a binary file object that compresses its input.*
- [pmath\\_symbol\\_t pmath\\_file\\_create\\_uncompressor](#) ([pmath\\_t](#) srcfile)  
*Create a binary file object that uncompresses its input.*

## 12.16 pmath-util/concurrency/atomic.h File Reference

### Data Structures

- struct [pmath\\_atomic\\_t](#)  
*A pointer-sized atomic variable type.*
- struct [pmath\\_atomic2\\_t](#)  
*A 2-pointer-sized atomic variable type.*

### Defines

- `#define PMATH\_ATOMIC\_STATIC\_INIT {0};`
- `#define PMATH\_ATOMIC\_FASTLOOP\_COUNT (1000)`
- `#define pmath\_atomic\_loop\_yield() (sched_yield())`
- `#define pmath\_atomic\_loop\_nop()`

#### 12.16.1 Define Documentation

##### 12.16.1.1 `#define PMATH\_ATOMIC\_FASTLOOP\_COUNT (1000)`

##### 12.16.1.2 `#define pmath\_atomic\_loop\_nop(void)`

### Value:

```
do{ \
    struct timespec tm; \
    tm.tv_sec = 0; \
    tm.tv_nsec = 2000001; \
    nanosleep(&tm, NULL); \
}while(0)
```

12.16.1.3 `#define pmath_atomic_loop_yield(void) (sched_yield())`

12.16.1.4 `#define PMATH_ATOMIC_STATIC_INIT {0};`

## 12.17 pmath-util/concurrency/atomic/non-atomic.h File Reference

### Defines

- `#define PMATH_ATOMIC_FASTLOOP_COUNT (0)`  
*Loop iterations in spinlocks before yielding control.*
- `#define PMATH_DECLARE_ALIGNED(TYPE, NAME, ALIGNMENT) TYPE NAME`  
*Declares a variable with specified alignment.*

### Functions

- `intptr_t pmath_atomic_fetch_add (pmath_atomic_t *atom, intptr_t delta)`  
*Add a value to another.*
- `intptr_t pmath_atomic_fetch_set (pmath_atomic_t *atom, intptr_t new_value)`  
*Exchange a value.*
- `intptr_t pmath_atomic_fetch_compare_and_set (pmath_atomic_t *atom, intptr_t old_value, intptr_t new_value)`  
*Exchange a value if it equals another value.*
- `pmath_bool_t pmath_atomic_compare_and_set (pmath_atomic_t *atom, intptr_t old_value, intptr_t new_value)`  
*Exchange a value if it equals another value.*
- `pmath_bool_t pmath_atomic_compare_and_set_2 (pmath_atomic2_t *atom, intptr_t old_value_fst, intptr_t old_value_snd, intptr_t new_value_fst, intptr_t new_value_snd)`  
*Exchange two values value if they equal another two values.*
- `pmath_bool_t pmath_atomic_have_cas2 (void)`  
*Check, whether the CPU supports `pmath_atomic_compare_and_set_2()`.*
- `void pmath_atomic_barrier (void)`  
*Insert an explicit memory barrier.*

- void [pmath\\_atomic\\_lock](#) ([pmath\\_atomic\\_t](#) \*atom)  
*Try to aquire a lock.*
- void [pmath\\_atomic\\_unlock](#) ([pmath\\_atomic\\_t](#) \*atom)  
*Release a previously aquired lock.*
- void [pmath\\_atomic\\_loop\\_yield](#) (void)  
*Yield control to another thread (used in spinlocks).*
- void [pmath\\_atomic\\_loop\\_nop](#) (void)  
*A no-operation or short wait for use in spin locks.*

## 12.18 pmath-util/concurrency/threadlocks.h File Reference

### Typedefs

- typedef struct \_pmath\_threadlock\_t \* [pmath\\_threadlock\\_t](#)

## 12.19 pmath-util/concurrency/threadmsg.h File Reference

### Typedefs

- typedef [pmath\\_custom\\_t](#) [pmath\\_messages\\_t](#)

### Functions

- double [pmath\\_tickcount](#) (void)  
*Gives the seconds since January 1, 1970 (UTC).*

## 12.20 pmath-util/concurrency/threadpool.h File Reference

### Typedefs

- typedef struct \_pmath\_task\_t \* [pmath\\_task\\_t](#)

### Functions

- [pmath\\_task\\_t](#) [pmath\\_task\\_ref](#) ([pmath\\_task\\_t](#) task)
- void [pmath\\_task\\_unref](#) ([pmath\\_task\\_t](#) task)
- void \* [pmath\\_task\\_get\\_data](#) ([pmath\\_task\\_t](#) task)
- [pmath\\_bool\\_t](#) [pmath\\_task\\_has\\_destructor](#) ([pmath\\_task\\_t](#) task, [pmath\\_callback\\_t](#) dtor)

- [pmath\\_task\\_t](#) [pmath\\_task\\_new](#) ([pmath\\_callback\\_t](#) *run*, [pmath\\_callback\\_t](#) *destroy*, void \**data*)
- void [pmath\\_task\\_wait](#) ([pmath\\_task\\_t](#) *task*)

### 12.20.1 Typedef Documentation

#### 12.20.1.1 typedef struct \_pmath\_task\_t\* [pmath\\_task\\_t](#)

#### Todo

document [pmath-util/concurrency/threadpool.h](#)

### 12.20.2 Function Documentation

#### 12.20.2.1 void\* [pmath\\_task\\_get\\_data](#) ([pmath\\_task\\_t](#) *task*)

#### 12.20.2.2 [pmath\\_bool\\_t](#) [pmath\\_task\\_has\\_destructor](#) ([pmath\\_task\\_t](#) *task*, [pmath\\_callback\\_t](#) *dtor*)

#### 12.20.2.3 [pmath\\_task\\_t](#) [pmath\\_task\\_new](#) ([pmath\\_callback\\_t](#) *run*, [pmath\\_callback\\_t](#) *destroy*, void \* *data*)

#### 12.20.2.4 [pmath\\_task\\_t](#) [pmath\\_task\\_ref](#) ([pmath\\_task\\_t](#) *task*)

#### 12.20.2.5 void [pmath\\_task\\_unref](#) ([pmath\\_task\\_t](#) *task*)

#### 12.20.2.6 void [pmath\\_task\\_wait](#) ([pmath\\_task\\_t](#) *task*)



## 12.21 pmath-util/concurrency/threads.h File Reference

### Typedefs

- typedef struct \_pmath\_thread\_t \* [pmath\\_thread\\_t](#)

### Functions

- [pmath\\_t pmath\\_thread\\_local\\_save](#) ([pmath\\_t](#) key, [pmath\\_t](#) value)  
*Store a thread/thread-local value.*
- [pmath\\_t pmath\\_thread\\_local\\_load](#) ([pmath\\_t](#) key)  
*Load a thread/thread-local value.*
- void [pmath\\_throw](#) ([pmath\\_t](#) exception)  
*Throw an exception.*
- [pmath\\_t pmath\\_catch](#) (void)  
*Catch any exception.*
- [pmath\\_bool\\_t pmath\\_aborting](#) (void)  
*Queries whether pMath was requested to abort the evaluation of the current thread.*
- void [pmath\\_abort\\_please](#) (void)  
*Requests pMath to abort the current evaluation.*
- [pmath\\_bool\\_t pmath\\_continue\\_after\\_abort](#) (void)  
*Requests pMath to stop aborting the current evaluation.*
- void [pmath\\_suspend\\_all\\_please](#) (void)  
*Suspend all other threads. This function does not really suspend threads immediately. Any other thread, that calls [pmath\\_aborting\(\)](#) (or [pmath\\_thread\\_aborting\(\)](#)), will block until we call [pmath\\_resume\\_all\(\)](#).*
- void [pmath\\_resume\\_all](#) (void)  
*Resume all other threads.*

## 12.22 pmath-util/debug.h File Reference

### Functions

- void [pmath\\_debug\\_print](#) (const char \*fmt,...)  
*Print out a simple debug message.*
- void [pmath\\_debug\\_print\\_object](#) (const char \*pre, [pmath\\_t](#) obj, const char \*post)

*Print a pMath object to the debug log.*

- void [pmath\\_debug\\_print\\_stack](#) (void)  
*Print the current pMath stack trace to the debug log.*

## 12.23 pmath-util/dlmalloc.h File Reference

## 12.24 pmath-util/emit-and-gather.h File Reference

## 12.25 pmath-util/evaluation.h File Reference

## 12.26 pmath-util/files.h File Reference

### Data Structures

- class [pmath\\_binary\\_file\\_api\\_t](#)  
*Access functions for binary files.*
- class [pmath\\_text\\_file\\_api\\_t](#)  
*Access functions for text files.*

### Enumerations

- enum [pmath\\_files\\_status\\_t](#) {  
    [PMATH\\_FILE\\_OK](#) = 0, [PMATH\\_FILE\\_INVALID](#) = 1,  
    [PMATH\\_FILE\\_ENDOFFILE](#) = 2, [PMATH\\_FILE\\_OTHERERROR](#) = 3,  
    [PMATH\\_FILE\\_RECURSIVE](#) = 4 }  
*The status of a file.*

### Functions

- [pmath\\_bool\\_t pmath\\_file\\_test](#) ([pmath\\_t](#) file, int properties)  
*Check whether a file supports a set of properties.*
- [pmath\\_files\\_status\\_t pmath\\_file\\_status](#) ([pmath\\_t](#) file)  
*Get the current status of a readable file.*
- [size\\_t pmath\\_file\\_read](#) ([pmath\\_t](#) file, void \*buffer, [size\\_t](#) buffer\_size, [pmath\\_bool\\_t](#) preserve\_internal\_buffer)  
*Read some bytes from a binary file.*
- [pmath\\_string\\_t pmath\\_file\\_readline](#) ([pmath\\_t](#) file)

*Read one line from a text file.*

- void `pmath_file_set_textbuffer` (`pmath_t` file, `pmath_string_t` buffer)  
*Set a file's internal text buffer.*
- size\_t `pmath_file_write` (`pmath_t` file, const void \*buffer, size\_t buffer\_size)  
*Write some bytes to a binary file.*
- `pmath_bool_t` `pmath_file_writetext` (`pmath_t` file, const uint16\_t \*str, int len)  
*Write to a text file.*
- void `pmath_file_flush` (`pmath_t` file)  
*Flush the output buffer of a writeable file.*
- `pmath_bool_t` `pmath_file_write_object` (`pmath_t` file, `pmath_t` obj, `pmath_write_options_t` options)  
*Write an object to a text file.*
- `pmath_bool_t` `pmath_file_set_binbuffer` (`pmath_t` file, size\_t size)  
*Set a binary file's buffer size.*
- void `pmath_file_manipulate` (`pmath_t` file, void(\*type)(void \*), void(\*callback)(void \*, void \*), void \*data)  
*Manipulate a file's internal representation.*
- `pmath_bool_t` `pmath_file_close` (`pmath_t` file)  
*Closes a file.*
- `pmath_symbol_t` `pmath_file_create_binary` (void \*extra, void(\*extra\_destructor)(void \*), `pmath_binary_file_api_t` \*api)  
*Create a binary file object.*
- `pmath_symbol_t` `pmath_file_create_text` (void \*extra, void(\*extra\_destructor)(void \*), `pmath_text_file_api_t` \*api)  
*Create a text file object.*
- `pmath_symbol_t` `pmath_file_create_text_from_binary` (`pmath_t` binfile, const char \*encoding)  
*Create a text file object operating on a binary file.*
- `pmath_symbol_t` `pmath_file_create_binary_buffer` (size\_t mincapacity)  
*Create a byte-stream file object.*
- size\_t `pmath_file_binary_buffer_size` (`pmath_t` binfile)  
*Get The number of readable bytes in a binary buffer.*

- void [pmath\\_file\\_binary\\_buffer\\_manipulate](#) ([pmath\\_t](#) binfile, void(\*callback)(uint8\_t \*readable, uint8\_t \*\*writable, const uint8\_t \*end, void \*closure), void \*closure)

*Manipulate the content of a binary buffer.*

## 12.27 pmath-util/hashtables.h File Reference

### Data Structures

- class [pmath\\_ht\\_class\\_t](#)  
*A hashtable interface.*

### Typedefs

- typedef struct \_pmath\_hashtable\_t \* [pmath\\_hashtable\\_t](#)
- typedef void(\* [pmath\\_ht\\_entry\\_callback\\_t](#) )(void \*entry, void \*data)  
*A callback function for hashtable entries.*
- typedef void \*(\* [pmath\\_ht\\_entry\\_copy\\_t](#) )(void \*entry)  
*An entry copy function.*
- typedef unsigned int(\* [pmath\\_ht\\_entry\\_hash\\_func\\_t](#) )(void \*entry)  
*An entry hash function.*
- typedef unsigned int(\* [pmath\\_ht\\_key\\_hash\\_func\\_t](#) )(void \*key)  
*A key hash function.*
- typedef [pmath\\_bool\\_t](#)(\* [pmath\\_ht\\_entry\\_equal\\_func\\_t](#) )(void \*entry1, void \*entry2)  
*An entry comparision function.*
- typedef [pmath\\_bool\\_t](#)(\* [pmath\\_ht\\_entry\\_equals\\_key\\_func\\_t](#) )(void \*entry, void \*key)  
*An entry to key comparision function.*

