

SParse

Generated by Doxygen 1.9.7

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	7
3.1 Allocator Struct Reference	7
3.2 AllocInfo Struct Reference	7
3.3 Array Class Reference	8
3.3.1 Member Function Documentation	8
3.3.1.1 Array_compare()	8
3.3.1.2 Array_copy()	8
3.3.1.3 Array_new()	9
3.3.1.4 Array_newFromFile()	9
3.4 ArrayParam Struct Reference	9
3.5 BTree Class Reference	9
3.6 Buffer Struct Reference	10
3.7 Class Struct Reference	10
3.8 Configuration Class Reference	10
3.8.1 Member Function Documentation	11
3.8.1.1 Configuration_new()	11
3.9 ConnectionParam Struct Reference	11
3.10 Declarator Struct Reference	11
3.11 FileDesc Class Reference	12
3.12 Fileilo Class Reference	12
3.13 Fileilo Struct Reference	12
3.13.1 Member Function Documentation	13
3.13.1.1 Fileilo_comp()	13
3.13.1.2 Fileilo_copy()	13
3.13.1.3 Fileilo_createFile()	13
3.13.1.4 Fileilo_delete()	14
3.13.1.5 Fileilo_getSize()	14
3.13.1.6 Fileilo_new()	14
3.13.1.7 Fileilo_openDir()	14
3.13.1.8 Fileilo_openFile()	15
3.13.1.9 Fileilo_print()	15
3.14 FileMgr Class Reference	15
3.14.1 Member Function Documentation	16
3.14.1.1 FileMgr_addDirectory()	16
3.14.1.2 FileMgr_addFile()	16
3.14.1.3 FileMgr_copy()	17
3.14.1.4 FileMgr_filterFiles()	17

3.14.1.5 FileMgr_getRef()	17
3.14.1.6 FileMgr_getRootLocation()	17
3.14.1.7 FileMgr_getSize()	18
3.14.1.8 FileMgr_load()	18
3.14.1.9 FileMgr_setRootLocation()	18
3.14.1.10 FileMgr_write()	18
3.15 FileReader Class Reference	19
3.15.1 Member Function Documentation	19
3.15.1.1 FileReader_addFile()	19
3.15.1.2 FileReader_copy()	20
3.15.1.3 FileReader_getBuffer()	20
3.15.1.4 FileReader_getName()	20
3.15.1.5 FileReader_getSize()	20
3.15.1.6 FileReader_new()	21
3.16 Grammar Struct Reference	21
3.17 Grammar2 Class Reference	21
3.17.1 Member Function Documentation	22
3.17.1.1 Grammar2_copy()	22
3.17.1.2 Grammar2_new()	22
3.18 GrammarC99 Class Reference	22
3.18.1 Member Function Documentation	23
3.18.1.1 GrammarC99_new()	23
3.19 GrammarContext Struct Reference	23
3.20 HTTPRequest Class Reference	23
3.20.1 Member Function Documentation	24
3.20.1.1 HTTPRequest_compare()	24
3.20.1.2 HTTPRequest_copy()	24
3.20.1.3 HTTPRequest_getSize()	24
3.21 HTTPResponse Class Reference	25
3.22 HTTPServer Class Reference	25
3.22.1 Member Function Documentation	25
3.22.1.1 HTTPRequest_new()	25
3.22.1.2 HTTPResponse_compare()	26
3.22.1.3 HTTPResponse_getSize()	26
3.22.1.4 HTTPServer_new()	26
3.23 IncludeInfo Struct Reference	27
3.24 List Class Reference	27
3.24.1 Member Function Documentation	28
3.24.1.1 List_compare()	28
3.24.1.2 List_copy()	28
3.24.1.3 List_forEach()	28
3.24.1.4 List_getNbNodes()	29

3.24.1.5 List_getSize()	29
3.24.1.6 List_insertHead()	29
3.24.1.7 List_insertTail()	29
3.24.1.8 List_merge()	30
3.24.1.9 List_new()	30
3.24.1.10 List_newFromAllocator()	30
3.24.1.11 ListNode_compare()	30
3.24.1.12 ListNode_copy()	31
3.24.1.13 ListNode_new()	31
3.24.1.14 ListNode_newFromAllocator()	31
3.25 MacroDefinition Struct Reference	31
3.26 MacroStore Struct Reference	32
3.27 MacroStoreNode Struct Reference	32
3.28 Malloc Struct Reference	32
3.29 Map Class Reference	33
3.29.1 Member Function Documentation	33
3.29.1.1 Map_copy()	33
3.29.1.2 Map_getAll()	33
3.29.1.3 Map_getSize()	34
3.29.1.4 Map_insert()	34
3.29.1.5 Map_new()	34
3.29.1.6 Map_newFromAllocator()	34
3.29.1.7 TaskMgr_new()	35
3.30 MapEntry Struct Reference	35
3.31 mem_align Union Reference	35
3.32 Mutex Struct Reference	36
3.33 MyAllocator Struct Reference	36
3.34 MyType Struct Reference	36
3.35 Node Struct Reference	37
3.36 Object Struct Reference	37
3.36.1 Member Function Documentation	38
3.36.1.1 Object_comp()	38
3.36.1.2 Object_copy()	38
3.36.1.3 Object_getRef()	38
3.36.1.4 Object_isValid()	38
3.36.1.5 Object_new()	38
3.36.1.6 Object_newFromAllocator()	39
3.36.1.7 Object_print()	39
3.37 ObjectInfo Struct Reference	39
3.38 ObjectMgr Class Reference	40
3.38.1 Member Function Documentation	40
3.38.1.1 ObjectMgr_allocate()	40

3.38.1.2 ObjectMgr_copy()	41
3.38.1.3 ObjectMgr_deallocate()	41
3.38.1.4 ObjectMgr_getRef()	41
3.38.2 Member Data Documentation	41
3.38.2.1 maxNbObjectAllocated	41
3.39 ObjectStore Class Reference	42
3.39.1 Member Function Documentation	42
3.39.1.1 ObjectStore_compare()	42
3.39.1.2 ObjectStore_copy()	43
3.39.1.3 ObjectStore_createAllocator()	43
3.39.1.4 ObjectStore_createObject()	43
3.39.1.5 ObjectStore_delete()	43
3.39.1.6 ObjectStore_deleteAllocator()	43
3.39.1.7 ObjectStore_deleteObject()	44
3.39.1.8 ObjectStore_getNbAllocatedObjects()	44
3.39.1.9 ObjectStore_getRef()	44
3.40 OptionDefault Struct Reference	44
3.41 OptionMgr Class Reference	45
3.41.1 Member Function Documentation	45
3.41.1.1 OptionMgr_getRef()	45
3.41.1.2 OptionMgr_readFromCmdLine()	45
3.42 PoolCache Struct Reference	46
3.43 Product Class Reference	46
3.44 SdbMgr Class Reference	46
3.44.1 Member Function Documentation	47
3.44.1.1 SdbMgr_copy()	47
3.44.1.2 SdbMgr_execute()	47
3.44.1.3 SdbMgr_getRef()	47
3.45 SdbRequest Class Reference	48
3.45.1 Member Function Documentation	48
3.45.1.1 SdbRequest_delete()	48
3.45.1.2 SdbRequest_execute()	48
3.45.1.3 SdbRequest_new()	49
3.46 SkipList Class Reference	49
3.46.1 Member Function Documentation	50
3.46.1.1 SkipList_add()	50
3.46.1.2 SkipList_compare()	51
3.46.1.3 SkipList_copy()	51
3.46.1.4 SkipList_delete()	51
3.46.1.5 SkipList_getSize()	52
3.46.1.6 SkipList_new()	52
3.46.1.7 SkipList_newFromAllocator()	52

3.46.1.8 SkipList_print()	53
3.46.1.9 SkipList_remove()	53
3.47 SkipNode Struct Reference	53
3.48 SParse Class Reference	54
3.48.1 Member Function Documentation	54
3.48.1.1 SParse_copy()	54
3.48.1.2 SParse_delete()	54
3.48.1.3 SParse_new()	55
3.48.1.4 SParse_parse()	55
3.49 String Struct Reference	55
3.49.1 Detailed Description	56
3.49.2 Member Function Documentation	56
3.49.2.1 String_compare()	56
3.49.2.2 String_copy()	56
3.49.2.3 String_getRef()	57
3.50 stub_data Struct Reference	57
3.51 Task Struct Reference	57
3.51.1 Member Function Documentation	58
3.51.1.1 Task_create()	58
3.51.1.2 Task_executeBody()	58
3.51.1.3 Task_isCompleted()	58
3.51.1.4 Task_isReady()	59
3.51.1.5 Task_isRunning()	59
3.52 TaskMgr Class Reference	59
3.52.1 Member Function Documentation	60
3.52.1.1 TaskMgr_createWorkerThreads()	60
3.52.1.2 TaskMgr_delete()	60
3.52.1.3 TaskMgr_destroySemaphores()	61
3.52.1.4 TaskMgr_findAvailableTask()	61
3.52.1.5 TaskMgr_getRef()	61
3.52.1.6 TaskMgr_getSize()	61
3.52.1.7 TaskMgr_initSemaphores()	62
3.52.1.8 TaskMgr_signalNotEmpty()	62
3.52.1.9 TaskMgr_signalNotFull()	62
3.52.1.10 TaskMgr_start()	62
3.52.1.11 TaskMgr_stop()	63
3.52.1.12 TaskMgr_waitForThread()	63
3.52.1.13 TaskMgr_waitNotEmpty()	63
3.52.1.14 TaskMgr_waitNotFull()	63
3.53 TestClass Struct Reference	64
3.54 TestFileMgr Struct Reference	64
3.55 TestObject Struct Reference	64

3.56 testOptionMgr Struct Reference	64
3.57 TestSdbMgr Struct Reference	65
3.58 TestTimeMgr Struct Reference	65
3.59 TimeMgr Class Reference	65
3.59.1 Member Function Documentation	66
3.59.1.1 TimeMgr_copy()	66
3.59.1.2 TimeMgr_delete()	66
3.59.1.3 TimeMgr_getRef()	66
3.59.1.4 TimeMgr_getSize()	66
3.59.1.5 TimeMgr_latchTime()	66
3.60 Timer Class Reference	67
3.60.1 Member Function Documentation	67
3.60.1.1 Timer_copy()	67
3.60.1.2 Timer_new()	68
3.61 TransUnit Class Reference	68
3.61.1 Member Function Documentation	69
3.61.1.1 TransUnit_getName()	69
3.61.1.2 TransUnit_getNextBuffer()	69
3.61.1.3 TransUnit_getSize()	69
3.61.1.4 TransUnit_new()	69
3.62 yy_buffer_state Struct Reference	70
3.62.1 Member Data Documentation	70
3.62.1.1 yy_bs_column	70
3.62.1.2 yy_bs_lineno	70
3.63 yy_trans_info Struct Reference	70
3.64 yyallocc Union Reference	71
3.65 yyguts_t Struct Reference	71
3.65.1 Member Data Documentation	71
3.65.1.1 yy_buffer_stack	71
3.65.1.2 yy_buffer_stack_max	72
3.65.1.3 yy_buffer_stack_top	72
3.66 YYSTYPE Union Reference	72
4 File Documentation	73
4.1 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/FileDesc.c File Reference	73
4.1.1 Detailed Description	73
4.2 FileDesc.h	74
4.3 FileDesc.h	74
4.4 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/FileMgr.c File Reference	74
4.4.1 Detailed Description	75
4.5 FileMgr.h	75
4.6 FileMgr.h	76

4.7 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/OptionMgr/OptionMgr.c File Reference	76
4.7.1 Detailed Description	76
4.8 OptionMgr.h	77
4.9 OptionMgr.h	77
4.10 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/SdbMgr/SdbMgr.c File Reference	77
4.10.1 Detailed Description	78
4.11 SdbMgr.h	78
4.12 SdbMgr.h	78
4.13 SdbRequest.h	79
4.14 SdbRequest.h	79
4.15 Storage.h	79
4.16 Mutex.h	80
4.17 Task.h	80
4.18 Task.h	81
4.19 TaskMgr.h	81
4.20 TaskMgr.h	81
4.21 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/TimeMgr/TimeMgr.c File Reference	82
4.21.1 Detailed Description	82
4.22 TimeMgr.h	82
4.23 TimeMgr.h	83
4.24 Timer.h	83
4.25 Timer.h	83
4.26 Allocator.h	83
4.27 Allocator.h	84
4.28 Malloc.h	84
4.29 Malloc.h	84
4.30 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Array/Array.c File Reference	85
4.30.1 Detailed Description	85
4.31 Array.h	85
4.32 Array.h	86
4.33 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/BTree.c File Reference	86
4.33.1 Detailed Description	87
4.34 BTree.h	87
4.35 BTree.h	87
4.36 Node.h	88
4.37 Node.h	88
4.38 TestObject.h	88
4.39 TestObject.h	89
4.40 TestObject.h	89
4.41 TestObject.h	89
4.42 Words1000.h	90
4.43 Words1000.h	91

4.44 Words1000.h	92
4.45 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Debug/Debug.c File Reference . . .	94
4.45.1 Detailed Description	94
4.46 Debug.h	94
4.47 Debug.h	95
4.48 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Error/Error.c File Reference	95
4.48.1 Detailed Description	95
4.48.2 Function Documentation	95
4.48.2.1 Error_new()	95
4.49 Error.h	96
4.50 Error.h	96
4.51 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Filelo/Filelo.c File Reference	96
4.51.1 Detailed Description	97
4.52 Filelo.h	97
4.53 Filelo.h	98
4.54 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/List.c File Reference	98
4.54.1 Detailed Description	99
4.55 List.h	99
4.56 List.h	99
4.57 ListNode.h	100
4.58 MyAllocator.h	101
4.59 MyAllocator.h	101
4.60 MyAllocator.h	102
4.61 MyAllocator.h	102
4.62 MyAllocator.h	102
4.63 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/Map.c File Reference	102
4.63.1 Detailed Description	103
4.64 Map.h	103
4.65 Map.h	104
4.66 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/MapEntry.c File Reference . . .	104
4.66.1 Detailed Description	104
4.67 MapEntry.h	105
4.68 MapEntry.h	105
4.69 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Memory/Memory.c File Reference . .	105
4.69.1 Detailed Description	106
4.70 Memory.h	106
4.71 Memory.h	106
4.72 Class.h	107
4.73 Class.h	107
4.74 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/Object.c File Reference . . .	107
4.74.1 Detailed Description	108
4.75 Object.h	108

4.76 Object.h	108
4.77 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectMgr/ObjectMgr.c File Reference	109
4.77.1 Detailed Description	110
4.78 ObjectMgr.h	110
4.79 ObjectMgr.h	110
4.80 ObjectStore.h	110
4.81 ObjectStore.h	111
4.82 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Pool/Pool.c File Reference	111
4.82.1 Detailed Description	112
4.82.2 Function Documentation	113
4.82.2.1 Pool_alloc()	113
4.82.2.2 Pool_allocInFile()	113
4.82.2.3 Pool_dealloc()	113
4.82.2.4 Pool_deallocInFile()	114
4.82.2.5 Pool_deallocInMemory()	114
4.82.2.6 Pool_delete()	114
4.82.2.7 Pool_new()	115
4.82.2.8 Pool_newFromFile()	115
4.82.2.9 Pool_read()	115
4.82.2.10 Pool_readInFile()	116
4.82.2.11 Pool_readInMemory()	116
4.82.2.12 Pool_report()	116
4.82.2.13 Pool_reportInFile()	117
4.82.2.14 Pool_reportInMemory()	117
4.82.2.15 Pool_reportNbNodes()	117
4.82.2.16 Pool_reportSizeInBytes()	118
4.82.2.17 Pool_write()	118
4.82.2.18 Pool_writeInFile()	118
4.82.2.19 Pool_writeInMemory()	119
4.83 Pool.h	119
4.84 Pool.h	119
4.85 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/SkipList.c File Reference	120
4.85.1 Detailed Description	121
4.86 SkipList.h	121
4.87 SkipList.h	121
4.88 SkipNode.h	121
4.89 String2.h	123
4.90 String2.h	123
4.91 Times.h	124
4.92 Times.h	124
4.93 Types.h	124
4.94 Types.h	124

4.95 UserTypes.h	125
4.96 UserTypes.h	125
4.97 /home/thomas/Projects/SParse-master/SParse/src/main.c File Reference	125
4.97.1 Detailed Description	126
4.97.2 Function Documentation	126
4.97.2.1 main()	126
4.97.2.2 print_usage()	126
4.97.2.3 sighandler()	127
4.98 Ast.h	128
4.99 Declarator.h	128
4.100 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/Configuration/Configuration.c File Reference	128
4.100.1 Detailed Description	129
4.100.2 Macro Definition Documentation	129
4.100.2.1 IS_KEY	129
4.100.2.2 IS_STRING	129
4.101 Configuration.h	130
4.102 Configuration.h	130
4.103 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/Configuration/Product.c File Reference	130
4.103.1 Detailed Description	131
4.104 Product.h	131
4.105 Product.h	131
4.106 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/FileReader/FileReader.c File Reference	132
4.106.1 Detailed Description	132
4.106.2 Variable Documentation	132
4.106.2.1 includeInfoClass	132
4.107 FileReader.h	133
4.108 FileReader.h	133
4.109 Grammar.h	133
4.110 Grammar.h	133
4.111 Grammar2.h	134
4.112 Grammar2.h	134
4.113 Grammar2.parse.h	134
4.114 GrammarC99.h	136
4.115 GrammarC99.h	136
4.116 GrammarC99.parse.h	136
4.117 MyType.h	138
4.118 HTTPRequest.h	138
4.119 HTTPResponse.h	140
4.120 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPServer.c File Reference	142
4.120.1 Detailed Description	143
4.121 HTTPServer.h	143

4.122 HTTPServer.h	143
4.123 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/SParse/SParse.c File Reference . . .	144
4.123.1 Detailed Description	144
4.124 SParse.h	144
4.125 SParse.h	145
4.126 Buffer.h	145
4.127 MacroDefinition.h	146
4.128 MacroStore.h	147
4.129 test.h	150
4.130 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/TransUnit.c File Reference .	150
4.130.1 Detailed Description	151
4.131 TransUnit.h	151
4.132 TransUnit.h	151
Index	153

Chapter 1

Class Index

1.1 Class List

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

Allocator	7
AllocInfo	7
Array	8
ArrayParam	9
BTree	9
Buffer	10
Class	10
Configuration	10
ConnectionParam	11
Declarator	11
FileDesc	12
Fileilo	12
Filelo	12
FileMgr	15
FileReader	19
Grammar	21
Grammar2	21
GrammarC99	22
GrammarContext	23
HTTPRequest	23
HTTPResponse	25
HTTPServer	25
IncludeInfo	27
List	27
MacroDefinition	31
MacroStore	32
MacroStoreNode	32
Malloc	32
Map	33
MapEntry	35
mem_align	35
Mutex	36
MyAllocator	36
MyType	36
Node	37

Object	37
ObjectInfo	39
ObjectMgr	40
ObjectStore	42
OptionDefault	44
OptionMgr	45
PoolCache	46
Product	46
SdbMgr	46
SdbRequest	48
SkipList	49
SkipNode	53
SParse	54
String	55
stub_data	57
Task	57
TaskMgr	59
TestClass	64
TestFileMgr	64
TestObject	64
testOptionMgr	64
TestSdbMgr	65
TestTimeMgr	65
TimeMgr	65
Timer	67
TransUnit	68
yy_buffer_state	70
yy_trans_info	70
yyalloc	71
yyguts_t	71
YYSTYPE	72

Chapter 2

File Index

2.1 File List

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

/home/thomas/Projects/SParse-master/SParse/src/ main.c	
Contains the main() function	125
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/ FileDesc.c	
The FileDesc class describe a File in the FileMgr	73
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/ FileDesc.h	74
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/ FileMgr.c	
The FileMgr class manages a list of files contained in a group of locations	74
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/ FileMgr.h	75
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/ FileDesc.h	74
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/ FileMgr.h	76
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/ OptionMgr.h	77
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/ SdbMgr.h	78
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/ SdbRequest.h	79
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/ Task.h	80
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/ TaskMgr.h	81
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/ TimeMgr.h	82
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/ Timer.h	83
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/OptionMgr/ OptionMgr.c	
The OptionMgr class manages the application configuration	76
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/OptionMgr/ OptionMgr.h	77
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/SdbMgr/ SdbMgr.c	
TBD	77
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/SdbMgr/ SdbMgr.h	78
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/SdbMgr/ SdbRequest.h	79
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/Storage/ Storage.h	79
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/TaskMgr/ Mutex.h	80
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/TaskMgr/ Task.h	81
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/TaskMgr/ TaskMgr.h	81
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/TimeMgr/ TimeMgr.c	
This file contains the implementation for the class TimeMgr	82
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/TimeMgr/ TimeMgr.h	83
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/TimeMgr/ Timer.h	83
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Allocator/ Allocator.h	83
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Allocator/ Malloc.h	84
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Array/ Array.c	
This file contains the implementation of the class Array	85

/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Array/Array.h	85
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/BTree.c	
This file contains the implementation of the class BTree	86
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/BTree.h	87
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/Node.h	88
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/tests/TestObject.h	88
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/tests/Words1000.h	90
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Debug/Debug.c	
This file contains debugging functions	94
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Debug/Debug.h	94
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Error/Error.c	
Reports errors	95
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Error/Error.h	96
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/FileIo/FileIo.c	
A FileIo class. This class provides a status and operation for various File I/O operations	96
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/FileIo/FileIo.h	97
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Allocator.h	84
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Array.h	86
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/BTree.h	87
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Class.h	107
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Debug.h	95
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Error.h	96
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/FileIo.h	98
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/List.h	99
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Malloc.h	84
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Map.h	103
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/MapEntry.h	105
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Memory.h	106
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Node.h	88
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Object.h	108
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ObjectMgr.h	110
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ObjectStore.h	110
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Pool.h	119
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/SkipList.h	121
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/String2.h	123
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Times.h	124
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Types.h	124
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/UserTypes.h	125
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/List.c	
This file contains the implementation of the class List	98
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/List.h	99
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/ListNode.h	100
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/tests/MyAllocator.h	101
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/tests/TestObject.h	89
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/Map.c	
A Map class. This class provides a container indexed by a string	102
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/Map.h	104
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/MapEntry.c	
A support class for the Map class	104
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/MapEntry.h	105
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/tests/MyAllocator.h	101
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/tests/TestObject.h	89
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/tests/Words1000.h	91
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Memory/Memory.c	
This file provides the implementation of the memory functions	105
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Memory/Memory.h	106
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/Class.h	107

/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/Object.c	
This file contains the implementation for the class Object	107
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/Object.h	108
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectMgr/ObjectMgr.c	
An object management class	109
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectMgr/ObjectMgr.h	110
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectStore/ObjectStore.h	111
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectStore/tests/MyAllocator.h	102
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Pool/Pool.c	
This file contains the implementation of the class Pool	111
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Pool/Pool.h	119
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/SkipList.c	
This file contains the implementation of the class SkipList . The class List implement the SkipList operations	120
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/SkipList.h	121
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/SkipNode.h	121
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/tests/MyAllocator.h	102
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/tests/TestObject.h	89
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/tests/Words1000.h	92
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/String/String2.h	123
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/String/tests/MyAllocator.h	102
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Types/Types.h	124
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Types/UserTypes.h	125
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Ast/Ast.h	128
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/C89Grammar/Declarator.h	128
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Configuration/Configuration.c	
This file contains the implementation for the class Configuration . The class Configuration lists all the SW products to parse including the path to the source files, any dependency	128
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Configuration/Configuration.h	130
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Configuration/Product.c	
This file contains the implementation for the class Product	130
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Configuration/Product.h	131
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/FileReader/FileReader.c	
This file contains the implementation for the class FileReader	132
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/FileReader/FileReader.h	133
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar/Grammar.h	133
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar2/Grammar2.h	134
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar2/Grammar2.parse.h	134
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/GrammarC99/GrammarC99.h	136
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/GrammarC99/GrammarC99.parse.h	136
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/GrammarC99/MyType.h	138
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPRequest.h	138
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPResponse.h	140
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPServer.c	
A HTTP Server class. This class provides server function to create, start HTML pages	142
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPServer.h	143
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/Configuration.h	130
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/FileReader.h	133
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/Grammar.h	133
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/Grammar2.h	134
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/GrammarC99.h	136
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/HTTPServer.h	143
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/Product.h	131
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/SParse.h	144
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/TransUnit.h	151
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/SParse/SParse.c	
This file contains the implementation for the class SParse	144

/home/thomas/Projects/SParse-master/SParse/src/ParseLib/SParse/SParse.h	145
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/Buffer.h	145
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/MacroDefinition.h	146
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/MacroStore.h	147
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/TransUnit.c	
This file implements a class that extract C code	150
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/TransUnit.h	151
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/tests/test.h	150

Chapter 3

Class Documentation

3.1 Allocator Struct Reference

Public Attributes

- NewFunction **new**
- DeleteFunction **delete**
- AllocateFunction **allocate**
- DeAllocateFunction **deallocate**
- ReportFunction **report**
- unsigned int **nbAllocatedObjects**

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Allocator/Allocator.h
- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Allocator.h

3.2 AllocInfo Struct Reference

Public Attributes

- [Allocator](#) * **ptr**
- [AllocInfo](#) * **prev**
- [AllocInfo](#) * **next**

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectStore/ObjectStore.c

3.3 Array Class Reference

Public Member Functions

- PUBLIC [Array](#) * [Array_new](#) ([ArrayParam](#) *param)
Create a new instance of the class [Array](#).
- PUBLIC [Array](#) * [Array_newFromFile](#) ([FileIo](#) *fileIo, [ArrayParam](#) *param)
Create a new instance of the class [Array](#) from a fileIo stream.
- PUBLIC void [Array_delete](#) ([Array](#) *this)
Delete an instance of the class [Array](#).
- PUBLIC [Array](#) * [Array_copy](#) ([Array](#) *this)
Copy an instance of the class [Array](#).
- PUBLIC int [Array_compare](#) ([Array](#) *this, [Array](#) *compared)
Compare 2 instances of the class [Array](#).
- PUBLIC void [Array_print](#) ([Array](#) *this)
Print an instance of the class [Array](#).

Public Attributes

- [Object](#) object
- unsigned int nbElements

3.3.1 Member Function Documentation

3.3.1.1 [Array_compare\(\)](#)

```
PUBLIC int Array\_compare (  
    Array * this,  
    Array * compared )
```

Compare 2 instances of the class [Array](#).

Returns

0 if different, 1 if equal.

3.3.1.2 [Array_copy\(\)](#)

```
PUBLIC Array * Array\_copy (  
    Array * this )
```

Copy an instance of the class [Array](#).

Returns

Copy of the given instance.

3.3.1.3 Array_new()

```
PUBLIC Array * Array_new (
    ArrayParam * param )
```

Create a new instance of the class [Array](#).

Returns

New instance.

3.3.1.4 Array_newFromFile()

```
PUBLIC Array * Array_newFromFile (
    FileIo * fileIo,
    ArrayParam * param )
```

Create a new instance of the class [Array](#) from a fileIo stream.

Returns

New instance.

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

- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Array/Array.c](#)

3.4 ArrayParam Struct Reference

Public Attributes

- unsigned int **defaultSize**
- unsigned int **storageMode**
- unsigned int **autoresize**

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

- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Array/Array.h](#)
- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Array.h](#)

3.5 BTree Class Reference

Public Attributes

- [Object](#) **object**
- [Node](#) * **root**
- unsigned int **order**
- unsigned int **depth**
- unsigned short int **nbObjects**
- unsigned int **nodeSize**

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

- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/BTree.c](#)
- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/BTree.rescue.c](#)

3.6 Buffer Struct Reference

Public Attributes

- [Object](#) **object**
- [String](#) * **string**
- char * **currentPtr**
- char * **startPtr**
- int **nbCharRead**

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/Buffer.h

3.7 Class Struct Reference

Public Attributes

- Constructor **f_new**
- Destructor **f_delete**
- Copy_Operator **f_copy**
- Comp_Operator **f_comp**
- Printer **f_print**
- Sizer **f_size**

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Class.h
- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/Class.h

3.8 Configuration Class Reference

Public Member Functions

- PUBLIC [Configuration](#) * **Configuration_new** ([String](#) *input)
Create an instance of configuration class from th input string.
- PUBLIC void **Configuration_delete** ([Configuration](#) *this)
Destroy an instance of configuration class.
- PUBLIC void **Configuration_print** ([Configuration](#) *this)
Print an instance of configuration class.
- PUBLIC unsigned int **Configuration_getSize** ([Configuration](#) *this)
Destroy an instance of configuration class.
- PUBLIC [List](#) * **Configuration_getProducts** ([Configuration](#) *this)
TBD.
- PUBLIC void **Configuration_parseProducts** ([Configuration](#) *this)
TBD.

Public Attributes

- [Object](#) **object**
- [List](#) * **products**

3.8.1 Member Function Documentation

3.8.1.1 Configuration_new()

```
PUBLIC Configuration * Configuration_new (  
    String * input )
```

Create an instance of configuration class from th input string.

Returns

Status.

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/Configuration/[Configuration.c](#)

3.9 ConnectionParam Struct Reference

Public Attributes

- int * **client_fd**

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/[HTTPServer.c](#)

3.10 Declarator Struct Reference

Public Attributes

- DeclaratorType **type**
- DeclaratorScope **scope**
- char * **name**

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/C89Grammar/Declarator.c

3.11 FileDesc Class Reference

Public Member Functions

- PUBLIC [FileDesc](#) * **FileDesc_new** ()
TBD.
- PUBLIC void **FileDesc_delete** ([FileDesc](#) *this)
TBD.
- PUBLIC [FileDesc](#) * **FileDesc_copy** ([FileDesc](#) *this)
TBD.
- PUBLIC void **FileDesc_setFullName** ([FileDesc](#) *this, [String](#) *fullName)
TBD.
- PUBLIC [String](#) * **FileDesc_getFullName** ([FileDesc](#) *this)
TBD.
- PUBLIC [String](#) * **FileDesc_getName** ([FileDesc](#) *this)
TBD.
- PUBLIC [String](#) * **FileDesc_load** ([FileDesc](#) *this)
Load the content of a file.