## 12.28 pmath-util/helpers.h File Reference

### Typedefs

- typedef [pmath\\_bool\\_t](#)(\* [pmath\\_stack\\_walker\\_t](#) )(pmath\_t head, void \*closure)  
*A stack walker function.*

## Functions

- [pmath\\_bool\\_t pmath\\_is\\_expr\\_of](#) ([pmath\\_t](#) obj, [pmath\\_symbol\\_t](#) head)  
*Check if an object is an expression with a specified head.*
- [pmath\\_bool\\_t pmath\\_is\\_expr\\_of\\_len](#) ([pmath\\_t](#) obj, [pmath\\_symbol\\_t](#) head, [size\\_t](#) length)  
*Check if an object is an expression with a specified head and length.*
- [pmath\\_t pmath\\_current\\_head](#) (void)  
*Get the currently evaluated function.*
- void [pmath\\_walk\\_stack](#) ([pmath\\_stack\\_walker\\_t](#) walker, void \*closure)  
*Walk up the current thread's and its parents' stack.*
- [pmath\\_t pmath\\_session\\_execute](#) ([pmath\\_t](#) input, [pmath\\_bool\\_t](#) \*aborted)  
*Execute an expression and change \$History and \$Line appropriately.*
- [pmath\\_t pmath\\_session\\_start](#) (void)  
*Saves some global state when an interactive dialog session starts.*
- void [pmath\\_session\\_end](#) ([pmath\\_t](#) old\_state)  
*Restore some global state when an interactive dialog session ends.*

## 12.29 pmath-util/line-writer.h File Reference

## 12.30 pmath-util/memory.h File Reference

## Functions

- void \* [pmath\\_mem\\_alloc](#) ([size\\_t](#) size)  
*Allocate some amount of memory.*
- void \* [pmath\\_mem\\_realloc](#) (void \*p, [size\\_t](#) new\_size)  
*Change the size of a memory-chunk.*
- void \* [pmath\\_mem\\_realloc\\_no\\_failfree](#) (void \*p, [size\\_t](#) new\_size)  
*Change the size of a memory-chunk.*
- void [pmath\\_mem\\_free](#) (void \*p)  
*Free a memory-chunk.*
- void [pmath\\_mem\\_usage](#) ([size\\_t](#) \*current, [size\\_t](#) \*max)  
*Get memory usage information.*

## 12.31 pmath-util/messages.h File Reference

### Functions

- void [pmath\\_message](#) ([pmath\\_symbol\\_t](#) symbol, const char \*tag, size\_t argcount,...)  
*Print a message with pMath object arguments.*
- void [pmath\\_message\\_argxxx](#) (size\_t given, size\_t min, size\_t max)  
*Generate a General::arg\* message (invalid argument count).*
- [pmath\\_string\\_t](#) [pmath\\_message\\_find\\_text](#) ([pmath\\_t](#) name)  
*Find a message's text.*
- void [pmath\\_message\\_syntax\\_error](#) ([pmath\\_string\\_t](#) code, int position, [pmath\\_string\\_t](#) filename, int lines\_before\_code)  
*Print a syntax warning or error message.*

## 12.32 pmath-util/mixed-file.h File Reference

### Functions

- void [pmath\\_file\\_create\\_mixed\\_buffer](#) (const char \*encoding, [pmath\\_symbol\\_t](#) \*out\_textfile, [pmath\\_symbol\\_t](#) \*out\_binfile)  
*Creates a mixed binary/text file double ended queue.*

## 12.33 pmath-util/serialize.h File Reference

### Enumerations

- enum [pmath\\_serialize\\_error\\_t](#) {  
    [PMATH\\_SERIALIZE\\_OK](#) = 0, [PMATH\\_SERIALIZE\\_NO\\_MEMORY](#) = 1,  
    [PMATH\\_SERIALIZE\\_BAD\\_OBJECT](#) = 2, [PMATH\\_SERIALIZE\\_EOF](#) = 3,  
    [PMATH\\_SERIALIZE\\_BAD\\_BYTE](#) = 4, [PMATH\\_SERIALIZE\\_BAD\\_REF](#) = 5  
}  
*(De-)Serialization error codes.*

### Functions

- [pmath\\_serialize\\_error\\_t](#) [pmath\\_serialize](#) ([pmath\\_t](#) file, [pmath\\_t](#) object)  
*Write an object to a binary file.*

- [pmath\\_t pmath\\_deserialize](#) ([pmath\\_t file](#), [pmath\\_serialize\\_error\\_t \\*error](#))

*Write an object to a binary file.*

### 12.33.1 Enumeration Type Documentation

#### 12.33.1.1 enum pmath\_serialize\_error\_t

(De-)Serialization error codes.

##### Enumerator:

***PMATH\_SERIALIZE\_OK*** No error occurred.

***PMATH\_SERIALIZE\_NO\_MEMORY***

***PMATH\_SERIALIZE\_BAD\_OBJECT*** The object cannot be serialized (e.g. custom objects).

***PMATH\_SERIALIZE\_EOF*** Unexpected end of file.

***PMATH\_SERIALIZE\_BAD\_BYTE*** Unexpected byte.

***PMATH\_SERIALIZE\_BAD\_REF*** Unknown back reference.

### 12.33.2 Function Documentation

#### 12.33.2.1 [pmath\\_t pmath\\_deserialize](#) ([pmath\\_t file](#), [pmath\\_serialize\\_error\\_t \\*error](#))

Write an object to a binary file.

##### Parameters:

***file*** A [file object](#). It wont be freed.

***error*** Where to put the error code (optional).

##### Returns:

The deserialized object.

#### 12.33.2.2 [pmath\\_serialize\\_error\\_t pmath\\_serialize](#) ([pmath\\_t file](#), [pmath\\_t object](#))

Write an object to a binary file.

**Parameters:**

- file* A [file object](#). It wont be freed.  
*object* A pMath object. It will be freed.

**Returns:**

An error code.

## 12.34 pmath-util/stacks.h File Reference

**Typedefs**

- typedef struct \_pmath\_stack\_t \* [pmath\\_stack\\_t](#)

**Functions**

- [pmath\\_stack\\_t](#) [pmath\\_stack\\_new](#) (void)  
*Create an empty stack.*
- void [pmath\\_stack\\_free](#) ([pmath\\_stack\\_t](#) stack)  
*Destroy a stack.*
- void [pmath\\_stack\\_push](#) ([pmath\\_stack\\_t](#) stack, void \*item)  
*Push an item onto a stack.*
- void \* [pmath\\_stack\\_pop](#) ([pmath\\_stack\\_t](#) stack)  
*Pop an item from a stack.*

## 12.35 pmath-util/strtod.h File Reference

**Functions**

- double [pmath\\_strtod](#) (const char \*str, const char \*\*end)  
*locale-neutral strtod*

### 12.35.1 Function Documentation

#### 12.35.1.1 double pmath\_strtod (const char \* str, const char \*\* end)

locale-neutral strtod



## 12.36 pmath-util/version.h File Reference

### Functions

- void [pmath\\_version\\_datetime](#) (int \*year, int \*month, int \*day, int \*hour, int \*minute, int \*second)  
*Get the date and time when pMath was compiled.*
- double [pmath\\_version\\_number](#) (void)  
*Get version number (major + minor/100).*
- long [pmath\\_version\\_number\\_part](#) (int index)  
*Get version number part.*

## 12.37 version.h File Reference

### Defines

- #define [PMATH\\_VERSION\\_RC\\_FILEVERSION](#) 0,1,0,0
- #define [PMATH\\_VERSION\\_RC\\_FILEVERSION\\_STRING](#) "0, 1, 0, 0\0"

### Variables

- static const char [PMATH\\_VERSION\\_DATE](#) [] = "18"
- static const char [PMATH\\_VERSION\\_MONTH](#) [] = "09"
- static const char [PMATH\\_VERSION\\_YEAR](#) [] = "2010"
- static const char [PMATH\\_VERSION\\_UBUNTU\\_VERSION\\_STYLE](#) [] = "10.09"
- static const char [PMATH\\_VERSION\\_STATUS](#) [] = "Alpha"
- static const char [PMATH\\_VERSION\\_STATUS\\_SHORT](#) [] = "a"
- static const long [PMATH\\_VERSION\\_MAJOR](#) = 0
- static const long [PMATH\\_VERSION\\_MINOR](#) = 1
- static const long [PMATH\\_VERSION\\_BUILD](#) = 0
- static const long [PMATH\\_VERSION\\_REVISION](#) = 0
- static const long [PMATH\\_VERSION\\_BUILDS\\_COUNT](#) = 0
- static const char [PMATH\\_VERSION\\_FULLVERSION\\_STRING](#) [] = "0.1.0.0"
- static const char [PMATH\\_VERSION\\_SVN\\_REVISION](#) [] = "0"
- static const char [PMATH\\_VERSION\\_SVN\\_DATE](#) [] = "unknown date"
- static const long [PMATH\\_VERSION\\_BUILD\\_HISTORY](#) = 0

### 12.37.1 Define Documentation

#### 12.37.1.1 #define PMATH\_VERSION\_RC\_FILEVERSION 0,1,0,0

**12.37.1.2** `#define PMATH_VERSION_RC_FILEVERSION_STRING "0, 1, 0, 0\0"`

## 12.37.2 Variable Documentation

**12.37.2.1** `const long PMATH_VERSION_BUILD = 0` [static]

**12.37.2.2** `const long PMATH_VERSION_BUILD_HISTORY = 0` [static]

**12.37.2.3** `const long PMATH_VERSION_BUILDS_COUNT = 0` [static]

**12.37.2.4** `const char PMATH_VERSION_DATE[] = "18"` [static]

**12.37.2.5** `const char PMATH_VERSION_FULLVERSION_STRING[] = "0.1.0.0"` [static]

**12.37.2.6** `const long PMATH_VERSION_MAJOR = 0` [static]

**12.37.2.7** `const long PMATH_VERSION_MINOR = 1` [static]

**12.37.2.8** `const char PMATH_VERSION_MONTH[] = "09"` [static]

**12.37.2.9** `const long PMATH_VERSION_REVISION = 0` [static]

**12.37.2.10** `const char PMATH_VERSION_STATUS[] = "Alpha"` `[static]`

**12.37.2.11** `const char PMATH_VERSION_STATUS_SHORT[] = "a"`  
`[static]`

**12.37.2.12** `const char PMATH_VERSION_SVN_DATE[] = "unknown date"`  
`[static]`

**12.37.2.13** `const char PMATH_VERSION_SVN_REVISION[] = "0"`  
`[static]`

**12.37.2.14** `const char PMATH_VERSION_UBUNTU_VERSION_STYLE[] = "10.09"` `[static]`

**12.37.2.15** `const char PMATH_VERSION_YEAR[] = "2010"` `[static]`

## 12.38 pmath.h File Reference

### Functions

- [pmath\\_bool\\_t pmath\\_init](#) (void)  
*Initialize the pMath CAS library.*
- void [pmath\\_done](#) (void)  
*Free all resources used by the pMath CAS library and unload all modules.*

## Index

- ~Expr
  - pmath::Expr, 158
- ~Gather
  - pmath::Gather, 168
- ~UserStream
  - pmath::UserStream, 211
- \_obj
  - pmath::Expr, 165
- Abs
  - pmath, 144
- ArcCos
  - pmath, 144
- ArcSin
  - pmath, 144
- ArcTan
  - pmath, 144
- Arg
  - pmath, 144
- as\_bits
  - pmath\_t, 191
- as\_chars
  - pmath\_t, 191
- as\_double
  - pmath\_t, 191
- as\_int32
  - pmath\_t, 191
- Atomic Operations, 87
- atomic.h
  - PMATH\_ATOMIC\_FASTLOOP\_COUNT, 236
  - pmath\_atomic\_loop\_nop, 236
  - pmath\_atomic\_loop\_yield, 236
  - PMATH\_ATOMIC\_STATIC\_INIT, 237
- atomic\_ops
  - pmath\_atomic\_barrier, 90
  - pmath\_atomic\_compare\_and\_set, 90
  - pmath\_atomic\_compare\_and\_set\_2, 90
  - PMATH\_ATOMIC\_FASTLOOP\_COUNT, 89
  - pmath\_atomic\_fetch\_add, 91
  - pmath\_atomic\_fetch\_compare\_and\_set, 91
  - pmath\_atomic\_fetch\_set, 92
  - pmath\_atomic\_have\_cas2, 92
  - pmath\_atomic\_lock, 92
  - pmath\_atomic\_loop\_nop, 93
  - pmath\_atomic\_loop\_yield, 93
  - pmath\_atomic\_unlock, 93
  - PMATH\_DECLARE\_ALIGNED, 90
- BinaryFile
  - pmath::BinaryFile, 152
- buffer
  - pmath::String, 203
- C++ Binding, 66
- Call
  - pmath, 145, 146
- Ceiling
  - pmath, 146
- close
  - pmath::File, 166
- compare
  - pmath::Expr, 159
- Complex
  - pmath, 146
- convert\_to\_file
  - pmath::BinaryUserStream, 154
  - pmath::TextUserStream, 209
- convert\_to\_file\_readonly
  - pmath::BinaryUserStream, 154
  - pmath::TextUserStream, 209
- convert\_to\_file\_writeonly
  - pmath::BinaryUserStream, 154
  - pmath::TextUserStream, 209
- Cos
  - pmath, 146
- create\_buffer
  - pmath::BinaryFile, 153
- create\_compressor
  - pmath::WriteableBinaryFile, 213
- create\_from\_binary
  - pmath::ReadableTextFile, 200
  - pmath::TextFile, 207
  - pmath::WriteableTextFile, 215
- create\_uncompressor
  - pmath::ReadableBinaryFile, 198
- custom
  - pmath\_custom\_get\_data, 9
  - pmath\_custom\_has\_destructor, 10