Public Attributes

- [Object](#) **object**
- [String](#) * **name**
- [String](#) * **fullName**

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

- /home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/[FileDesc.c](#)

3.12 Fileilo Class Reference

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Filelo/[Filelo.c](#)

3.13 Filelo Struct Reference

Public Member Functions

- PUBLIC [Filelo](#) * **Filelo_new** ()
Create a new instance of the class [Filelo](#).
- PUBLIC void **Filelo_delete** ([Filelo](#) *this)
Delte an instance of the class [Filelo](#).
- PUBLIC [Filelo](#) * **Filelo_copy** ([Filelo](#) *this)
Copy an instance of the class [Filelo](#).
- PUBLIC int **Filelo_comp** ([Filelo](#) *this, [Filelo](#) *compare)

Compare an instance of the class [FileIo](#) to another one.

- PUBLIC void [FileIo_print](#) ([FileIo](#) *this)

Print an instance of the class [FileIo](#).

- PUBLIC unsigned int [FileIo_getSize](#) ([FileIo](#) *this)

Return the size of an instance of the class [FileIo](#).

- PUBLIC void [FileIo_openFile](#) ([FileIo](#) *this, [String](#) *fullFileName)

Open an instance of the class [FileIo](#) for reading/writing.

- PUBLIC void [FileIo_createFile](#) ([FileIo](#) *this, [String](#) *fullFileName)

Create a new file.

- PUBLIC void [FileIo_openDir](#) ([FileIo](#) *this, [String](#) *fullFileName)

Create a new file.

Public Attributes

- [Object](#) object
- FILE * f
- int status

3.13.1 Member Function Documentation

3.13.1.1 FileIo_comp()

```
PUBLIC int FileIo_comp (
    FileIo * this,
    FileIo * compare )
```

Compare an instance of the class [FileIo](#) to another one.

Returns

0 if equal.

3.13.1.2 FileIo_copy()

```
PUBLIC FileIo * FileIo_copy (
    FileIo * this )
```

Copy an instance of the class [FileIo](#).

Returns

Copy of the instance.

3.13.1.3 FileIo_createFile()

```
PUBLIC void FileIo_createFile (
    FileIo * this,
    String * fullFileName )
```

Create a new file.

Parameters

in	<i>String</i>	Full path of file to create
----	---------------	-----------------------------

Returns

none

3.13.1.4 FileIo_delete()

```
PUBLIC void FileIo_delete (  
    FileIo * this )
```

Delte an instance of the class *FileIo*.

Returns

none

3.13.1.5 FileIo_getSize()

```
PUBLIC unsigned int FileIo_getSize (  
    FileIo * this )
```

Return the size of an instance of the class *FileIo*.

Returns

Size in bytes.

3.13.1.6 FileIo_new()

```
PUBLIC FileIo * FileIo_new ( )
```

Create a new instance of the class *FileIo*.

Returns

New *FileIo* instance or NULL if failed to allocate.

3.13.1.7 FileIo_openDir()

```
PUBLIC void FileIo_openDir (  
    FileIo * this,  
    String * fullFileName )
```

Create a new file.

Parameters

in	String	Full path of file to create
----	------------------------	-----------------------------

Returns

none

3.13.1.8 FileIo_openFile()

```
PUBLIC void FileIo_openFile (  
    FileIo * this,  
    String * fullFileName )
```

Open an instance of the class [FileIo](#) for reading/writing.

Parameters

in	String	Full path of file to open
----	------------------------	---------------------------

Returns

none

3.13.1.9 FileIo_print()

```
PUBLIC void FileIo_print (  
    FileIo * this )
```

Print an instance of the class [FileIo](#).

Returns

none.

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

- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/FileIo/FileIo.c](#)

3.14 FileMgr Class Reference

Public Member Functions

- PUBLIC void **FileMgr_delete** ([FileMgr](#) **this*)
Delete an instance of the class [FileMgr](#).
- PUBLIC [FileMgr](#) * **FileMgr_copy** ([FileMgr](#) **this*)

- Copy an instance of the class [FileMgr](#).*

 - PUBLIC [FileMgr](#) * [FileMgr_getRef](#) ()

Get a reference to the singleton instance of [FileMgr](#).
- PUBLIC unsigned int [FileMgr_getSize](#) ([FileMgr](#) *this)

Return the size in byte of the class or object.
- PUBLIC unsigned int [FileMgr_setRootLocation](#) ([FileMgr](#) *this, const char *location)

Set the root location.
- PUBLIC char * [FileMgr_getRootLocation](#) ([FileMgr](#) *this)

TBD.
- PUBLIC unsigned int [FileMgr_addDirectory](#) ([FileMgr](#) *this, const char *directoryName)

Add all files in the given directory to the list of managed files.
- PUBLIC [FileDesc](#) * [FileMgr_addFile](#) ([FileMgr](#) *this, const char *fileName)

Add a files to the list of managed files.
- PUBLIC [String](#) * [FileMgr_load](#) ([FileMgr](#) *this, const char *fileName)

Load a managed file into a [String](#).

@parameter File Name.
- PUBLIC void [FileMgr_write](#) ([FileMgr](#) *this, const char *fileName, [String](#) *content)

Write a string into a file.
- PUBLIC [List](#) * [FileMgr_filterFiles](#) ([FileMgr](#) *this, const char *pattern)

TBD.

Public Attributes

- [Object](#) object
- [List](#) * files
- [List](#) * directories
- char * separator
- [String](#) * rootLocation

3.14.1 Member Function Documentation

3.14.1.1 FileMgr_addDirectory()

```
PUBLIC unsigned int FileMgr_addDirectory (
    FileMgr * this,
    const char * directoryName )
```

Add all files in the given directory to the list of managed files.

Returns

Status.

3.14.1.2 FileMgr_addFile()

```
PUBLIC FileDesc * FileMgr_addFile (
    FileMgr * this,
    const char * fileName )
```

Add a files to the list of managed files.

Returns

Status.

3.14.1.3 FileMgr_copy()

```
PUBLIC FileMgr * FileMgr_copy (
    FileMgr * this )
```

Copy an instance of the class [FileMgr](#).

Returns

New instance

3.14.1.4 FileMgr_filterFiles()

```
PUBLIC List * FileMgr_filterFiles (
    FileMgr * this,
    const char * pattern )
```

TBD.

Returns

TBD

3.14.1.5 FileMgr_getRef()

```
PUBLIC FileMgr * FileMgr_getRef ( )
```

Get a reference to the singleton instance of [FileMgr](#).

Returns

Reference to the singleton.

3.14.1.6 FileMgr_getRootLocation()

```
PUBLIC char * FileMgr_getRootLocation (
    FileMgr * this )
```

TBD.

Returns

Status.

3.14.1.7 FileMgr_getSize()

```
PUBLIC unsigned int FileMgr_getSize (  
    FileMgr * this )
```

Return the size in byte of the class or object.

Returns

Size in byte of [Class](#) or instance.

3.14.1.8 FileMgr_load()

```
PUBLIC String * FileMgr_load (  
    FileMgr * this,  
    const char * fileName )
```

Load a managed file into a [String](#).

@parameter File Name.

Returns

Content of file.

3.14.1.9 FileMgr_setRootLocation()

```
PUBLIC unsigned int FileMgr_setRootLocation (  
    FileMgr * this,  
    const char * location )
```

Set the root location.

Returns

Status.

3.14.1.10 FileMgr_write()

```
PUBLIC void FileMgr_write (  
    FileMgr * this,  
    const char * fileName,  
    String * content )
```

Write a string into a file.

Returns

None

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

- [/home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/FileMgr.c](#)

3.15 FileReader Class Reference

Public Member Functions

- PUBLIC [FileReader](#) * [FileReader_new](#) ([FileDesc](#) *fileDesc, [FileMgr](#) *fileMgr)
Create a new [FileReader](#) object.
- PUBLIC void [FileReader_delete](#) ([FileReader](#) *this)
Delete an instance of a [FileReader](#) object.
- PUBLIC [FileReader](#) * [FileReader_copy](#) ([FileReader](#) *this)
Copy an instance of a [FileReader](#) object.
- PUBLIC void [FileReader_print](#) ([FileReader](#) *this)
Print an instance of a [FileReader](#) object.
- PUBLIC unsigned int [FileReader_getSize](#) ([FileReader](#) *this)
Return the size in bytes of an instance of a [FileReader](#) object.
- PUBLIC char * [FileReader_getBuffer](#) ([FileReader](#) *this)
Returns the buffer of a [FileReader](#) object.
- PUBLIC [String](#) * [FileReader_getName](#) ([FileReader](#) *this)
Returns the name of a [FileReader](#) object.
- PUBLIC char * [FileReader_addFile](#) ([FileReader](#) *this, [String](#) *fileName)
Add a new file buffer for filename.

Public Attributes

- [Object](#) object
- [List](#) * buffers
- [FileDesc](#) * fileDesc
- [FileMgr](#) * fileMgr
- [String](#) * currentBuffer
- [List](#) * preferredDirs

3.15.1 Member Function Documentation

3.15.1.1 [FileReader_addFile\(\)](#)

```
PUBLIC char * FileReader_addFile (  
    FileReader * this,  
    String * fileName )
```

Add a new file buffer for filename.

Returns

File buffer

3.15.1.2 FileReader_copy()

```
PUBLIC FileReader * FileReader_copy (
    FileReader * this )
```

Copy an instance of a [FileReader](#) object.

Returns

New instance

3.15.1.3 FileReader_getBuffer()

```
PUBLIC char * FileReader_getBuffer (
    FileReader * this )
```

Returns the buffer of a [FileReader](#) object.

Returns

[Buffer](#) of characters

3.15.1.4 FileReader_getName()

```
PUBLIC String * FileReader_getName (
    FileReader * this )
```

Returns the name of a [FileReader](#) object.

Returns

File name

3.15.1.5 FileReader_getSize()

```
PUBLIC unsigned int FileReader_getSize (
    FileReader * this )
```

Return the size in bytes of an instance of a [FileReader](#) object.

Returns

Size in bytes

3.15.1.6 FileReader_new()

```
PUBLIC FileReader * FileReader_new (
    FileDesc * fileDesc,
    FileMgr * fileMgr )
```

Create a new `FileReader` object.

Returns

Created `FileReader` object.

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

- `/home/thomas/Projects/SParse-master/SParse/src/ParseLib/FileReader/FileReader.c`

3.16 Grammar Struct Reference

Public Attributes

- `Object` `object`
- `Grammar` `*(* new)(void)`
- `void(* delete)(Grammar *this)`
- `Grammar` `*(* copy)(Grammar *this)`
- `void(* print)(Grammar *this)`

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

- `/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar/Grammar.h`
- `/home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/Grammar.h`

3.17 Grammar2 Class Reference

Public Member Functions

- PUBLIC `Grammar2` * `Grammar2_new` (`FileReader` *fr, `SdbMgr` *sdbMgr)
Create an instance of the class `Grammar2`.
- PUBLIC void `Grammar2_delete` (`Grammar2` *this)
Delete an instance of the class `Grammar2`.
- PUBLIC `Grammar2` * `Grammar2_copy` (`Grammar2` *this)
Copy an instance of the class `Grammar2`.

Public Attributes

- [Object](#) **object**
- void * **scanner**
- [SdbMgr](#) * **sdbMgr**
- [FileReader](#) * **reader**
- [TransUnit](#) * **unit**
- char * **buffer**
- int **node_text_position**
- [GrammarContext](#) * **current**
- [List](#) * **contexts**

3.17.1 Member Function Documentation

3.17.1.1 Grammar2_copy()

```
PUBLIC Grammar2 * Grammar2_copy (
    Grammar2 * this )
```

Copy an instance of the class [Grammar2](#).

Returns

Copied instance.

3.17.1.2 Grammar2_new()

```
PUBLIC Grammar2 * Grammar2_new (
    FileReader * fr,
    SdbMgr * sdbMgr )
```

Create an instance of the class [Grammar2](#).

Returns

New instance.

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar2/Grammar2.c

3.18 GrammarC99 Class Reference

Public Member Functions

- PUBLIC [Grammar](#) * [GrammarC99_new](#) ([FileDesc](#) *fileDesc, [FileMgr](#) *fm)
Create an instance of the class [GrammarC99](#).
- PUBLIC void [GrammarC99_delete](#) ([Grammar](#) *this)
TBC.
- PUBLIC void [GrammarC99_print](#) ([Grammar](#) *this)
TBC.
- PUBLIC unsigned int [GrammarC99_getSize](#) ([Grammar](#) *this)
TBC.
- PUBLIC void [GrammarC99_process](#) ([GrammarC99](#) *this)
TBC.

Public Attributes

- [Grammar](#) **grammar**
- [TransUnit](#) * **transUnit**
- [FileMgr](#) * **fm**
- void * **scanner**

3.18.1 Member Function Documentation**3.18.1.1 GrammarC99_new()**

```
PUBLIC Grammar * GrammarC99_new (
    FileDesc * fileDesc,
    FileMgr * fm )
```

Create an instance of the class [GrammarC99](#).

Returns

New instance.

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/GrammarC99/GrammarC99.c

3.19 GrammarContext Struct Reference**Public Attributes**

- [Object](#) **object**
- unsigned int **lastNode**
- unsigned int **includeNodeBranch**

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar2/Grammar2.c

3.20 HTTPRequest Class Reference**Public Member Functions**

- PRIVATE void **HTTPRequest_delete** ([HTTPRequest](#) *this)
Delete an instance of the class [HTTPRequest](#).
- PRIVATE [HTTPRequest](#) * **HTTPRequest_copy** ([HTTPRequest](#) *this)
Copy an instance of the class [HTTPRequest](#).
- PRIVATE int **HTTPRequest_compare** ([HTTPRequest](#) *this, [HTTPRequest](#) *compared)
Compare 2 instances of the class [HTTPRequest](#).
- PRIVATE void **HTTPRequest_print** ([HTTPRequest](#) *this)
Print an instance of the class [HTTPRequest](#).
- PRIVATE unsigned int **HTTPRequest_getSize** ([HTTPRequest](#) *this)
Get the size of an [HTTPRequest](#). If parameter is 0 return the size of the class.

Public Attributes

- [Object](#) **object**
- enum Method **method**
- [String](#) * **path**
- int **majorVersion**
- int **minorVersion**
- [Map](#) * **headers**
- [String](#) * **body**
- int **isValid**

3.20.1 Member Function Documentation

3.20.1.1 HTTPRequest_compare()

```
PRIVATE int HTTPRequest_compare (
    HTTPRequest * this,
    HTTPRequest * compared )
```

Compare 2 instances of the class [HTTPRequest](#).

Returns

0 if different, 1 if equal.

3.20.1.2 HTTPRequest_copy()

```
PRIVATE HTTPRequest * HTTPRequest_copy (
    HTTPRequest * this )
```

Copy an instance of the class [HTTPRequest](#).

Returns

Copy of the instance

3.20.1.3 HTTPRequest_getSize()

```
PRIVATE unsigned int HTTPRequest_getSize (
    HTTPRequest * this )
```

Get the size of an [HTTPRequest](#). If parameter is 0 return the size of the class.

Returns

Number of items.

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPRequest.h

3.21 HTTPResponse Class Reference

Public Attributes

- [Object](#) **object**
- int **statusCode**
- enum Reason **reason**
- int **majorVersion**
- int **minorVersion**
- [Map](#) * **headers**
- [String](#) * **body**
- int **isValid**

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPResponse.h

3.22 HTTPServer Class Reference

Public Member Functions

- PRIVATE [HTTPRequest](#) * [HTTPRequest_new](#) (char *buffer)
Create a new instance of the class [HTTPRequest](#).
- PRIVATE int [HTTPResponse_compare](#) ([HTTPResponse](#) *this, [HTTPResponse](#) *compared)
Compare 2 instances of the class [HTTPResponse](#).
- PRIVATE void [HTTPResponse_print](#) ([HTTPResponse](#) *this)
Print an instance of the class [HTTPResponse](#).
- PRIVATE unsigned int [HTTPResponse_getSize](#) ([HTTPResponse](#) *this)
Get the size of an [HTTPResponse](#). If parameter is 0 return the size of the class.
- PUBLIC [HTTPServer](#) * [HTTPServer_new](#) ()
Create a new instance of the class [HTTPServer](#).

Public Attributes

- [Object](#) **object**
- int **port**
- struct sockaddr_in **server_addr**
- int **fd**

3.22.1 Member Function Documentation

3.22.1.1 HTTPRequest_new()

```
PRIVATE HTTPRequest * HTTPRequest_new (  
    char * buffer )
```

Create a new instance of the class [HTTPRequest](#).

Parameters

in	<i>none</i>	
----	-------------	--

Returns

New instance of class [HTTPRequest](#).

3.22.1.2 HTTPResponse_compare()

```
PRIVATE int HTTPResponse_compare (
    HTTPResponse * this,
    HTTPResponse * compared )
```

Compare 2 instances of the class [HTTPResponse](#).

Returns

0 if different, 1 if equal.

3.22.1.3 HTTPResponse_getSize()

```
PRIVATE unsigned int HTTPResponse_getSize (
    HTTPResponse * this )
```

Get the size of an [HTTPResponse](#). If parameter is 0 return the size of the class.

Returns

Number of items.

3.22.1.4 HTTPServer_new()

```
PUBLIC HTTPServer * HTTPServer_new ( )
```

Create a new instance of the class [HTTPServer](#).

Parameters

in	<i>none</i>	
----	-------------	--

Returns

New instance of class [HTTPServer](#).

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

- [/home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPServer.c](#)
- [/home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPRequest.h](#)
- [/home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPResponse.h](#)

3.23 IncludeInfo Struct Reference

Public Attributes

- [Object](#) **object**
- [String](#) * **pattern**
- [List](#) * **dirs**

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

- [/home/thomas/Projects/SParse-master/SParse/src/ParseLib/FileReader/FileReader.c](#)

3.24 List Class Reference

Public Member Functions

- PUBLIC [List](#) * **List_new** ()
Create a new instance of the class [List](#).
- PUBLIC [List](#) * **List_newFromAllocator** ([Allocator](#) *allocator)
Create a new instance of the class [List](#) using a custom allocator.
- PUBLIC void **List_delete** ([List](#) *this)
Delete an instance of the class [List](#).
- PUBLIC [List](#) * **List_copy** ([List](#) *this)
Copy an instance of the class [List](#).
- PUBLIC int **List_compare** ([List](#) *this, [List](#) *compared)
Compare 2 instances of the class [List](#).
- PUBLIC void **List_print** ([List](#) *this)
Print an instance of the class [List](#).
- PUBLIC void **List_insertHead** ([List](#) *this, void *item, int isOwner)
Insert an item at the head of a list instance.
- PUBLIC void **List_insertTail** ([List](#) *this, void *item, int isOwner)
Insert an item at the tail of a [List](#) instance.
- PUBLIC void **List_merge** ([List](#) *this, [List](#) *l1)
Merge a list into a [List](#) instance.
- PUBLIC void **List_forEach** ([List](#) *this, void(*method)(void *o))
Execute a given function for each item in an instance of [List](#).
- PUBLIC unsigned int **List_getNbNodes** ([List](#) *this)
Get the number of items in [List](#) instance.
- PUBLIC unsigned int **List_getSize** ([List](#) *this)
Get the size of a [List](#) object. If parameter is 0 return the size of the class.
- PUBLIC void * **List_removeHead** ([List](#) *this)
Remove the head item in an instance of [List](#)
- PUBLIC void * **List_removeTail** ([List](#) *this)

- Remove the tail item in an instance of [List](#)*
- PUBLIC void * **List_getHead** ([List](#) *this)
- Get the head item in an insatnce of [List](#)*
- PRIVATE ListNode * **ListNode_new** ([Object](#) *object, int isOwner)
- Create a new instance of the class [ListNode](#).*
- PRIVATE ListNode * **ListNode_newFromAllocator** ([Allocator](#) *allocator, [Object](#) *object, int isOwner)
- Create a new instance of the class [List](#) using a custom allocator.*
- PRIVATE void **ListNode_delete** (ListNode *this)
- Delete an instance of the class [List](#).*
- PRIVATE ListNode * **ListNode_copy** (ListNode *this)
- Copy an instance of the class [List](#).*
- PRIVATE int **ListNode_compare** (ListNode *this, ListNode *compared)
- Compare 2 instances of the class [List](#).*
- PRIVATE void **ListNode_print** (ListNode *this)
- Print an instance of the class [List](#).*

Public Attributes

- [Object](#) **object**
- ListNode * **head**
- ListNode * **tail**
- ListNode * **iterator**
- unsigned int **nbNodes**

3.24.1 Member Function Documentation

3.24.1.1 List_compare()

```
PUBLIC int List_compare (
    List * this,
    List * compared )
```

Compare 2 instances of the class [List](#).

Returns

0 if different, 1 if equal.

3.24.1.2 List_copy()

```
PUBLIC List * List_copy (
    List * this )
```

Copy an instance of the class [List](#).

Returns

Copy of the given instance.

3.24.1.3 List_forEach()

```
PUBLIC void List_forEach (
    List * this,
    void(*) (void *o) method )
```

Execute a given function for each item in an instance of [List](#)..

Parameters

in	<i>f</i>	Pointer to function.
----	----------	----------------------

3.24.1.4 List_getNbNodes()

```
PUBLIC unsigned int List_getNbNodes (  
    List * this )
```

Get the number of items in [List](#) instance.

Returns

Number of items.

3.24.1.5 List_getSize()

```
PUBLIC unsigned int List_getSize (  
    List * this )
```

Get the size of a [List](#) object. If parameter is 0 return the size of the class.

Returns

Number of items.

3.24.1.6 List_insertHead()

```
PUBLIC void List_insertHead (  
    List * this,  
    void * item,  
    int isOwner )
```

Insert an item at the head of a list instance.

Parameters

in	<i>item</i>	Reference to item.
----	-------------	--------------------

3.24.1.7 List_insertTail()

```
PUBLIC void List_insertTail (  
    List * this,  
    void * item,  
    int isOwner )
```

Insert an item at the tail of a [List](#) instance.

Parameters

in	<i>item</i>	Reference to item.
----	-------------	--------------------

3.24.1.8 List_merge()

```
PUBLIC void List_merge (  
    List * this,  
    List * ll )
```

Merge a list into a [List](#) instance.

Parameters

in	<i>ll</i>	Reference to list to merge.
----	-----------	-----------------------------

3.24.1.9 List_new()

```
PUBLIC List * List_new ( )
```

Create a new instance of the class [List](#).

Returns

New instance.

3.24.1.10 List_newFromAllocator()

```
PUBLIC List * List_newFromAllocator (  
    Allocator * allocator )
```

Create a new instance of the class [List](#) using a custom allocator.

Returns

New instance.

3.24.1.11 ListNode_compare()

```
PRIVATE int ListNode_compare (  
    ListNode * this,  
    ListNode * compared )
```

Compare 2 instances of the class [List](#).

Returns

0 if different, 1 if equal.

3.24.1.12 ListNode_copy()

```
PRIVATE ListNode * ListNode_copy (
    ListNode * this )
```

Copy an instance of the class [List](#).

Returns

Copy of the given instance.

3.24.1.13 ListNode_new()

```
PRIVATE ListNode * ListNode_new (
    Object * object,
    int isOwner )
```

Create a new instance of the class [ListNode](#).

Returns

New instance.

3.24.1.14 ListNode_newFromAllocator()

```
PRIVATE ListNode * ListNode_newFromAllocator (
    Allocator * allocator,
    Object * object,
    int isOwner )
```

Create a new instance of the class [List](#) using a custom allocator.

Returns

New instance.

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

- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/List.c](#)
- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/ListNode.h](#)

3.25 MacroDefinition Struct Reference

Public Attributes

- [Object](#) **object**
- [String](#) * **body**
- [List](#) * **parameters**

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

- [/home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/MacroDefinition.h](#)

3.26 MacroStore Struct Reference

Public Attributes

- [Object](#) **object**
- struct [MacroStoreNode](#) * **root**

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/MacroStore.h

3.27 MacroStoreNode Struct Reference

Public Attributes

- int **isLeaf**
- [MacroDefinition](#) * **def**
- void * **children** [MAX_CHILDREN]

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/MacroStore.h

3.28 Malloc Struct Reference

Public Attributes

- [Allocator](#) **allocator**

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Allocator/Malloc.c

3.29 Map Class Reference

Public Member Functions

- PRIVATE [TaskMgr](#) * [TaskMgr_new](#) ()
Create a new instance of the class [TaskMgr](#).
- PUBLIC [Map](#) * [Map_new](#) ()
Create a new instance of the class [Map](#).
- PUBLIC [Map](#) * [Map_newFromAllocator](#) ([Allocator](#) *allocator)
Create a new instance of the class [Map](#) using a specific allocator.
- PUBLIC void [Map_delete](#) ([Map](#) *this)
Delete an instance of the class [Map](#).
- PUBLIC [Map](#) * [Map_copy](#) ([Map](#) *this)
Copy an instance of the class [Map](#).
- PUBLIC unsigned int [Map_insert](#) ([Map](#) *this, [String](#) *s, void *p, int isOwner)
Insert an object into a [Map](#) instance.
- PUBLIC unsigned int [Map_find](#) ([Map](#) *this, [String](#) *s, void **p)
TBD.
- PUBLIC void [Map_print](#) ([Map](#) *this)
Print a [Map](#) instance.
- PUBLIC unsigned int [Map_getSize](#) ([Map](#) *this)
Provide the size of a [Map](#) instance.
- PUBLIC [List](#) * [Map_getAll](#) ([Map](#) *this)
Get all the entries in an instance of a [Map](#).

Public Attributes

- [Object](#) object
- [List](#) *htable [HTABLE_SIZE]

3.29.1 Member Function Documentation

3.29.1.1 Map_copy()

```
PUBLIC Map * Map_copy (
    Map * this )
```

Copy an instance of the class [Map](#).

Returns

Copy of instance of NULL if failed to allocate.

3.29.1.2 Map_getAll()

```
PUBLIC List * Map_getAll (
    Map * this )
```

Get all the entries in an instance of a [Map](#).

Returns

[List](#) of map objects

3.29.1.3 Map_getSize()

```
PUBLIC unsigned int Map_getSize (
    Map * this )
```

Provide the size of a [Map](#) instance.

Returns

Size in bytes

3.29.1.4 Map_insert()

```
PUBLIC unsigned int Map_insert (
    Map * this,
    String * s,
    void * p,
    int isOwner )
```

Insert an object into a [Map](#) instance.

Returns

1 is inserted

3.29.1.5 Map_new()

```
PUBLIC Map * Map_new ( )
```

Create a new instance of the class [Map](#).

Returns

New [Map](#) instance or NULL if failed to allocate.

3.29.1.6 Map_newFromAllocator()

```
PUBLIC Map * Map_newFromAllocator (
    Allocator * allocator )
```

Create a new instance of the class [Map](#) using a specific allocator.

Returns

New [Map](#) instance or NULL if failed to allocate.

3.29.1.7 TaskMgr_new()

```
PRIVATE TaskMgr * TaskMgr_new ( )
```

Create a new instance of the class [TaskMgr](#).

Returns

New taskMgr instance or NULL if failed to allocate.

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

- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/Map.c](#)
- [/home/thomas/Projects/SParse-master/SParse/src/AppliLib/TaskMgr/TaskMgr.c](#)

3.30 MapEntry Struct Reference

Public Attributes

- [Object](#) **object**
- [String](#) * **s**
- void * **item**
- int **isOwner**

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

- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/MapEntry.c](#)

3.31 mem_align Union Reference

Public Attributes

- void * **a**
- long int **b**
- long long **c**

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

- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Types.h](#)
- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Types/Types.h](#)

3.32 Mutex Struct Reference

Public Attributes

- pthread_mutex_t * **mutex**
- pthread_cond_t **cond**

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

- /home/thomas/Projects/SParse-master/SParse/src/AppliLib/TaskMgr/Mutex.h

3.33 MyAllocator Struct Reference

Public Attributes

- [Allocator](#) **allocator**
- unsigned int **size**
- void * **memory_start**
- void * **memory_end**
- void * **pointer**
- void * **memory**

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/tests/MyAllocator.c
- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/tests/MyAllocator.c
- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectStore/tests/MyAllocator.c
- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/tests/MyAllocator.c
- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/String/tests/MyAllocator.c

3.34 MyType Struct Reference

Public Attributes

- char * **sval**

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/GrammarC99/MyType.h

3.35 Node Struct Reference

Public Attributes

- unsigned int **nbKeyUsed**
- unsigned int **isLeaf**
- [Object](#) ** **keys**
- [Object](#) ** **leaves**
- [Node](#) ** **children**
- [Object](#) * **buffer** [18]

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/Node.h
- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Node.h

3.36 Object Struct Reference

Public Member Functions

- PUBLIC [Object](#) * [Object_new](#) (unsigned int size, [Class](#) *class)
Create an instance of the class [Object](#).
- PUBLIC [Object](#) * [Object_newFromAllocator](#) ([Class](#) *class, [Allocator](#) *allocator)
TBD.
- PUBLIC void [Object_delete](#) ([Object](#) *this)
Delete an instance of the class [Object](#).
- PUBLIC void [Object_deallocate](#) ([Object](#) *this)
De-allocate an instance of the class [Object](#).
- PUBLIC [Object](#) * [Object_copy](#) ([Object](#) *this)
Copy an instance of the class [Object](#).
- PUBLIC int [Object_comp](#) ([Object](#) *this, [Object](#) *compared)
Compare 2 instances of the class [Object](#).
- PUBLIC char * [Object_print](#) ([Object](#) *this)
Print an instance of the class [Object](#) into a buffer of characters.
- PUBLIC [Object](#) * [Object_getRef](#) ([Object](#) *this)
Get a reference to an instance of the class [Object](#).
- PUBLIC void [Object_deRef](#) ([Object](#) *this)
De-reference to an instance of the class [Object](#).
- PUBLIC int [Object_isValid](#) ([Object](#) *this)
Check the pointed object is allocated.

Public Attributes

- int **marker**
- unsigned int **id**
- unsigned int **uniqId**
- [Class](#) * **class**
- void(* **delete**)([Object](#) *this)
- [Object](#) *(* **copy**)([Object](#) *this)
- unsigned int **refCount**
- unsigned int **size**
- [Allocator](#) * **allocator**

3.36.1 Member Function Documentation

3.36.1.1 Object_comp()

```
PUBLIC int Object_comp (
    Object * this,
    Object * compared )
```

Compare 2 instances of the class [Object](#).

Returns

0 if O1=O2, negative if O1<O2, positive if O1>O2

3.36.1.2 Object_copy()

```
PUBLIC Object * Object_copy (
    Object * this )
```

Copy an instance of the class [Object](#).

Returns

New instance

3.36.1.3 Object_getRef()

```
PUBLIC Object * Object_getRef (
    Object * this )
```

Get a reference to an instance of the class [Object](#).

Returns

Reference to instance

3.36.1.4 Object_isValid()

```
PUBLIC int Object_isValid (
    Object * this )
```

Check the pointed object is allocated.

Returns

1 if valid, 0 otherwise

3.36.1.5 Object_new()

```
PUBLIC Object * Object_new (
    unsigned int size,
    Class * class )
```

Create an instance of the class [Object](#).

Parameters

in	Class	to instanciate
----	-----------------------	----------------

3.36.1.6 Object_newFromAllocator()

```
PUBLIC Object * Object_newFromAllocator (
    Class * class,
    Allocator * allocator )
```

TBD.

Parameters

in	Class	to instanciate
----	-----------------------	----------------

3.36.1.7 Object_print()

```
PUBLIC char * Object_print (
    Object * this )
```

Print an instance of the class [Object](#) into a buffer of characters.

Returns

[Buffer](#) of characters

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Object.h
- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/Object.h
- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/[Object.c](#)

3.37 ObjectInfo Struct Reference

Public Attributes

- [Object](#) * **ptr**
- unsigned int **prevId**
- unsigned int **nextId**

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectMgr/[ObjectMgr.c](#)

3.38 ObjectMgr Class Reference

Public Member Functions

- PUBLIC void **ObjectMgr_delete** ([ObjectMgr](#) *this)
Delete an instance of the class [ObjectMgr](#).
- PUBLIC [ObjectMgr](#) * **ObjectMgr_copy** ([ObjectMgr](#) *this)
Copy an instance of the class [ObjectMgr](#).
- PUBLIC [ObjectMgr](#) * **ObjectMgr_getRef** ()
Get a reference to the singleton instance of [ObjectMgr](#).
- PUBLIC unsigned int **ObjectMgr_report** ([ObjectMgr](#) *this)
Reports the usage statistics for an instance of [ObjectMgr](#).
- PUBLIC [Object](#) * **ObjectMgr_allocate** ([ObjectMgr](#) *this, unsigned int size)
Allocate a new object memory footprint of a given size.
- PUBLIC void **ObjectMgr_deallocate** ([ObjectMgr](#) *this, [Object](#) *object)
De Allocate a given object.
- PUBLIC void **ObjectMgr_reportUnallocated** ([ObjectMgr](#) *this)
Report objects not deallocated.

Public Attributes

- [Object](#) **object**
- unsigned int **maxNbObjectAllocated**
- unsigned int **allocRequestId**
- unsigned int **freeRequestId**
- unsigned int **nbAllocatedObjects**
- [ObjectInfo](#) **allocatedObjects** [MAX_NB_OBJECTS]
- unsigned int **freeSpace**
- unsigned int **usedSpace**
- unsigned int **nextId**

3.38.1 Member Function Documentation

3.38.1.1 ObjectMgr_allocate()

```
PUBLIC Object * ObjectMgr_allocate (
    ObjectMgr * this,
    unsigned int size )
```

Allocate a new object memory footprint of a given size.

Parameters

in	size	size in bytes of the memory footprint.
----	------	--

Returns

Reference to a instance of [Object](#).

3.38.1.2 ObjectMgr_copy()

```
PUBLIC ObjectMgr * ObjectMgr_copy (
    ObjectMgr * this )
```

Copy an instance of the class [ObjectMgr](#).

Returns

New instance

3.38.1.3 ObjectMgr_deallocate()

```
PUBLIC void ObjectMgr_deallocate (
    ObjectMgr * this,
    Object * object )
```

De Allocate a given object.

Parameters

in	<i>object</i>	Reference to instance of Object .
----	---------------	---

3.38.1.4 ObjectMgr_getRef()

```
PUBLIC ObjectMgr * ObjectMgr_getRef ( )
```

Get a reference to the singleton instance of [ObjectMgr](#).

Returns

Reference to the singleton.

3.38.2 Member Data Documentation

3.38.2.1 maxNbObjectAllocated

```
unsigned int ObjectMgr::maxNbObjectAllocated
```

This is member B

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

- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectMgr/ObjectMgr.c](#)

3.39 ObjectStore Class Reference

Public Member Functions

- PUBLIC void [ObjectStore_delete](#) ([ObjectStore](#) *this)
Delete an instance of the class [ObjectMgr](#).
- PUBLIC [ObjectStore](#) * [ObjectStore_copy](#) ([ObjectStore](#) *this)
Copy an instance of the class [ObjectStore](#).
- PUBLIC [ObjectStore](#) * [ObjectStore_getRef](#) ()
Obtain the reference to the object store.
- PUBLIC [AllocInfo](#) * [ObjectStore_createAllocator](#) ([ObjectStore](#) *this, [Allocator](#) *allocator)
Register an [Allocator](#) with the objectStore.
- PUBLIC void [ObjectStore_deleteAllocator](#) ([ObjectStore](#) *this, [AllocInfo](#) *allocInfo)
TBD.
- PUBLIC [Object](#) * [ObjectStore_createObject](#) ([ObjectStore](#) *this, [Class](#) *class, [Allocator](#) *allocator)
TBD.
- PUBLIC void [ObjectStore_deleteObject](#) ([ObjectStore](#) *this, [Object](#) *object)
Delete an object from the object store.
- PUBLIC void [ObjectStore_report](#) ([ObjectStore](#) *this)
Reports the usage statistics for an instance of [ObjectStore](#).
- PUBLIC unsigned int [ObjectStore_getNbAllocatedObjects](#) ([ObjectStore](#) *this)
Reports the number of allocated objects in the [ObjectStore](#).
- PUBLIC int [ObjectStore_compare](#) ([ObjectStore](#) *this, [ObjectStore](#) *compared)
Compare 2 instances of the class [ObjectStore](#). Since there is only one [ObjectStore](#) instance, always return 1.
- PUBLIC void [ObjectStore_print](#) ([ObjectStore](#) *this)
Print an instance of the class [ObjectStore](#).

Public Attributes

- [Object](#) object
- unsigned int nbAllocatedObjects
- [AllocInfo](#) * allocList

3.39.1 Member Function Documentation

3.39.1.1 ObjectStore_compare()

```
PUBLIC int ObjectStore_compare (
    ObjectStore * this,
    ObjectStore * compared )
```

Compare 2 instances of the class [ObjectStore](#). Since there is only one [ObjectStore](#) instance, always return 1.

Returns

0 if different, 1 if equal.

3.39.1.2 ObjectStore_copy()

```
PUBLIC ObjectStore * ObjectStore_copy (
    ObjectStore * this )
```

Copy an instance of the class [ObjectStore](#).

Returns

Copy of the given instance.

3.39.1.3 ObjectStore_createAllocator()

```
PUBLIC AllocInfo * ObjectStore_createAllocator (
    ObjectStore * this,
    Allocator * allocator )
```

Register an [Allocator](#) with the objectStore.

TBD

3.39.1.4 ObjectStore_createObject()

```
PUBLIC Object * ObjectStore_createObject (
    ObjectStore * this,
    Class * class,
    Allocator * allocator )
```

TBD.

TBD

3.39.1.5 ObjectStore_delete()

```
PUBLIC void ObjectStore_delete (
    ObjectStore * this )
```

Delete an instance of the class [ObjectMgr](#).

TBD

3.39.1.6 ObjectStore_deleteAllocator()

```
PUBLIC void ObjectStore_deleteAllocator (
    ObjectStore * this,
    AllocInfo * allocInfo )
```

TBD.

TBD

3.39.1.7 ObjectStore_deleteObject()

```
PUBLIC void ObjectStore_deleteObject (
    ObjectStore * this,
    Object * object )
```

Delete an object from the object store.

TBD

3.39.1.8 ObjectStore_getNbAllocatedObjects()

```
PUBLIC unsigned int ObjectStore_getNbAllocatedObjects (
    ObjectStore * this )
```

Reports the number of allocated objects in the [ObjectStore](#).

TBD

3.39.1.9 ObjectStore_getRef()

```
PUBLIC ObjectStore * ObjectStore_getRef ( )
```

Obtain the reference to the object store.

TBD

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

- [/home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectStore/ObjectStore.c](#)

3.40 OptionDefault Struct Reference

Public Attributes

- char * **name**
- char * **flag**
- char * **value**

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

- [/home/thomas/Projects/SParse-master/SParse/src/AppliLib/OptionMgr/OptionMgr.c](#)

3.41 OptionMgr Class Reference

Public Member Functions

- PUBLIC void **OptionMgr_delete** ([OptionMgr](#) *this)
TBD.
- PUBLIC [OptionMgr](#) * **OptionMgr_copy** ([OptionMgr](#) *this)
TBD.
- PUBLIC [OptionMgr](#) * **OptionMgr_getRef** ()
TBD.
- PUBLIC unsigned int **OptionMgr_getSize** ([OptionMgr](#) *this)
TBD.
- PUBLIC [String](#) * **OptionMgr_getOption** ([OptionMgr](#) *this, const char *name)
TBD.
- PUBLIC void **OptionMgr_setOption** ([OptionMgr](#) *this, const char *optionName, [String](#) *value)
TBD.
- PUBLIC unsigned int **OptionMgr_readFromFile** ([OptionMgr](#) *this)
TBD.
- PUBLIC unsigned int **OptionMgr_readFromCmdLine** ([OptionMgr](#) *this, const int argc, const char **argv)
TBD.

Public Attributes

- [Object](#) object
- [Map](#) * options

3.41.1 Member Function Documentation

3.41.1.1 OptionMgr_getRef()

```
PUBLIC OptionMgr * OptionMgr_getRef ( )
```

TBD.

TBD

3.41.1.2 OptionMgr_readFromCmdLine()

```
PUBLIC unsigned int OptionMgr_readFromCmdLine (
    OptionMgr * this,
    const int argc,
    const char ** argv )
```

TBD.

Parameters

in	<i>argc</i>	Number of commandline arguments.
in	<i>argv</i>	List os commandline arguments.

Returns

Status of operation.

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

- </home/thomas/Projects/SParse-master/SParse/src/AppliLib/OptionMgr/OptionMgr.c>

3.42 PoolCache Struct Reference

Public Attributes

- unsigned int **idx**
- unsigned int **isUsed**
- void * **cache**

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

- </home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Pool.h>
- </home/thomas/Projects/SParse-master/SParse/src/CommonLib/Pool/Pool.h>

3.43 Product Class Reference

Public Attributes

- [Object](#) **object**
- [String](#) * **name**
- [String](#) * **location**
- [List](#) * **sources**
- [List](#) * **includes**
- [List](#) * **uses**

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

- </home/thomas/Projects/SParse-master/SParse/src/ParseLib/Configuration/Product.c>

3.44 SdbMgr Class Reference

Public Member Functions

- PUBLIC void **SdbMgr_delete** ([SdbMgr](#) *this)
Destroy an instance of the class [SdbMgr](#).
- PUBLIC [SdbMgr](#) * **SdbMgr_copy** ([SdbMgr](#) *this)
Create a copy of an [SdbMgr](#) object.
- PUBLIC [SdbMgr](#) * **SdbMgr_getRef** ()
Get a reference to an object.
- PUBLIC unsigned int **SdbMgr_execute** ([SdbMgr](#) *this, const char *statement, [List](#) *result)
Execute a Sdb request.

Public Attributes

- [Object](#) `object`
- `sqlite3 * db`
- [String](#) * `name`

3.44.1 Member Function Documentation

3.44.1.1 SdbMgr_copy()

```
PUBLIC SdbMgr * SdbMgr_copy (  
    SdbMgr * this )
```

Create a copy of an [SdbMgr](#) object.

Returns

A copy of the [SdbMgr](#) object.

3.44.1.2 SdbMgr_execute()

```
PUBLIC unsigned int SdbMgr_execute (  
    SdbMgr * this,  
    const char * statement,  
    List * result )
```

Execute a Sdb request.

Returns

status

3.44.1.3 SdbMgr_getRef()

```
PUBLIC SdbMgr * SdbMgr_getRef ( )
```

Get a reference to an object.

Returns

A reference to a [SdbMgr](#) object.

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

- `/home/thomas/Projects/SParse-master/SParse/src/AppliLib/SdbMgr/SdbMgr.c`

3.45 SdbRequest Class Reference

Public Member Functions

- PUBLIC [SdbRequest](#) * [SdbRequest_new](#) (const char *fmt)
Create a new [SdbRequest](#) instance
@parameter SQL statement template.
- PUBLIC void [SdbRequest_delete](#) ([SdbRequest](#) *this)
Create a new [SdbRequest](#) instance
@parameter SQL statement template.
- PUBLIC void [SdbRequest_execute](#) ([SdbRequest](#) *this,...)
Execute a [SdbRequest](#)
@parameter Variable list of parameter to use with SQL template.

Public Attributes

- [Object](#) **object**
- char * **buffer**
- unsigned int **size**
- const char * **fmt**
- [List](#) * **result**
- unsigned int **nbResults**
- unsigned int **nbColumns**

3.45.1 Member Function Documentation

3.45.1.1 SdbRequest_delete()

```
PUBLIC void SdbRequest_delete (  
    SdbRequest * this )
```

Create a new [SdbRequest](#) instance
@parameter SQL statement template.

Returns

Instance of an [SdbRequest](#)

3.45.1.2 SdbRequest_execute()

```
PUBLIC void SdbRequest_execute (  
    SdbRequest * this,  
    ... )
```

Execute a [SdbRequest](#)
@parameter Variable list of parameter to use with SQL template.

Returns

Instance of an [SdbRequest](#)

3.45.1.3 SdbRequest_new()

```
PUBLIC SdbRequest * SdbRequest_new (
    const char * fmt )
```

Create a new [SdbRequest](#) instance

@parameter SQL statement template.

Returns

Instance of an [SdbRequest](#)

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

- /home/thomas/Projects/SParse-master/SParse/src/AppliLib/SdbMgr/SdbRequest.c

3.46 SkipList Class Reference

Public Member Functions

- PUBLIC [SkipList](#) * [SkipList_new](#) ()
Create a new instance of the class [SkipList](#).
- PUBLIC [SkipList](#) * [SkipList_newFromAllocator](#) ([Allocator](#) *allocator)
Create a new instance of the class [SkipList](#) from an specified allocator.
- PUBLIC void [SkipList_delete](#) ([SkipList](#) *this)
[SkipList_delete](#).
- PUBLIC [SkipList](#) * [SkipList_copy](#) ([SkipList](#) *this)
[SkipList_copy](#).
- PUBLIC void [SkipList_add](#) ([SkipList](#) *this, [Object](#) *key, [Object](#) *item)
[SkipList_add](#).
- PUBLIC [Object](#) * [SkipList_remove](#) ([SkipList](#) *this, [Object](#) *key)
[SkipList_remove](#).
- PUBLIC int [SkipList_compare](#) ([SkipList](#) *this, [SkipList](#) *compared)
[SkipList_compare](#).
- PUBLIC void [SkipList_print](#) ([SkipList](#) *this)
[SkipList_print](#).
- PUBLIC unsigned int [SkipList_getSize](#) ([SkipList](#) *this)
[SkipList_getSize](#).

Public Attributes

- [Object](#) **object**
- unsigned int **level**
- unsigned int **nbObjects**
- unsigned int **pack**
- void * **headerPtr**
- void * **endPtr**

3.46.1 Member Function Documentation

3.46.1.1 SkipList_add()

```
PUBLIC void SkipList_add (  
    SkipList * this,  
    Object * key,  
    Object * item )
```

SkipList_add.

Parameters

in	<i>Key</i>	to index object
in	<i>Object</i>	to add to SkipList object.

Returns

None

3.46.1.2 SkipList_compare()

```
PUBLIC int SkipList_compare (
    SkipList * this,
    SkipList * compared )
```

SkipList_compare.

Parameters

in	<i>Instance</i>	to be compared to.
----	-----------------	--------------------

Returns

0 if equal, <0 if S1<S2, >0 if S1>S2

3.46.1.3 SkipList_copy()

```
PUBLIC SkipList * SkipList_copy (
    SkipList * this )
```

SkipList_copy.

Parameters

in	<i>Instance</i>	to copy
----	-----------------	---------

Returns

A copy of the [SkipList](#) instance.

3.46.1.4 SkipList_delete()

```
PUBLIC void SkipList_delete (
    SkipList * this )
```

SkipList_delete.

Parameters

in	<i>Instance</i>	to destroy
----	-----------------	------------

Returns

None

3.46.1.5 SkipList_getSize()

```
PUBLIC unsigned int SkipList_getSize (  
    SkipList * this )
```

SkipList_getSize.

Parameters

in	<i>None</i>	
----	-------------	--

Returns

None

3.46.1.6 SkipList_new()

```
PUBLIC SkipList * SkipList_new ( )
```

Create a new instance of the class [SkipList](#).

Parameters

in	<i>none</i>	
----	-------------	--

Returns

New instance of class [SkipList](#).

3.46.1.7 SkipList_newFromAllocator()

```
PUBLIC SkipList * SkipList_newFromAllocator (  
    Allocator * allocator )
```

Create a new instance of the class SkipList from an specified allocator.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

New instance of class [SkipList](#).

3.46.1.8 SkipList_print()

```
PUBLIC void SkipList_print (
    SkipList * this )
```

SkipList_print.

Parameters

in	<i>None</i>	
----	-------------	--

Returns

None

3.46.1.9 SkipList_remove()

```
PUBLIC Object * SkipList_remove (
    SkipList * this,
    Object * key )
```

SkipList_remove.

Parameters

in	<i>Key</i>	of object to remove
----	------------	---------------------

Returns

[Object](#) removed from [SkipList](#) object.

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/[SkipList.c](#)

3.47 SkipNode Struct Reference**Public Attributes**

- [Object](#) **object**
- [Object](#) * **key**
- [Object](#) * **item**
- unsigned int **level**
- void * **forward** [SKIPLIST_MAX_LEVEL]

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/SkipNode.h

3.48 SParse Class Reference

Public Member Functions

- PUBLIC [SParse](#) * [SParse_new](#) ([String](#) *sdbName)
Create a new [SParse](#) object.
- PUBLIC void [SParse_delete](#) ([SParse](#) *this)
Delete a [SParse](#) object.
- PUBLIC [SParse](#) * [SParse_copy](#) ([SParse](#) *this)
Copy a [SParse](#) object instance.
- PUBLIC void [SParse_print](#) ([SParse](#) *this)
Print a [SParse](#) object.
- PUBLIC unsigned int [SParse_parse](#) ([SParse](#) *this, const char *extension)
Parse all files with a given extension.

Public Attributes

- [Object](#) **object**
- [Configuration](#) * **configuration**
- char * **extension**
- [SdbMgr](#) * **sdbMgr**

3.48.1 Member Function Documentation

3.48.1.1 SParse_copy()

```
PUBLIC SParse * SParse_copy (
    SParse * this )
```

Copy a [SParse](#) object instance.

Returns

Copy of instance.

3.48.1.2 SParse_delete()

```
PUBLIC void SParse_delete (
    SParse * this )
```

Delete a [SParse](#) object.

Parameters

Object	to delete.
------------------------	------------

3.48.1.3 SParse_new()

```
PUBLIC SParse * SParse_new (
    String * sdbName )
```

Create a new [SParse](#) object.

Returns

New [SParse](#) object.

3.48.1.4 SParse_parse()

```
PUBLIC unsigned int SParse_parse (
    SParse * this,
    const char * extension )
```

Parse all files with a given extension.

Parameters

<code>in</code>	<code>extension</code>	Extension of the files to parse.
-----------------	------------------------	----------------------------------

Returns

Status of the operation.

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

- `/home/thomas/Projects/SParse-master/SParse/src/ParseLib/SParse/SParse.c`

3.49 String Struct Reference

Public Member Functions

- PUBLIC void **String_delete** ([String](#) *this)
Delete an instance of class [String](#).
- PUBLIC [String](#) * **String_copy** ([String](#) *this)
Copy an instance of class [String](#).
- PUBLIC [String](#) * **String_getRef** ([String](#) *this)
Copy an instance of class [String](#).
- PUBLIC int **String_compare** ([String](#) *this, [String](#) *compared)
Compare this [String](#) with another [String](#).
- PUBLIC [String](#) * **String_subString** ([String](#) *this, unsigned int idx, unsigned int length)
TBD.
- PUBLIC int **String_tolnt** ([String](#) *this)
TBD.

- PUBLIC unsigned int **String_getLength** ([String](#) *this)
TBD.
- PUBLIC char * **String_getBuffer** ([String](#) *this)
TBD.
- PUBLIC void **String_setBuffer** ([String](#) *this, char *buffer, int isOwned)
TBD.
- PUBLIC unsigned int **String_isContained** ([String](#) *this, [String](#) *s2)
TBD.

Public Attributes

- [Object](#) **object**
- int **isOwned**
- char * **buffer**
- unsigned int **length**

3.49.1 Detailed Description

/file String2.c

/brief The [String](#) class provide a dynamic array of char terminated by 0.

The class [String](#) is a container for text data. /class [String](#)

3.49.2 Member Function Documentation

3.49.2.1 String_compare()

```
PUBLIC int String_compare (
    String * this,
    String * compared )
```

Compare this [String](#) with another [String](#).

Parameters

in	<i>compared</i>	String to compare
----	-----------------	-----------------------------------

Returns

0 if S1=S2, negative if S1<S2, positive if S1>S2

3.49.2.2 String_copy()

```
PUBLIC String * String_copy (
    String * this )
```

Copy an instance of class [String](#).

Returns

Copy of instance.

3.49.2.3 String_getRef()

```
PUBLIC String * String_getRef (
    String * this )
```

Copy an instance of class [String](#).

Returns

Copy of instance.

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/String/String2.c

3.50 stub_data Struct Reference**Public Attributes**

- void * **malloc_result**
- int **malloc_nb_calls**
- int **free_nb_calls**

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Allocator/tests/Stub_Malloc.c

3.51 Task Struct Reference**Public Member Functions**

- PUBLIC [Task](#) * **Task_create** (void *(*body)(void *p), int nbParams, void **params)
Create a task object.
- PUBLIC void **Task_destroy** ([Task](#) *this)
Destroy a task object.
- PUBLIC int **Task_isReady** ([Task](#) *this)
Obtain the status of the ready flag.
- PUBLIC void **Task_setReady** ([Task](#) *this)
Mark a task as ready.
- PUBLIC int **Task_isRunning** ([Task](#) *this)
Obtain the status of the running flag.
- PUBLIC void **Task_setRunning** ([Task](#) *this)
Mark a task as running.
- PUBLIC int **Task_isCompleted** ([Task](#) *this)
Obtain the status of the completion flag.
- PUBLIC void **Task_setCompleted** ([Task](#) *this)
Mark a task as completed.
- PUBLIC void **Task_start** ([Task](#) *this)
Start a [Task](#).
- PUBLIC void **Task_executeBody** ([Task](#) *this)
Execute the body of the task.

Public Attributes

- void ***(* body)**(void *p)
- int **nbParams**
- void * **params** [5]
- int **isReady**
- int **isRunning**
- int **isCompleted**
- int **execTime**

3.51.1 Member Function Documentation

3.51.1.1 Task_create()

```
PUBLIC Task * Task_create (
    void (*)(void *) body,
    int nbParams,
    void ** params )
```

Create a task object.

Returns

The new instance of a task.

3.51.1.2 Task_executeBody()

```
PUBLIC void Task_executeBody (
    Task * this )
```

Execute the body of the task.

Returns

TBD

3.51.1.3 Task_isCompleted()

```
PUBLIC int Task_isCompleted (
    Task * this )
```

Obtain the status of the completion flag.

Returns

Completion flag.

3.51.1.4 Task_isReady()

```
PUBLIC int Task_isReady (
    Task * this )
```

Obtain the status of the ready flag.

Returns

Ready flag.

3.51.1.5 Task_isRunning()

```
PUBLIC int Task_isRunning (
    Task * this )
```

Obtain the status of the running flag.

Returns

Running flag.

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

- /home/thomas/Projects/SParse-master/SParse/src/AppliLib/TaskMgr/Task.c

3.52 TaskMgr Class Reference

Public Member Functions

- PUBLIC TaskMgr * TaskMgr_getRef ()
Get reference to singleton TaskMgr.
- PUBLIC void TaskMgr_delete (TaskMgr *this)
TBD.
- PUBLIC int TaskMgr_start (TaskMgr *this, Task *task)
Queue a task for execution.
- PUBLIC void TaskMgr_stop (TaskMgr *this)
Request all worker threads to stop.
- PUBLIC void TaskMgr_print (TaskMgr *this)
Print the content of the TaskMgr.
- PUBLIC unsigned int TaskMgr_getSize (TaskMgr *this)
TBD.
- PRIVATE void TaskMgr_waitForThread (TaskMgr *this)
TBD.
- PRIVATE int TaskMgr_createWorkerThreads (TaskMgr *this)
Create all worker threads.
- PRIVATE int TaskMgr_findAvailableTask (TaskMgr *this)
TBD.
- PRIVATE int TaskMgr_initSemaphores (TaskMgr *this)

Initialise empty and full semaphores.

- PRIVATE int `TaskMgr_destroySemaphores` (`TaskMgr *this`)

Destroy empty and full semaphores.

- PRIVATE int `TaskMgr_waitNotFull` (`TaskMgr *this`)

Wait until there is a space to add a task.

- PRIVATE int `TaskMgr_waitNotEmpty` (`TaskMgr *this`)

Wait until there is a task in the queue.

- PRIVATE int `TaskMgr_signalNotFull` (`TaskMgr *this`)

Signal that task can be added to the queue.

- PRIVATE int `TaskMgr_signalNotEmpty` (`TaskMgr *this`)

Signal that there are task to process.

Public Attributes

- Object `object`
- int `nbThreads`
- `Task * taskId` [MAX_TASKS]
- `pthread_t threadHandle` [MAX_THREADS]
- `sem_t semEmpty`
- `sem_t semFull`
- `pthread_mutex_t mutex`
- int `isStopping`

3.52.1 Member Function Documentation

3.52.1.1 TaskMgr_createWorkerThreads()

```
PRIVATE int TaskMgr_createWorkerThreads (
    TaskMgr * this )
```

Create all worker threads.

Returns

TBD

3.52.1.2 TaskMgr_delete()

```
PUBLIC void TaskMgr_delete (
    TaskMgr * this )
```

TBD.

Returns

none

3.52.1.3 TaskMgr_destroySemaphores()

```
PRIVATE int TaskMgr_destroySemaphores (
    TaskMgr * this )
```

Destroy empty and full semaphores.

Returns

1 indicates if operation was successful.

3.52.1.4 TaskMgr_findAvailableTask()

```
PRIVATE int TaskMgr_findAvailableTask (
    TaskMgr * this )
```

TBD.

Returns

TBD

3.52.1.5 TaskMgr_getRef()

```
PUBLIC TaskMgr * TaskMgr_getRef ( )
```

Get reference to singleton [TaskMgr](#).

Returns

Reference to the [TaskMgr](#).

3.52.1.6 TaskMgr_getSize()

```
PUBLIC unsigned int TaskMgr_getSize (
    TaskMgr * this )
```

TBD.

Parameters

in	TBD	
----	-----	--

Returns

TBD

3.52.1.7 TaskMgr_initSemaphores()

```
PRIVATE int TaskMgr_initSemaphores (
    TaskMgr * this )
```

Initialise empty and full semaphores.

Returns

1 indicates if operation was successful.

3.52.1.8 TaskMgr_signalNotEmpty()

```
PRIVATE int TaskMgr_signalNotEmpty (
    TaskMgr * this )
```

Signal that there are task to process.

Returns

1 indicates if operation was successful.

3.52.1.9 TaskMgr_signalNotFull()

```
PRIVATE int TaskMgr_signalNotFull (
    TaskMgr * this )
```

Signal that task can be added to the queue.

Returns

1 indicates if operation was successful.

3.52.1.10 TaskMgr_start()

```
PUBLIC int TaskMgr_start (
    TaskMgr * this,
    Task * task )
```

Queue a task for execution.

Parameters

in	<i>task</i>	
----	-------------	--

Returns

1 if successful.

3.52.1.11 TaskMgr_stop()

```
PUBLIC void TaskMgr_stop (
    TaskMgr * this )
```

Request all worker threads to stop.

Returns

1 if successful.

3.52.1.12 TaskMgr_waitForThread()

```
PRIVATE void TaskMgr_waitForThread (
    TaskMgr * this )
```

TBD.

Parameters

in	TBD	
----	-----	--

Returns

TBD

3.52.1.13 TaskMgr_waitNotEmpty()

```
PRIVATE int TaskMgr_waitNotEmpty (
    TaskMgr * this )
```

Wait until there is a task in the queue.

Returns

1 indicates if operation was successful.

3.52.1.14 TaskMgr_waitNotFull()

```
PRIVATE int TaskMgr_waitNotFull (
    TaskMgr * this )
```

Wait until there is a space to add a task.

Returns

1 indicates if operation was successful.

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

- /home/thomas/Projects/SParse-master/SParse/src/AppliLib/TaskMgr/TaskMgr.c

3.53 TestClass Struct Reference

Public Attributes

- [Object](#) **object**

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectStore/tests/UT_ObjectStore.c

3.54 TestFileMgr Struct Reference

Public Attributes

- [Object](#) **object**
- [List](#) * **files**
- [List](#) * **directories**
- char * **separator**
- [String](#) * **rootLocation**

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

- /home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/tests/UT_FileMgr_01.c

3.55 TestObject Struct Reference

Public Attributes

- [Object](#) **object**
- int **id**
- int **testValue**

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Array/tests/UT_Array_01.c
- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/tests/TestObject.c
- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/tests/TestObject.c
- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/tests/TestObject.c
- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/tests/TestObject.c

3.56 testOptionMgr Struct Reference

Public Attributes

- [Object](#) **object**
- [Map](#) * **options**

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

- /home/thomas/Projects/SParse-master/SParse/src/AppliLib/OptionMgr/tests/UT_OptionMgr_01.c

3.57 TestSdbMgr Struct Reference

Public Attributes

- [Object](#) object

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

- /home/thomas/Projects/SParse-master/SParse/src/AppliLib/SdbMgr/tests/UT_SdbMgr_01.c

3.58 TestTimeMgr Struct Reference

Public Attributes

- [Object](#) object
- [Map](#) * timers

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

- /home/thomas/Projects/SParse-master/SParse/src/AppliLib/TimeMgr/tests/UT_TimeMgr_01.c

3.59 TimeMgr Class Reference

Public Member Functions

- PUBLIC void [TimeMgr_delete](#) ([TimeMgr](#) *this)
Delete a [TimeMgr](#) object.
- PUBLIC [TimeMgr](#) * [TimeMgr_copy](#) ([TimeMgr](#) *this)
Copy an instance of the class [TimeMgr](#).
- PUBLIC [TimeMgr](#) * [TimeMgr_getRef](#) ()
Get a reference to the singleton instance of [TimeMgr](#).
- PUBLIC unsigned int [TimeMgr_getSize](#) ([TimeMgr](#) *this)
Provide the size of the class or an instance.
- PUBLIC void [TimeMgr_latchTime](#) ([TimeMgr](#) *this, [String](#) *s)
Latch the current time under the specified name.