- pmath\_custom\_new, [10](#)
  - pmath\_custom\_t, [9](#)
- Custom Objects, [8](#)
- debug
  - pmath\_debug\_print, [106](#)
  - pmath\_debug\_print\_object, [107](#)
  - pmath\_debug\_print\_stack, [107](#)
- Debugging, [106](#)
- dereference
  - pmath::UserStream, [211](#)
- deserialize
  - pmath::Expr, [159](#)
- destructor\_function
  - pmath::UserStream, [211](#)
- Divide
  - pmath, [146](#)
- emit
  - pmath::Gather, [168](#)
- end
  - pmath::Gather, [169](#)
- ended
  - pmath::Gather, [169](#)
- entry\_destructor
  - pmath\_ht\_class\_t, [177](#)
- entry\_equals\_key
  - pmath\_ht\_class\_t, [177](#)
- entry\_hash
  - pmath\_ht\_class\_t, [177](#)
- entry\_keys\_equal
  - pmath\_ht\_class\_t, [177](#)
- equals
  - pmath::String, [203](#)
- Evaluate
  - pmath, [146](#)
- Exp
  - pmath, [146](#)
- Expr
  - pmath::Expr, [158](#)
- expr\_length
  - pmath::Expr, [159](#)
- Expressions, [10](#)
- expressions
  - pmath\_expr\_append, [12](#)
  - pmath\_expr\_extract\_item, [12](#)
  - pmath\_expr\_flatten, [13](#)
  - pmath\_expr\_get\_item, [13](#)
  - pmath\_expr\_get\_item\_range, [13](#)
  - pmath\_expr\_length, [14](#)
  - pmath\_expr\_new, [14](#)
  - pmath\_expr\_new\_extended, [14](#)
  - pmath\_expr\_remove\_all, [15](#)
  - pmath\_expr\_resize, [15](#)
  - pmath\_expr\_set\_item, [16](#)
  - pmath\_expr\_sort, [16](#)
  - pmath\_expr\_t, [12](#)
- FALSE
  - general\_types, [87](#)
- File
  - pmath::File, [166](#)
- File API, [107](#)
- file\_api
  - PMATH\_FILE\_ENDOFFILE, [110](#)
  - PMATH\_FILE\_INVALID, [110](#)
  - PMATH\_FILE\_OK, [110](#)
  - PMATH\_FILE\_OTHERERROR, [110](#)
  - PMATH\_FILE\_RECURSIVE, [110](#)
- file\_api
  - pmath\_file\_binary\_buffer\_-  
manipulate, [110](#)
  - pmath\_file\_binary\_buffer\_size, [110](#)
  - pmath\_file\_close, [110](#)
  - pmath\_file\_create\_binary, [111](#)
  - pmath\_file\_create\_binary\_buffer,  
[111](#)
  - pmath\_file\_create\_compressor, [111](#)
  - pmath\_file\_create\_mixed\_buffer,  
[112](#)
  - pmath\_file\_create\_text, [112](#)
  - pmath\_file\_create\_text\_from\_-  
binary, [113](#)
  - pmath\_file\_create\_uncompressor,  
[113](#)
  - pmath\_file\_flush, [113](#)
  - pmath\_file\_manipulate, [113](#)
  - pmath\_file\_read, [114](#)
  - pmath\_file\_readline, [114](#)
  - pmath\_file\_set\_binbuffer, [115](#)
  - pmath\_file\_set\_textbuffer, [115](#)
  - pmath\_file\_status, [115](#)
  - pmath\_file\_test, [115](#)
  - pmath\_file\_write, [116](#)
  - pmath\_file\_write\_object, [116](#)
  - pmath\_file\_writetext, [117](#)
  - pmath\_files\_status\_t, [109](#)
- file\_wraps
  - pmath::UserStream, [211](#)

- Floor
  - pmath, 146
- flush
  - pmath::BinaryUserStream, 154
  - pmath::File, 166
  - pmath::TextUserStream, 209
- flush\_function
  - pmath\_binary\_file\_api\_t, 170
  - pmath\_text\_file\_api\_t, 192
- FromChar
  - pmath::String, 203
- FromUcs2
  - pmath::String, 203
- FromUtf8
  - pmath::String, 203
- Front-ends, 138
- frontend
  - pmath\_continue\_after\_abort, 139
  - pmath\_done, 139
  - pmath\_init, 140
  - pmath\_session\_end, 140
  - pmath\_session\_execute, 140
  - pmath\_session\_start, 141
- Gather
  - pmath::Gather, 168
- General Purpose Types, 86
- general\_types
  - FALSE, 87
  - pmath\_bool\_t, 87
  - pmath\_callback\_t, 87
  - PMATH\_INVALID\_PTR, 87
  - TRUE, 87
- get
  - pmath::Expr, 159
- get\_as\_string
  - pmath::String, 203
- get\_buffer\_size
  - pmath::BinaryFile, 153
- has\_capabilities
  - pmath::File, 166
- hash
  - pmath::Expr, 159
- Hashtables, 117
- hashtables
  - pmath\_hashtable\_t, 119
  - pmath\_ht\_capacity, 121
  - pmath\_ht\_clear, 121
  - pmath\_ht\_copy, 121
  - pmath\_ht\_count, 122
  - pmath\_ht\_create, 122
  - pmath\_ht\_destroy, 122
  - pmath\_ht\_entry, 122
  - pmath\_ht\_entry\_callback\_t, 119
  - pmath\_ht\_entry\_copy\_t, 119
  - pmath\_ht\_entry\_equal\_func\_t, 119
  - pmath\_ht\_entry\_equals\_key\_func\_t, 120
  - pmath\_ht\_entry\_hash\_func\_t, 120
  - pmath\_ht\_insert, 123
  - pmath\_ht\_key\_hash\_func\_t, 120
  - pmath\_ht\_remove, 123
  - pmath\_ht\_search, 123
- helpers
  - pmath\_build\_value, 125
  - pmath\_build\_value\_v, 127
  - pmath\_current\_head, 127
  - pmath\_emit, 127
  - pmath\_gather\_begin, 128
  - pmath\_gather\_end, 128
  - pmath\_is\_expr\_of, 129
  - pmath\_is\_expr\_of\_len, 129
  - pmath\_option\_value, 129
  - pmath\_options\_extract, 130
  - pmath\_stack\_walker\_t, 125
  - pmath\_walk\_stack, 130
- Im
  - pmath, 146
- Imaginary
  - pmath, 147
- insert
  - pmath::String, 203, 204
- Inv
  - pmath, 147
- is\_binary
  - pmath::File, 166
- is\_custom
  - pmath::Expr, 159
- is\_double
  - pmath::Expr, 159
- is\_evaluated
  - pmath::Expr, 159
- is\_expr
  - pmath::Expr, 160
- is\_file
  - pmath::File, 167
- is\_float
  - pmath::Expr, 160

- is\_int32
  - pmath::Expr, 160
- is\_integer
  - pmath::Expr, 160
- is\_magic
  - pmath::Expr, 160
- is\_mpfloaf
  - pmath::Expr, 160
- is\_null
  - pmath::Expr, 160
- is\_number
  - pmath::Expr, 160
- is\_pointer
  - pmath::Expr, 160
- is\_pointer\_of
  - pmath::Expr, 160
- is\_quotient
  - pmath::Expr, 160
- is\_rational
  - pmath::Expr, 161
- is\_readable
  - pmath::File, 167
- is\_string
  - pmath::Expr, 161
- is\_symbol
  - pmath::Expr, 161
- is\_text
  - pmath::File, 167
- is\_valid
  - pmath::Expr, 161
- is\_writeable
  - pmath::File, 167
- key\_hash
  - pmath\_ht\_class\_t, 177
- length
  - pmath::String, 204
- List
  - pmath, 147, 148
- Log
  - pmath, 148
- MakeList
  - pmath, 148
- manipulate
  - pmath::UserStream, 211, 212
- memory
  - pmath\_mem\_alloc, 131
  - pmath\_mem\_free, 131
  - pmath\_mem\_realloc, 132
  - pmath\_mem\_realloc\_no\_failfree, 132
  - pmath\_mem\_usage, 132
- Memory Management, 130
- Messages, 133
- messages
  - pmath\_message, 134
  - pmath\_message\_argxxx, 134
  - pmath\_message\_find\_text, 135
  - pmath\_message\_syntax\_error, 135
- Minus
  - pmath, 148
- Mod
  - pmath, 148
- Multithreading with pMath, 99
- Number
  - pmath, 148
- Numbers, 16
- numbers
  - PMATH\_PREC\_CTRL\_AUTO, 26
  - PMATH\_PREC\_CTRL\_GIVEN\_-ACC, 27
  - PMATH\_PREC\_CTRL\_GIVEN\_-PREC, 26
  - PMATH\_PREC\_CTRL\_-MACHINE\_PREC, 26
  - pmath\_accuracy, 27
  - pmath\_approximate, 27
  - PMATH\_AUTO\_PRECISION, 21
  - pmath\_float\_new\_str, 27
  - pmath\_float\_t, 26
  - pmath\_integer\_fits\_si32, 21
  - pmath\_integer\_fits\_si64, 28
  - pmath\_integer\_fits\_siptr, 21, 22
  - pmath\_integer\_fits\_ui32, 29
  - pmath\_integer\_fits\_ui64, 29
  - pmath\_integer\_fits\_uiptr, 22
  - pmath\_integer\_get\_si32, 29
  - pmath\_integer\_get\_si64, 30
  - pmath\_integer\_get\_siptr, 22, 23
  - pmath\_integer\_get\_ui32, 30
  - pmath\_integer\_get\_ui64, 30
  - pmath\_integer\_get\_uiptr, 23
  - pmath\_integer\_new\_data, 31
  - pmath\_integer\_new\_si32, 24
  - pmath\_integer\_new\_si64, 31
  - pmath\_integer\_new\_siptr, 24, 25
  - pmath\_integer\_new\_slong, 32

- pmath\_integer\_new\_str, 32
- pmath\_integer\_new\_ui32, 32
- pmath\_integer\_new\_ui64, 33
- pmath\_integer\_new\_uiptr, 25
- pmath\_integer\_new\_ulong, 33
- pmath\_integer\_t, 26
- PMATH\_MACHINE\_PRECISION, 25
- pmath\_mpf\_float\_t, 26
- pmath\_mpint\_t, 26
- pmath\_number\_get\_d, 33
- pmath\_number\_neg, 34
- pmath\_number\_sign, 34
- pmath\_number\_t, 26
- pmath\_precision, 34
- pmath\_precision\_control\_t, 26
- pmath\_quotient\_t, 26
- pmath\_rational\_denominator, 35
- pmath\_rational\_new, 35
- pmath\_rational\_numerator, 35
- pmath\_rational\_t, 26
- pmath\_set\_accuracy, 36
- pmath\_set\_precision, 36
- Object Utility Functions, 124
- objects
  - PMATH\_TYPE\_BIGSTRING, 44
  - PMATH\_TYPE\_CUSTOM, 44
  - PMATH\_TYPE\_EXPRESSION, 44
  - PMATH\_TYPE\_EXPRESSION\_-GENERAL, 44
  - PMATH\_TYPE\_EXPRESSION\_-GENERAL\_PART, 44
  - PMATH\_TYPE\_MP\_FLOAT, 44
  - PMATH\_TYPE\_MP\_INT, 44
  - PMATH\_TYPE\_QUOTIENT, 44
  - PMATH\_TYPE\_SHIFT\_-BIGSTRING, 44
  - PMATH\_TYPE\_SHIFT\_COUNT, 44
  - PMATH\_TYPE\_SHIFT\_CUSTOM, 44
  - PMATH\_TYPE\_SHIFT\_-EXPRESSION\_GENERAL, 44
  - PMATH\_TYPE\_SHIFT\_-EXPRESSION\_GENERAL\_-PART, 44
  - PMATH\_TYPE\_SHIFT\_MP\_-FLOAT, 44
  - PMATH\_TYPE\_SHIFT\_MP\_INT, 44
  - PMATH\_TYPE\_SHIFT\_-QUOTIENT, 44
  - PMATH\_TYPE\_SHIFT\_-RESERVED\_1, 44
  - PMATH\_TYPE\_SHIFT\_SYMBOL, 44
  - PMATH\_TYPE\_SYMBOL, 44
  - PMATH\_WRITE\_OPTIONS\_-FULLEXPR, 45
  - PMATH\_WRITE\_OPTIONS\_-FULLNAME, 45
  - PMATH\_WRITE\_OPTIONS\_-FULLSTR, 45
  - PMATH\_WRITE\_OPTIONS\_-INPUTEXPR, 45
  - PMATH\_ABORT\_EXCEPTION, 40
  - pmath\_compare, 45
  - pmath\_compare\_func\_t, 42
  - pmath\_equal\_func\_t, 42
  - PMATH\_FROM\_INT32, 40
  - PMATH\_FROM\_PTR, 45
  - PMATH\_FROM\_TAG, 45
  - pmath\_func\_t, 42
  - pmath\_hash, 45
  - pmath\_hash\_func\_t, 42
  - pmath\_is\_evaluated, 46
  - PMATH\_NULL, 47
  - pmath\_param\_proc\_t, 42
  - pmath\_proc\_t, 42
  - PMATH\_STATIC\_NULL, 40
  - PMATH\_STATIC\_UNDEFINED, 40
  - PMATH\_TAG\_INT32, 40
  - PMATH\_TAG\_INVALID, 40
  - PMATH\_TAG\_MAGIC, 40
  - PMATH\_TAG\_STR0, 41
  - PMATH\_TAG\_STR1, 41
  - PMATH\_TAG\_STR2, 41
  - PMATH\_TAGMASK\_BITCOUNT, 41
  - PMATH\_TAGMASK\_-NONDOUBLE, 41
  - PMATH\_TAGMASK\_POINTER, 41
  - PMATH\_THREAD\_KEY\_-PARSERARGUMENTS, 41



- PMATH\_THREAD\_KEY\_-
  - PARSESYMBOLS, [41](#)
- pmath\_type\_t, [43](#)
- PMATH\_UNDEFINED, [47](#)
- pmath\_write, [46](#)
- pmath\_write\_ex, [46](#)
- pmath\_write\_options\_t, [43](#)
- pmath\_write\_with\_pagewidth, [47](#)
- Objects - the Base of pMath, [36](#)
- objects-inline.h
  - PMATH\_AS\_DOUBLE, [223](#)
  - PMATH\_AS\_INT32, [220](#)
  - PMATH\_AS\_PTR, [223](#)
  - PMATH\_AS\_TAG, [220](#)
  - PMATH\_FROM\_DOUBLE, [223](#)
  - pmath\_is\_bigstr, [220](#)
  - pmath\_is\_custom, [220](#)
  - pmath\_is\_double, [220](#)
  - pmath\_is\_evaluable, [223](#)
  - pmath\_is\_expr, [220](#)
  - pmath\_is\_float, [221](#)
  - pmath\_is\_int32, [221](#)
  - pmath\_is\_integer, [221](#)
  - pmath\_is\_magic, [221](#)
  - pmath\_is\_ministr, [221](#)
  - pmath\_is\_mpffloat, [221](#)
  - pmath\_is\_mfpint, [221](#)
  - pmath\_is\_null, [221](#)
  - pmath\_is\_number, [221](#)
  - pmath\_is\_pointer, [222](#)
  - pmath\_is\_pointer\_of, [223](#)
  - pmath\_is\_quotient, [222](#)
  - pmath\_is\_rational, [222](#)
  - pmath\_is\_str0, [222](#)
  - pmath\_is\_str1, [222](#)
  - pmath\_is\_str2, [222](#)
  - pmath\_is\_string, [222](#)
  - pmath\_is\_symbol, [222](#)
  - pmath\_refcount, [223](#)
  - pmath\_same, [222](#)
- operator<
  - pmath::Expr, [161](#)
- operator<=
  - pmath::Expr, [161](#)
- operator>
  - pmath::Expr, [162](#)
- operator>=
  - pmath::Expr, [162](#)
- operator+
  - pmath::String, [204](#)
- operator+=
  - pmath::String, [204](#), [205](#)
- operator=
  - pmath::Expr, [162](#)
  - pmath::String, [205](#)
- operator==
  - pmath::Expr, [162](#)
- options
  - pmath\_write\_ex\_t, [196](#)
- Parse
  - pmath, [148](#), [149](#)
- ParseArgs
  - pmath, [149](#)
- parser
  - PMATH\_PREC\_ADD, [76](#)
  - PMATH\_PREC\_ALT, [76](#)
  - PMATH\_PREC\_AND, [76](#)
  - PMATH\_PREC\_ANY, [76](#)
  - PMATH\_PREC\_APL, [76](#)
  - PMATH\_PREC\_ARROW, [76](#)
  - PMATH\_PREC\_ASS, [76](#)
  - PMATH\_PREC\_CALL, [77](#)
  - PMATH\_PREC\_CIRCADD, [76](#)
  - PMATH\_PREC\_CIRCMUL, [76](#)
  - PMATH\_PREC\_COND, [76](#)
  - PMATH\_PREC\_CROSS, [76](#)
  - PMATH\_PREC\_DIFF, [77](#)
  - PMATH\_PREC\_DIV, [76](#)
  - PMATH\_PREC\_EVAL, [76](#)
  - PMATH\_PREC\_FAC, [76](#)
  - PMATH\_PREC\_FUNC, [76](#)
  - PMATH\_PREC\_INC, [77](#)
  - PMATH\_PREC\_ISECT, [76](#)
  - PMATH\_PREC\_LAZY, [76](#)
  - PMATH\_PREC\_MAP, [76](#)
  - PMATH\_PREC\_MIDDOT, [76](#)
  - PMATH\_PREC\_MODY, [76](#)
  - PMATH\_PREC\_MUL, [76](#)
  - PMATH\_PREC\_MUL2, [76](#)
  - PMATH\_PREC\_OR, [76](#)
  - PMATH\_PREC\_PLUMI, [76](#)
  - PMATH\_PREC\_POW, [76](#)
  - PMATH\_PREC\_PRIM, [77](#)
  - PMATH\_PREC\_RANGE, [76](#)
  - PMATH\_PREC\_REL, [76](#)
  - PMATH\_PREC\_REPEAT, [77](#)
  - PMATH\_PREC\_REPL, [76](#)
  - PMATH\_PREC\_RULE, [76](#)
  - PMATH\_PREC\_SEQ, [76](#)

- PMATH\_PREC\_STR, [76](#)
- PMATH\_PREC\_TEST, [77](#)
- PMATH\_PREC\_UNION, [76](#)
- PMATH\_PREC\_XOR, [76](#)
- PMATH\_TOK\_ASSIGNTAG, [77](#)
- PMATH\_TOK\_BINARY\_LEFT, [77](#)
- PMATH\_TOK\_BINARY\_LEFT\_-  
  AUTOARG, [77](#)
- PMATH\_TOK\_BINARY\_LEFT\_-  
  OR\_PREFIX, [77](#)
- PMATH\_TOK\_BINARY\_RIGHT,  
  [77](#)
- PMATH\_TOK\_CALL, [77](#)
- PMATH\_TOK\_COLON, [77](#)
- PMATH\_TOK\_COMMENTEND,  
  [78](#)
- PMATH\_TOK\_DIGIT, [77](#)
- PMATH\_TOK\_INTEGRAL, [78](#)
- PMATH\_TOK\_LEFT, [77](#)
- PMATH\_TOK\_LEFTCALL, [77](#)
- PMATH\_TOK\_NAME, [77](#)
- PMATH\_TOK\_NAME2, [77](#)
- PMATH\_TOK\_NARY, [77](#)
- PMATH\_TOK\_NARY\_AUTOARG,  
  [77](#)
- PMATH\_TOK\_NARY\_OR\_-  
  PREFIX, [77](#)
- PMATH\_TOK\_NONE, [77](#)
- PMATH\_TOK\_PLUSPLUS, [77](#)
- PMATH\_TOK\_POSTFIX, [77](#)
- PMATH\_TOK\_POSTFIX\_OR\_-  
  PREFIX, [77](#)
- PMATH\_TOK\_PREFIX, [77](#)
- PMATH\_TOK\_PRETEXT, [77](#)
- PMATH\_TOK\_QUESTION, [78](#)
- PMATH\_TOK\_RIGHT, [77](#)
- PMATH\_TOK\_SLOT, [78](#)
- PMATH\_TOK\_SPACE, [77](#)
- PMATH\_TOK\_STRING, [77](#)
- PMATH\_TOK\_TILDES, [77](#)
- pmath\_boxes\_from\_spans, [78](#)
- PMATH\_CHAR\_-  
  ALIASDELIMITER, [73](#)
- PMATH\_CHAR\_-  
  ALIASINDICATOR, [73](#)
- PMATH\_CHAR\_ASSIGN, [73](#)
- PMATH\_CHAR\_-  
  ASSIGNDELAYED, [73](#)
- PMATH\_CHAR\_BOX, [73](#)
- PMATH\_CHAR\_INTEGRAL\_D,  
  [73](#)
- PMATH\_CHAR\_-  
  INVISIBLECALL, [74](#)
- pmath\_char\_is\_36digit, [78](#)
- pmath\_char\_is\_basedigit, [78](#)
- pmath\_char\_is\_digit, [78](#)
- pmath\_char\_is\_hexdigit, [79](#)
- pmath\_char\_is\_integral, [79](#)
- pmath\_char\_is\_left, [79](#)
- pmath\_char\_is\_name, [79](#)
- pmath\_char\_is\_right, [79](#)
- PMATH\_CHAR\_LEFT\_BOX, [74](#)
- pmath\_char\_maybe\_bigop, [79](#)
- PMATH\_CHAR\_PIECEWISE, [74](#)
- PMATH\_CHAR\_PLACEHOLDER,  
  [74](#)
- PMATH\_CHAR\_RIGHT\_BOX, [74](#)
- PMATH\_CHAR\_RULE, [74](#)
- PMATH\_CHAR\_RULEDELAYED,  
  [74](#)
- PMATH\_CHAR\_VECTOR, [74](#)
- pmath\_parse\_string, [79](#)
- pmath\_parse\_string\_args, [80](#)
- pmath\_right\_fence, [80](#)
- PMATH\_RUN, [75](#)
- PMATH\_RUN\_ARGS, [75](#)
- pmath\_span\_array\_free, [81](#)
- pmath\_span\_array\_is\_operand\_start,  
  [81](#)
- pmath\_span\_array\_is\_token\_end, [81](#)
- pmath\_span\_array\_length, [81](#)
- pmath\_span\_array\_t, [75](#)
- pmath\_span\_at, [82](#)
- pmath\_span\_end, [82](#)
- pmath\_span\_next, [82](#)
- pmath\_span\_t, [75](#)
- pmath\_spans\_from\_boxes, [83](#)
- pmath\_spans\_from\_string, [83](#)
- pmath\_string\_expand\_boxes, [84](#)
- pmath\_token\_analyse, [84](#)
- pmath\_token\_maybe\_bigop, [85](#)
- pmath\_token\_maybe\_first, [85](#)
- pmath\_token\_maybe\_rest, [85](#)
- pmath\_token\_prefix\_precedence, [85](#)
- pmath\_token\_t, [77](#)
- Parsing Code, [68](#)
- part
  - pmath::String, [205](#)
- Plus