Public Attributes

- [Object](#) object
- [Map](#) * timers

3.59.1 Member Function Documentation

3.59.1.1 TimeMgr_copy()

```
PUBLIC TimeMgr * TimeMgr_copy (
    TimeMgr * this )
```

Copy an instance of the class [TimeMgr](#).

Returns

New instance

3.59.1.2 TimeMgr_delete()

```
PUBLIC void TimeMgr_delete (
    TimeMgr * this )
```

Delete a [TimeMgr](#) object.

Parameters

Object	to delete.
------------------------	------------

3.59.1.3 TimeMgr_getRef()

```
PUBLIC TimeMgr * TimeMgr_getRef ( )
```

Get a reference to the singleton instance of [TimeMgr](#).

Returns

Reference to the singleton.

3.59.1.4 TimeMgr_getSize()

```
PUBLIC unsigned int TimeMgr_getSize (
    TimeMgr * this )
```

Provide the size of the class or an instance.

Returns

Size in byte

3.59.1.5 TimeMgr_latchTime()

```
PUBLIC void TimeMgr_latchTime (
    TimeMgr * this,
    String * s )
```

Latch the current time under the specified name.

Parameters

<i>name</i>	of the timer to create
-------------	------------------------

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

- /home/thomas/Projects/SParse-master/SParse/src/AppliLib/TimeMgr/[TimeMgr.c](#)

3.60 Timer Class Reference

Public Member Functions

- PUBLIC [Timer](#) * [Timer_new](#) ([String](#) *name)
Create an instance of the class [Timer](#).
- PUBLIC void [Timer_delete](#) ([Timer](#) *this)
Delete an instance of the class [Timer](#).
- PUBLIC [Timer](#) * [Timer_copy](#) ([Timer](#) *this)
Copy an instance of the class [Timer](#).
- PUBLIC unsigned int [Timer_getSize](#) ([Timer](#) *this)
TBD.
- PUBLIC unsigned int [Timer_isEqual](#) ([Timer](#) *this, [Timer](#) *compared)
TBD.
- PUBLIC void [Timer_print](#) ([Timer](#) *this)
TBD.
- PUBLIC void [Timer_latchTime](#) ([Timer](#) *this)
TBD.

Public Attributes

- [Object](#) **object**
- [String](#) * **name**
- unsigned int **state**
- unsigned int **nbCalls**
- long double **cpuDurationS**
- long double **wallDurationS**
- long double **cpuLatchedTimeS**
- long double **wallLatchedTimeS**

3.60.1 Member Function Documentation

3.60.1.1 [Timer_copy\(\)](#)

```
PUBLIC Timer * Timer\_copy (  
    Timer * this )
```

Copy an instance of the class [Timer](#).

Returns

Copied instance.

3.60.1.2 Timer_new()

```
PUBLIC Timer * Timer_new (
    String * name )
```

Create an instance of the class [Timer](#).

Returns

New instance.

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

- /home/thomas/Projects/SParse-master/SParse/src/AppliLib/TimeMgr/Timer.c

3.61 TransUnit Class Reference

Public Member Functions

- PUBLIC [TransUnit](#) * [TransUnit_new](#) ([FileDesc](#) *file, [FileMgr](#) *fileMgr)
Create a new [TransUnit](#) object.
- PUBLIC void [TransUnit_delete](#) ([TransUnit](#) *this)
Delete an instance of a [TransUnit](#) object.
- PUBLIC void [TransUnit_print](#) ([TransUnit](#) *this)
Print a new [TransUnit](#) object.
- PUBLIC unsigned int [TransUnit_getSize](#) ([TransUnit](#) *this)
Returns the size a new [TransUnit](#) object.
- PUBLIC char * [TransUnit_getName](#) ([TransUnit](#) *this)
Returns the filename a new [TransUnit](#) object.
- PUBLIC [String](#) * [TransUnit_getNextBuffer](#) ([TransUnit](#) *this)
Returns the buffer of a new [TransUnit](#) object.

Public Attributes

- [Object](#) object
- [FileDesc](#) * file
- [FileMgr](#) * fm
- [List](#) * buffers
- [MacroStore](#) * store
- [Buffer](#) * currentBuffer
- int nbCharRead
- char * outputBuffer
- int outputBufferSize
- int nbCharWritten

3.61.1 Member Function Documentation

3.61.1.1 TransUnit_getName()

```
PUBLIC char * TransUnit_getName (
    TransUnit * this )
```

Returns the filename a new [TransUnit](#) object.

Returns

Filename

3.61.1.2 TransUnit_getNextBuffer()

```
PUBLIC String * TransUnit_getNextBuffer (
    TransUnit * this )
```

Returns the buffer of a new [TransUnit](#) object.

Returns

[Buffer](#)

3.61.1.3 TransUnit_getSize()

```
PUBLIC unsigned int TransUnit_getSize (
    TransUnit * this )
```

Returns the size a new [TransUnit](#) object.

Returns

Size in bytes.

3.61.1.4 TransUnit_new()

```
PUBLIC TransUnit * TransUnit_new (
    FileDesc * file,
    FileMgr * fileMgr )
```

Create a new [TransUnit](#) object.

Returns

Created [TransUnit](#) instance.

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

- [/home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/TransUnit.c](#)

3.62 yy_buffer_state Struct Reference

Public Attributes

- FILE * **yy_input_file**
- char * **yy_ch_buf**
- char * **yy_buf_pos**
- int **yy_buf_size**
- int **yy_n_chars**
- int **yy_is_our_buffer**
- int **yy_is_interactive**
- int **yy_at_bol**
- int [yy_bs_lineno](#)
- int [yy_bs_column](#)
- int **yy_fill_buffer**
- int **yy_buffer_status**

3.62.1 Member Data Documentation

3.62.1.1 yy_bs_column

```
int yy_buffer_state::yy_bs_column
```

The column count.

3.62.1.2 yy_bs_lineno

```
int yy_buffer_state::yy_bs_lineno
```

The line count.

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar2/Grammar2.lex.c
- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/GrammarC99/GrammarC99.lex.c

3.63 yy_trans_info Struct Reference

Public Attributes

- flex_int16_t **yy_verify**
- flex_int16_t **yy_nxt**

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

- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar2/Grammar2.lex.c
- /home/thomas/Projects/SParse-master/SParse/src/ParseLib/GrammarC99/GrammarC99.lex.c

3.64 yyallocc Union Reference

Public Attributes

- `yy_state_t yyss_alloc`
- `YYSTYPE yyvs_alloc`

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

- `/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar2/Grammar2.parse.c`
- `/home/thomas/Projects/SParse-master/SParse/src/ParseLib/GrammarC99/GrammarC99.parse.c`

3.65 yyguts_t Struct Reference

Public Attributes

- `YY_EXTRA_TYPE yyextra_r`
- `FILE * yyin_r`
- `FILE * yyout_r`
- `size_t yy_buffer_stack_top`
- `size_t yy_buffer_stack_max`
- `YY_BUFFER_STATE * yy_buffer_stack`
- `char yy_hold_char`
- `int yy_n_chars`
- `int yyleng_r`
- `char * yy_c_buf_p`
- `int yy_init`
- `int yy_start`
- `int yy_did_buffer_switch_on_eof`
- `int yy_start_stack_ptr`
- `int yy_start_stack_depth`
- `int * yy_start_stack`
- `yy_state_type yy_last_accepting_state`
- `char * yy_last_accepting_cpos`
- `int yylineno_r`
- `int yy_flex_debug_r`
- `char * yytext_r`
- `int yy_more_flag`
- `int yy_more_len`
- `YYSTYPE * yylval_r`

3.65.1 Member Data Documentation

3.65.1.1 yy_buffer_stack

`YY_BUFFER_STATE * yyguts_t::yy_buffer_stack`

Stack as an array.

3.65.1.2 yy_buffer_stack_max

`size_t yyguts_t::yy_buffer_stack_max`

capacity of stack.

3.65.1.3 yy_buffer_stack_top

`size_t yyguts_t::yy_buffer_stack_top`

index of top of stack.

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

- `/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar2/Grammar2.lex.c`
- `/home/thomas/Projects/SParse-master/SParse/src/ParseLib/GrammarC99/GrammarC99.lex.c`

3.66 YYSTYPE Union Reference

Public Attributes

- [String](#) * `text`

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

- `/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar2/Grammar2.parse.h`

Chapter 4

File Documentation

4.1 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/↵ FileDesc.c File Reference

The [FileDesc](#) class describe a File in the FileMgr.

```
#include "FileDesc.h"
#include "String2.h"
#include "FileIo.h"
#include "Class.h"
#include "Object.h"
#include "Memory.h"
```

Classes

- class [FileDesc](#)

Functions

- PRIVATE [String](#) * [FileDesc_getBasename](#) ([FileDesc](#) *this)
- PUBLIC unsigned int [FileDesc_getSize](#) ([FileDesc](#) *this)

4.1.1 Detailed Description

The [FileDesc](#) class describe a File in the FileMgr.

The class [FileDesc](#) is TBD

4.2 FileDesc.h

```

00001 /* FileDesc.h */
00002
00003 #ifndef _FILEDESC_H_
00004 #define _FILEDESC_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008
00009 typedef struct FileDesc FileDesc;
00010
00011 PUBLIC FileDesc * FileDesc_new();
00012 PUBLIC void FileDesc_delete(FileDesc * this);
00013 PUBLIC FileDesc * FileDesc_copy(FileDesc * this);
00014 PUBLIC unsigned int FileDesc_getSize(FileDesc* this);
00015 PUBLIC void FileDesc_setFullName(FileDesc * this, String * fullName);
00016 PUBLIC String * FileDesc_getFullName(FileDesc * this);
00017 PUBLIC void FileDesc_setName(FileDesc * this, String * name);
00018 PUBLIC String * FileDesc_getName(FileDesc * this);
00019 PUBLIC String * FileDesc_load(FileDesc * this);
00020
00021 #endif /* _FILEDESC_H_ */

```

4.3 FileDesc.h

```

00001 /* FileDesc.h */
00002
00003 #ifndef _FILEDESC_H_
00004 #define _FILEDESC_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008
00009 typedef struct FileDesc FileDesc;
00010
00011 PUBLIC FileDesc * FileDesc_new();
00012 PUBLIC void FileDesc_delete(FileDesc * this);
00013 PUBLIC FileDesc * FileDesc_copy(FileDesc * this);
00014 PUBLIC unsigned int FileDesc_getSize(FileDesc* this);
00015 PUBLIC void FileDesc_setFullName(FileDesc * this, String * fullName);
00016 PUBLIC String * FileDesc_getFullName(FileDesc * this);
00017 PUBLIC void FileDesc_setName(FileDesc * this, String * name);
00018 PUBLIC String * FileDesc_getName(FileDesc * this);
00019 PUBLIC String * FileDesc_load(FileDesc * this);
00020
00021 #endif /* _FILEDESC_H_ */

```

4.4 /home/thomas/Projects/SParse-master/SParse/src/AppLib/FileMgr/↵ FileMgr.c File Reference

The [FileMgr](#) class manages a list of files contained in a group of locations.

```

#include "FileMgr.h"
#include "String2.h"
#include "Class.h"
#include "Object.h"
#include "List.h"
#include "FileDesc.h"
#include "Memory.h"
#include "Error.h"
#include "Debug.h"
#include "FileIo.h"

```

Classes

- class [FileMgr](#)

Macros

- `#define DEBUG (0)`
- `#define FILEMGR_MAX_PATH (1024)`

Functions

- PRIVATE void **FileMgr_listFiles** (**FileMgr** *this, **String** *directory)
- PRIVATE unsigned int **FileMgr_existFS** (**FileMgr** *this, **String** *fullName)
- PUBLIC void **FileMgr_print** (**FileMgr** *this)
- PUBLIC **FileDesc** * **FileMgr_createFile** (**FileMgr** *this, const char *fileName)
- PUBLIC **FileDesc** * **FileMgr_searchFile** (**FileMgr** *this, **String** *name, **List** *preferredDir)
- PUBLIC **FileDesc** * **FileMgr_isManaged** (**FileMgr** *this, **String** *fullName)

4.4.1 Detailed Description

The **FileMgr** class manages a list of files contained in a group of locations.

The class **FileMgr** is TBD

4.5 FileMgr.h

```

00001  /* FileMgr.h */
00002
00003  #ifndef _FILEMGR_H_
00004  #define _FILEMGR_H_
00005
00006  #include "Types.h"
00007  #include "List.h"
00008  #include "String2.h"
00009  #include "FileDesc.h"
00010
00011  typedef struct FileMgr FileMgr;
00012
00013  PUBLIC FileMgr* FileMgr_new();
00014  PUBLIC void FileMgr_delete(FileMgr * this);
00015  PUBLIC FileMgr * FileMgr_copy(FileMgr * this);
00016  PUBLIC void FileMgr_print(FileMgr * this);
00017  PUBLIC String* FileMgr_load(FileMgr* this, const char * fileName);
00018  PUBLIC void FileMgr_write(FileMgr* this, const char* fileName, String* buffer);
00019  PUBLIC void FileMgr_close(FileMgr* this, String* fileName);
00020  PUBLIC unsigned int FileMgr_setRootLocation(FileMgr* this, const char * location);
00021  PUBLIC char * FileMgr_getRootLocation(FileMgr* this);
00022  PUBLIC FileMgr* FileMgr_getRef();
00023  PUBLIC unsigned int FileMgr_getSize(FileMgr * this);
00024  PUBLIC unsigned int FileMgr_addDirectory(FileMgr * this, const char * directoryName);
00025  PUBLIC FileDesc * FileMgr_addFile(FileMgr * this, const char * fileName);
00026  PUBLIC FileDesc* FileMgr_createFile(FileMgr* this, const char* fileName);
00027  PUBLIC List * FileMgr_filterFiles(FileMgr * this, const char * pattern);
00028  PUBLIC FileDesc * FileMgr_searchFile(FileMgr * this, String * name, List * preferredDir);
00029  PUBLIC FileDesc* FileMgr_isManaged(FileMgr* this, String* fullName);
00030  #endif  /* _FILEMGR_H_ */

```

4.6 FileMgr.h

```

00001 /* FileMgr.h */
00002
00003 #ifndef _FILEMGR_H_
00004 #define _FILEMGR_H_
00005
00006 #include "Types.h"
00007 #include "List.h"
00008 #include "String2.h"
00009 #include "FileDesc.h"
00010
00011 typedef struct FileMgr FileMgr;
00012
00013 PUBLIC FileMgr* FileMgr_new();
00014 PUBLIC void FileMgr_delete(FileMgr * this);
00015 PUBLIC FileMgr * FileMgr_copy(FileMgr * this);
00016 PUBLIC void FileMgr_print(FileMgr * this);
00017 PUBLIC String* FileMgr_load(FileMgr* this, const char * fileName);
00018 PUBLIC void FileMgr_write(FileMgr* this, const char* fileName, String* buffer);
00019 PUBLIC void FileMgr_close(FileMgr* this, String* fileName);
00020 PUBLIC unsigned int FileMgr_setRootLocation(FileMgr* this, const char * location);
00021 PUBLIC char * FileMgr_getRootLocation(FileMgr* this);
00022 PUBLIC FileMgr* FileMgr_getRef();
00023 PUBLIC unsigned int FileMgr_getSize(FileMgr * this);
00024 PUBLIC unsigned int FileMgr_addDirectory(FileMgr * this, const char * directoryName);
00025 PUBLIC FileDesc * FileMgr_addFile(FileMgr * this, const char * fileName);
00026 PUBLIC FileDesc* FileMgr_createFile(FileMgr* this, const char* fileName);
00027 PUBLIC List * FileMgr_filterFiles(FileMgr * this, const char * pattern);
00028 PUBLIC FileDesc * FileMgr_searchFile(FileMgr * this, String * name, List * preferredDir);
00029 PUBLIC FileDesc* FileMgr_isManaged(FileMgr* this, String* fullName);
00030 #endif /* _FILEMGR_H_ */

```

4.7 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/OptionMgr/OptionMgr.c File Reference ↩↪

The [OptionMgr](#) class manages the application configuration.

```

#include "OptionMgr.h"
#include "Class.h"
#include "Object.h"
#include "String2.h"
#include "Map.h"
#include "FileMgr.h"
#include "Memory.h"
#include "Error.h"
#include "Debug.h"

```

Classes

- class [OptionMgr](#)
- struct [OptionDefault](#)

Functions

- PRIVATE unsigned int [OptionMgr_parseFile](#) ([OptionMgr](#) *this, [String](#) *fileContent)

4.7.1 Detailed Description

The [OptionMgr](#) class manages the application configuration.

The class [OptionMgr](#) is TBD

4.8 OptionMgr.h

```

00001 /* OptionMgr.h */
00002
00003 #ifndef _OPTIONMGR_H_
00004 #define _OPTIONMGR_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008
00009 typedef struct OptionMgr OptionMgr;
00010
00011 PUBLIC void OptionMgr_delete(OptionMgr * this);
00012 PUBLIC OptionMgr * OptionMgr_copy(OptionMgr * this);
00013 PUBLIC PUBLIC OptionMgr* OptionMgr_getRef();
00014 PUBLIC unsigned int OptionMgr_getSize(OptionMgr * this);
00015 PUBLIC String * OptionMgr_getOption(OptionMgr * this, const char * name);
00016 PUBLIC void OptionMgr_setOption(OptionMgr * this, const char * optionName, String * value);
00017 PUBLIC unsigned int OptionMgr_readFromFile(OptionMgr * this);
00018 PUBLIC unsigned int OptionMgr_readFromCmdLine(OptionMgr * this, const int argc, const char ** argv);
00019 PUBLIC unsigned int OptionMgr_isOptionEnabled(OptionMgr* this, const char * optionName);
00020
00021 #endif /* _OPTIONMGR_H_ */

```

4.9 OptionMgr.h

```

00001 /* OptionMgr.h */
00002
00003 #ifndef _OPTIONMGR_H_
00004 #define _OPTIONMGR_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008
00009 typedef struct OptionMgr OptionMgr;
00010
00011 PUBLIC void OptionMgr_delete(OptionMgr * this);
00012 PUBLIC OptionMgr * OptionMgr_copy(OptionMgr * this);
00013 PUBLIC PUBLIC OptionMgr* OptionMgr_getRef();
00014 PUBLIC unsigned int OptionMgr_getSize(OptionMgr * this);
00015 PUBLIC String * OptionMgr_getOption(OptionMgr * this, const char * name);
00016 PUBLIC void OptionMgr_setOption(OptionMgr * this, const char * optionName, String * value);
00017 PUBLIC unsigned int OptionMgr_readFromFile(OptionMgr * this);
00018 PUBLIC unsigned int OptionMgr_readFromCmdLine(OptionMgr * this, const int argc, const char ** argv);
00019 PUBLIC unsigned int OptionMgr_isOptionEnabled(OptionMgr* this, const char * optionName);
00020
00021 #endif /* _OPTIONMGR_H_ */

```

4.10 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/SdbMgr/SdbMgr.c File Reference

TBD.

```

#include "SdbMgr.h"
#include "Class.h"
#include "Object.h"
#include "String2.h"
#include "Memory.h"
#include "Error.h"
#include "List.h"
#include <sqlite3.h>

```

Classes

- class [SdbMgr](#)

Functions

- PRIVATE unsigned int **SdbMgr_open** ([SdbMgr](#) *this, [String](#) *sdbName)
- PRIVATE void **SdbMgr_close** ([SdbMgr](#) *this)
- PUBLIC unsigned int **SdbMgr_getSize** ([SdbMgr](#) *this)

Variables

- PRIVATE [SdbMgr](#) * **sdbMgr** = 0

4.10.1 Detailed Description

TBD.

TBD

4.11 SdbMgr.h

```
00001 /* SdbMgr.h */
00002
00003 #ifndef _SDBMGR_H_
00004 #define _SDBMGR_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008 #include "List.h"
00009
00010 typedef struct SdbMgr SdbMgr;
00011
00012 PUBLIC SdbMgr * SdbMgr_new(String * name);
00013 PUBLIC void SdbMgr_delete(SdbMgr* this);
00014 PUBLIC SdbMgr * SdbMgr_copy(SdbMgr* this);
00015 PUBLIC SdbMgr * SdbMgr_getRef();
00016 PUBLIC unsigned int SdbMgr_getSize(SdbMgr* this);
00017 PUBLIC unsigned int SdbMgr_execute(SdbMgr* this, const char* statement, List * result);
00018
00019 #endif /* _SDBMGR_H_ */
```

4.12 SdbMgr.h

```
00001 /* SdbMgr.h */
00002
00003 #ifndef _SDBMGR_H_
00004 #define _SDBMGR_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008 #include "List.h"
00009
00010 typedef struct SdbMgr SdbMgr;
00011
00012 PUBLIC SdbMgr * SdbMgr_new(String * name);
00013 PUBLIC void SdbMgr_delete(SdbMgr* this);
00014 PUBLIC SdbMgr * SdbMgr_copy(SdbMgr* this);
00015 PUBLIC SdbMgr * SdbMgr_getRef();
00016 PUBLIC unsigned int SdbMgr_getSize(SdbMgr* this);
00017 PUBLIC unsigned int SdbMgr_execute(SdbMgr* this, const char* statement, List * result);
00018
00019 #endif /* _SDBMGR_H_ */
```

4.13 SdbRequest.h

```

00001 /* SdbRequest.h */
00002 #ifndef _SDBREQUEST_H_
00003 #define _SDBREQUEST_H_
00004
00005 #include "Types.h"
00006 #include "List.h"
00007
00008 typedef struct SdbRequest SdbRequest;
00009
00010 PUBLIC SdbRequest * SdbRequest_new(const char * fmt);
00011 PUBLIC void SdbRequest_delete(SdbRequest * this);
00012 PUBLIC SdbRequest * SdbRequest_copy(SdbRequest * this);
00013 PUBLIC unsigned int SdbRequest_getSize(SdbRequest * this);
00014 PUBLIC void SdbRequest_execute(SdbRequest * this, ...);
00015 PUBLIC unsigned int SdbRequest_getNbResult(SdbRequest * this);
00016 PUBLIC List * SdbRequest_getResults(SdbRequest * this);
00017
00018 #endif /* _SDBREQUEST_H_ */

```

4.14 SdbRequest.h

```

00001 /* SdbRequest.h */
00002 #ifndef _SDBREQUEST_H_
00003 #define _SDBREQUEST_H_
00004
00005 #include "Types.h"
00006 #include "List.h"
00007
00008 typedef struct SdbRequest SdbRequest;
00009
00010 PUBLIC SdbRequest * SdbRequest_new(const char * fmt);
00011 PUBLIC void SdbRequest_delete(SdbRequest * this);
00012 PUBLIC SdbRequest * SdbRequest_copy(SdbRequest * this);
00013 PUBLIC unsigned int SdbRequest_getSize(SdbRequest * this);
00014 PUBLIC void SdbRequest_execute(SdbRequest * this, ...);
00015 PUBLIC unsigned int SdbRequest_getNbResult(SdbRequest * this);
00016 PUBLIC List * SdbRequest_getResults(SdbRequest * this);
00017
00018 #endif /* _SDBREQUEST_H_ */

```

4.15 Storage.h

```

00001 /* Storage.h */
00002 #ifndef _STORAGE_H_
00003 #define _STORAGE_H_
00004
00005 #include "Types.h"
00006
00007 typedef struct Storage Storage;
00008
00009 PUBLIC Storage * Storage_new();
00010 PUBLIC void Storage_delete(Storage * this);
00011 // "SELECT * FROM Include_Nodes WHERE Name='%s'
00012 // "SELECT * FROM Nodes WHERE NodeId=%d;
00013 PUBLIC void Storage_select();
00014 // INSERT INTO Include_Nodes (NodeId, Name, EntryNode)
00015 // INSERT INTO Code_Nodes (NodeId, Code)
00016 // "INSERT INTO Comment_Nodes (NodeId, Comment) "
00017 PUBLIC void Storage_insert();
00018 // UPDATE Include_Nodes SET EntryNode = %d WHERE NodeId = %d;
00019
00020 // UPDATE Nodes SET NodeNext = %d WHERE NodeId = %d;
00021 PUBLIC void Storage_update();
00022
00023 /* "CREATE TABLE Nodes ( "
00024 "NodeId integer PRIMARY_KEY, "
00025 "NodeType integer NOT NULL, "
00026 "NodePtr integer NOT NULL, "
00027 "NodeNext integer, "
00028 "NodePrev integer"
00029 ");");*/
00030 PUBLIC void Storage_create();
00031 // "DROP TABLE "
00032 PUBLIC void Storage_drop();
00033 #endif /* _STORAGE_H_ */
00034

```

4.16 Mutex.h

```

00001 /* Mutex.h */
00002 #ifndef _MUTEX_H_
00003 #define _MUTEX_H_
00004
00005 #include "Types.h"
00006 #ifndef WIN32
00007 #include <pthread.h>
00008 #else
00009 #include <windows.h>
00010 #endif
00011
00012 typedef struct Mutex
00013 {
00014     #ifndef WIN32
00015         pthread_mutex_t *mutex;
00016         pthread_cond_t cond;
00017     #else
00018         HANDLE mutex;
00019     #endif
00020 } Mutex;
00021
00022 PRIVATE void Mutex_new(Mutex * this, int initState);
00023 PRIVATE void Mutex_delete(Mutex * this);
00024 PRIVATE void Mutex_take(Mutex * this);
00025 PRIVATE void Mutex_release(Mutex * this);
00026 PRIVATE void Mutex_waitAvailability(Mutex * this);
00027
00028 PRIVATE void Mutex_new(Mutex * this, int initState)
00029 {
00030     #ifndef WIN32
00031         //int pthread_mutex_init(this->mutex, const pthread_mutexattr_t *restrict attr);
00032         //int pthread_cond_init(pthread_cond_t *restrict cond, const pthread_condattr_t *restrict attr);
00033         //this->cond = PTHREAD_COND_INITIALIZER;
00034     #else
00035         //this->mutex = CreateMutexW(NULL, TRUE, NULL);          // Set
00036     #endif
00037 }
00038
00039 PRIVATE void Mutex_delete(Mutex * this)
00040 {
00041     //int pthread_mutex_destroy(this->mutex);
00042     //int pthread_cond_destroy(pthread_cond_t *cond);
00043     //CloseHandle(this->mutex);
00044     //if (hScreenMutex) CloseHandle(hScreenMutex);
00045     //if (hRunMutex) CloseHandle(hRunMutex);
00046 }
00047
00048 PRIVATE void Mutex_take(Mutex * this)
00049 {
00050     #ifndef WIN32
00051         //int pthread_mutex_lock(pthread_mutex_t *mutex);
00052     #else
00053     #endif
00054 }
00055
00056 PRIVATE void Mutex_release(Mutex * this)
00057 {
00058     #ifndef WIN32
00059         //int pthread_mutex_unlock(pthread_mutex_t *mutex)
00060         //pthread_cond_signal(&condition); //wake up thread 1
00061     #else
00062         //ReleaseMutex(this->mutex);
00063     #endif
00064 }
00065
00066 PRIVATE void Mutex_waitAvailability(Mutex * this)
00067 {
00068     //dwWaitResult = WaitForSingleObject(this->mutex, INFINITE); // no time-out interval
00069     //pthread_cond_wait(&cond, &lock);
00070     //int pthread_cond_timedwait(pthread_cond_t *restrict cond,
00071     //    pthread_mutex_t *restrict mutex,
00072     //    const struct timespec *restrict abstime
00073 }
00074 #endif /* _MUTEX_H_ */

```

4.17 Task.h

```

00001 /* Task.h */
00002 #ifndef _TASK_H_
00003 #define _TASK_H_
00004

```

```

00005 #include "Types.h"
00006
00007 typedef struct Task Task;
00008
00009 PUBLIC Task* Task_create(void * (*body)(void* p), int nbParams, void ** params);
00010 PUBLIC void Task_destroy(Task * this);
00011 PUBLIC void Task_start(Task * this);
00012 PUBLIC int Task_isReady(Task * this);
00013 PUBLIC void Task_setReady(Task * this);
00014 PUBLIC int Task_isRunning(Task * this);
00015 PUBLIC void Task_setRunning(Task * this);
00016 PUBLIC int Task_isCompleted(Task * this);
00017 PUBLIC void Task_setCompleted(Task * this);
00018 PUBLIC void Task_destroy(Task * this);
00019 PUBLIC void Task_executeBody(Task * this);
00020
00021 #endif /* _TASK_H_ */

```

4.18 Task.h

```

00001 /* Task.h */
00002 #ifndef _TASK_H_
00003 #define _TASK_H_
00004
00005 #include "Types.h"
00006
00007 typedef struct Task Task;
00008
00009 PUBLIC Task* Task_create(void * (*body)(void* p), int nbParams, void ** params);
00010 PUBLIC void Task_destroy(Task * this);
00011 PUBLIC void Task_start(Task * this);
00012 PUBLIC int Task_isReady(Task * this);
00013 PUBLIC void Task_setReady(Task * this);
00014 PUBLIC int Task_isRunning(Task * this);
00015 PUBLIC void Task_setRunning(Task * this);
00016 PUBLIC int Task_isCompleted(Task * this);
00017 PUBLIC void Task_setCompleted(Task * this);
00018 PUBLIC void Task_destroy(Task * this);
00019 PUBLIC void Task_executeBody(Task * this);
00020
00021 #endif /* _TASK_H_ */

```

4.19 TaskMgr.h

```

00001 /* TaskMgr.h */
00002 #ifndef _TASKMGR_H_
00003 #define _TASKMGR_H_
00004
00005 #include "Types.h"
00006
00007 typedef struct Task Task;
00008 typedef struct TaskMgr TaskMgr;
00009
00010 PUBLIC TaskMgr * TaskMgr_getRef();
00011 PUBLIC void TaskMgr_delete(TaskMgr * this);
00012 PUBLIC void TaskMgr_print(TaskMgr * this);
00013 PUBLIC unsigned int TaskMgr_getSize(TaskMgr * this);
00014 PUBLIC int TaskMgr_start(TaskMgr * this, Task * task);
00015 PUBLIC void TaskMgr_stop(TaskMgr * this);
00016 #endif /* _TASKMGR_H_ */

```

4.20 TaskMgr.h

```

00001 /* TaskMgr.h */
00002 #ifndef _TASKMGR_H_
00003 #define _TASKMGR_H_
00004
00005 #include "Types.h"
00006
00007 typedef struct Task Task;
00008 typedef struct TaskMgr TaskMgr;
00009
00010 PUBLIC TaskMgr * TaskMgr_getRef();
00011 PUBLIC void TaskMgr_delete(TaskMgr * this);
00012 PUBLIC void TaskMgr_print(TaskMgr * this);

```

```

00013 PUBLIC unsigned int TaskMgr_getSize(TaskMgr * this);
00014 PUBLIC int TaskMgr_start(TaskMgr * this, Task * task);
00015 PUBLIC void TaskMgr_stop(TaskMgr* this);
00016 #endif /* _TASKMGR_H_ */

```

4.21 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/TimeMgr/TimeMgr.c File Reference

This file contains the implementation for the class [TimeMgr](#).

```

#include "TimeMgr.h"
#include "Timer.h"
#include "Class.h"
#include "Object.h"
#include "Map.h"

```

Classes

- class [TimeMgr](#)

Functions

- PUBLIC void [TimeMgr_report](#) ([TimeMgr](#) *this)

Variables

- PRIVATE [TimeMgr](#) * [timeMgr](#) = 0

4.21.1 Detailed Description

This file contains the implementation for the class [TimeMgr](#).