- pmath, 149
- pmath, 141
  - Abs, 144
  - ArcCos, 144
  - ArcSin, 144
  - ArcTan, 144
  - Arg, 144
  - Call, 145, 146
  - Ceiling, 146
  - Complex, 146
  - Cos, 146
  - Divide, 146
  - Evaluate, 146
  - Exp, 146
  - Floor, 146
  - Im, 146
  - Imaginary, 147
  - Inv, 147
  - List, 147, 148
  - Log, 148
  - MakeList, 148
  - Minus, 148
  - Mod, 148
  - Number, 148
  - Parse, 148, 149
  - ParseArgs, 149
  - Plus, 149
  - Power, 149
  - Quotient, 150
  - Rational, 150
  - Re, 150
  - Ref, 150
  - Round, 150
  - Rule, 150
  - RuleDelayed, 150
  - Sign, 150
  - Sin, 150
  - Sqrt, 151
  - Symbol, 151
  - SymbolPi, 151
  - Tan, 151
  - Times, 151
- pmath-config.h, 215
  - PMATH\_API, 216
  - PMATH\_ATTRIBUTE\_-  
NONNULL, 216
  - PMATH\_ATTRIBUTE\_PURE, 216
  - PMATH\_ATTRIBUTE\_USE\_-  
RESULT, 216
  - PMATH\_BYTE\_ORDER, 216
  - PMATH\_CONCAT, 216
  - PMATH\_CONCAT\_, 216
  - PMATH\_DEPRECATED, 217
  - PMATH\_EXTERN\_C, 217
  - PMATH\_FORCE\_INLINE, 217
  - PMATH\_INLINE, 217
  - PMATH\_LIKELY, 217
  - PMATH\_MODULE, 217
  - PMATH\_NEED\_GNUC, 217
  - PMATH\_STATIC\_ASSERT, 217
  - PMATH\_UNLIKELY, 217
  - PMATH\_UNUSED, 217
- pmath-core/custom.h, 218
- pmath-core/expressions.h, 218
- pmath-core/numbers.h, 218
- pmath-core/objects-inline.h, 219
- pmath-core/objects.h, 223
- pmath-core/strings.h, 226
- pmath-core/symbols.h, 226
- pmath-cpp.h, 227
- pmath-language/charnames.h, 230
- pmath-language/scanner.h, 231
- pmath-language/tokens.h, 231
- pmath-types.h, 235
- pmath-util/approximate.h, 235
- pmath-util/compression.h, 236
- pmath-util/concurrency/atomic.h, 236
- pmath-util/concurrency/atomic/non-  
atomic.h, 237
- pmath-util/concurrency/threadlocks.h,  
238
- pmath-util/concurrency/threadmsg.h, 238
- pmath-util/concurrency/threadpool.h, 238
- pmath-util/concurrency/threads.h, 240
- pmath-util/debug.h, 240
- pmath-util/dlmalloc.h, 241
- pmath-util/emit-and-gather.h, 241
- pmath-util/evaluation.h, 241
- pmath-util/files.h, 241
- pmath-util/hashtables.h, 243
- pmath-util/helpers.h, 243
- pmath-util/line-writer.h, 244
- pmath-util/memory.h, 244
- pmath-util/messages.h, 245
- pmath-util/mixed-file.h, 245
- pmath-util/serialize.h, 245
- pmath-util/stacks.h, 247
- pmath-util/strtod.h, 247
- pmath-util/version.h, 248
- pmath.h, 250

- pmath::BinaryFile, 151
  - BinaryFile, 152
  - create\_buffer, 153
  - get\_buffer\_size, 153
  - set\_buffer\_size, 153
- pmath::BinaryUserStream, 153
  - convert\_to\_file, 154
  - convert\_to\_file\_readonly, 154
  - convert\_to\_file\_writeonly, 154
  - flush, 154
  - read, 154
  - status, 154
  - write, 155
- pmath::Expr, 155
  - ~Expr, 158
  - \_obj, 165
  - compare, 159
  - deserialize, 159
  - Expr, 158
  - expr\_length, 159
  - get, 159
  - hash, 159
  - is\_custom, 159
  - is\_double, 159
  - is\_evaluated, 159
  - is\_expr, 160
  - is\_float, 160
  - is\_int32, 160
  - is\_integer, 160
  - is\_magic, 160
  - is\_mpffloat, 160
  - is\_null, 160
  - is\_number, 160
  - is\_pointer, 160
  - is\_pointer\_of, 160
  - is\_quotient, 160
  - is\_rational, 161
  - is\_string, 161
  - is\_symbol, 161
  - is\_valid, 161
  - operator<, 161
  - operator<=, 161
  - operator>, 162
  - operator>=, 162
  - operator=, 162
  - operator==, 162
  - release, 163
  - serialize, 163
  - set, 163
  - to\_double, 163
  - to\_string, 164
  - write\_to\_file, 164
  - write\_to\_string, 164
- pmath::File, 165
  - close, 166
  - File, 166
  - flush, 166
  - has\_capabilities, 166
  - is\_binary, 166
  - is\_file, 167
  - is\_readable, 167
  - is\_text, 167
  - is\_writeable, 167
  - status, 167
- pmath::Gather, 167
  - ~Gather, 168
  - emit, 168
  - end, 169
  - ended, 169
  - Gather, 168
- pmath::ReadableBinaryFile, 197
  - create\_uncompressor, 198
  - read, 198
  - ReadableBinaryFile, 198
- pmath::ReadableTextFile, 198
  - create\_from\_binary, 200
  - ReadableTextFile, 199
  - readline, 200
- pmath::String, 200
  - buffer, 203
  - equals, 203
  - FromChar, 203
  - FromUcs2, 203
  - FromUtf8, 203
  - get\_as\_string, 203
  - insert, 203, 204
  - length, 204
  - operator+, 204
  - operator+=, 204, 205
  - operator=, 205
  - part, 205
  - remove, 205
  - starts\_with, 205, 206
  - String, 202
  - trim, 206
- pmath::TextFile, 206
  - create\_from\_binary, 207
  - set\_buffer, 207
  - TextFile, 207
- pmath::TextUserStream, 208

- convert\_to\_file, 209
- convert\_to\_file\_readonly, 209
- convert\_to\_file\_writeonly, 209
- flush, 209
- readline, 209
- status, 209
- write, 209
- pmath::UserStream, 210
  - ~UserStream, 211
  - dereference, 211
  - destructor\_function, 211
  - file\_wraps, 211
  - manipulate, 211, 212
- pmath::WriteableBinaryFile, 212
  - create\_compressor, 213
  - write, 213
  - WriteableBinaryFile, 213
- pmath::WriteableTextFile, 214
  - create\_from\_binary, 215
  - write, 215
  - WriteableTextFile, 214, 215
- PMATH\_FILE\_ENDOFFILE
  - file\_api, 110
- PMATH\_FILE\_INVALID
  - file\_api, 110
- PMATH\_FILE\_OK
  - file\_api, 110
- PMATH\_FILE\_OTHERERROR
  - file\_api, 110
- PMATH\_FILE\_RECURSIVE
  - file\_api, 110
- PMATH\_PREC\_ADD
  - parser, 76
- PMATH\_PREC\_ALT
  - parser, 76
- PMATH\_PREC\_AND
  - parser, 76
- PMATH\_PREC\_ANY
  - parser, 76
- PMATH\_PREC\_APL
  - parser, 76
- PMATH\_PREC\_ARROW
  - parser, 76
- PMATH\_PREC\_ASS
  - parser, 76
- PMATH\_PREC\_CALL
  - parser, 77
- PMATH\_PREC\_CIRCADD
  - parser, 76
- PMATH\_PREC\_CIRCMUL
  - parser, 76
- PMATH\_PREC\_COND
  - parser, 76
- PMATH\_PREC\_CROSS
  - parser, 76
- PMATH\_PREC\_CTRL\_AUTO
  - numbers, 26
- PMATH\_PREC\_CTRL\_GIVEN\_ACC
  - numbers, 27
- PMATH\_PREC\_CTRL\_GIVEN\_PREC
  - numbers, 26
- PMATH\_PREC\_CTRL\_MACHINE\_-PREC
  - numbers, 26
- PMATH\_PREC\_DIFF
  - parser, 77
- PMATH\_PREC\_DIV
  - parser, 76
- PMATH\_PREC\_EVAL
  - parser, 76
- PMATH\_PREC\_FAC
  - parser, 76
- PMATH\_PREC\_FUNC
  - parser, 76
- PMATH\_PREC\_INC
  - parser, 77
- PMATH\_PREC\_ISECT
  - parser, 76
- PMATH\_PREC\_LAZY
  - parser, 76
- PMATH\_PREC\_MAP
  - parser, 76
- PMATH\_PREC\_MIDDOT
  - parser, 76
- PMATH\_PREC\_MODY
  - parser, 76
- PMATH\_PREC\_MUL
  - parser, 76
- PMATH\_PREC\_MUL2
  - parser, 76
- PMATH\_PREC\_OR
  - parser, 76
- PMATH\_PREC\_PLUMI
  - parser, 76
- PMATH\_PREC\_POW
  - parser, 76
- PMATH\_PREC\_PRIM
  - parser, 77
- PMATH\_PREC\_RANGE
  - parser, 76

- PMATH\_PREC\_REL
  - parser, [76](#)
- PMATH\_PREC\_REPEAT
  - parser, [77](#)
- PMATH\_PREC\_REPL
  - parser, [76](#)
- PMATH\_PREC\_RULE
  - parser, [76](#)
- PMATH\_PREC\_SEQ
  - parser, [76](#)
- PMATH\_PREC\_STR
  - parser, [76](#)
- PMATH\_PREC\_TEST
  - parser, [77](#)
- PMATH\_PREC\_UNION
  - parser, [76](#)
- PMATH\_PREC\_XOR
  - parser, [76](#)
- PMATH\_SERIALIZE\_BAD\_BYTE
  - serialize.h, [246](#)
- PMATH\_SERIALIZE\_BAD\_OBJECT
  - serialize.h, [246](#)
- PMATH\_SERIALIZE\_BAD\_REF
  - serialize.h, [246](#)
- PMATH\_SERIALIZE\_EOF
  - serialize.h, [246](#)
- PMATH\_SERIALIZE\_NO\_MEMORY
  - serialize.h, [246](#)
- PMATH\_SERIALIZE\_OK
  - serialize.h, [246](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-ASSOCIATIVE
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-DEEPHOLDALL
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-DEFINITEFUNCTION
  - symbols, [62](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-HOLDALL
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-HOLDALLCOMPLETE
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-HOLDFIRST
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-HOLDREST
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-LISTABLE
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-NHOLDALL
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-NHOLDFIRST
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-NHOLDREST
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-NUMERICFUNCTION
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-ONEIDENTITY
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-PROTECTED
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-READPROTECTED
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-REMOVED
  - symbols, [62](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-SEQUENCEHOLD
  - symbols, [62](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-SYMMETRIC
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-TEMPORARY
  - symbols, [61](#)
- PMATH\_SYMBOL\_ATTRIBUTE\_-THREADLOCAL
  - symbols, [61](#)
- PMATH\_TOK\_ASSIGNTAG
  - parser, [77](#)
- PMATH\_TOK\_BINARY\_LEFT
  - parser, [77](#)
- PMATH\_TOK\_BINARY\_LEFT\_-AUTOARG
  - parser, [77](#)
- PMATH\_TOK\_BINARY\_LEFT\_OR\_-PREFIX
  - parser, [77](#)

- PMATH\_TOK\_BINARY\_RIGHT
  - parser, [77](#)
- PMATH\_TOK\_CALL
  - parser, [77](#)
- PMATH\_TOK\_COLON
  - parser, [77](#)
- PMATH\_TOK\_COMMENTEND
  - parser, [78](#)
- PMATH\_TOK\_DIGIT
  - parser, [77](#)
- PMATH\_TOK\_INTEGRAL
  - parser, [78](#)
- PMATH\_TOK\_LEFT
  - parser, [77](#)
- PMATH\_TOK\_LEFTCALL
  - parser, [77](#)
- PMATH\_TOK\_NAME
  - parser, [77](#)
- PMATH\_TOK\_NAME2
  - parser, [77](#)
- PMATH\_TOK\_NARY
  - parser, [77](#)
- PMATH\_TOK\_NARY\_AUTOARG
  - parser, [77](#)
- PMATH\_TOK\_NARY\_OR\_PREFIX
  - parser, [77](#)
- PMATH\_TOK\_NONE
  - parser, [77](#)
- PMATH\_TOK\_PLUSPLUS
  - parser, [77](#)
- PMATH\_TOK\_POSTFIX
  - parser, [77](#)
- PMATH\_TOK\_POSTFIX\_OR\_PREFIX
  - parser, [77](#)
- PMATH\_TOK\_PREFIX
  - parser, [77](#)
- PMATH\_TOK\_PRETEXT
  - parser, [77](#)
- PMATH\_TOK\_QUESTION
  - parser, [78](#)
- PMATH\_TOK\_RIGHT
  - parser, [77](#)
- PMATH\_TOK\_SLOT
  - parser, [78](#)
- PMATH\_TOK\_SPACE
  - parser, [77](#)
- PMATH\_TOK\_STRING
  - parser, [77](#)
- PMATH\_TOK\_TILDES
  - parser, [77](#)
- PMATH\_TYPE\_BIGSTRING
  - objects, [44](#)
- PMATH\_TYPE\_CUSTOM
  - objects, [44](#)
- PMATH\_TYPE\_EXPRESSION
  - objects, [44](#)
- PMATH\_TYPE\_EXPRESSION\_-GENERAL
  - objects, [44](#)
- PMATH\_TYPE\_EXPRESSION\_-GENERAL\_PART
  - objects, [44](#)
- PMATH\_TYPE\_MP\_FLOAT
  - objects, [44](#)
- PMATH\_TYPE\_MP\_INT
  - objects, [44](#)
- PMATH\_TYPE\_QUOTIENT
  - objects, [44](#)
- PMATH\_TYPE\_SHIFT\_BIGSTRING
  - objects, [44](#)
- PMATH\_TYPE\_SHIFT\_COUNT
  - objects, [44](#)
- PMATH\_TYPE\_SHIFT\_CUSTOM
  - objects, [44](#)
- PMATH\_TYPE\_SHIFT\_-EXPRESSION\_GENERAL
  - objects, [44](#)
- PMATH\_TYPE\_SHIFT\_-EXPRESSION\_GENERAL\_-PART
  - objects, [44](#)
- PMATH\_TYPE\_SHIFT\_MP\_FLOAT
  - objects, [44](#)
- PMATH\_TYPE\_SHIFT\_MP\_INT
  - objects, [44](#)
- PMATH\_TYPE\_SHIFT\_QUOTIENT
  - objects, [44](#)
- PMATH\_TYPE\_SHIFT\_RESERVED\_1
  - objects, [44](#)
- PMATH\_TYPE\_SHIFT\_SYMBOL
  - objects, [44](#)
- PMATH\_TYPE\_SYMBOL
  - objects, [44](#)
- PMATH\_WRITE\_OPTIONS\_-FULLEXPR
  - objects, [45](#)
- PMATH\_WRITE\_OPTIONS\_-FULLNAME
  - objects, [45](#)
- PMATH\_WRITE\_OPTIONS\_FULLSTR

- objects, [45](#)
- PMATH\_WRITE\_OPTIONS\_-  
INPUTEXPR
  - objects, [45](#)
- PMATH\_ABORT\_EXCEPTION
  - objects, [40](#)
- pmath\_abort\_please
  - threads, [101](#)
- pmath\_aborting
  - threads, [101](#)
- pmath\_accuracy
  - numbers, [27](#)
- PMATH\_API
  - pmath-config.h, [216](#)
- pmath\_approximate
  - numbers, [27](#)
- PMATH\_AS\_DOUBLE
  - objects-inline.h, [223](#)
- PMATH\_AS\_INT32
  - objects-inline.h, [220](#)
- PMATH\_AS\_PTR
  - objects-inline.h, [223](#)
- PMATH\_AS\_TAG
  - objects-inline.h, [220](#)
- pmath\_atomic2\_t, [169](#)
- pmath\_atomic\_barrier
  - atomic\_ops, [90](#)
- pmath\_atomic\_compare\_and\_set
  - atomic\_ops, [90](#)
- pmath\_atomic\_compare\_and\_set\_2
  - atomic\_ops, [90](#)
- PMATH\_ATOMIC\_FASTLOOP\_-  
COUNT
  - atomic.h, [236](#)
  - atomic\_ops, [89](#)
- pmath\_atomic\_fetch\_add
  - atomic\_ops, [91](#)
- pmath\_atomic\_fetch\_compare\_and\_set
  - atomic\_ops, [91](#)
- pmath\_atomic\_fetch\_set
  - atomic\_ops, [92](#)
- pmath\_atomic\_have\_cas2
  - atomic\_ops, [92](#)
- pmath\_atomic\_lock
  - atomic\_ops, [92](#)
- pmath\_atomic\_loop\_nop
  - atomic.h, [236](#)
  - atomic\_ops, [93](#)
- pmath\_atomic\_loop\_yield
  - atomic.h, [236](#)
- atomic\_ops, [93](#)
- PMATH\_ATOMIC\_STATIC\_INIT
  - atomic.h, [237](#)
- pmath\_atomic\_t, [169](#)
- pmath\_atomic\_unlock
  - atomic\_ops, [93](#)
- PMATH\_ATTRIBUTE\_NONNULL
  - pmath-config.h, [216](#)
- PMATH\_ATTRIBUTE\_PURE
  - pmath-config.h, [216](#)
- PMATH\_ATTRIBUTE\_USE\_RESULT
  - pmath-config.h, [216](#)
- PMATH\_AUTO\_PRECISION
  - numbers, [21](#)
- pmath\_binary\_file\_api\_t, [170](#)
  - flush\_function, [170](#)
  - read\_function, [170](#)
  - status\_function, [170](#)
  - struct\_size, [171](#)
  - write\_function, [171](#)
- pmath\_bool\_t
  - general\_types, [87](#)
- pmath\_boxes\_from\_spans
  - parser, [78](#)
- pmath\_build\_value
  - helpers, [125](#)
- pmath\_build\_value\_v
  - helpers, [127](#)
- PMATH\_BYTE\_ORDER
  - pmath-config.h, [216](#)
- PMATH\_C\_STRING
  - strings, [50](#)
- pmath\_callback\_t
  - general\_types, [87](#)
- pmath\_catch
  - threads, [102](#)
- PMATH\_CHAR\_ALIASDELIMITER
  - parser, [73](#)
- PMATH\_CHAR\_ALIASINDICATOR
  - parser, [73](#)
- PMATH\_CHAR\_ASSIGN
  - parser, [73](#)
- PMATH\_CHAR\_ASSIGNDELAYED
  - parser, [73](#)
- PMATH\_CHAR\_BOX
  - parser, [73](#)
- pmath\_char\_from\_name
  - strings, [51](#)
- PMATH\_CHAR\_INTEGRAL\_D
  - parser, [73](#)



- PMATH\_CHAR\_INVISIBLECALL
  - parser, [74](#)
- pmath\_char\_is\_36digit
  - parser, [78](#)
- pmath\_char\_is\_basedigit
  - parser, [78](#)
- pmath\_char\_is\_digit
  - parser, [78](#)
- pmath\_char\_is\_hexdigit
  - parser, [79](#)
- pmath\_char\_is\_integral
  - parser, [79](#)
- pmath\_char\_is\_left
  - parser, [79](#)
- pmath\_char\_is\_name
  - parser, [79](#)
- pmath\_char\_is\_right
  - parser, [79](#)
- PMATH\_CHAR\_LEFT\_BOX
  - parser, [74](#)
- pmath\_char\_maybe\_bigop
  - parser, [79](#)
- PMATH\_CHAR\_PIECEWISE
  - parser, [74](#)
- PMATH\_CHAR\_PLACEHOLDER
  - parser, [74](#)
- PMATH\_CHAR\_RIGHT\_BOX
  - parser, [74](#)
- PMATH\_CHAR\_RULE
  - parser, [74](#)
- PMATH\_CHAR\_RULEDELAYED
  - parser, [74](#)
- pmath\_char\_to\_name
  - strings, [51](#)
- PMATH\_CHAR\_VECTOR
  - parser, [74](#)
- pmath\_collect\_temporary\_symbols
  - pmath\_symbol\_t, [187](#)
- pmath\_compare
  - objects, [45](#)
- pmath\_compare\_func\_t
  - objects, [42](#)
- PMATH\_CONCAT
  - pmath-config.h, [216](#)
- PMATH\_CONCAT\_
  - pmath-config.h, [216](#)
- pmath\_continue\_after\_abort
  - frontend, [139](#)
- pmath\_cstr\_writer\_info\_t, [171](#)
  - user, [171](#)
- write\_cstr, [171](#)
- pmath\_current\_head
  - helpers, [127](#)
- pmath\_custom\_get\_data
  - custom, [9](#)
- pmath\_custom\_has\_destructor
  - custom, [10](#)
- pmath\_custom\_new
  - custom, [10](#)
- pmath\_custom\_t, [172](#)
  - custom, [9](#)
- pmath\_debug\_print
  - debug, [106](#)
- pmath\_debug\_print\_object
  - debug, [107](#)
- pmath\_debug\_print\_stack
  - debug, [107](#)
- PMATH\_DECLARE\_ALIGNED
  - atomic\_ops, [90](#)
- PMATH\_DEPRECATED
  - pmath-config.h, [217](#)
- pmath\_deserialize
  - serialize.h, [246](#)
- pmath\_done
  - frontend, [139](#)
- pmath\_emit
  - helpers, [127](#)
- pmath\_equal\_func\_t
  - objects, [42](#)
- pmath\_equals
  - pmath\_t, [190](#)
- pmath\_evaluate
  - pmath\_t, [191](#)
- pmath\_evaluate\_expression
  - pmath\_expr\_t, [174](#)
- pmath\_expr\_append
  - expressions, [12](#)
- pmath\_expr\_extract\_item
  - expressions, [12](#)
- pmath\_expr\_flatten
  - expressions, [13](#)
- pmath\_expr\_get\_item
  - expressions, [13](#)
- pmath\_expr\_get\_item\_range
  - expressions, [13](#)
- pmath\_expr\_length
  - expressions, [14](#)
- pmath\_expr\_new
  - expressions, [14](#)
- pmath\_expr\_new\_extended

- expressions, [14](#)
- `pmath_expr_remove_all`
  - expressions, [15](#)
- `pmath_expr_resize`
  - expressions, [15](#)
- `pmath_expr_set_item`
  - expressions, [16](#)
- `pmath_expr_sort`
  - expressions, [16](#)
- `pmath_expr_t`, [172](#)
  - expressions, [12](#)
  - `pmath_evaluate_expression`, [174](#)
- `PMATH_EXTERN_C`
  - `pmath-config.h`, [217](#)
- `pmath_file_binary_buffer_manipulate`
  - `file_api`, [110](#)
- `pmath_file_binary_buffer_size`
  - `file_api`, [110](#)
- `pmath_file_close`
  - `file_api`, [110](#)
- `pmath_file_create_binary`
  - `file_api`, [111](#)
- `pmath_file_create_binary_buffer`
  - `file_api`, [111](#)
- `pmath_file_create_compressor`
  - `file_api`, [111](#)
- `pmath_file_create_mixed_buffer`
  - `file_api`, [112](#)
- `pmath_file_create_text`
  - `file_api`, [112](#)
- `pmath_file_create_text_from_binary`
  - `file_api`, [113](#)
- `pmath_file_create_uncompressor`
  - `file_api`, [113](#)
- `pmath_file_flush`
  - `file_api`, [113](#)
- `pmath_file_manipulate`
  - `file_api`, [113](#)
- `pmath_file_read`
  - `file_api`, [114](#)
- `pmath_file_readline`
  - `file_api`, [114](#)
- `pmath_file_set_binbuffer`
  - `file_api`, [115](#)
- `pmath_file_set_textbuffer`
  - `file_api`, [115](#)
- `pmath_file_status`
  - `file_api`, [115](#)
- `pmath_file_test`
  - `file_api`, [115](#)
- `pmath_file_write`
  - `file_api`, [116](#)
- `pmath_file_write_object`
  - `file_api`, [116](#)
- `pmath_file_writetext`
  - `file_api`, [117](#)
- `pmath_files_status_t`
  - `file_api`, [109](#)
- `pmath_float_new_str`
  - numbers, [27](#)
- `pmath_float_t`, [175](#)
  - numbers, [26](#)
- `PMATH_FORCE_INLINE`
  - `pmath-config.h`, [217](#)
- `PMATH_FROM_DOUBLE`
  - `objects-inline.h`, [223](#)
- `PMATH_FROM_INT32`
  - objects, [40](#)
- `PMATH_FROM_PTR`
  - objects, [45](#)
- `PMATH_FROM_TAG`
  - objects, [45](#)
- `pmath_func_t`
  - objects, [42](#)
- `pmath_gather_begin`
  - helpers, [128](#)
- `pmath_gather_end`
  - helpers, [128](#)
- `pmath_hash`
  - objects, [45](#)
- `pmath_hash_func_t`
  - objects, [42](#)
- `pmath_hashtable_t`, [175](#)
  - hashtables, [119](#)
- `pmath_ht_capacity`
  - hashtables, [121](#)
- `pmath_ht_class_t`, [176](#)
  - `entry_destructor`, [177](#)
  - `entry_equals_key`, [177](#)
  - `entry_hash`, [177](#)
  - `entry_keys_equal`, [177](#)
  - `key_hash`, [177](#)
- `pmath_ht_clear`
  - hashtables, [121](#)
- `pmath_ht_copy`
  - hashtables, [121](#)
- `pmath_ht_count`
  - hashtables, [122](#)
- `pmath_ht_create`
  - hashtables, [122](#)