The class [TimeMgr](#) provides an interface to the creation of timers.

4.22 TimeMgr.h

```

00001 /* TimeMgr.h */
00002
00003 #ifndef _TIMEMGR_H_
00004 #define _TIMEMGR_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008
00009 typedef struct TimeMgr TimeMgr;
00010
00011 PUBLIC void TimeMgr_delete(TimeMgr * this);
00012 PUBLIC TimeMgr * TimeMgr_copy(TimeMgr * this);
00013 PUBLIC TimeMgr * TimeMgr_getRef();
00014 PUBLIC unsigned int TimeMgr_getSize(TimeMgr * this);
00015 PUBLIC void TimeMgr_latchTime(TimeMgr * this, String * s);
00016 PUBLIC void TimeMgr_report(TimeMgr * this);
00017
00018 #endif /* _TIMEMGR_H_ */

```


4.23 TimeMgr.h

```

00001 /* TimeMgr.h */
00002
00003 #ifndef _TIMEMGR_H_
00004 #define _TIMEMGR_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008
00009 typedef struct TimeMgr TimeMgr;
00010
00011 PUBLIC void TimeMgr_delete(TimeMgr * this);
00012 PUBLIC TimeMgr * TimeMgr_copy(TimeMgr * this);
00013 PUBLIC TimeMgr * TimeMgr_getRef();
00014 PUBLIC unsigned int TimeMgr_getSize(TimeMgr * this);
00015 PUBLIC void TimeMgr_latchTime(TimeMgr * this, String * s);
00016 PUBLIC void TimeMgr_report(TimeMgr * this);
00017
00018 #endif /* _TIMEMGR_H_ */

```

4.24 Timer.h

```

00001 /* Timer.h */
00002
00003 #include "Types.h"
00004
00005 typedef struct Timer Timer;
00006
00007 PUBLIC Timer * Timer_new();
00008 PUBLIC void Timer_delete(Timer * this);
00009 PUBLIC Timer * Timer_copy(Timer * this);
00010 PUBLIC unsigned int Timer_getSize(Timer * this);
00011 PUBLIC unsigned int Timer_isEqual(Timer * this, Timer * compared);
00012 PUBLIC void Timer_print(Timer * this);
00013 PUBLIC void Timer_latchTime(Timer * this);

```

4.25 Timer.h

```

00001 /* Timer.h */
00002
00003 #include "Types.h"
00004
00005 typedef struct Timer Timer;
00006
00007 PUBLIC Timer * Timer_new();
00008 PUBLIC void Timer_delete(Timer * this);
00009 PUBLIC Timer * Timer_copy(Timer * this);
00010 PUBLIC unsigned int Timer_getSize(Timer * this);
00011 PUBLIC unsigned int Timer_isEqual(Timer * this, Timer * compared);
00012 PUBLIC void Timer_print(Timer * this);
00013 PUBLIC void Timer_latchTime(Timer * this);

```

4.26 Allocator.h

```

00001 /* Allocator.h */
00002 #ifndef _ALLOCATOR_H_
00003 #define _ALLOCATOR_H_
00004
00005 typedef struct Allocator Allocator;
00006
00007 typedef void * (*NewFunction)();
00008 typedef void (*DeleteFunction)(Allocator * allocator);
00009 typedef void* (*AllocateFunction)(Allocator * allocator, unsigned int size);
00010 typedef void (*DeAllocateFunction)(Allocator * allocator, void * ptr);
00011 typedef unsigned int (*ReportFunction)(Allocator * allocator);
00012
00013 struct Allocator
00014 {
00015     NewFunction new;
00016     DeleteFunction delete;
00017     AllocateFunction allocate;
00018     DeAllocateFunction deallocate;
00019     ReportFunction report;
00020     unsigned int nbAllocatedObjects;

```

```

00021 };
00022
00023 Allocator * Allocator_new();
00024 void * Allocator_allocate(Allocator * this, unsigned int size);
00025 //void * Allocator_allocFromClass(Allocator * this, /* Class class*/);
00026 void Allocator_deallocate(Allocator * this, void * ptr);
00027 void Allocator_delete(Allocator * this);
00028 #endif /* _ALLOCATOR_H_ */

```

4.27 Allocator.h

```

00001 /* Allocator.h */
00002 #ifndef _ALLOCATOR_H_
00003 #define _ALLOCATOR_H_
00004
00005 typedef struct Allocator Allocator;
00006
00007 typedef void * (*NewFunction)();
00008 typedef void (*DeleteFunction)(Allocator * allocator);
00009 typedef void* (*AllocateFunction)(Allocator * allocator, unsigned int size);
00010 typedef void (*DeAllocateFunction)(Allocator * allocator, void * ptr);
00011 typedef unsigned int (*ReportFunction)(Allocator * allocator);
00012
00013 struct Allocator
00014 {
00015     NewFunction new;
00016     DeleteFunction delete;
00017     AllocateFunction allocate;
00018     DeAllocateFunction deallocate;
00019     ReportFunction report;
00020     unsigned int nbAllocatedObjects;
00021 };
00022
00023 Allocator * Allocator_new();
00024 void * Allocator_allocate(Allocator * this, unsigned int size);
00025 //void * Allocator_allocFromClass(Allocator * this, /* Class class*/);
00026 void Allocator_deallocate(Allocator * this, void * ptr);
00027 void Allocator_delete(Allocator * this);
00028 #endif /* _ALLOCATOR_H_ */

```

4.28 Malloc.h

```

00001 #ifndef _MALLOC_H_
00002 #define _MALLOC_H_
00003 #include "Allocator.h"
00004 #include "Types.h"
00005
00006 typedef struct Malloc Malloc;
00007
00008 PUBLIC Malloc * Malloc_getRef();
00009 PUBLIC void Malloc_delete(Allocator * this);
00010 PUBLIC void * Malloc_allocate(Allocator * this, unsigned int size);
00011 PUBLIC void Malloc_deallocate(Allocator * this, void * ptr);
00012 PUBLIC unsigned int Malloc_report(Allocator * this);
00013 #endif /* _MALLOC_H_ */

```

4.29 Malloc.h

```

00001 #ifndef _MALLOC_H_
00002 #define _MALLOC_H_
00003 #include "Allocator.h"
00004 #include "Types.h"
00005
00006 typedef struct Malloc Malloc;
00007
00008 PUBLIC Malloc * Malloc_getRef();
00009 PUBLIC void Malloc_delete(Allocator * this);
00010 PUBLIC void * Malloc_allocate(Allocator * this, unsigned int size);
00011 PUBLIC void Malloc_deallocate(Allocator * this, void * ptr);
00012 PUBLIC unsigned int Malloc_report(Allocator * this);
00013 #endif /* _MALLOC_H_ */

```

4.30 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Array/Array.c File Reference

This file contains the implementation of the class [Array](#).

```
#include "Array.h"
#include "Class.h"
#include "Object.h"
#include "Debug.h"
```

Classes

- class [Array](#)

Macros

- #define **NB_ELEMENT_MAX** (100)
- #define **ELEMENT_SIZE_BYTES** (100)

Functions

- PUBLIC unsigned int **Array_getSize** ([Array](#) *this)

4.30.1 Detailed Description

This file contains the implementation of the class [Array](#).

The class [Array](#) implement the [Array](#) operations:

- init
- put
- get

4.31 Array.h

```
00001 #ifndef _ARRAY_H_
00002 #define _ARRAY_H_
00003 /*****
00004  * Array.h
00005  *
00006  *****/
00007 #include "Types.h"
00008 #include "Object.h"
00009 #include "FileIo.h"
00010
00011 typedef struct Array Array;
00012
00013 typedef struct ArrayParam
00014 {
00015     unsigned int defaultSize;
00016     unsigned int storageMode;
00017     unsigned int autoresize;
00018 } ArrayParam;
00019
00020 PUBLIC Array * Array_new(ArrayParam * param);
00021 PUBLIC Array * Array_newFromFile(FileIo * fileIo, ArrayParam * param);
00022 PUBLIC void Array_delete(Array* this);
00023 PUBLIC Array * Array_copy(Array* this);
00024 PUBLIC int Array_compare(Array * this, Array * compared);
00025 PUBLIC void Array_print(Array * this);
00026 PUBLIC void Array_put(Array * this, unsigned int index);
00027 PUBLIC Object * Array_get(Array * this, unsigned int index);
00028 PUBLIC unsigned int Array_getSize(Array * this);
00029
00030 #endif /* _ARRAY_H_ */
```

4.32 Array.h

```

00001 #ifndef _ARRAY_H_
00002 #define _ARRAY_H_
00003 /*****
00004  * Array.h
00005  *
00006  *****/
00007 #include "Types.h"
00008 #include "Object.h"
00009 #include "FileIo.h"
00010
00011 typedef struct Array Array;
00012
00013 typedef struct ArrayParam
00014 {
00015     unsigned int defaultSize;
00016     unsigned int storageMode;
00017     unsigned int autoresize;
00018 } ArrayParam;
00019
00020 PUBLIC Array * Array_new(ArrayParam * param);
00021 PUBLIC Array * Array_newFromFile(FileIo * fileIo, ArrayParam * param);
00022 PUBLIC void Array_delete(Array* this);
00023 PUBLIC Array * Array_copy(Array* this);
00024 PUBLIC int Array_compare(Array * this, Array * compared);
00025 PUBLIC void Array_print(Array * this);
00026 PUBLIC void Array_put(Array * this, unsigned int index);
00027 PUBLIC Object * Array_get(Array * this, unsigned int index);
00028 PUBLIC unsigned int Array_getSize(Array * this);
00029
00030 #endif /* _ARRAY_H_ */

```

4.33 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/BTree.c File Reference

This file contains the implementation of the class [BTree](#).

```

#include "BTree.h"
#include "Node.h"
#include "Memory.h"
#include "Debug.h"

```

Classes

- class [BTree](#)

Functions

- PUBLIC [BTree](#) * **BTree_new** (unsigned int order)
- PUBLIC void **BTree_delete** ([BTree](#) *this)
- PUBLIC [BTree](#) * **BTree_copy** ([BTree](#) *this)
- PUBLIC int **BTree_comp** ([BTree](#) *this, [BTree](#) *compared)
- PUBLIC void **BTree_add** ([BTree](#) *tree, [Object](#) *key, [Object](#) *object, int isOwner)
- PUBLIC [Object](#) * **BTree_get** ([BTree](#) *tree, [Object](#) *key)
- PUBLIC [Object](#) * **BTree_remove** ([BTree](#) *tree, [Object](#) *key)
- PUBLIC void **BTree_print** ([BTree](#) *tree)
- PUBLIC [BTree](#) * **BTree_newFromFile** (char *fileName)
- PUBLIC unsigned int **BTree_getSize** ([BTree](#) *this)
- PUBLIC unsigned int **BTree_getNbNodes** ([BTree](#) *this)

4.33.1 Detailed Description

This file contains the implementation of the class [BTree](#).

The class [BTree](#) implements the [BTree](#) operations:

- init
- add
- remove

4.34 BTree.h

```
00001 #ifndef _BTREE_
00002 #define _BTREE_
00003 /*****
00004  * BTree.h
00005  *
00006  *****/
00007 #include "Types.h"
00008 #include "Object.h"
00009 #include "Allocator.h"
00010
00011 typedef struct BTree BTree;
00012
00013 PUBLIC BTree * BTree_new(unsigned int order);
00014 PUBLIC BTree * BTree_newFromAllocator(Allocator * allocator);
00015 PUBLIC BTree * BTree_newFromFile(char* fileName);
00016 PUBLIC void BTree_delete(BTree * tree);
00017 PUBLIC BTree * BTree_copy(BTree * this);
00018 PUBLIC int BTree_comp(BTree * this, BTree * compared);
00019 PUBLIC void BTree_add(BTree * tree, Object * key, Object * object, int isOwner);
00020 PUBLIC Object * BTree_get(BTree * tree, Object * key);
00021 PUBLIC Object * BTree_remove(BTree * tree, Object * key);
00022 PUBLIC void BTree_print(BTree * tree);
00023 PUBLIC unsigned int BTree_sizeof(BTree* tree);
00024 PUBLIC unsigned int BTree_reportSizeInBytes(BTree * tree);
00025 PUBLIC unsigned int BTree_getSize(BTree * this);
00026 PUBLIC unsigned int BTree_getNbNodes(BTree* this);
00027
00028 #endif /* _BTREE_ */
00029
```

4.35 BTree.h

```
00001 #ifndef _BTREE_
00002 #define _BTREE_
00003 /*****
00004  * BTree.h
00005  *
00006  *****/
00007 #include "Types.h"
00008 #include "Object.h"
00009 #include "Allocator.h"
00010
00011 typedef struct BTree BTree;
00012
00013 PUBLIC BTree * BTree_new(unsigned int order);
00014 PUBLIC BTree * BTree_newFromAllocator(Allocator * allocator);
00015 PUBLIC BTree * BTree_newFromFile(char* fileName);
00016 PUBLIC void BTree_delete(BTree * tree);
00017 PUBLIC BTree * BTree_copy(BTree * this);
00018 PUBLIC int BTree_comp(BTree * this, BTree * compared);
00019 PUBLIC void BTree_add(BTree * tree, Object * key, Object * object, int isOwner);
00020 PUBLIC Object * BTree_get(BTree * tree, Object * key);
00021 PUBLIC Object * BTree_remove(BTree * tree, Object * key);
00022 PUBLIC void BTree_print(BTree * tree);
00023 PUBLIC unsigned int BTree_sizeof(BTree* tree);
00024 PUBLIC unsigned int BTree_reportSizeInBytes(BTree * tree);
00025 PUBLIC unsigned int BTree_getSize(BTree * this);
00026 PUBLIC unsigned int BTree_getNbNodes(BTree* this);
00027
00028 #endif /* _BTREE_ */
00029
```

4.36 Node.h

```

00001 /*
00002  * Node.h
00003  */
00004 #ifndef _NODE_
00005 #define _NODE_
00006
00007 #include "Types.h"
00008 #include "Object.h"
00009
00010 typedef struct Node Node;
00011
00012 typedef struct Node
00013 {
00014     unsigned int nbKeyUsed;
00015     unsigned int isLeaf;
00016     Object ** keys;
00017     Object ** leaves;
00018     Node ** children;
00019     Object* buffer[18];
00020 } Node;
00021
00022 PUBLIC Node * Node_new(unsigned short int isLeaf, unsigned int order);
00023 PUBLIC Node* Node_splitNode(Node * node, unsigned int order, Node* nodeToSplit, Object * key);
00024 PUBLIC void Node_insert(Node * node, unsigned int order, Object * key, Object * object, int isOwner);
00025 PUBLIC Object * Node_remove(Node * node, unsigned int order, Object * key, Object ** keyToUpdate);
00026 PUBLIC Object * Node_search(Node * node, unsigned int order, Object * key, unsigned int
    isFoundAlready);
00027 PUBLIC void Node_free(Node * node, unsigned int order);
00028 PUBLIC void Node_print(Node * node, unsigned int order, unsigned int depth);
00029 PUBLIC unsigned int Node_getNbNodes(Node * node);
00030 #endif /* _NODE_ */
00031

```

4.37 Node.h

```

00001 /*
00002  * Node.h
00003  */
00004 #ifndef _NODE_
00005 #define _NODE_
00006
00007 #include "Types.h"
00008 #include "Object.h"
00009
00010 typedef struct Node Node;
00011
00012 typedef struct Node
00013 {
00014     unsigned int nbKeyUsed;
00015     unsigned int isLeaf;
00016     Object ** keys;
00017     Object ** leaves;
00018     Node ** children;
00019     Object* buffer[18];
00020 } Node;
00021
00022 PUBLIC Node * Node_new(unsigned short int isLeaf, unsigned int order);
00023 PUBLIC Node* Node_splitNode(Node * node, unsigned int order, Node* nodeToSplit, Object * key);
00024 PUBLIC void Node_insert(Node * node, unsigned int order, Object * key, Object * object, int isOwner);
00025 PUBLIC Object * Node_remove(Node * node, unsigned int order, Object * key, Object ** keyToUpdate);
00026 PUBLIC Object * Node_search(Node * node, unsigned int order, Object * key, unsigned int
    isFoundAlready);
00027 PUBLIC void Node_free(Node * node, unsigned int order);
00028 PUBLIC void Node_print(Node * node, unsigned int order, unsigned int depth);
00029 PUBLIC unsigned int Node_getNbNodes(Node * node);
00030 #endif /* _NODE_ */
00031

```

4.38 TestObject.h

```

00001 /* TestObject.h */
00002 #ifndef _TESTOBJECT_H_
00003 #define _TESTOBJECT_H_
00004
00005 #include "Types.h"
00006 #include "Allocator.h"
00007

```

```

00008 typedef struct TestObject TestObject;
00009
00010 PUBLIC TestObject * TestObject_new();
00011 PUBLIC TestObject * TestObject_newFromAllocator(Allocator* allocator);
00012 PUBLIC void TestObject_delete(TestObject * this);
00013 PUBLIC int TestObject_compare(TestObject* this, TestObject* compare);
00014 PUBLIC TestObject* TestObject_copy();
00015 PUBLIC void TestObject_print(TestObject* this);
00016 PUBLIC unsigned int TestObject_getSize(TestObject* this);
00017 #endif /* _TESTOBJECT_H_ */

```

4.39 TestObject.h

```

00001 /* TestObject.h */
00002 #ifndef _TESTOBJECT_H_
00003 #define _TESTOBJECT_H_
00004
00005 #include "Types.h"
00006 #include "Allocator.h"
00007
00008 typedef struct TestObject TestObject;
00009
00010 PUBLIC TestObject * TestObject_new();
00011 PUBLIC TestObject * TestObject_newFromAllocator(Allocator* allocator);
00012 PUBLIC void TestObject_delete(TestObject * this);
00013 PUBLIC int TestObject_compare(TestObject* this, TestObject* compare);
00014 PUBLIC TestObject* TestObject_copy();
00015 PUBLIC void TestObject_print(TestObject* this);
00016 PUBLIC unsigned int TestObject_getSize(TestObject* this);
00017 #endif /* _TESTOBJECT_H_ */

```

4.40 TestObject.h

```

00001 /* TestObject.h */
00002 #ifndef _TESTOBJECT_H_
00003 #define _TESTOBJECT_H_
00004
00005 #include "Types.h"
00006 #include "Allocator.h"
00007
00008 typedef struct TestObject TestObject;
00009
00010 PUBLIC TestObject * TestObject_new();
00011 PUBLIC TestObject * TestObject_newFromAllocator(Allocator* allocator);
00012 PUBLIC void TestObject_delete(TestObject * this);
00013 PUBLIC int TestObject_compare(TestObject* this, TestObject* compare);
00014 PUBLIC TestObject* TestObject_copy();
00015 PUBLIC void TestObject_print(TestObject* this);
00016 PUBLIC unsigned int TestObject_getSize(TestObject* this);
00017 #endif /* _TESTOBJECT_H_ */

```

4.41 TestObject.h

```

00001 /* TestObject.h */
00002 #ifndef _TESTOBJECT_H_
00003 #define _TESTOBJECT_H_
00004
00005 #include "Types.h"
00006 #include "Allocator.h"
00007
00008 typedef struct TestObject TestObject;
00009
00010 PUBLIC TestObject * TestObject_new();
00011 PUBLIC TestObject * TestObject_newFromAllocator(Allocator* allocator);
00012 PUBLIC void TestObject_delete(TestObject * this);
00013 PUBLIC int TestObject_compare(TestObject* this, TestObject* compare);
00014 PUBLIC TestObject* TestObject_copy();
00015 PUBLIC void TestObject_print(TestObject* this);
00016 PUBLIC unsigned int TestObject_getSize(TestObject* this);
00017 #endif /* _TESTOBJECT_H_ */

```

4.42 Words1000.h

```

00001 /* Words1000.h */
00002
00003 #ifndef _WORDS1000_H_
00004 #define _WORDS1000_H_
00005
00006 char words1000[] = "hang holistic oceanic development decorous fence question songs chase relation
adamant guitar writer ruthless "
00007 "harass unnatural invite bite aloof numberless answer yam carriage apparatus graceful wonderful
demonic gruesome "
00008 "imaginary bewildered hypnotic deer therapeutic nasty relax righteous apathetic wrong mother clever
impress mix harm "
00009 "aspiring hook drip frog fairies workable mass monkey key sneeze window chunky ambiguous ice well-off
matter thoughtful "
00010 "number cure messy frame mere fax cheat shoes voiceless alert kind woozy impulse wander kick beds
elated drop north babies "
00011 "cumbersome sense theory note calculator yummy paper enormous drink slope agreement jump incredible
oafish aboriginal deeply "
00012 "animal pump cooperative kittens letters support pies connection excellent peaceful fertile lavish
efficacious office "
00013 "statement puncture many tin good overwrought believe rush remarkable keen abounding uncle aromatic
tight dizzy taste lumber "
00014 "cars obey stitch poke embarrass hard-to-find glossy grumpy fortunate fire elderly puny repeat jumpy
erect ignorant tongue "
00015 "fine oranges rejoice madly border squirrel equable baseball size zoo umbrella belong hole mountain
cake volatile hungry "
00016 "flesh attract tumble advice vague unequal narrow victorious optimal pushy trashy hop blush warn
earthquake houses undesirable "
00017 "harmonious same punch motionless allow children color homely large subsequent instruct flowery
stretch property advertisement "
00018 "page straight frightening glistening joyous high-pitched ticket reading dream can uncovered low
unbecoming chivalrous belief "
00019 "pretty nose crush refuse fence stew blot colour envious wandering oven art famous witty two hard
kaput cat move delightful "
00020 "memorize berserk inject wise kind monkey flow release industrious substantial rainy last agreeable
wealth celery choke protect "
00021 "existence pick far-flung ruddy fire trick knee humor mess up occur grade back nail help zonked spare
crowd right sail left "
00022 "stare crooked instrument night recondite subdued pray deadpan knotty locket creator pastoral
plausible ambiguous rest nondescript "
00023 "fowl unused macabre innate push torpid educated penitent chase governor plantation debonair irate
faint crawl thundering judicious "
00024 "righteous sleep bury mind button wax effect afterthought pushy radiate tender glass business stale
filthy tranquil jellyfish "
00025 "rule gaping finicky fall suggestion diligent sordid bless gate grey whole eight willing scratch
houses hall north notice "
00026 "distribution song nasty amazing fax card fog pie whip utter troubled cause gifted paddle difficult
fish bee engine enchanting "
00027 "ring familiar glove design save medical mark obeisant functional yam likeable kneel desire ossified
toe subtract maniacal analyze "
00028 "long-term run available enter alive vulgar grouchy thinkable pan fragile laughable quarrelsome tasty
money finger tongue confuse "
00029 "beam calculator return industry gratis whine neighborly coherent prevent past haircut flame gaze
reading dream can uncovered "
00030 "low unbecoming chivalrous belief pretty nose crush refuse fence stew blot colour envious wandering
oven art famous witty two "
00031 "hard kaput cat move delightful memorize berserk inject wise kind monkey flow release industrious
substantial rainy last agreeable "
00032 "wealth celery choke protect existence pick far-flung ruddy fire trick knee humor mess up occur grade
back nail help zonked spare "
00033 "crowd right sail left stare crooked instrument night recondite subdued pray deadpan knotty locket
creator pastoral plausible "
00034 "ambiguous rest nondescript fowl unused macabre innate push torpid educated penitent chase governor
plantation debonair irate "
00035 "faint crawl thundering judicious righteous sleep bury mind button wax effect afterthought pushy
radiate tender glass business "
00036 "stale filthy tranquil jellyfish rule gaping finicky fall suggestion diligent sordid bless gate grey
whole eight willing scratch "
00037 "houses hall north notice distribution song nasty amazing fax card fog pie whip utter troubled cause
gifted paddle difficult fish "
00038 "bee engine enchanting ring familiar glove design save medical mark obeisant functional yam likeable
kneel desire ossified toe "
00039 "subtract maniacal analyze long-term run available enter alive vulgar grouchy thinkable pan fragile
laughable quarrelsome tasty "
00040 "money finger tongue confuse beam calculator return industry gratis whine neighborly coherent prevent
past haircut flame gaze "
00041 "yak delay duck confuse wave subdued trot cumbersome burst rainy bang dysfunctional remain lyrical
yummy red fluttering plucky "
00042 "winter scandalous grotesque celery nimble powder spiffy hope fabulous terrible interrupt diligent
daffy watch short flash ambiguous "
00043 "distinct geese offend scarecrow jump quaint inquisitive hysterical wide bait wax wish children tap
infamous pale flagrant "
00044 "unbecoming rescue canvas cast harass snails person distance succinct watery curious familiar lonely
oatmeal approve spell meddle "
00045 "kettle vein plausible jazzy quicksand hard object luxuriant instinctive approval important spill
bitter fixed level gusty concern "

```



```

00046 "courageous argument name hydrant gruesome fireman mellow invite woozy risk quarter zoo bed rigid
insurance puzzling amusement "
00047 "spooky weather authority bite sincere scale quince grouchy left song aware wretched ear magenta story
blue holiday sudden "
00048 "direction friendly gray earn befitting cap fall suggestion regular tumble park slip jeans stomach
warn appliance pink false "
00049 "double airport cheer glib demonic hollow teeny-tiny year dance advertisement tub brown thankful
chicken book attend crook "
00050 "saw wry entertaining pets spurious chess wood uninterested macho lacking hands stingy fuzzy super
error clover agreement erratic "
00051 "explain drip four finger condemned creature spade root downtown materialistic concerned decide bat
mate silly breathe psychotic "
00052 "stiff phobic front disarm rob pop health assorted public known clean lunchroom cause scissors dynamic
tearful scene range "
00053 "elite outgoing giraffe payment seat acrid trees entertain rabid north sidewalk song cloth tasty
grotesque icky stew grip cherry "
00054 "mighty push hushed short ashamed plate teeny-tiny sparkle lopsided spoon shelf release puffy abaft
record deeply zephyr found "
00055 "lame rude colorful advise kill border big line slim unhealthy bake love unique crash hospitable warn
vast picture oatmeal "
00056 "harbor ill crazy orange expensive afford whimsical scream ship sudden relieved grubby last add blind
zealous obsequious screw "
00057 "jar science throat answer daughter annoyed riddle itchy questionable throne shiny confused vulgar
squirrel jaded vivacious "
00058 "cute snow order chief smash wing night defeated murky keen hesitant neat believe rampant lake voyage
bubble tick current "
00059 "quarter dirty blot wander care rambunctious spoil scary treatment trashy mine expect trick laughable
undress rub selective "
00060 "channel delay hammer overconfident quince craven befitting colour arch mask knit jumpy bouncy
psychotic belligerent stare "
00061 "understood freezing milky long accidental attach hot discreet ready house string maddening loutish
spurious door birthday gabby "
00062 "mute step spotless scrape explain used carpenter heavy argument bump paste feigned coherent driving
stroke separate tenuous "
00063 "letters analyze unarmed bawdy different absurd six hurt shame fast pour church vague real whole";
00064
00065
00066 #endif /* _WORDS1000_H_ */

```

4.43 Words1000.h

```

00001 /* Words1000.h */
00002
00003 #ifndef _WORDS1000_H_
00004 #define _WORDS1000_H_
00005
00006 char words1000[] = "hang holistic oceanic development decorous fence question songs chase relation
adamant guitar writer ruthless "
00007 "harass unnatural invite bite aloof numberless answer yam carriage apparatus graceful wonderful
demonic gruesome "
00008 "imaginary bewildered hypnotic deer therapeutic nasty relax righteous apathetic wrong mother clever
impress mix harm "
00009 "aspiring hook drip frog fairies workable mass monkey key sneeze window chunky ambiguous ice well-off
matter thoughtful "
00010 "number cure messy frame mere fax cheat shoes voiceless alert kind woozy impulse wander kick beds
elated drop north babies "
00011 "cumbersome sense theory note calculator yummy paper enormous drink slope agreement jump incredible
oafish aboriginal deeply "
00012 "animal pump cooperative kittens letters support pies connection excellent peaceful fertile lavish
efficacious office "
00013 "statement puncture many tin good overwrought believe rush remarkable keen abounding uncle aromatic
tight dizzy taste lumber "
00014 "cars obey stitch poke embarrass hard-to-find glossy grumpy fortunate fire elderly puny repeat jumpy
erect ignorant tongue "
00015 "fine oranges rejoice madly border squirrel equable baseball size zoo umbrella belong hole mountain
cake volatile hungry "
00016 "flesh attract tumble advice vague unequal narrow victorious optimal pushy trashy hop blush warn
earthquake houses undesirable "
00017 "harmonious same punch motionless allow children color homely large subsequent instruct flowery
stretch property advertisement "
00018 "page straight frightening glistening joyous high-pitched ticket reading dream can uncovered low
unbecoming chivalrous belief "
00019 "pretty nose crush refuse fence stew blot colour envious wandering oven art famous witty two hard
kaput cat move delightful "
00020 "memorize berserk inject wise kind monkey flow release industrious substantial rainy last agreeable
wealth celery choke protect "
00021 "existence pick far-flung ruddy fire trick knee humor mess up occur grade back nail help zonked spare
crowd right sail left "
00022 "stare crooked instrument night recondite subdued pray deadpan knotty locket creator pastoral
plausible ambiguous rest nondescript "
00023 "fowl unused macabre innate push torpid educated penitent chase governor plantation debonair irate
faint crawl thundering judicious "
00024 "righteous sleep bury mind button wax effect afterthought pushy radiate tender glass business stale
filthy tranquil jellyfish "