- pmath\_ht\_destroy
  - hashtables, [122](#)
- pmath\_ht\_entry
  - hashtables, [122](#)
- pmath\_ht\_entry\_callback\_t
  - hashtables, [119](#)
- pmath\_ht\_entry\_copy\_t
  - hashtables, [119](#)
- pmath\_ht\_entry\_equal\_func\_t
  - hashtables, [119](#)
- pmath\_ht\_entry\_equals\_key\_func\_t
  - hashtables, [120](#)
- pmath\_ht\_entry\_hash\_func\_t
  - hashtables, [120](#)
- pmath\_ht\_insert
  - hashtables, [123](#)
- pmath\_ht\_key\_hash\_func\_t
  - hashtables, [120](#)
- pmath\_ht\_remove
  - hashtables, [123](#)
- pmath\_ht\_search
  - hashtables, [123](#)
- pmath\_init
  - frontend, [140](#)
- PMATH\_INLINE
  - pmath-config.h, [217](#)
- pmath\_integer\_fits\_si32
  - numbers, [21](#)
- pmath\_integer\_fits\_si64
  - numbers, [28](#)
- pmath\_integer\_fits\_siptr
  - numbers, [21](#), [22](#)
- pmath\_integer\_fits\_ui32
  - numbers, [29](#)
- pmath\_integer\_fits\_ui64
  - numbers, [29](#)
- pmath\_integer\_fits\_uiptr
  - numbers, [22](#)
- pmath\_integer\_get\_si32
  - numbers, [29](#)
- pmath\_integer\_get\_si64
  - numbers, [30](#)
- pmath\_integer\_get\_siptr
  - numbers, [22](#), [23](#)
- pmath\_integer\_get\_ui32
  - numbers, [30](#)
- pmath\_integer\_get\_ui64
  - numbers, [30](#)
- pmath\_integer\_get\_uiptr
  - numbers, [23](#)
- pmath\_integer\_new\_data
  - numbers, [31](#)
- pmath\_integer\_new\_si32
  - numbers, [24](#)
- pmath\_integer\_new\_si64
  - numbers, [31](#)
- pmath\_integer\_new\_siptr
  - numbers, [24](#), [25](#)
- pmath\_integer\_new\_slong
  - numbers, [32](#)
- pmath\_integer\_new\_str
  - numbers, [32](#)
- pmath\_integer\_new\_ui32
  - numbers, [32](#)
- pmath\_integer\_new\_ui64
  - numbers, [33](#)
- pmath\_integer\_new\_uiptr
  - numbers, [25](#)
- pmath\_integer\_new\_ulong
  - numbers, [33](#)
- pmath\_integer\_t, [177](#)
  - numbers, [26](#)
- PMATH\_INVALID\_PTR
  - general\_types, [87](#)
- pmath\_is\_bigstr
  - objects-inline.h, [220](#)
- pmath\_is\_custom
  - objects-inline.h, [220](#)
- pmath\_is\_double
  - objects-inline.h, [220](#)
- pmath\_is\_evaluable
  - objects-inline.h, [223](#)
- pmath\_is\_evaluated
  - objects, [46](#)
- pmath\_is\_expr
  - objects-inline.h, [220](#)
- pmath\_is\_expr\_of
  - helpers, [129](#)
- pmath\_is\_expr\_of\_len
  - helpers, [129](#)
- pmath\_is\_float
  - objects-inline.h, [221](#)
- pmath\_is\_int32
  - objects-inline.h, [221](#)
- pmath\_is\_integer
  - objects-inline.h, [221](#)
- pmath\_is\_magic
  - objects-inline.h, [221](#)
- pmath\_is\_message\_queue
  - threadmsg, [95](#)

- pmath\_is\_ministr
  - objects-inline.h, 221
- pmath\_is\_mpffloat
  - objects-inline.h, 221
- pmath\_is\_mpint
  - objects-inline.h, 221
- pmath\_is\_null
  - objects-inline.h, 221
- pmath\_is\_number
  - objects-inline.h, 221
- pmath\_is\_pointer
  - objects-inline.h, 222
- pmath\_is\_pointer\_of
  - objects-inline.h, 223
- pmath\_is\_quotient
  - objects-inline.h, 222
- pmath\_is\_rational
  - objects-inline.h, 222
- pmath\_is\_str0
  - objects-inline.h, 222
- pmath\_is\_str1
  - objects-inline.h, 222
- pmath\_is\_str2
  - objects-inline.h, 222
- pmath\_is\_string
  - objects-inline.h, 222
- pmath\_is\_symbol
  - objects-inline.h, 222
- PMATH\_LIKELY
  - pmath-config.h, 217
- PMATH\_MACHINE\_PRECISION
  - numbers, 25
- pmath\_mem\_alloc
  - memory, 131
- pmath\_mem\_free
  - memory, 131
- pmath\_mem\_realloc
  - memory, 132
- pmath\_mem\_realloc\_no\_failfree
  - memory, 132
- pmath\_mem\_usage
  - memory, 132
- pmath\_message
  - messages, 134
- pmath\_message\_argxxx
  - messages, 134
- pmath\_message\_find\_text
  - messages, 135
- pmath\_message\_syntax\_error
  - messages, 135
- pmath\_messages\_t, 179
  - threadmsg, 95
- PMATH\_MODULE
  - pmath-config.h, 217
- pmath\_mpffloat\_t
  - numbers, 26
- pmath\_mpint\_t
  - numbers, 26
- pmath\_native\_writer
  - strings, 51
- PMATH\_NEED\_GNUC
  - pmath-config.h, 217
- PMATH\_NULL
  - objects, 47
- pmath\_number\_get\_d
  - numbers, 33
- pmath\_number\_neg
  - numbers, 34
- pmath\_number\_sign
  - numbers, 34
- pmath\_number\_t, 180
  - numbers, 26
- pmath\_option\_value
  - helpers, 129
- pmath\_options\_extract
  - helpers, 130
- pmath\_param\_proc\_t
  - objects, 42
- pmath\_parse\_string
  - parser, 79
- pmath\_parse\_string\_args
  - parser, 80
- pmath\_precision
  - numbers, 34
- pmath\_precision\_control\_t
  - numbers, 26
- pmath\_proc\_t
  - objects, 42
- pmath\_quotient\_t, 181
  - numbers, 26
- pmath\_rational\_denominator
  - numbers, 35
- pmath\_rational\_new
  - numbers, 35
- pmath\_rational\_numerator
  - numbers, 35
- pmath\_rational\_t, 181
  - numbers, 26
- pmath\_ref
  - pmath\_t, 190

- pmath\_refcount
  - objects-inline.h, 223
- pmath\_resume\_all
  - threads, 102
- pmath\_right\_fence
  - parser, 80
- PMATH\_RUN
  - parser, 75
- PMATH\_RUN\_ARGS
  - parser, 75
- pmath\_same
  - objects-inline.h, 222
- pmath\_serialize
  - serialize.h, 246
- pmath\_serialize\_error\_t
  - serialize.h, 246
- pmath\_session\_end
  - frontend, 140
- pmath\_session\_execute
  - frontend, 140
- pmath\_session\_start
  - frontend, 141
- pmath\_set\_accuracy
  - numbers, 36
- pmath\_set\_precision
  - numbers, 36
- pmath\_span\_array\_free
  - parser, 81
- pmath\_span\_array\_is\_operand\_start
  - parser, 81
- pmath\_span\_array\_is\_token\_end
  - parser, 81
- pmath\_span\_array\_length
  - parser, 81
- pmath\_span\_array\_t, 182
  - parser, 75
- pmath\_span\_at
  - parser, 82
- pmath\_span\_end
  - parser, 82
- pmath\_span\_next
  - parser, 82
- pmath\_span\_t, 183
  - parser, 75
- pmath\_spans\_from\_boxes
  - parser, 83
- pmath\_spans\_from\_string
  - parser, 83
- pmath\_stack\_free
  - stacks, 137
- pmath\_stack\_new
  - stacks, 137
- pmath\_stack\_pop
  - stacks, 137
- pmath\_stack\_push
  - stacks, 137
- pmath\_stack\_t, 184
  - stacks, 136
- pmath\_stack\_walker\_t
  - helpers, 125
- PMATH\_STATIC\_ASSERT
  - pmath-config.h, 217
- PMATH\_STATIC\_NULL
  - objects, 40
- PMATH\_STATIC\_UNDEFINED
  - objects, 40
- pmath\_string\_buffer
  - strings, 51
- pmath\_string\_concat
  - strings, 52
- pmath\_string\_equals\_latin1
  - strings, 52
- pmath\_string\_expand\_boxes
  - parser, 84
- pmath\_string\_from\_native
  - strings, 52
- pmath\_string\_from\_utf8
  - strings, 53
- pmath\_string\_insert
  - strings, 53
- pmath\_string\_insert\_codepage
  - strings, 53
- pmath\_string\_insert\_latin1
  - strings, 54
- pmath\_string\_insert\_ucs2
  - strings, 54
- pmath\_string\_length
  - strings, 55
- pmath\_string\_new
  - strings, 55
- pmath\_string\_part
  - strings, 55
- pmath\_string\_t, 184
  - strings, 51
- pmath\_string\_to\_native
  - strings, 56
- pmath\_string\_to\_utf8
  - strings, 56
- pmath\_strtod
  - strtod.h, 247

- `pmath_suspend_all_please`
  - threads, [102](#)
- `pmath_symbol_attributes_t`
  - symbols, [59](#)
- `pmath_symbol_create_temporary`
  - symbols, [62](#)
- `pmath_symbol_find`
  - symbols, [62](#)
- `pmath_symbol_get`
  - symbols, [62](#)
- `pmath_symbol_get_attributes`
  - symbols, [63](#)
- `pmath_symbol_get_value`
  - symbols, [63](#)
- `pmath_symbol_iter_next`
  - symbols, [63](#)
- `pmath_symbol_name`
  - symbols, [64](#)
- `pmath_symbol_remove`
  - symbols, [64](#)
- `pmath_symbol_set_attributes`
  - symbols, [65](#)
- `pmath_symbol_set_value`
  - symbols, [65](#)
- `pmath_symbol_synchronized`
  - symbols, [65](#)
- `pmath_symbol_t`, [186](#)
  - `pmath_collect_temporary_symbols`, [187](#)
  - symbols, [61](#)
- `pmath_symbol_update`
  - symbols, [66](#)
- `pmath_t`, [188](#)
  - `as_bits`, [191](#)
  - `as_chars`, [191](#)
  - `as_double`, [191](#)
  - `as_int32`, [191](#)
  - `pmath_equals`, [190](#)
  - `pmath_evaluate`, [191](#)
  - `pmath_ref`, [190](#)
  - `pmath_unref`, [190](#)
  - `s`, [191](#)
  - `tag`, [191](#)
  - `u`, [191](#)
- `PMATH_TAG_INT32`
  - objects, [40](#)
- `PMATH_TAG_INVALID`
  - objects, [40](#)
- `PMATH_TAG_MAGIC`
  - objects, [40](#)
- `PMATH_TAG_STR0`
  - objects, [41](#)
- `PMATH_TAG_STR1`
  - objects, [41](#)
- `PMATH_TAG_STR2`
  - objects, [41](#)
- `PMATH_TAGMASK_BITCOUNT`
  - objects, [41](#)
- `PMATH_TAGMASK_NONDOUBLE`
  - objects, [41](#)
- `PMATH_TAGMASK_POINTER`
  - objects, [41](#)
- `pmath_task_get_data`
  - threadpool.h, [239](#)
- `pmath_task_has_destructor`
  - threadpool.h, [239](#)
- `pmath_task_new`
  - threadpool.h, [239](#)
- `pmath_task_ref`
  - threadpool.h, [239](#)
- `pmath_task_t`
  - threadpool.h, [239](#)
- `pmath_task_unref`
  - threadpool.h, [239](#)
- `pmath_task_wait`
  - threadpool.h, [239](#)
- `pmath_text_file_api_t`, [192](#)
  - `flush_function`, [192](#)
  - `readln_function`, [192](#)
  - `status_function`, [193](#)
  - `struct_size`, [193](#)
  - `write_function`, [193](#)
- `pmath_thread_aborting`
  - threads, [102](#)
- `pmath_thread_call_locked`
  - threads, [102](#)
- `pmath_thread_fork_daemon`
  - `pmath_thread_t`, [194](#)
- `pmath_thread_get_current`
  - threads, [103](#)
- `pmath_thread_get_parent`
  - threads, [103](#)
- `pmath_thread_get_queue`
  - threadmsg, [95](#)
- `pmath_thread_is_parent`
  - threads, [103](#)
- `PMATH_THREAD_KEY_-PARSERARGUMENTS`
  - objects, [41](#)