```

```

00025 "rule gaping finicky fall suggestion diligent sordid bless gate grey whole eight willing scratch
houses hall north notice "
00026 "distribution song nasty amazing fax card fog pie whip utter troubled cause gifted paddle difficult
fish bee engine enchanting "
00027 "ring familiar glove design save medical mark obeisant functional yam likeable kneel desire ossified
toe subtract maniacal analyze "
00028 "long-term run available enter alive vulgar grouchy thinkable pan fragile laughable quarrelsome tasty
money finger tongue confuse "
00029 "beam calculator return industry gratis whine neighborly coherent prevent past haircut flame gaze
reading dream can uncovered "
00030 "low unbecoming chivalrous belief pretty nose crush refuse fence stew blot colour envious wandering
oven art famous witty two "
00031 "hard kaput cat move delightful memorize berserk inject wise kind monkey flow release industrious
substantial rainy last agreeable "
00032 "wealth celery choke protect existence pick far-flung ruddy fire trick knee humor mess up occur grade
back nail help zonked spare "
00033 "crowd right sail left stare crooked instrument night recondite subdued pray deadpan knotty locket
creator pastoral plausible "
00034 "ambiguous rest nondescript fowl unused macabre innate push torpid educated penitent chase governor
plantation debonair irate "
00035 "faint crawl thundering judicious righteous sleep bury mind button wax effect afterthought pushy
radiate tender glass business "
00036 "stale filthy tranquil jellyfish rule gaping finicky fall suggestion diligent sordid bless gate grey
whole eight willing scratch "
00037 "houses hall north notice distribution song nasty amazing fax card fog pie whip utter troubled cause
gifted paddle difficult fish "
00038 "bee engine enchanting ring familiar glove design save medical mark obeisant functional yam likeable
kneel desire ossified toe "
00039 "subtract maniacal analyze long-term run available enter alive vulgar grouchy thinkable pan fragile
laughable quarrelsome tasty "
00040 "money finger tongue confuse beam calculator return industry gratis whine neighborly coherent prevent
past haircut flame gaze "
00041 "yak delay duck confuse wave subdued trot cumbersome burst rainy bang dysfunctional remain lyrical
yummy red fluttering plucky "
00042 "winter scandalous grotesque celery nimble powder spiffy hope fabulous terrible interrupt diligent
daffy watch short flash ambiguous "
00043 "distinct geese offend scarecrow jump quaint inquisitive hysterical wide bait wax wish children tap
infamous pale flagrant "
00044 "unbecoming rescue canvas cast harass snails person distance succinct watery curious familiar lonely
oatmeal approve spell meddle "
00045 "kettle vein plausible jazzy quicksand hard object luxuriant instinctive approval important spill
bitter fixed level gusty concern "
00046 "courageous argument name hydrant gruesome fireman mellow invite woozy risk quarter zoo bed rigid
insurance puzzling amusement "
00047 "spooky weather authority bite sincere scale quince grouchy left song aware wretched ear magenta story
blue holiday sudden "
00048 "direction friendly gray earn befitting cap fall suggestion regular tumble park slip jeans stomach
warn appliance pink false "
00049 "double airport cheer glib demonic hollow teeny-tiny year dance advertisement tub brown thankful
chicken book attend crook "
00050 "saw wry entertaining pets spurious chess wood uninterested macho lacking hands stingy fuzzy super
error clover agreement erratic "
00051 "explain drip four finger condemned creature spade root downtown materialistic concerned decide bat
mate silly breathe psychotic "
00052 "stiff phobic front disarm rob pop health assorted public known clean lunchroom cause scissors dynamic
tearful scene range "
00053 "elite outgoing giraffe payment seat acrid trees entertain rabid north sidewalk song cloth tasty
grotesque icky stew grip cherry "
00054 "mighty push hushed short ashamed plate teeny-tiny sparkle lopsided spoon shelf release puffy abaft
record deeply zephyr found "
00055 "lame rude colorful advise kill border big line slim unhealthy bake love unique crash hospitable warn
vast picture oatmeal "
00056 "harbor ill crazy orange expensive afford whimsical scream ship sudden relieved grubby last add blind
zealous obsequious screw "
00057 "jar science throat answer daughter annoyed riddle itchy questionable throne shiny confused vulgar
squirrel jaded vivacious "
00058 "cute snow order chief smash wing night defeated murky keen hesitant neat believe rampant lake voyage
bubble tick current "
00059 "quarter dirty blot wander care rambunctious spoil scary treatment trashy mine expect trick laughable
undress rub selective "
00060 "channel delay hammer overconfident quince craven befitting colour arch mask knit jumpy bouncy
psychotic belligerent stare "
00061 "understood freezing milky long accidental attach hot discreet ready house string maddening loutish
spurious door birthday gabby "
00062 "mute step spotless scrape explain used carpenter heavy argument bump paste feigned coherent driving
stroke separate tenuous "
00063 "letters analyze unarmed bawdy different absurd six hurt shame fast pour church vague real whole";
00064
00065
00066 #endif /* _WORDS1000_H_ */

```

4.44 Words1000.h

```
00001 /* Words1000.h */
```

```

00002
00003 #ifndef _WORDS1000_H_
00004 #define _WORDS1000_H_
00005
00006 const char words1000[] = "hang holistic oceanic development decorous fence question songs chase
relation adamant guitar writer ruthless "
00007 "harass unnatural invite bite aloof numberless answer yam carriage apparatus graceful wonderful
demonic gruesome "
00008 "imaginary bewildered hypnotic deer therapeutic nasty relax righteous apathetic wrong mother clever
impress mix harm "
00009 "aspiring hook drip frog fairies workable mass monkey key sneeze window chunky ambiguous ice well-off
matter thoughtful "
00010 "number cure messy frame mere fax cheat shoes voiceless alert kind woozy impulse wander kick beds
elated drop north babies "
00011 "cumbersome sense theory note calculator yummy paper enormous drink slope agreement jump incredible
oafish aboriginal deeply "
00012 "animal pump cooperative kittens letters support pies connection excellent peaceful fertile lavish
efficacious office "
00013 "statement puncture many tin good overwrought believe rush remarkable keen abounding uncle aromatic
tight dizzy taste lumber "
00014 "cars obey stitch poke embarrass hard-to-find glossy grumpy fortunate fire elderly puny repeat jumpy
erect ignorant tongue "
00015 "fine oranges rejoice madly border squirrel equable baseball size zoo umbrella belong hole mountain
cake volatile hungry "
00016 "flesh attract tumble advice vague unequal narrow victorious optimal pushy trashy hop blush warn
earthquake houses undesirable "
00017 "harmonious same punch motionless allow children color homely large subsequent instruct flowery
stretch property advertisement "
00018 "page straight frightening glistening joyous high-pitched ticket reading dream can uncovered low
unbecoming chivalrous belief "
00019 "pretty nose crush refuse fence stew blot colour envious wandering oven art famous witty two hard
kaput cat move delightful "
00020 "memorize berserk inject wise kind monkey flow release industrious substantial rainy last agreeable
wealth celery choke protect "
00021 "existence pick far-flung ruddy fire trick knee humor mess up occur grade back nail help zonked spare
crowd right sail left "
00022 "stare crooked instrument night recondite subdued pray deadpan knotty locket creator pastoral
plausible ambiguous rest nondescript "
00023 "fowl unused macabre innate push torpid educated penitent chase governor plantation debonair irate
faint crawl thundering judicious "
00024 "righteous sleep bury mind button wax effect afterthought pushy radiate tender glass business stale
filthy tranquil jellyfish "
00025 "rule gaping finicky fall suggestion diligent sordid bless gate grey whole eight willing scratch
houses hall north notice "
00026 "distribution song nasty amazing fax card fog pie whip utter troubled cause gifted paddle difficult
fish bee engine enchanting "
00027 "ring familiar glove design save medical mark obeisant functional yam likeable kneel desire ossified
toe subtract maniacal analyze "
00028 "long-term run available enter alive vulgar grouchy thinkable pan fragile laughable quarrelsome tasty
money finger tongue confuse "
00029 "beam calculator return industry gratis whine neighborly coherent prevent past haircut flame gaze
reading dream can uncovered "
00030 "low unbecoming chivalrous belief pretty nose crush refuse fence stew blot colour envious wandering
oven art famous witty two "
00031 "hard kaput cat move delightful memorize berserk inject wise kind monkey flow release industrious
substantial rainy last agreeable "
00032 "wealth celery choke protect existence pick far-flung ruddy fire trick knee humor mess up occur grade
back nail help zonked spare "
00033 "crowd right sail left stare crooked instrument night recondite subdued pray deadpan knotty locket
creator pastoral plausible "
00034 "ambiguous rest nondescript fowl unused macabre innate push torpid educated penitent chase governor
plantation debonair irate "
00035 "faint crawl thundering judicious righteous sleep bury mind button wax effect afterthought pushy
radiate tender glass business "
00036 "stale filthy tranquil jellyfish rule gaping finicky fall suggestion diligent sordid bless gate grey
whole eight willing scratch "
00037 "houses hall north notice distribution song nasty amazing fax card fog pie whip utter troubled cause
gifted paddle difficult fish "
00038 "bee engine enchanting ring familiar glove design save medical mark obeisant functional yam likeable
kneel desire ossified toe "
00039 "subtract maniacal analyze long-term run available enter alive vulgar grouchy thinkable pan fragile
laughable quarrelsome tasty "
00040 "money finger tongue confuse beam calculator return industry gratis whine neighborly coherent prevent
past haircut flame gaze "
00041 "yak delay duck confuse wave subdued trot cumbersome burst rainy bang dysfunctional remain lyrical
yummy red fluttering plucky "
00042 "winter scandalous grotesque celery nimble powder spiffy hope fabulous terrible interrupt diligent
daffy watch short flash ambiguous "
00043 "distinct geese offend scarecrow jump quaint inquisitive hysterical wide bait wax wish children tap
infamous pale flagrant "
00044 "unbecoming rescue canvas cast harass snails person distance succinct watery curious familiar lonely
oatmeal approve spell meddle "
00045 "kettle vein plausible jazzy quicksand hard object luxuriant instinctive approval important spill
bitter fixed level gusty concern "
00046 "courageous argument name hydrant gruesome fireman mellow invite woozy risk quarter zoo bed rigid
insurance puzzling amusement "
00047 "spooky weather authority bite sincere scale quince grouchy left song aware wretched ear magenta story

```

```

    blue holiday sudden "
00048 "direction friendly gray earn befitting cap fall suggestion regular tumble park slip jeans stomach
    warn appliance pink false "
00049 "double airport cheer glib demonic hollow teeny-tiny year dance advertisement tub brown thankful
    chicken book attend crook "
00050 "saw wry entertaining pets spurious chess wood uninterested macho lacking hands stingy fuzzy super
    error clover agreement erratic "
00051 "explain drip four finger condemned creature spade root downtown materialistic concerned decide bat
    mate silly breathe psychotic "
00052 "stiff phobic front disarm rob pop health assorted public known clean lunchroom cause scissors dynamic
    tearful scene range "
00053 "elite outgoing giraffe payment seat acrid trees entertain rabid north sidewalk song cloth tasty
    grotesque icky stew grip cherry "
00054 "mighty push hushed short ashamed plate teeny-tiny sparkle lopsided spoon shelf release puffy abaft
    record deeply zephyr found "
00055 "lame rude colorful advise kill border big line slim unhealthy bake love unique crash hospitable warn
    vast picture oatmeal "
00056 "harbor ill crazy orange expensive afford whimsical scream ship sudden relieved grubby last add blind
    zealous obsequious screw "
00057 "jar science throat answer daughter annoyed riddle itchy questionable throne shiny confused vulgar
    squirrel jaded vivacious "
00058 "cute snow order chief smash wing night defeated murky keen hesitant neat believe rampant lake voyage
    bubble tick current "
00059 "quarter dirty blot wander care rambunctious spoil scary treatment trashy mine expect trick laughable
    undress rub selective "
00060 "channel delay hammer overconfident quince craven befitting colour arch mask knit jumpy bouncy
    psychotic belligerent stare "
00061 "understood freezing milky long accidental attach hot discreet ready house string maddening loutish
    spurious door birthday gabby "
00062 "mute step spotless scrape explain used carpenter heavy argument bump paste feigned coherent driving
    stroke separate tenuous "
00063 "letters analyze unarmed bawdy different absurd six hurt shame fast pour church vague real whole";
00064
00065
00066 #endif /* _WORDS1000_H_ */

```

4.45 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Debug/Debug.c File Reference

This file contains debugging functions.

```

#include "Debug.h"
#include <stdio.h>

```

Functions

- void **dbg_printf** (const char *fmt,...)

4.45.1 Detailed Description

This file contains debugging functions.

The debugging function are TBD

4.46 Debug.h

```

00001 /* Debug.h */
00002
00003 #include <stdarg.h>
00004 #include <stdio.h>
00005
00006 #define TRACE(x) do { if (DEBUG) dbg_printf x; } while (0)
00007
00008 #define PRINT(x) do { dbg_printf x; } while (0)
00009
00010 void dbg_printf(const char *fmt, ...);

```

4.47 Debug.h

```

00001 /* Debug.h */
00002
00003 #include <stdarg.h>
00004 #include <stdio.h>
00005
00006 #define TRACE(x) do { if (DEBUG) dbg_printf x; } while (0)
00007
00008 #define PRINT(x) do { dbg_printf x; } while (0)
00009
00010 void dbg_printf(const char *fmt, ...);

```

4.48 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Error/Error.c File Reference

Reports errors.

```

#include "Error.h"
#include "Debug.h"
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>

```

Macros

- #define **DEBUG** (1)

Functions

- PUBLIC void [Error_new](#) (ErrorSeverity severity, char *msg,...)
Reports errors.

4.48.1 Detailed Description

Reports errors.

This file contains error reporting functions.

4.48.2 Function Documentation

4.48.2.1 Error_new()

```

PUBLIC void Error_new (
    ErrorSeverity severity,
    char * msg,
    ... )

```

Reports errors.

Parameters

<i>severity</i>	Enum
<i>msg</i>	Variable list of parameters

This function reports errors using different formatting according to severity.

4.49 Error.h

```

00001 /* Error.h */
00002
00003 #include "Types.h"
00004
00005 typedef enum
00006 {
00007     ERROR_DBG,
00008     ERROR_INFO,
00009     ERROR_NORMAL,
00010     ERROR_FATAL
00011 } ErrorSeverity;
00012
00013 PUBLIC void Error_new(ErrorSeverity severity, char * msg, ...);

```

4.50 Error.h

```

00001 /* Error.h */
00002
00003 #include "Types.h"
00004
00005 typedef enum
00006 {
00007     ERROR_DBG,
00008     ERROR_INFO,
00009     ERROR_NORMAL,
00010     ERROR_FATAL
00011 } ErrorSeverity;
00012
00013 PUBLIC void Error_new(ErrorSeverity severity, char * msg, ...);

```

**4.51 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/↵
Filelo/Filelo.c File Reference**

A **Filelo** class. This class provides a status and operation for various File I/O operations.

```

#include "FileIo.h"
#include "String2.h"
#include "List.h"
#include "Object.h"
#include "Memory.h"
#include "Error.h"
#include "Debug.h"
#include <stdio.h>
#include <limits.h>
#include <stdlib.h>
#include <unistd.h>
#include <dirent.h>
#include <sys/stat.h>

```

Classes

- struct [FileIo](#)

Macros

- #define **DEBUG** (0)

Functions

- PUBLIC void **FileIo_write** ([FileIo](#) *this, char *buffer, int length)
- PUBLIC void **FileIo_read** ([FileIo](#) *this, char *buffer, int length)
- PUBLIC void **FileIo_remove** ([FileIo](#) *this, [String](#) *fullFileName)
- PUBLIC void **FileIo_createDir** ([FileIo](#) *this, [String](#) *fullDirName)
- PUBLIC [List](#) * **FileIo_listDirs** ([FileIo](#) *this, [String](#) *directory)
- PUBLIC [List](#) * **FileIo_listFiles** ([FileIo](#) *this, [String](#) *directory)
- PUBLIC int **FileIo_fSeekEnd** ([FileIo](#) *this, int pos)
- PUBLIC [String](#) * **FileIo_getCwd** ([FileIo](#) *this)
- PUBLIC int **FileIo_fSeekSet** ([FileIo](#) *this, int pos)
- PUBLIC int **FileIo_ftell** ([FileIo](#) *this)
- PUBLIC FileIoStatus **FileIo_isOpen** ([FileIo](#) *this)

4.51.1 Detailed Description

A [FileIo](#) class. This class provides a status and operation for various File I/O operations.

4.52 FileIo.h

```

00001 #ifndef _FILEIO_H_
00002 #define _FILEIO_H_
00003 #include "String2.h"
00004 #include "Types.h"
00005
00006 typedef enum FileIoStatus
00007 {
00008     UNKNOWN=0,
00009     FILE_OPEN,
00010     DIR_OPEN
00011 } FileIoStatus;
00012
00013 typedef struct FileIo FileIo;
00014
00015 PUBLIC FileIo * FileIo_new();
00016 PUBLIC void FileIo_delete();
00017 PUBLIC FileIo* FileIo_copy(FileIo* this);
00018 PUBLIC int FileIo_comp(FileIo* this, FileIo* compare);
00019 PUBLIC void FileIo_print(FileIo* this);
00020 PUBLIC unsigned int FileIo_getSize(FileIo* this);
00021 PUBLIC void FileIo_openFile(FileIo* this, String* fullFileName);
00022 PUBLIC void FileIo_createFile(FileIo* this, String* fullFileName);
00023 PUBLIC void FileIo_openDir(FileIo* this, String* fullFileName);
00024 PUBLIC void FileIo_createDir(FileIo* this, String* fullDirName);
00025 PUBLIC void FileIo_write(FileIo* this, char* buffer, int length);
00026 PUBLIC void FileIo_read(FileIo* this, char* buffer, int length);
00027 PUBLIC void FileIo_remove(FileIo* this, String* fullFileName);
00028 PUBLIC String * FileIo_getCwd(FileIo* this);
00029 PUBLIC List * FileIo_listDirs(FileIo * this, String * directory);
00030 PUBLIC List* FileIo_listFiles(FileIo* this, String * directory);
00031 PUBLIC int FileIo_fSeekEnd(FileIo * this, int pos);
00032 PUBLIC int FileIo_fSeekSet(FileIo * this, int pos);
00033 PUBLIC int FileIo_ftell(FileIo* this);
00034 PUBLIC FileIoStatus FileIo_isOpen(FileIo * this);
00035 //Opendir
00036 //Readdir
00037
00038 #endif /* _FILEIO_H_ */

```

4.53 FileIo.h

```

00001 #ifndef _FILEIO_H_
00002 #define _FILEIO_H_
00003 #include "String2.h"
00004 #include "Types.h"
00005
00006 typedef enum FileIoStatus
00007 {
00008     UNKNOWN=0,
00009     FILE_OPEN,
00010     DIR_OPEN
00011 } FileIoStatus;
00012
00013 typedef struct FileIo FileIo;
00014
00015 PUBLIC FileIo * FileIo_new();
00016 PUBLIC void FileIo_delete();
00017 PUBLIC FileIo* FileIo_copy(FileIo* this);
00018 PUBLIC int FileIo_comp(FileIo* this, FileIo* compare);
00019 PUBLIC void FileIo_print(FileIo* this);
00020 PUBLIC unsigned int FileIo_getSize(FileIo* this);
00021 PUBLIC void FileIo_openFile(FileIo* this, String* fullFileName);
00022 PUBLIC void FileIo_createFile(FileIo* this, String* fullFileName);
00023 PUBLIC void FileIo_openDir(FileIo* this, String* fullFileName);
00024 PUBLIC void FileIo_createDir(FileIo* this, String* fullDirName);
00025 PUBLIC void FileIo_write(FileIo* this, char* buffer, int length);
00026 PUBLIC void FileIo_read(FileIo* this, char* buffer, int length);
00027 PUBLIC void FileIo_remove(FileIo* this, String* fullFileName);
00028 PUBLIC String * FileIo_getCwd(FileIo* this);
00029 PUBLIC List * FileIo_listDirs(FileIo * this, String * directory);
00030 PUBLIC List* FileIo_listFiles(FileIo* this, String * directory);
00031 PUBLIC int FileIo_fSeekEnd(FileIo * this, int pos);
00032 PUBLIC int FileIo_fSeekSet(FileIo * this, int pos);
00033 PUBLIC int FileIo_ftell(FileIo* this);
00034 PUBLIC FileIoStatus FileIo_isOpen(FileIo * this);
00035 //Opendir
00036 //Readdir
00037
00038 #endif /* _FILEIO_H_ */

```

4.54 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/↵ List/List.c File Reference

This file contains the implementation of the class [List](#).

```

#include "List.h"
#include "Class.h"
#include "Object.h"
#include "Memory.h"
#include "Debug.h"
#include "ListNode.h"

```

Classes

- class [List](#)

Functions

- PUBLIC void * [List_getNext](#) ([List](#) *this)
- PUBLIC void [List_resetIterator](#) ([List](#) *this)

4.54.1 Detailed Description

This file contains the implementation of the class [List](#).

The class [List](#) implement the [List](#) operations:

- [init](#)
- [add](#)

4.55 List.h

```
00001 /* List.h */
00002
00003 #ifndef _LIST_H_
00004 #define _LIST_H_
00005
00006 #include "Types.h"
00007 #include "Allocator.h"
00008
00009 typedef struct List List;
00010
00011 PUBLIC List * List_new();
00012 PUBLIC List * List_newFromAllocator(Allocator * allocator);
00013 PUBLIC void List_delete(List* this);
00014 PUBLIC List * List_copy(List* this);
00015 PUBLIC int List_compare(List * this, List * compared);
00016 PUBLIC void List_print(List * this);
00017 PUBLIC void List_insertHead(List* this, void* item, int isOwner);
00018 PUBLIC void List_insertTail(List* this, void* item, int isOwner);
00019 PUBLIC void List_merge(List* this, List* ll);
00020 PUBLIC void List_forEach(List* this, void (*method)(void* o));
00021 PUBLIC void * List_getNext(List* this);
00022 PUBLIC void * List_removeHead(List * this);
00023 PUBLIC void* List_removeTail(List* this);
00024 PUBLIC void * List_getHead(List * this);
00025 PUBLIC unsigned int List_getSize(List * this);
00026 PUBLIC unsigned int List_getNbNodes(List * this);
00027 PUBLIC void List_resetIterator(List * this);
00028
00029 #endif /* _LIST_H_ */
```

4.56 List.h

```
00001 /* List.h */
00002
00003 #ifndef _LIST_H_
00004 #define _LIST_H_
00005
00006 #include "Types.h"
00007 #include "Allocator.h"
00008
00009 typedef struct List List;
00010
00011 PUBLIC List * List_new();
00012 PUBLIC List * List_newFromAllocator(Allocator * allocator);
00013 PUBLIC void List_delete(List* this);
00014 PUBLIC List * List_copy(List* this);
00015 PUBLIC int List_compare(List * this, List * compared);
00016 PUBLIC void List_print(List * this);
00017 PUBLIC void List_insertHead(List* this, void* item, int isOwner);
00018 PUBLIC void List_insertTail(List* this, void* item, int isOwner);
00019 PUBLIC void List_merge(List* this, List* ll);
00020 PUBLIC void List_forEach(List* this, void (*method)(void* o));
00021 PUBLIC void * List_getNext(List* this);
00022 PUBLIC void * List_removeHead(List * this);
00023 PUBLIC void* List_removeTail(List* this);
00024 PUBLIC void * List_getHead(List * this);
00025 PUBLIC unsigned int List_getSize(List * this);
00026 PUBLIC unsigned int List_getNbNodes(List * this);
00027 PUBLIC void List_resetIterator(List * this);
00028
00029 #endif /* _LIST_H_ */
```

4.57 ListNode.h

```

00001  /* ListNode.h */
00002
00003  #ifndef _LISTNODE_H_
00004  #define _LISTNODE_H_
00005
00006  #include "Types.h"
00007  #include "Allocator.h"
00008  #include "Class.h"
00009  #include "Object.h"
00010  #include "Memory.h"
00011  #include "ObjectStore.h"
00012  #include "Error.h"
00013
00014  typedef struct ListNode ListNode;
00015
00016  PRIVATE ListNode * ListNode_new();
00017  PRIVATE ListNode * ListNode_newFromAllocator(Allocator * allocator, Object * object, int isOwner);
00018  PRIVATE void ListNode_delete(ListNode* this);
00019  PRIVATE ListNode * ListNode_copy(ListNode* this);
00020  PRIVATE int ListNode_compare(ListNode * this, ListNode * compared);
00021  PRIVATE void ListNode_print(ListNode * this);
00022  PRIVATE unsigned int ListNode_getSize(ListNode * this);
00023
00024  /*****/
00027  struct ListNode
00028  {
00029      Object object;
00030      void* item;
00031      int isOwner;
00032      ListNode* next;
00033      ListNode* prev;
00034  };
00035
00036  /*****/
00039  PRIVATE Class listNodeClass =
00040  {
00041      .f_new = (Constructor)0,
00042      .f_delete = (Destructor)&ListNode_delete,
00043      .f_copy = (Copy_Operator)&ListNode_copy,
00044      .f_comp = (Comp_Operator)&ListNode_compare,
00045      .f_print = (Printer)&ListNode_print,
00046      .f_size = (Sizer)&ListNode_getSize
00047  };
00048
00049  /*****/
00055  PRIVATE ListNode* ListNode_new(Object* object, int isOwner)
00056  {
00057      ListNode* this = 0;
00058
00059      this = (ListNode*)Object_new(sizeof(ListNode), &listNodeClass);
00060
00061      if (this != 0)
00062      {
00063          if (isOwner)
00064              this->item = object;
00065          else
00066              this->item = Object_getRef(object);
00067          this->isOwner = isOwner;
00068          this->next = 0;
00069          this->prev = 0;
00070      }
00071
00072      return this;
00073  }
00074
00075  /*****/
00081  PRIVATE ListNode* ListNode_newFromAllocator(Allocator* allocator, Object* object, int isOwner)
00082  {
00083      ListNode* this = 0;
00084
00085      this = (ListNode*)Object_newFromAllocator(&listNodeClass, allocator);
00086
00087      if (this != 0)
00088      {
00089          if (isOwner)
00090              this->item = object;
00091          else
00092              this->item = Object_getRef(object);
00093          this->isOwner = isOwner;
00094          this->next = 0;
00095          this->prev = 0;
00096      }
00097
00098      return this;
00099  }

```

```

00100
00101 /*****/
00106 PRIVATE void ListNode_delete(ListNode* this)
00107 {
00108     if (this != 0)
00109     {
00110         if ((this->item) && ((Object*)this->item)->delete != 0)
00111         {
00112             if ((Object*)this->item->marker != 0x0B5EC7)
00113             {
00114                 Error_new(ERROR_NORMAL, "List Node: item is not an object and is being deleted\n");
00115             }
00116             else
00117             {
00118                 if (this->isOwner)
00119                     ((Object*)this->item)->delete(this->item);
00120                 else
00121                     Object_deRef((Object*)this->item);
00122             }
00123         }
00124         Object_deallocate(&this->object);
00125     }
00126 }
00127
00128 /*****/
00134 PRIVATE ListNode* ListNode_copy(ListNode* this)
00135 {
00136     ListNode* copy = 0;
00137
00138     if (this != 0)
00139     {
00140         copy = ListNode_new((Object*)this->item, 0);
00141     }
00142
00143     return copy;
00144 }
00145
00146 /*****/
00152 PRIVATE int ListNode_compare(ListNode* this, ListNode* compared)
00153 {
00154     unsigned int result = 0;
00155
00156     return result;
00157 }
00158
00159 /*****/
00164 PRIVATE void ListNode_print(ListNode* this)
00165 {
00166 }
00167
00168 PRIVATE unsigned int ListNode_getSize(ListNode* this)
00169 {
00170     return sizeof(ListNode);
00171 }
00172
00173 #endif /* _LISTNODE_H_ */

```

4.58 MyAllocator.h

```

00001 /* MyAllocator.h */
00002 #ifndef _MYALLOCATOR_H_
00003 #define _MYALLOCATOR_H_
00004
00005 #include "Allocator.h"
00006 #include "Types.h"
00007
00008 typedef struct MyAllocator MyAllocator;
00009
00010 PUBLIC MyAllocator * MyAllocator_new(unsigned int size);
00011 PUBLIC void MyAllocator_delete(MyAllocator * this);
00012 void MyAllocator_reset(Allocator * this);
00013 PUBLIC void * MyAllocator_allocate(Allocator * allocator, unsigned int size);
00014 PUBLIC void MyAllocator_deallocate(Allocator * allocator, void * ptr);
00015 PUBLIC unsigned int MyAllocator_report(Allocator * this);
00016
00017 #endif /* _MYALLOCATOR_H_ */

```

4.59 MyAllocator.h

```

00001 /* MyAllocator.h */

```

```

00002 #ifndef _MYALLOCATOR_H_
00003 #define _MYALLOCATOR_H_
00004
00005 #include "Allocator.h"
00006 #include "Types.h"
00007
00008 typedef struct MyAllocator MyAllocator;
00009
00010 PUBLIC MyAllocator * MyAllocator_new(unsigned int size);
00011 PUBLIC void MyAllocator_delete(MyAllocator * this);
00012 void MyAllocator_reset(Allocator * this);
00013 PUBLIC void * MyAllocator_allocate(Allocator * allocator, unsigned int size);
00014 PUBLIC void MyAllocator_deallocate(Allocator * allocator, void * ptr);
00015 PUBLIC unsigned int MyAllocator_report(Allocator * this);
00016
00017 #endif /* _MYALLOCATOR_H_ */

```

4.60 MyAllocator.h

```

00001 /* MyAllocator.h */
00002 #ifndef _MYALLOCATOR_H_
00003 #define _MYALLOCATOR_H_
00004
00005 #include "Allocator.h"
00006 #include "Types.h"
00007
00008 typedef struct MyAllocator MyAllocator;
00009
00010 PUBLIC MyAllocator * MyAllocator_new();
00011 PUBLIC void MyAllocator_delete(MyAllocator * this);
00012 PUBLIC void * MyAllocator_allocate(Allocator * allocator, unsigned int size);
00013 PUBLIC void MyAllocator_deallocate(Allocator * allocator, void * ptr);
00014 PUBLIC unsigned int MyAllocator_report(Allocator * this);
00015
00016 #endif /* _MYALLOCATOR_H_ */

```

4.61 MyAllocator.h

```

00001 /* MyAllocator.h */
00002 #ifndef _MYALLOCATOR_H_
00003 #define _MYALLOCATOR_H_
00004
00005 #include "Allocator.h"
00006 #include "Types.h"
00007
00008 typedef struct MyAllocator MyAllocator;
00009
00010 PUBLIC MyAllocator * MyAllocator_new(unsigned int size);
00011 PUBLIC void MyAllocator_delete(MyAllocator * this);
00012 void MyAllocator_reset(Allocator * this);
00013 PUBLIC void * MyAllocator_allocate(Allocator * allocator, unsigned int size);
00014 PUBLIC void MyAllocator_deallocate(Allocator * allocator, void * ptr);
00015 PUBLIC unsigned int MyAllocator_report(Allocator * this);
00016
00017 #endif /* _MYALLOCATOR_H_ */

```

4.62 MyAllocator.h

4.63 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/↵ Map/Map.c File Reference

A [Map](#) class. This class provides a container indexed by a string.

```

#include "Map.h"
#include "MapEntry.h"
#include "List.h"

```

```
#include "Class.h"
#include "Object.h"
#include "String2.h"
#include "Memory.h"
#include "Debug.h"
#include "Error.h"
```

Classes

- class [Map](#)

Macros

- #define **DEBUG** (0)
- #define **HTABLE_SIZE** (50)

Functions

- PRIVATE [MapEntry](#) * **Map_findEntry** ([Map](#) *this, [String](#) *s)
- PUBLIC int **Map_comp** ([Map](#) *this, [Map](#) *compared)

4.63.1 Detailed Description

A [Map](#) class. This class provides a container indexed by a string.

4.64 Map.h

```
00001 /* Map.h */
00002
00003 #ifndef _MAP_H_
00004 #define _MAP_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008 #include "List.h"
00009 #include "Allocator.h"
00010
00011 typedef struct Map Map;
00012
00013 PUBLIC Map * Map_new();
00014 PUBLIC Map* Map_newFromAllocator(Allocator * allocator);
00015 PUBLIC void Map_delete(Map * this);
00016 PUBLIC Map * Map_copy(Map * this);
00017 PUBLIC int Map_comp(Map* this, Map* compared);
00018 PUBLIC unsigned int Map_insert(Map * this, String* s, void * p, int isOwner);
00019 PUBLIC unsigned int Map_find(Map * this, String* s, void ** p);
00020 PUBLIC void Map_print(Map * this);
00021 PUBLIC unsigned int Map_getSize(Map * this);
00022 PUBLIC List * Map_getAll(Map * this);
00023
00024 #endif /* _MAP_H_ */
```

4.65 Map.h

```

00001 /* Map.h */
00002
00003 #ifndef _MAP_H_
00004 #define _MAP_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008 #include "List.h"
00009 #include "Allocator.h"
00010
00011 typedef struct Map Map;
00012
00013 PUBLIC Map * Map_new();
00014 PUBLIC Map* Map_newFromAllocator(Allocator * allocator);
00015 PUBLIC void Map_delete(Map * this);
00016 PUBLIC Map * Map_copy(Map * this);
00017 PUBLIC int Map_comp(Map* this, Map* compared);
00018 PUBLIC unsigned int Map_insert(Map * this, String* s, void * p, int isOwner);
00019 PUBLIC unsigned int Map_find(Map * this, String* s, void ** p);
00020 PUBLIC void Map_print(Map * this);
00021 PUBLIC unsigned int Map_getSize(Map * this);
00022 PUBLIC List * Map_getAll(Map * this);
00023
00024 #endif /* _MAP_H_ */

```

4.66 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/↵ Map/MapEntry.c File Reference

A support class for the [Map](#) class.

```

#include "MapEntry.h"
#include "Class.h"
#include "Object.h"
#include "String2.h"

```

Classes

- struct [MapEntry](#)

Functions

- PUBLIC [MapEntry](#) * **MapEntry_new** ([String](#) *s, void *item, int isOwner)
- PUBLIC [MapEntry](#) * **MapEntry_newFromAllocator** ([Allocator](#) *allocator, [String](#) *s, void *item, int isOwner)
- PUBLIC void **MapEntry_delete** ([MapEntry](#) *this)
- PUBLIC [MapEntry](#) * **MapEntry_copy** ([MapEntry](#) *this)
- PUBLIC unsigned int **MapEntry_getSize** ([MapEntry](#) *this)
- PUBLIC [String](#) * **MapEntry_getString** ([MapEntry](#) *this)
- PUBLIC void **MapEntry_setString** ([MapEntry](#) *this, [String](#) *s)
- PUBLIC void * **MapEntry_getItem** ([MapEntry](#) *this)
- PUBLIC void **MapEntry_setItem** ([MapEntry](#) *this, void *item)

4.66.1 Detailed Description

A support class for the [Map](#) class.

4.67 MapEntry.h

```

00001 /* MapEntry.h */
00002
00003 #ifndef _MAPENTRY_H_
00004 #define _MAPENTRY_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008 #include "Allocator.h"
00009
00010 typedef struct MapEntry MapEntry;
00011
00012 PUBLIC MapEntry * MapEntry_new(String * s, void * p, int isOwner);
00013 PUBLIC MapEntry * MapEntry_newFromAllocator(Allocator * allocator, String *s, void * p, int isOwner);
00014 PUBLIC void MapEntry_delete(MapEntry * this);
00015 PUBLIC MapEntry * MapEntry_copy(MapEntry * this);
00016 PUBLIC unsigned int MapEntry_getSize(MapEntry * this);
00017 PUBLIC String * MapEntry_getString(MapEntry * this);
00018 PUBLIC void * MapEntry_getItem(MapEntry * this);
00019 PUBLIC void MapEntry_setString(MapEntry * this, String * s);
00020 PUBLIC void MapEntry_setItem(MapEntry * this, void * item);
00021
00022 #endif /* _MAPENTRY_H_ */

```

4.68 MapEntry.h

```

00001 /* MapEntry.h */
00002
00003 #ifndef _MAPENTRY_H_
00004 #define _MAPENTRY_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008 #include "Allocator.h"
00009
00010 typedef struct MapEntry MapEntry;
00011
00012 PUBLIC MapEntry * MapEntry_new(String * s, void * p, int isOwner);
00013 PUBLIC MapEntry * MapEntry_newFromAllocator(Allocator * allocator, String *s, void * p, int isOwner);
00014 PUBLIC void MapEntry_delete(MapEntry * this);
00015 PUBLIC MapEntry * MapEntry_copy(MapEntry * this);
00016 PUBLIC unsigned int MapEntry_getSize(MapEntry * this);
00017 PUBLIC String * MapEntry_getString(MapEntry * this);
00018 PUBLIC void * MapEntry_getItem(MapEntry * this);
00019 PUBLIC void MapEntry_setString(MapEntry * this, String * s);
00020 PUBLIC void MapEntry_setItem(MapEntry * this, void * item);
00021
00022 #endif /* _MAPENTRY_H_ */

```

4.69 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/↵ Memory/Memory.c File Reference

This file provides the implementation of the memory functions.

```

#include "Memory.h"
#include "Debug.h"
#include "Error.h"
#include <stdlib.h>
#include <string.h>

```

Macros

- #define DEBUG (0)

Variables

- PRIVATE unsigned int **Memory_allocRequestId** = 0
- PRIVATE unsigned int **Memory_freeRequestId** = 0
- PRIVATE unsigned int **Memory_nbBytesAllocated** = 0

4.69.1 Detailed Description

This file provides the implementation of the memory functions.

TBD

4.70 Memory.h

```
00001 /* Memory.h */
00002
00003 #ifndef _MEMORY_H_
00004 #define _MEMORY_H_
00005
00006 #include "Types.h"
00007
00008 #ifdef _WIN32
00009 #define MEMORY_ISVALID(p) (*(long int*)p!=0xCDCDCDCD)
00010 #elif _WIN64
00011 #define MEMORY_ISVALID(p) (p!=0xCDCDCDCD)
00012 #else
00013 #define MEMORY_ISVALID(p) (p!=0)
00014 #endif
00015
00016 PUBLIC void* Memory_alloc(unsigned int nbBytes);
00017 PUBLIC void * Memory_realloc(void * pointer, unsigned int prevSizeBytes, unsigned int newSizeBytes);
00018 PUBLIC void Memory_free(void* pointer, unsigned int nbBytes);
00019 PUBLIC void Memory_set(void * pointer, unsigned char val, unsigned int nbBytes);
00020 PUBLIC void Memory_copy(void * pointer, void * src, unsigned int nbBytes);
00021 PUBLIC int Memory_ncmp(void * pointer, void * compared, unsigned int nbBytes);
00022 PUBLIC int Memory_cmp(void * pointer, void * compared);
00023 PUBLIC unsigned int Memory_len(const void * pointer);
00024 PUBLIC void Memory_report();
00025 PUBLIC int Memory_getAllocRequestNb();
00026 PUBLIC int Memory_getFreeRequestNb();
00027
00028 #endif /* _MEMORY_H_ */
```

4.71 Memory.h

```
00001 /* Memory.h */
00002
00003 #ifndef _MEMORY_H_
00004 #define _MEMORY_H_
00005
00006 #include "Types.h"
00007
00008 #ifdef _WIN32
00009 #define MEMORY_ISVALID(p) (*(long int*)p!=0xCDCDCDCD)
00010 #elif _WIN64
00011 #define MEMORY_ISVALID(p) (p!=0xCDCDCDCD)
00012 #else
00013 #define MEMORY_ISVALID(p) (p!=0)
00014 #endif
00015
00016 PUBLIC void* Memory_alloc(unsigned int nbBytes);
00017 PUBLIC void * Memory_realloc(void * pointer, unsigned int prevSizeBytes, unsigned int newSizeBytes);
00018 PUBLIC void Memory_free(void* pointer, unsigned int nbBytes);
00019 PUBLIC void Memory_set(void * pointer, unsigned char val, unsigned int nbBytes);
00020 PUBLIC void Memory_copy(void * pointer, void * src, unsigned int nbBytes);
00021 PUBLIC int Memory_ncmp(void * pointer, void * compared, unsigned int nbBytes);
00022 PUBLIC int Memory_cmp(void * pointer, void * compared);
00023 PUBLIC unsigned int Memory_len(const void * pointer);
00024 PUBLIC void Memory_report();
00025 PUBLIC int Memory_getAllocRequestNb();
00026 PUBLIC int Memory_getFreeRequestNb();
00027
00028 #endif /* _MEMORY_H_ */
```


4.72 Class.h

```

00001  /* Class. h */
00002
00003  #ifndef _CLASS_H_
00004  #define _CLASS_H_
00005
00006  typedef struct Class Class;
00007
00008  struct Object;
00009  typedef struct Object* (*Constructor)();
00010  typedef void (*Destructor)(struct Object*);
00011  typedef struct Object* (*Copy_Operator)(struct Object*);
00012  typedef int (*Comp_Operator)(struct Object*, struct Object*);
00013  typedef char* (*Printer)(struct Object*);
00014  typedef unsigned int (*Sizer)();
00015
00016  struct Class
00017  {
00018      Constructor f_new;
00019      Destructor f_delete;
00020      Copy_Operator f_copy;
00021      Comp_Operator f_comp;
00022      Printer f_print;
00023      Sizer f_size;
00024  };
00025
00026  #endif /* _CLASS_H_ */

```

4.73 Class.h

```

00001  /* Class. h */
00002
00003  #ifndef _CLASS_H_
00004  #define _CLASS_H_
00005
00006  typedef struct Class Class;
00007
00008  struct Object;
00009  typedef struct Object* (*Constructor)();
00010  typedef void (*Destructor)(struct Object*);
00011  typedef struct Object* (*Copy_Operator)(struct Object*);
00012  typedef int (*Comp_Operator)(struct Object*, struct Object*);
00013  typedef char* (*Printer)(struct Object*);
00014  typedef unsigned int (*Sizer)();
00015
00016  struct Class
00017  {
00018      Constructor f_new;
00019      Destructor f_delete;
00020      Copy_Operator f_copy;
00021      Comp_Operator f_comp;
00022      Printer f_print;
00023      Sizer f_size;
00024  };
00025
00026  #endif /* _CLASS_H_ */

```

4.74 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/Object.c File Reference

This file contains the implementation for the class [Object](#).

```

#include "Class.h"
#include "Object.h"
#include "ObjectMgr.h"
#include "ObjectStore.h"
#include "Allocator.h"
#include "Memory.h"
#include "Error.h"

```

Macros

- `#define OBJECT_MARKER (0x0B5EC7)`

Variables

- `PRIVATE ObjectStore * Object_objectStore = 0`

4.74.1 Detailed Description

This file contains the implementation for the class `Object`.

The class `Object` is TBD

4.75 Object.h

```
00001 /* Object.h */
00002
00003 #ifndef _OBJECT_H_
00004 #define _OBJECT_H_
00005
00006 #include "Types.h"
00007 #include "Class.h"
00008 #include "Allocator.h"
00009
00010 typedef struct Object Object;
00011
00012 struct Object
00013 {
00014     int marker;
00015     unsigned int id;
00016     unsigned int uniqId;
00017     Class * class;
00018     void (*delete)(Object * this);
00019     Object * (*copy)(Object * this);
00020     unsigned int refCount;
00021     unsigned int size;
00022     Allocator * allocator;
00023 };
00024
00025 PUBLIC Object * Object_new(unsigned int size, Class * class);
00026 PUBLIC Object* Object_newFromAllocator(Class* class, Allocator* allocator);
00027 PUBLIC void Object_delete(Object * this);
00028 PUBLIC void Object_deallocate(Object* this);
00029 PUBLIC Object * Object_copy(Object * this);
00030 PUBLIC int Object_comp(Object * this, Object * compared);
00031 PUBLIC char * Object_print(Object * this);
00032 PUBLIC Object* Object_getRef(Object* this);
00033 PUBLIC void Object_deRef(Object * this);
00034 PUBLIC int Object_isValid(Object* this);
00035
00036 #endif /* _OBJECT_H_ */
```

4.76 Object.h

```
00001 /* Object.h */
00002
00003 #ifndef _OBJECT_H_
00004 #define _OBJECT_H_
00005
00006 #include "Types.h"
00007 #include "Class.h"
00008 #include "Allocator.h"
00009
00010 typedef struct Object Object;
00011
00012 struct Object
00013 {
```

```

00014     int marker;
00015     unsigned int id;
00016     unsigned int uniqId;
00017     Class * class;
00018     void (*delete)(Object * this);
00019     Object * (*copy)(Object * this);
00020     unsigned int refCount;
00021     unsigned int size;
00022     Allocator * allocator;
00023 };
00024
00025 PUBLIC Object * Object_new(unsigned int size, Class * class);
00026 PUBLIC Object* Object_newFromAllocator(Class* class, Allocator* allocator);
00027 PUBLIC void Object_delete(Object * this);
00028 PUBLIC void Object_deallocate(Object* this);
00029 PUBLIC Object * Object_copy(Object * this);
00030 PUBLIC int Object_comp(Object * this, Object * compared);
00031 PUBLIC char * Object_print(Object * this);
00032 PUBLIC Object* Object_getRef(Object* this);
00033 PUBLIC void Object_deRef(Object * this);
00034 PUBLIC int Object_isValid(Object* this);
00035
00036 #endif /* _OBJECT_H_ */

```

4.77 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectMgr/ObjectMgr.c File Reference

An object management class.

```

#include "ObjectMgr.h"
#include "Object.h"
#include "Memory.h"
#include "Debug.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

```

Classes

- struct [ObjectInfo](#)
- class [ObjectMgr](#)

Macros

- #define **MAX_NB_OBJECTS** (40000)
- #define **END_OF_QUEUE** (0xFFFFFFFF)

Typedefs

- typedef struct [ObjectInfo](#) [ObjectInfo](#)

Variables

- PRIVATE [ObjectMgr](#) * **objectMgr** = 0

4.77.1 Detailed Description

An object management class.

This class provides an object allocation and de-allocation service. Only one instance of this class can be created.

4.78 ObjectMgr.h

```
00001 /* ObjectMgr.h */
00002
00003 #ifndef _OBJECTMGR_H_
00004 #define _OBJECTMGR_H_
00005
00006 #include "Object.h"
00007 #include "Types.h"
00008
00009 typedef struct ObjectMgr ObjectMgr;
00010
00011 PUBLIC void ObjectMgr_delete(ObjectMgr * this);
00012 PUBLIC ObjectMgr * ObjectMgr_copy(ObjectMgr * this);
00013 PUBLIC ObjectMgr * ObjectMgr_getRef();
00014 PUBLIC Object * ObjectMgr_allocate(ObjectMgr * this, unsigned int size);
00015 PUBLIC void ObjectMgr_deallocate(ObjectMgr * this, Object * object);
00016 PUBLIC void ObjectMgr_reportUnallocated(ObjectMgr* this);
00017 PUBLIC unsigned int ObjectMgr_report(ObjectMgr * this);
00018
00019 #endif /* _OBJECTMGR_H_ */
```

4.79 ObjectMgr.h

```
00001 /* ObjectMgr.h */
00002
00003 #ifndef _OBJECTMGR_H_
00004 #define _OBJECTMGR_H_
00005
00006 #include "Object.h"
00007 #include "Types.h"
00008
00009 typedef struct ObjectMgr ObjectMgr;
00010
00011 PUBLIC void ObjectMgr_delete(ObjectMgr * this);
00012 PUBLIC ObjectMgr * ObjectMgr_copy(ObjectMgr * this);
00013 PUBLIC ObjectMgr * ObjectMgr_getRef();
00014 PUBLIC Object * ObjectMgr_allocate(ObjectMgr * this, unsigned int size);
00015 PUBLIC void ObjectMgr_deallocate(ObjectMgr * this, Object * object);
00016 PUBLIC void ObjectMgr_reportUnallocated(ObjectMgr* this);
00017 PUBLIC unsigned int ObjectMgr_report(ObjectMgr * this);
00018
00019 #endif /* _OBJECTMGR_H_ */
```

4.80 ObjectStore.h

```
00001 /* ObjectStore.h */
00002
00003 #ifndef _OBJECTSTORE_H_
00004 #define _OBJECTSTORE_H_
00005
00006 #include "Types.h"
00007 #include "Object.h"
00008 #include "Allocator.h"
00009
00010 typedef struct AllocInfo AllocInfo;
00011 typedef struct ObjectStore ObjectStore;
00012
00013 PUBLIC void ObjectStore_delete(ObjectStore * this);
00014 PUBLIC ObjectStore * ObjectStore_copy(ObjectStore* this);
00015 PUBLIC ObjectStore * ObjectStore_getRef();
00016 PUBLIC AllocInfo * ObjectStore_createAllocator(ObjectStore * this, Allocator * allocator);
00017 PUBLIC void ObjectStore_deleteAllocator(ObjectStore * this, AllocInfo * allocInfo);
00018 PUBLIC Object * ObjectStore_createObject(ObjectStore * this, Class * class, Allocator * allocator);
00019 PUBLIC void ObjectStore_deleteObject(ObjectStore * this, Object * object);
00020 PUBLIC void ObjectStore_reportUnallocated(ObjectStore * this);
```

```

00021 PUBLIC void ObjectStore_report(ObjectStore * this);
00022 PUBLIC unsigned int ObjectStore_getNbAllocatedObjects(ObjectStore * this);
00023 PUBLIC int ObjectStore_compare(ObjectStore * this, ObjectStore * compared);
00024 PUBLIC void ObjectStore_print(ObjectStore * this);
00025
00026 #endif /* _OBJECTSTORE_H_ */

```

4.81 ObjectStore.h

```

00001 /* ObjectStore.h */
00002
00003 #ifndef _OBJECTSTORE_H_
00004 #define _OBJECTSTORE_H_
00005
00006 #include "Types.h"
00007 #include "Object.h"
00008 #include "Allocator.h"
00009
00010 typedef struct AllocInfo AllocInfo;
00011 typedef struct ObjectStore ObjectStore;
00012
00013 PUBLIC void ObjectStore_delete(ObjectStore * this);
00014 PUBLIC ObjectStore * ObjectStore_copy(ObjectStore* this);
00015 PUBLIC ObjectStore * ObjectStore_getRef();
00016 PUBLIC AllocInfo * ObjectStore_createAllocator(ObjectStore * this, Allocator * allocator);
00017 PUBLIC void ObjectStore_deleteAllocator(ObjectStore * this, AllocInfo * allocInfo);
00018 PUBLIC Object * ObjectStore_createObject(ObjectStore * this, Class * class, Allocator * allocator);
00019 PUBLIC void ObjectStore_deleteObject(ObjectStore * this, Object * object);
00020 PUBLIC void ObjectStore_reportUnallocated(ObjectStore * this);
00021 PUBLIC void ObjectStore_report(ObjectStore * this);
00022 PUBLIC unsigned int ObjectStore_getNbAllocatedObjects(ObjectStore * this);
00023 PUBLIC int ObjectStore_compare(ObjectStore * this, ObjectStore * compared);
00024 PUBLIC void ObjectStore_print(ObjectStore * this);
00025
00026 #endif /* _OBJECTSTORE_H_ */

```

4.82 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Pool/Pool.c File Reference

This file contains the implementation of the class Pool.

```

#include "Pool.h"
#include "Memory.h"
#include "Error.h"
#include <stdio.h>

```

Macros

- #define **CACHE_NB** (6)
- #define **END_OF_QUEUE** (0xFFFFFFFF)
- #define **END_OF_ALLOC** (0xFFFFFFFFE)
- #define **START_OF_AVAIL** (0xFFFFFFFFD)

Functions

- PRIVATE AllocStatus **Pool_allocInFile** (Pool *pool, unsigned int *ptrIdx)
Pool_allocInFile.
- PRIVATE void **Pool_deallocInMemory** (Pool *pool, unsigned int idx)
Pool_deallocInMemory.
- PRIVATE void **Pool_deallocInFile** (Pool *pool, unsigned int idx)

- Pool_deallocInFile.*
- PRIVATE void [Pool_reportInFile](#) (Pool *pool)
 - Pool_reportInFile.*
- PRIVATE void [Pool_reportInMemory](#) (Pool *pool)
 - Pool_reportInMemory.*
- PRIVATE void [Pool_readInFile](#) (Pool *pool, unsigned int idx, void *p)
 - Pool_readInFile.*
- PRIVATE void [Pool_readInMemory](#) (Pool *pool, unsigned int idx, void *p)
 - Pool_readInMemory.*
- PRIVATE void [Pool_writeInFile](#) (Pool *pool, unsigned int idx, void *p)
 - Pool_writeInFile.*
- PRIVATE void [Pool_writeInMemory](#) (Pool *pool, unsigned int idx, void *p)
 - Pool_writeInMemory.*
- PUBLIC Pool * [Pool_new](#) (unsigned int nbMemChunks, unsigned int memChunkSize)
 - Create a new instance of the class Pool in RAM.*
- PUBLIC Pool * [Pool_newFromFile](#) (char *fileName, unsigned int nbMemChunks, unsigned int memChunkSize)
 - Create a new instance of the class Pool in a file.*
- PUBLIC void [Pool_delete](#) (Pool *pool)
 - Pool_delete.*
- PUBLIC void * [Pool_alloc](#) (Pool *pool, unsigned int *ptrIdx)
 - Pool_alloc.*
- PUBLIC void [Pool_dealloc](#) (Pool *pool, unsigned int idx)
 - Pool_dealloc.*
- PUBLIC void [Pool_write](#) (Pool *pool, unsigned int idx, void *ptrContent)
 - Pool_writeCache.*
- PUBLIC void * [Pool_read](#) (Pool *pool, unsigned int idx)
 - Pool_read.*
- PUBLIC void [Pool_report](#) (Pool *pool)
 - Pool_report.*
- PUBLIC unsigned int [Pool_reportSizeInBytes](#) (Pool *pool)
 - Pool_reportSizeInBytes input: none.*
- PUBLIC unsigned int [Pool_reportNbNodes](#) (Pool *pool)
 - Pool_reportNbNodes.*
- PUBLIC void **Pool_discardCache** (Pool *pool, unsigned int idx)
- PUBLIC void **Pool_discardAllCache** (Pool *pool)
- PUBLIC unsigned int **Pool_reportCacheUsed** (Pool *pool)

4.82.1 Detailed Description

This file contains the implementation of the class Pool.

The class [List](#) implement the Pool operations

- Alloc
- De-alloc

4.82.2 Function Documentation

4.82.2.1 Pool_alloc()

```
PUBLIC void * Pool_alloc (
    Pool * pool,
    unsigned int * ptrIdx )
```

Pool_alloc.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

Reference to cache position, NULL is cache full

4.82.2.2 Pool_allocInFile()

```
PRIVATE AllocStatus Pool_allocInFile (
    Pool * pool,
    unsigned int * ptrIdx )
```

Pool_allocInFile.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.82.2.3 Pool_dealloc()

```
PUBLIC void Pool_dealloc (
    Pool * pool,
    unsigned int idx )
```

Pool_dealloc.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.82.2.4 Pool_deallocInFile()

```
PRIVATE void Pool_deallocInFile (
    Pool * pool,
    unsigned int idx )
```

Pool_deallocInFile.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.82.2.5 Pool_deallocInMemory()

```
PRIVATE void Pool_deallocInMemory (
    Pool * pool,
    unsigned int idx )
```

Pool_deallocInMemory.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.82.2.6 Pool_delete()

```
PUBLIC void Pool_delete (
    Pool * pool )
```

Pool_delete.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.82.2.7 Pool_new()

```
PUBLIC Pool * Pool_new (
    unsigned int nbMemChunks,
    unsigned int memChunkSize )
```

Create a new instance of the class Pool in RAM.

Parameters

in	<i>number</i>	of memory chunks to allocate.
in	<i>size</i>	of memory chunk.

Returns

New instance.

4.82.2.8 Pool_newFromFile()

```
PUBLIC Pool * Pool_newFromFile (
    char * fileName,
    unsigned int nbMemChunks,
    unsigned int memChunkSize )
```

Create a new instance of the class Pool in a file.

Parameters

in	<i>File</i>	name
in	<i>Number</i>	of memory chunks to allocate
in	<i>Size</i>	of memory chunk return A pool of memory

4.82.2.9 Pool_read()

```
PUBLIC void * Pool_read (
    Pool * pool,
    unsigned int idx )
```

Pool_read.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.82.2.10 Pool_readInFile()

```
PRIVATE void Pool_readInFile (
    Pool * pool,
    unsigned int idx,
    void * p )
```

Pool_readInFile.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.82.2.11 Pool_readInMemory()

```
PRIVATE void Pool_readInMemory (
    Pool * pool,
    unsigned int idx,
    void * p )
```

Pool_readInMemory.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.82.2.12 Pool_report()

```
PUBLIC void Pool_report (
    Pool * pool )
```

Pool_report.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.82.2.13 Pool_reportInFile()

```
PRIVATE void Pool_reportInFile (  
    Pool * pool )
```

Pool_reportInFile.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.82.2.14 Pool_reportInMemory()

```
PRIVATE void Pool_reportInMemory (  
    Pool * pool )
```

Pool_reportInMemory.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.82.2.15 Pool_reportNbNodes()

```
PUBLIC unsigned int Pool_reportNbNodes (  
    Pool * pool )
```

Pool_reportNbNodes.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.82.2.16 Pool_reportSizeInBytes()

```
PUBLIC unsigned int Pool_reportSizeInBytes (
    Pool * pool )
```

Pool_reportSizeInBytes input: none.

Returns

none

4.82.2.17 Pool_write()

```
PUBLIC void Pool_write (
    Pool * pool,
    unsigned int idx,
    void * ptrContent )
```

Pool_writeCache.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none none

4.82.2.18 Pool_writeInFile()

```
PRIVATE void Pool_writeInFile (
    Pool * pool,
    unsigned int idx,
    void * p )
```

Pool_writeInFile.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.82.2.19 Pool_writeInMemory()

```
PRIVATE void Pool_writeInMemory (
    Pool * pool,
    unsigned int idx,
    void * p )
```

Pool_writeInMemory.