- PMATH\_THREAD\_KEY\_-  
PARSESYMBOLS
  - objects, [41](#)
- pmath\_thread\_local\_load
  - threads, [104](#)
- pmath\_thread\_local\_save
  - threads, [104](#)
- pmath\_thread\_queue\_is\_blocked\_by
  - threads, [105](#)
- pmath\_thread\_run\_with\_interrupt\_-  
notifier
  - threads, [105](#)
- pmath\_thread\_send
  - threadmsg, [96](#)
- pmath\_thread\_send\_delayed
  - threadmsg, [96](#)
- pmath\_thread\_send\_wait
  - threadmsg, [96](#)
- pmath\_thread\_sleep
  - threadmsg, [97](#)
- pmath\_thread\_sleep\_timeout
  - threadmsg, [97](#)
- pmath\_thread\_t, [193](#)
  - pmath\_thread\_fork\_daemon, [194](#)
  - threads, [101](#)
- pmath\_thread\_wakeup
  - threadmsg, [98](#)
- pmath\_threadlock\_t, [195](#)
  - threads, [101](#)
- pmath\_throw
  - threads, [106](#)
- pmath\_tickcount
  - threadmsg, [98](#)
- pmath\_token\_analyse
  - parser, [84](#)
- pmath\_token\_maybe\_bigop
  - parser, [85](#)
- pmath\_token\_maybe\_first
  - parser, [85](#)
- pmath\_token\_maybe\_rest
  - parser, [85](#)
- pmath\_token\_prefix\_precedence
  - parser, [85](#)
- pmath\_token\_t
  - parser, [77](#)
- pmath\_type\_t
  - objects, [43](#)
- PMATH\_UNDEFINED
  - objects, [47](#)
- PMATH\_UNLIKELY
  - pmath-config.h, [217](#)
- pmath\_unref
  - pmath\_t, [190](#)
- PMATH\_UNUSED
  - pmath-config.h, [217](#)
- pmath\_utf8\_writer
  - strings, [57](#)
- PMATH\_VERSION\_BUILD
  - version.h, [249](#)
- PMATH\_VERSION\_BUILD\_HISTORY
  - version.h, [249](#)
- PMATH\_VERSION\_BUILDS\_COUNT
  - version.h, [249](#)
- PMATH\_VERSION\_DATE
  - version.h, [249](#)
- pmath\_version\_datetime
  - version, [138](#)
- PMATH\_VERSION\_FULLVERSION\_-  
STRING
  - version.h, [249](#)
- PMATH\_VERSION\_MAJOR
  - version.h, [249](#)
- PMATH\_VERSION\_MINOR
  - version.h, [249](#)
- PMATH\_VERSION\_MONTH
  - version.h, [249](#)
- pmath\_version\_number
  - version, [138](#)
- pmath\_version\_number\_part
  - version, [138](#)
- PMATH\_VERSION\_RC\_-  
FILEVERSION
  - version.h, [248](#)
- PMATH\_VERSION\_RC\_-  
FILEVERSION\_STRING
  - version.h, [248](#)
- PMATH\_VERSION\_REVISION
  - version.h, [249](#)
- PMATH\_VERSION\_STATUS
  - version.h, [249](#)
- PMATH\_VERSION\_STATUS\_SHORT
  - version.h, [250](#)
- PMATH\_VERSION\_SVN\_DATE
  - version.h, [250](#)
- PMATH\_VERSION\_SVN\_REVISION
  - version.h, [250](#)
- PMATH\_VERSION\_UBUNTU\_-  
VERSION\_STYLE
  - version.h, [250](#)
- PMATH\_VERSION\_YEAR

- version.h, 250
- pmath\_walk\_stack
  - helpers, 130
- pmath\_write
  - objects, 46
- pmath\_write\_ex
  - objects, 46
- pmath\_write\_ex\_t, 195
  - options, 196
  - post\_write, 196
  - pre\_write, 196
  - size, 196
  - user, 196
  - write, 197
- pmath\_write\_options\_t
  - objects, 43
- pmath\_write\_with\_pagewidth
  - objects, 47
- post\_write
  - pmath\_write\_ex\_t, 196
- Power
  - pmath, 149
- pre\_write
  - pmath\_write\_ex\_t, 196
- Quotient
  - pmath, 150
- Rational
  - pmath, 150
- Re
  - pmath, 150
- read
  - pmath::BinaryUserStream, 154
  - pmath::ReadableBinaryFile, 198
- read\_function
  - pmath\_binary\_file\_api\_t, 170
- ReadableBinaryFile
  - pmath::ReadableBinaryFile, 198
- ReadableTextFile
  - pmath::ReadableTextFile, 199
- readline
  - pmath::ReadableTextFile, 200
  - pmath::TextUserStream, 209
- readln\_function
  - pmath\_text\_file\_api\_t, 192
- Ref
  - pmath, 150
- release
  - pmath::Expr, 163
- remove
  - pmath::String, 205
- Round
  - pmath, 150
- Rule
  - pmath, 150
- RuleDelayed
  - pmath, 150
- s
  - pmath\_t, 191
- serialize
  - pmath::Expr, 163
- serialize.h
  - PMATH\_SERIALIZE\_BAD\_-  
BYTE, 246
  - PMATH\_SERIALIZE\_BAD\_-  
OBJECT, 246
  - PMATH\_SERIALIZE\_BAD\_REF,  
246
  - PMATH\_SERIALIZE\_EOF, 246
  - PMATH\_SERIALIZE\_NO\_-  
MEMORY, 246
  - PMATH\_SERIALIZE\_OK, 246
  - pmath\_deserialize, 246
  - pmath\_serialize, 246
  - pmath\_serialize\_error\_t, 246
- set
  - pmath::Expr, 163
- set\_buffer
  - pmath::TextFile, 207
- set\_buffer\_size
  - pmath::BinaryFile, 153
- Sign
  - pmath, 150
- Sin
  - pmath, 150
- size
  - pmath\_write\_ex\_t, 196
- Sqrt
  - pmath, 151
- stacks
  - pmath\_stack\_free, 137
  - pmath\_stack\_new, 137
  - pmath\_stack\_pop, 137
  - pmath\_stack\_push, 137
  - pmath\_stack\_t, 136
- starts\_with
  - pmath::String, 205, 206
- status



- pmath::BinaryUserStream, [154](#)
  - pmath::File, [167](#)
  - pmath::TextUserStream, [209](#)
- status\_function
  - pmath\_binary\_file\_api\_t, [170](#)
  - pmath\_text\_file\_api\_t, [193](#)
- String
  - pmath::String, [202](#)
- Strings, [48](#)
- strings
  - PMATH\_C\_STRING, [50](#)
  - pmath\_char\_from\_name, [51](#)
  - pmath\_char\_to\_name, [51](#)
  - pmath\_native\_writer, [51](#)
  - pmath\_string\_buffer, [51](#)
  - pmath\_string\_concat, [52](#)
  - pmath\_string\_equals\_latin1, [52](#)
  - pmath\_string\_from\_native, [52](#)
  - pmath\_string\_from\_utf8, [53](#)
  - pmath\_string\_insert, [53](#)
  - pmath\_string\_insert\_codepage, [53](#)
  - pmath\_string\_insert\_latin1, [54](#)
  - pmath\_string\_insert\_ucs2, [54](#)
  - pmath\_string\_length, [55](#)
  - pmath\_string\_new, [55](#)
  - pmath\_string\_part, [55](#)
  - pmath\_string\_t, [51](#)
  - pmath\_string\_to\_native, [56](#)
  - pmath\_string\_to\_utf8, [56](#)
  - pmath\_utf8\_writer, [57](#)
- strtod.h
  - pmath\_strtod, [247](#)
- struct\_size
  - pmath\_binary\_file\_api\_t, [171](#)
  - pmath\_text\_file\_api\_t, [193](#)
- Symbol
  - pmath, [151](#)
- SymbolPi
  - pmath, [151](#)
- Symbols, [57](#)
- symbols
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_ASSOCIATIVE, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_-
      - DEEPHOLDALL, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_-
      - DEFINITEFUNCTION, [62](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_-
      - ATTRIBUTE\_HOLDALL, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_-
      - ATTRIBUTE\_HOLDFIRST, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_-
      - ATTRIBUTE\_HOLDREST, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_-
      - ATTRIBUTE\_LISTABLE, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_-
      - ATTRIBUTE\_NHOLDALL, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_-
      - ATTRIBUTE\_NHOLDFIRST, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_-
      - ATTRIBUTE\_NHOLDREST, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_-
      - NUMERICFUNCTION, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_-
      - ONEIDENTITY, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_PROTECTED, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_-
      - READPROTECTED, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_REMOVED, [62](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_-
      - SEQUENCEHOLD, [62](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_SYMMETRIC, [61](#)
  - PMATH\_SYMBOL\_-
    - ATTRIBUTE\_TEMPORARY, [61](#)

- 61
- PMATH\_SYMBOL\_-
  - ATTRIBUTE\_-
  - THREADLOCAL, 61
- pmath\_symbol\_attributes\_t, 59
- pmath\_symbol\_create\_temporary, 62
- pmath\_symbol\_find, 62
- pmath\_symbol\_get, 62
- pmath\_symbol\_get\_attributes, 63
- pmath\_symbol\_get\_value, 63
- pmath\_symbol\_iter\_next, 63
- pmath\_symbol\_name, 64
- pmath\_symbol\_remove, 64
- pmath\_symbol\_set\_attributes, 65
- pmath\_symbol\_set\_value, 65
- pmath\_symbol\_synchronized, 65
- pmath\_symbol\_t, 61
- pmath\_symbol\_update, 66
- tag
  - pmath\_t, 191
- Tan
  - pmath, 151
- TextFile
  - pmath::TextFile, 207
- Thread Messaging, 93
- threadmsg
  - pmath\_is\_message\_queue, 95
  - pmath\_messages\_t, 95
  - pmath\_thread\_get\_queue, 95
  - pmath\_thread\_send, 96
  - pmath\_thread\_send\_delayed, 96
  - pmath\_thread\_send\_wait, 96
  - pmath\_thread\_sleep, 97
  - pmath\_thread\_sleep\_timeout, 97
  - pmath\_thread\_wakeup, 98
  - pmath\_tickcount, 98
- threadpool.h
  - pmath\_task\_get\_data, 239
  - pmath\_task\_has\_destructor, 239
  - pmath\_task\_new, 239
  - pmath\_task\_ref, 239
  - pmath\_task\_t, 239
  - pmath\_task\_unref, 239
  - pmath\_task\_wait, 239
- threads
  - pmath\_abort\_please, 101
  - pmath\_aborting, 101
  - pmath\_catch, 102
  - pmath\_resume\_all, 102
  - pmath\_suspend\_all\_please, 102
  - pmath\_thread\_aborting, 102
  - pmath\_thread\_call\_locked, 102
  - pmath\_thread\_get\_current, 103
  - pmath\_thread\_get\_parent, 103
  - pmath\_thread\_is\_parent, 103
  - pmath\_thread\_local\_load, 104
  - pmath\_thread\_local\_save, 104
  - pmath\_thread\_queue\_is\_blocked\_-
    - by, 105
  - pmath\_thread\_run\_with\_interrupt\_-
    - notifier, 105
  - pmath\_thread\_t, 101
  - pmath\_threadlock\_t, 101
  - pmath\_throw, 106
- Threadsafe Stacks, 135
- Times
  - pmath, 151
- to\_double
  - pmath::Expr, 163
- to\_string
  - pmath::Expr, 164
- trim
  - pmath::String, 206
- TRUE
  - general\_types, 87
- u
  - pmath\_t, 191
- user
  - pmath\_cstr\_writer\_info\_t, 171
  - pmath\_write\_ex\_t, 196
- version
  - pmath\_version\_datetime, 138
  - pmath\_version\_number, 138
  - pmath\_version\_number\_part, 138
- Version Information, 138
- version.h, 248
  - PMATH\_VERSION\_BUILD, 249
  - PMATH\_VERSION\_BUILD\_-
    - HISTORY, 249
  - PMATH\_VERSION\_BUILDS\_-
    - COUNT, 249
  - PMATH\_VERSION\_DATE, 249
  - PMATH\_VERSION\_-
    - FULLVERSION\_STRING, 249
  - PMATH\_VERSION\_MAJOR, 249

- PMATH\_VERSION\_MINOR, [249](#)
- PMATH\_VERSION\_MONTH, [249](#)
- PMATH\_VERSION\_RC\_-
  - FILEVERSION, [248](#)
- PMATH\_VERSION\_RC\_-
  - FILEVERSION\_STRING, [248](#)
- PMATH\_VERSION\_REVISION, [249](#)
- PMATH\_VERSION\_STATUS, [249](#)
- PMATH\_VERSION\_STATUS\_-
  - SHORT, [250](#)
- PMATH\_VERSION\_SVN\_DATE, [250](#)
- PMATH\_VERSION\_SVN\_-
  - REVISION, [250](#)
- PMATH\_VERSION\_UBUNTU\_-
  - VERSION\_STYLE, [250](#)
- PMATH\_VERSION\_YEAR, [250](#)
- write
  - pmath::BinaryUserStream, [155](#)
  - pmath::TextUserStream, [209](#)
  - pmath::WriteableBinaryFile, [213](#)
  - pmath::WriteableTextFile, [215](#)
  - pmath\_write\_ex\_t, [197](#)
- write\_cstr
  - pmath\_cstr\_writer\_info\_t, [171](#)
- write\_function
  - pmath\_binary\_file\_api\_t, [171](#)
  - pmath\_text\_file\_api\_t, [193](#)
- write\_to\_file
  - pmath::Expr, [164](#)
- write\_to\_string
  - pmath::Expr, [164](#)
- WriteableBinaryFile
  - pmath::WriteableBinaryFile, [213](#)
- WriteableTextFile
  - pmath::WriteableTextFile, [214](#), [215](#)