Parameters

in	<i>none</i>	
----	-------------	--

Returns

none

4.83 Pool.h

```
00001 #ifndef _POOL_
00002 #define _POOL_
00003 /*****
00004  * Pool.h
00005  *
00006  *****/
00007 #include "Types.h"
00008 #include "Pool.h"
00009
00010 typedef enum AllocStatus
00011 {
00012     ALLOC_OK = 0,
00013     ALLOC_FAIL = 1
00014 } AllocStatus;
00015
00016 typedef struct PoolCache
00017 {
00018     unsigned int idx;
00019     unsigned int isUsed;
00020     void* cache;
00021 } PoolCache;
00022
00023 typedef struct Pool Pool;
00024
00025 PUBLIC Pool* Pool_new(unsigned int nbMemChunks, unsigned int memChunkSize);
00026 PUBLIC Pool* Pool_newFixed(unsigned int nbMemChunks, unsigned int memChunkSize);
00027 PUBLIC Pool* Pool_newFromFile(char* fileName, unsigned int nbMemChunks, unsigned int memChunkSize);
00028 PUBLIC void Pool_delete(Pool* pool);
00029 PUBLIC void * Pool_alloc(Pool* pool, unsigned int * ptrIdx);
00030 PUBLIC void Pool_dealloc(Pool* pool, unsigned int p);
00031 PUBLIC void Pool_write(Pool* pool, unsigned int idx, void* ptrContent);
00032 PUBLIC void* Pool_read(Pool* pool, unsigned int idx);
00033 PUBLIC unsigned int Pool_addToChunkCache(Pool* pool, void* p, unsigned int length);
00034 PUBLIC void Pool_report(Pool* pool);
00035 PUBLIC unsigned int Pool_reportSizeInBytes(Pool* pool);
00036 PUBLIC unsigned int Pool_reportNbNodes(Pool* pool);
00037 PUBLIC void Pool_discardCache(Pool* pool, unsigned int idx);
00038 PUBLIC void Pool_discardAllCache(Pool* pool);
00039 PUBLIC unsigned int Pool_reportCacheUsed(Pool * pool);
00040 #endif /* _POOL_ */
```

4.84 Pool.h

```
00001 #ifndef _POOL_
00002 #define _POOL_
00003 /*****
00004  * Pool.h
00005  *
00006  *****/
00007 #include "Types.h"
```

```

00008 #include "Pool.h"
00009
00010 typedef enum AllocStatus
00011 {
00012     ALLOC_OK = 0,
00013     ALLOC_FAIL = 1
00014 } AllocStatus;
00015
00016 typedef struct PoolCache
00017 {
00018     unsigned int idx;
00019     unsigned int isUsed;
00020     void* cache;
00021 } PoolCache;
00022
00023 typedef struct Pool Pool;
00024
00025 PUBLIC Pool* Pool_new(unsigned int nbMemChunks, unsigned int memChunkSize);
00026 PUBLIC Pool* Pool_newFixed(unsigned int nbMemChunks, unsigned int memChunkSize);
00027 PUBLIC Pool* Pool_newFromFile(char* fileName, unsigned int nbMemChunks, unsigned int memChunkSize);
00028 PUBLIC void Pool_delete(Pool* pool);
00029 PUBLIC void * Pool_alloc(Pool* pool, unsigned int * ptrIdx);
00030 PUBLIC void Pool_dealloc(Pool* pool, unsigned int p);
00031 PUBLIC void Pool_write(Pool* pool, unsigned int idx, void* ptrContent);
00032 PUBLIC void* Pool_read(Pool* pool, unsigned int idx);
00033 PUBLIC unsigned int Pool_addToChunkCache(Pool* pool, void* p, unsigned int length);
00034 PUBLIC void Pool_report(Pool* pool);
00035 PUBLIC unsigned int Pool_reportSizeInBytes(Pool* pool);
00036 PUBLIC unsigned int Pool_reportNbNodes(Pool* pool);
00037 PUBLIC void Pool_discardCache(Pool* pool, unsigned int idx);
00038 PUBLIC void Pool_discardAllCache(Pool* pool);
00039 PUBLIC unsigned int Pool_reportCacheUsed(Pool * pool);
00040 #endif /* _POOL_ */

```

4.85 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/SkipList.c File Reference

This file contains the implementation of the class [SkipList](#). The class [List](#) implement the [SkipList](#) operations.

```

#include "SkipList.h"
#include "Pool.h"
#include "Class.h"
#include "Object.h"
#include "Debug.h"
#include <stdlib.h>
#include "SkipNode.h"

```

Classes

- class [SkipList](#)

Typedefs

- typedef struct [SkipList](#) [SkipList](#)

Functions

- PUBLIC [Object](#) * [SkipList_get](#) ([SkipList](#) *this, [Object](#) *key)

4.85.1 Detailed Description

This file contains the implementation of the class [SkipList](#). The class [List](#) implement the [SkipList](#) operations.

- Add
- Remove
- Get

4.86 SkipList.h

```
00001 /* SkipList.h */
00002
00003 #ifndef _SKIPLIST_
00004 #define _SKIPLIST_
00005
00006 #include "Types.h"
00007 #include "Object.h"
00008 #include "Allocator.h"
00009
00010 typedef struct SkipList SkipList;
00011
00012 PUBLIC SkipList * SkipList_new();
00013 PUBLIC SkipList * SkipList_newFromAllocator(Allocator * allocator);
00014 PUBLIC void SkipList_delete(SkipList* skipList);
00015 PUBLIC SkipList * SkipList_copy(SkipList * this);
00016 PUBLIC void SkipList_add(SkipList* this, Object * key, Object * item);
00017 PUBLIC Object * SkipList_remove(SkipList* this, Object * key);
00018 PUBLIC Object * SkipList_get(SkipList* this, Object * key);
00019 PUBLIC int SkipList_compare(SkipList * this, SkipList * compared);
00020 PUBLIC void SkipList_print(SkipList* this);
00021 PUBLIC unsigned int SkipList_getSize(SkipList* this);
00022
00023 #endif /* _SKIPLIST_ */
```

4.87 SkipList.h

```
00001 /* SkipList.h */
00002
00003 #ifndef _SKIPLIST_
00004 #define _SKIPLIST_
00005
00006 #include "Types.h"
00007 #include "Object.h"
00008 #include "Allocator.h"
00009
00010 typedef struct SkipList SkipList;
00011
00012 PUBLIC SkipList * SkipList_new();
00013 PUBLIC SkipList * SkipList_newFromAllocator(Allocator * allocator);
00014 PUBLIC void SkipList_delete(SkipList* skipList);
00015 PUBLIC SkipList * SkipList_copy(SkipList * this);
00016 PUBLIC void SkipList_add(SkipList* this, Object * key, Object * item);
00017 PUBLIC Object * SkipList_remove(SkipList* this, Object * key);
00018 PUBLIC Object * SkipList_get(SkipList* this, Object * key);
00019 PUBLIC int SkipList_compare(SkipList * this, SkipList * compared);
00020 PUBLIC void SkipList_print(SkipList* this);
00021 PUBLIC unsigned int SkipList_getSize(SkipList* this);
00022
00023 #endif /* _SKIPLIST_ */
```

4.88 SkipNode.h

```
00001 /* SkipNode.h */
00002
00003 #ifndef _SKIPNODE_H_
00004 #define _SKIPNODE_H_
00005
00006 #include "Types.h"
```

```

00007 #include "Object.h"
00008 #include "Allocator.h"
00009 #include <limits.h>
00010
00011 #define SKIPLIST_MAX_LEVEL (6)
00012 #define END_NODE (0)
00013
00014 typedef struct SkipNode SkipNode;
00015
00016 PUBLIC SkipNode * SkipNode_new();
00017 PUBLIC SkipNode * SkipNode_newFromAllocator();
00018 PUBLIC void SkipNode_delete(SkipNode * this);
00019 PUBLIC SkipNode * SkipNode_copy(SkipNode * this);
00020 PUBLIC int SkipNode_compare(SkipNode * this, SkipNode * compared);
00021 PUBLIC void SkipNode_print(SkipNode * this);
00022 PUBLIC unsigned int SkipNode_getSize(SkipNode * this);
00023
00024 /*****/
00027 PRIVATE Class skipNodeClass =
00028 {
00029     .f_new = 0,
00030     .f_delete = (Destructor)&SkipNode_delete,
00031     .f_copy = (Copy_Operator)&SkipNode_copy,
00032     .f_comp = (Comp_Operator)&SkipNode_compare,
00033     .f_print = (Printer)&SkipNode_print,
00034     .f_size = (Sizer)&SkipNode_getSize
00035 };
00036
00037 typedef struct SkipNode
00038 {
00039     Object object;
00040     Object * key;
00041     Object * item;
00042     unsigned int level;
00043     void * forward[SKIPLIST_MAX_LEVEL];
00044 } SkipNode;
00045
00046
00047 PUBLIC SkipNode * SkipNode_new()
00048 {
00049     SkipNode * this = 0;
00050
00051     this = (SkipNode*)Object_new(sizeof(SkipNode), &skipNodeClass);
00052     if (this==0) return 0;
00053
00054     this->item = 0;
00055     this->level = 1;
00056     this->key = END_NODE;
00057
00058     return this;
00059 }
00060
00061 PUBLIC SkipNode * SkipNode_newFromAllocator(Allocator * allocator)
00062 {
00063     SkipNode * this = 0;
00064
00065     this = (SkipNode*)Object_newFromAllocator(&skipNodeClass, allocator);
00066     if (this == 0) return 0;
00067     this->item = 0;
00068     this->level = 1;
00069     this->key = END_NODE;
00070     return this;
00071 }
00072
00073 PUBLIC void SkipNode_delete(SkipNode * this)
00074 {
00075     if (this==0) return;
00076
00077     //Object_delete(this->item);
00078     Object_deallocate(&this->object);
00079 }
00080
00081 PUBLIC SkipNode * SkipNode_copy(SkipNode * this)
00082 {
00083     SkipNode * copy = 0;
00084
00085     return copy;
00086 }
00087
00088 PUBLIC int SkipNode_compare(SkipNode * this, SkipNode * compared)
00089 {
00090     return 1;
00091 }
00092
00093 PUBLIC void SkipNode_print(SkipNode * this)
00094 {
00095     if (this==0) return;

```



```

00096
00097
00098
00099 }
00100
00101 PUBLIC unsigned int SkipNode_getSize(SkipNode * this)
00102 {
00103     return sizeof(SkipNode);
00104 }
00105 #endif /* _SKIPNODE_H_ */

```

4.89 String2.h

```

00001 /* String2.h */
00002
00003 #ifndef _STRING2_H_
00004 #define _STRING2_H_
00005
00006 #include "Types.h"
00007 #include "List.h"
00008
00009 typedef struct String String;
00010
00011 PUBLIC String * String_new(const char * constString);
00012 PUBLIC String * String_newByRef(const char * constString);
00013 PUBLIC void String_delete(String * this);
00014 PUBLIC String * String_copy(String * this);
00015 PUBLIC String * String_getRef(String * this);
00016 PUBLIC unsigned int String_getLength(String * this);
00017 PUBLIC char * String_getBuffer(String * this);
00018 PUBLIC void String_setBuffer(String* this, char* buffer, int isOwned);
00019 PUBLIC unsigned int String_isContained(String * this, String * s2);
00020 PUBLIC unsigned int String_prepend(String * this, const char * prefix);
00021 PUBLIC unsigned int String_append(String* this, const char* postfix);
00022 PUBLIC int String_compare(String * this, String * compared);
00023 PUBLIC String * String_subString(String * this, unsigned int idx, unsigned int length);
00024 PUBLIC unsigned int String_matchWildcard(String * this, const char * wildcard);
00025 PUBLIC int String_toInt(String* this);
00026 PUBLIC List* String_splitToken(String* this, const char* separator);
00027 PUBLIC void String_stealBuffer(String* this, String* s);
00028 PUBLIC unsigned int String_getSize(String* this);
00029 PUBLIC void String_print(String* this);
00030 #endif /* _STRING2_H_ */

```

4.90 String2.h

```

00001 /* String2.h */
00002
00003 #ifndef _STRING2_H_
00004 #define _STRING2_H_
00005
00006 #include "Types.h"
00007 #include "List.h"
00008
00009 typedef struct String String;
00010
00011 PUBLIC String * String_new(const char * constString);
00012 PUBLIC String * String_newByRef(const char * constString);
00013 PUBLIC void String_delete(String * this);
00014 PUBLIC String * String_copy(String * this);
00015 PUBLIC String * String_getRef(String * this);
00016 PUBLIC unsigned int String_getLength(String * this);
00017 PUBLIC char * String_getBuffer(String * this);
00018 PUBLIC void String_setBuffer(String* this, char* buffer, int isOwned);
00019 PUBLIC unsigned int String_isContained(String * this, String * s2);
00020 PUBLIC unsigned int String_prepend(String * this, const char * prefix);
00021 PUBLIC unsigned int String_append(String* this, const char* postfix);
00022 PUBLIC int String_compare(String * this, String * compared);
00023 PUBLIC String * String_subString(String * this, unsigned int idx, unsigned int length);
00024 PUBLIC unsigned int String_matchWildcard(String * this, const char * wildcard);
00025 PUBLIC int String_toInt(String* this);
00026 PUBLIC List* String_splitToken(String* this, const char* separator);
00027 PUBLIC void String_stealBuffer(String* this, String* s);
00028 PUBLIC unsigned int String_getSize(String* this);
00029 PUBLIC void String_print(String* this);
00030 #endif /* _STRING2_H_ */

```

4.91 Times.h

```
00001 /* Times.h */
00002
00003 long double get_wall_time();
00004 long double get_cpu_time();
00005
```

4.92 Times.h

```
00001 /* Times.h */
00002
00003 long double get_wall_time();
00004 long double get_cpu_time();
00005
```

4.93 Types.h

```
00001 /* Types.h */
00002
00003 #ifndef _TYPES_H_
00004 #define _TYPES_H_
00005
00006 #define PUBLIC
00007
00008 #define DECLARE_CLASS(x)
00009
00010 #ifndef UNIT_TEST
00011 #define PRIVATE static
00012 #else
00013 #define PRIVATE
00014 #endif
00015
00016 union mem_align
00017 {
00018     void * a;
00019     long int b;
00020     long long c;
00021 };
00022
00023 #define MEM_ALIGN (sizeof(union mem_align))
00024
00025 #include "UserTypes.h"
00026 #endif /* _TYPES_H_ */
```

4.94 Types.h

```
00001 /* Types.h */
00002
00003 #ifndef _TYPES_H_
00004 #define _TYPES_H_
00005
00006 #define PUBLIC
00007
00008 #define DECLARE_CLASS(x)
00009
00010 #ifndef UNIT_TEST
00011 #define PRIVATE static
00012 #else
00013 #define PRIVATE
00014 #endif
00015
00016 union mem_align
00017 {
00018     void * a;
00019     long int b;
00020     long long c;
00021 };
00022
00023 #define MEM_ALIGN (sizeof(union mem_align))
00024
00025 #include "UserTypes.h"
00026 #endif /* _TYPES_H_ */
```

4.95 UserTypes.h

```

00001 /* User Types */
00002 #ifndef _USERTYPES_
00003 #define _USERTYPES_
00004 #include "Class.h"
00005 extern Class listClass;
00006 extern Class stringClass;
00007
00008 enum ClassId
00009 {
00010     ListClass,
00011     StringClass,
00012     NB_CLASSES
00013 };
00014
00015 /*Class * userTypes[] =
00016 {
00017     &listClass,
00018     &stringClass,
00019 };*/
00020 #endif /* UserTypes.h */

```

4.96 UserTypes.h

```

00001 /* User Types */
00002 #ifndef _USERTYPES_
00003 #define _USERTYPES_
00004 #include "Class.h"
00005 extern Class listClass;
00006 extern Class stringClass;
00007
00008 enum ClassId
00009 {
00010     ListClass,
00011     StringClass,
00012     NB_CLASSES
00013 };
00014
00015 /*Class * userTypes[] =
00016 {
00017     &listClass,
00018     &stringClass,
00019 };*/
00020 #endif /* UserTypes.h */

```

4.97 /home/thomas/Projects/SParse-master/SParse/src/main.c File Reference

Contains the main() function.

```

#include "OptionMgr.h"
#include "FileMgr.h"
#include "TimeMgr.h"
#include "Error.h"
#include "Debug.h"
#include "SParse.h"
#include "Memory.h"
#include "ObjectMgr.h"
#include <signal.h>

```

Functions

- PRIVATE void [print_usage](#) ()

Prints the application help.

- PRIVATE void [sighandler](#) (int signum, siginfo_t *info, void *ptr)

Display and exit when signal is received.

- PUBLIC int [main](#) (const int argc, const char **argv)

Initial entry point for the application.

Variables

- struct sigaction **action**

4.97.1 Detailed Description

Contains the main() function.

This file contains only one function main() which initialises the [OptionMgr](#) and [FileMgr](#) objects. The function also processes each source file in turn.

4.97.2 Function Documentation

4.97.2.1 main()

```
PUBLIC int main (
    const int argc,
    const char ** argv )
```

Initial entry point for the application.

Parameters

<i>argc</i>	Number of arguments
<i>argv</i>	Array of arguments

The main function: 1) Reads the options from command line or file 2) Starts the application for a DB name and an input file directory.

4.97.2.2 print_usage()

```
PRIVATE void print_usage ( )
```

Prints the application help.

Prints the usage information for the application.

4.97.2.3 sighandler()

```
PRIVATE void sighandler (  
    int signum,  
    siginfo_t * info,  
    void * ptr )
```

Display and exit when signal is received.

Parameters

<i>signum</i>	TBC
<i>info</i>	TBC
<i>ptr</i>	TBC

This function displays a signal and exit the application.

4.98 Ast.h**4.99 Declarator.h**

```

00001  /* Declarator.h */
00002
00003  #ifndef _DECLARATOR_H_
00004  #define _DECLARATOR_H_
00005
00006  typedef enum
00007  {
00008      E_DEC_FUNCTION,
00009      E_DEC_VAR,
00010      E_DEC_TYPE
00011  } DeclaratorType;
00012
00013  typedef struct Declarator Declarator;
00014
00015  Declarator * Declarator_new(DeclaratorType * t)
00016  {
00017  }
00018
00019  void Declarator_delete(Declarator * this)
00020  {
00021  }
00022
00023  #endif /* #ifndef _DECLARATOR_H_

```

4.100 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/↵ Configuration/Configuration.c File Reference

This file contains the implementation for the class [Configuration](#). The class [Configuration](#) lists all the SW products to parse including the path to the source files, any dependency.

```

#include "Configuration.h"
#include "Product.h"
#include "TransUnit.h"
#include "Grammar.h"
#include "Object.h"
#include "Memory.h"
#include "Error.h"
#include "Debug.h"

```

Classes

- class [Configuration](#)

Macros

- `#define DEBUG (0)`
- `#define IS_KEY(C)`
- `#define IS_COLON(P) (Memory_ncmp(P, ":", 1))`
- `#define IS_LIST(P) (Memory_ncmp(P, "- ", 2))`
- `#define IS_LOCATION_KEY(P) (Memory_ncmp(P, "Location:", 9))`
- `#define IS_INCLUDES_KEY(P) (Memory_ncmp(P, "Includes:", 9))`
- `#define IS_USES_KEY(P) (Memory_ncmp(P, "Uses:", 5))`
- `#define IS_SOURCES_KEY(P) (Memory_ncmp(P, "Sources:", 8))`
- `#define IS_IGNORED(C) ((C==' ') || (C=='\n') || (C=='\r'))`
- `#define IS_STRING(C)`
- `#define IS_FORBIDDEN(C) (C=='\t')`
- `#define IS_EOL(P) (Memory_ncmp(P, "\n", 1))`

Functions

- `PRIVATE List * Configuration_readProducts (Configuration *this, String *s)`
- `PRIVATE String * Configuration_readLocation (Configuration *this, String *s, unsigned int *idx)`
- `PRIVATE List * Configuration_readIncludes (Configuration *this, String *s, unsigned int *idx)`
- `PRIVATE List * Configuration_readUses (Configuration *this, String *s, unsigned int *idx)`
- `PRIVATE List * Configuration_readSources (Configuration *this, String *s, unsigned int *idx)`
- `PRIVATE String * Configuration_readString (Configuration *this, String *s, unsigned int *idx)`
- `PRIVATE List * Configuration_readList (Configuration *this, String *s, unsigned int *idx)`
- `PRIVATE void Configuration_readEndOfLine (Configuration *this, String *s, unsigned int *idx)`
- `PRIVATE unsigned int Configuration_readIndent (Configuration *this, String *s, unsigned int *idx)`

4.100.1 Detailed Description

This file contains the implementation for the class `Configuration`. The class `Configuration` lists all the SW products to parse including the path to the source files, any dependency.

4.100.2 Macro Definition Documentation

4.100.2.1 IS_KEY

```
#define IS_KEY(  
    C )
```

Value:

```
(( (C>='A') && (C<='Z')) || ((C>='a') && (C<='z')) \
|| ((C>='0') && (C<='9')) || (C=='_'))
```

4.100.2.2 IS_STRING

```
#define IS_STRING(  
    C )
```

Value:

```
(( (C>='A') && (C<='Z')) || ((C>='a') && (C<='z')) \
|| ((C>='0') && (C<='9')) || (C=='_') || (C=='/') || (C=='\\') || (C=='-') || (C=='.'))
```

4.101 Configuration.h

```

00001 /* Configuration.h */
00002 #ifndef _CONFIGURATION_H_
00003 #define _CONFIGURATION_H_
00004
00005 #include "Types.h"
00006 #include "String2.h"
00007 #include "List.h"
00008 #include "FileMgr.h"
00009
00010 typedef struct Configuration Configuration;
00011
00012 PUBLIC Configuration * Configuration_new(String * input);
00013 PUBLIC void Configuration_delete(Configuration * this);
00014 PUBLIC void Configuration_print(Configuration * this);
00015 PUBLIC unsigned int Configuration_getSize(Configuration * this);
00016 PUBLIC List * Configuration_getProducts(Configuration* this);
00017 #endif /* _CONFIGURATION_H_ */
00018

```

4.102 Configuration.h

```

00001 /* Configuration.h */
00002 #ifndef _CONFIGURATION_H_
00003 #define _CONFIGURATION_H_
00004
00005 #include "Types.h"
00006 #include "String2.h"
00007 #include "List.h"
00008 #include "FileMgr.h"
00009
00010 typedef struct Configuration Configuration;
00011
00012 PUBLIC Configuration * Configuration_new(String * input);
00013 PUBLIC void Configuration_delete(Configuration * this);
00014 PUBLIC void Configuration_print(Configuration * this);
00015 PUBLIC unsigned int Configuration_getSize(Configuration * this);
00016 PUBLIC List * Configuration_getProducts(Configuration* this);
00017 #endif /* _CONFIGURATION_H_ */
00018

```

4.103 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/Configuration/Product.c File Reference

This file contains the implementation for the class [Product](#).

```

#include "Product.h"
#include "FileMgr.h"
#include "String2.h"
#include "Debug.h"

```

Classes

- class [Product](#)

Macros

- #define **DEBUG** (0)

Functions

- PUBLIC [Product](#) * [Product_new](#) ([String](#) *s)
- PUBLIC void [Product_delete](#) ([Product](#) *this)
- PUBLIC void [Product_print](#) ([Product](#) *this)
- PUBLIC unsigned int [Product_getSize](#) ([Product](#) *this)
- PUBLIC [String](#) * [Product_getName](#) ([Product](#) *this)
- PUBLIC void [Product_setLocation](#) ([Product](#) *this, [String](#) *s)
- PUBLIC [String](#) * [Product_getLocation](#) ([Product](#) *this)
- PUBLIC void [Product_setIncludes](#) ([Product](#) *this, [List](#) *l)
- PUBLIC void [Product_setUses](#) ([Product](#) *this, [List](#) *l)
- PUBLIC void [Product_setSources](#) ([Product](#) *this, [List](#) *l)
- PUBLIC [FileMgr](#) * [Product_getSourceFiles](#) ([Product](#) *this)

4.103.1 Detailed Description

This file contains the implementation for the class [Product](#).

The class [Product](#) contains the sources for a given product.

4.104 Product.h

```
00001 /* Product.h */
00002 #ifndef _PRODUCT_H_
00003 #define _PRODUCT_H_
00004
00005 #include "FileMgr.h"
00006 #include "List.h"
00007 #include "String2.h"
00008 #include "Object.h"
00009 #include "Debug.h"
00010
00011 typedef struct Product Product;
00012
00013 PUBLIC Product* Product_new(String * this);
00014 PUBLIC void Product_delete(Product * this);
00015 PUBLIC void Product_print(Product * this);
00016 PUBLIC unsigned int Product_getSize(Product * this);
00017 PUBLIC String* Product_getName(Product* this);
00018 PUBLIC void Product_setLocation(Product * this, String * s);
00019 PUBLIC String* Product_getLocation(Product* this);
00020 PUBLIC void Product_setIncludes(Product * this, List * l);
00021 PUBLIC void Product_setUses(Product * this, List * l);
00022 PUBLIC void Product_setSources(Product * this, List * l);
00023 PUBLIC FileMgr * Product_getSourceFiles(Product * this);
00024 #endif /* _PRODUCT_H_ */
```

4.105 Product.h

```
00001 /* Product.h */
00002 #ifndef _PRODUCT_H_
00003 #define _PRODUCT_H_
00004
00005 #include "FileMgr.h"
00006 #include "List.h"
00007 #include "String2.h"
00008 #include "Object.h"
00009 #include "Debug.h"
00010
00011 typedef struct Product Product;
00012
00013 PUBLIC Product* Product_new(String * this);
00014 PUBLIC void Product_delete(Product * this);
00015 PUBLIC void Product_print(Product * this);
00016 PUBLIC unsigned int Product_getSize(Product * this);
00017 PUBLIC String* Product_getName(Product* this);
00018 PUBLIC void Product_setLocation(Product * this, String * s);
00019 PUBLIC String* Product_getLocation(Product* this);
00020 PUBLIC void Product_setIncludes(Product * this, List * l);
00021 PUBLIC void Product_setUses(Product * this, List * l);
00022 PUBLIC void Product_setSources(Product * this, List * l);
00023 PUBLIC FileMgr * Product_getSourceFiles(Product * this);
00024 #endif /* _PRODUCT_H_ */
```

4.106 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/FileReader/FileReader.c File Reference

This file contains the implementation for the class [FileReader](#).

```
#include "FileReader.h"
#include "Class.h"
#include "Object.h"
#include "String2.h"
#include "FileMgr.h"
#include "FileDesc.h"
#include "OptionMgr.h"
#include "List.h"
#include "Error.h"
#include "Memory.h"
```

Classes

- struct [IncludeInfo](#)
- class [FileReader](#)

Typedefs

- typedef struct [IncludeInfo](#) [IncludeInfo](#)

Functions

- PRIVATE unsigned int [IncludeDir_getSize](#) ([IncludeInfo](#) *this)

Variables

- PRIVATE Class [includeInfoClass](#)

4.106.1 Detailed Description

This file contains the implementation for the class [FileReader](#).

The class [FileReader](#) is TBD

4.106.2 Variable Documentation

4.106.2.1 includeInfoClass

PRIVATE Class [includeInfoClass](#)

Initial value:

```
=
{
    .f_new = (Constructor)0,
    .f_delete = (Destructor)0,
    .f_copy = (Copy_Operator)0,
    .f_comp = (Comp_Operator)0,
    .f_print = (Printer)0,
    .f_size = (Sizer)&IncludeDir_getSize
}
```

4.107 FileReader.h

```

00001 /* FileReader.h */
00002
00003 #ifndef _FILEREADER_H_
00004 #define _FILEREADER_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008 #include "FileMgr.h"
00009
00010 typedef struct FileReader FileReader;
00011
00012 PUBLIC FileReader * FileReader_new(FileDesc * file, FileMgr * fileMgr);
00013 PUBLIC void FileReader_delete(FileReader * this);
00014 PUBLIC FileReader * FileReader_copy(FileReader * this);
00015 PUBLIC void FileReader_print(FileReader * this);
00016 PUBLIC unsigned int FileReader_getSize(FileReader * this);
00017 PUBLIC char * FileReader_getBuffer(FileReader * this);
00018 PUBLIC String * FileReader_getName(FileReader * this);
00019 PUBLIC char * FileReader_addFile(FileReader * this, String * fileName);
00020 #endif /* _FILEREADER_H_ */

```

4.108 FileReader.h

```

00001 /* FileReader.h */
00002
00003 #ifndef _FILEREADER_H_
00004 #define _FILEREADER_H_
00005
00006 #include "Types.h"
00007 #include "String2.h"
00008 #include "FileMgr.h"
00009
00010 typedef struct FileReader FileReader;
00011
00012 PUBLIC FileReader * FileReader_new(FileDesc * file, FileMgr * fileMgr);
00013 PUBLIC void FileReader_delete(FileReader * this);
00014 PUBLIC FileReader * FileReader_copy(FileReader * this);
00015 PUBLIC void FileReader_print(FileReader * this);
00016 PUBLIC unsigned int FileReader_getSize(FileReader * this);
00017 PUBLIC char * FileReader_getBuffer(FileReader * this);
00018 PUBLIC String * FileReader_getName(FileReader * this);
00019 PUBLIC char * FileReader_addFile(FileReader * this, String * fileName);
00020 #endif /* _FILEREADER_H_ */

```

4.109 Grammar.h

```

00001 /* Grammar.h */
00002 #ifndef _GRAMMAR_H_
00003 #define _GRAMMAR_H_
00004
00005 #include "Types.h"
00006 #include "Object.h"
00007
00008 typedef struct Grammar Grammar;
00009
00010 struct Grammar
00011 {
00012     Object object;
00013     Grammar * (*new)(void);
00014     void (*delete)(Grammar * this);
00015     Grammar * (*copy)(Grammar * this);
00016     void (*print)(Grammar * this);
00017 };
00018
00019 PUBLIC Grammar * Grammar_new();
00020 PUBLIC void Grammar_delete(Grammar * this);
00021 PUBLIC void Grammar_process(Grammar * this);
00022 PUBLIC void Grammar_print(Grammar * this);
00023
00024 #endif /* _GRAMMAR_H_ */

```

4.110 Grammar.h

```

00001 /* Grammar.h */

```

```

00002 #ifndef _GRAMMAR_H_
00003 #define _GRAMMAR_H_
00004
00005 #include "Types.h"
00006 #include "Object.h"
00007
00008 typedef struct Grammar Grammar;
00009
00010 struct Grammar
00011 {
00012     Object object;
00013     Grammar * (*new)(void);
00014     void (*delete)(Grammar * this);
00015     Grammar * (*copy)(Grammar * this);
00016     void (*print)(Grammar * this);
00017 };
00018
00019 PUBLIC Grammar * Grammar_new();
00020 PUBLIC void Grammar_delete(Grammar * this);
00021 PUBLIC void Grammar_process(Grammar * this);
00022 PUBLIC void Grammar_print(Grammar * this);
00023
00024 #endif /* _GRAMMAR_H_ */

```

4.111 Grammar2.h

```

00001 /* Grammar2.h */
00002
00003 #include "Types.h"
00004 #include "SdbMgr.h"
00005 #include "FileReader.h"
00006
00007 typedef struct Grammar2 Grammar2;
00008
00009 PUBLIC Grammar2 * Grammar2_new(FileReader * fr, SdbMgr * sdbMgr);
00010 PUBLIC void Grammar2_delete(Grammar2 * this);
00011 PUBLIC Grammar2 * Grammar2_copy(Grammar2 * this);
00012 PUBLIC void Grammar2_print(Grammar2 * this);
00013 PUBLIC unsigned int Grammar2_getSize(Grammar2 * this);
00014 PUBLIC void Grammar2_process(Grammar2 * this);
00015 PUBLIC FileReader * Grammar2_getFileReader(Grammar2 * grammar); // Not used
00016 PUBLIC SdbMgr * Grammar2_getSdbMgr(Grammar2 * grammar); // Not used
00017 PUBLIC void Grammar2_addToBuffer(Grammar2 * grammar, char * text); // Used by lex.c
00018 PUBLIC void Grammar2_addComment(Grammar2 * this); // Used by parse.c
00019 PUBLIC void Grammar2_addCodeNode(Grammar2 * this);
00020 PUBLIC void Grammar2_addIncludeNode(Grammar2 * this, char * name);
00021 PUBLIC char * Grammar2_processNewFile(Grammar2 * this, String * fileName); // Used by lex.c
00022 PUBLIC void Grammar2_returnToFile(Grammar2 * this); // Used by lex.c

```

4.112 Grammar2.h

```

00001 /* Grammar2.h */
00002
00003 #include "Types.h"
00004 #include "SdbMgr.h"
00005 #include "FileReader.h"
00006
00007 typedef struct Grammar2 Grammar2;
00008
00009 PUBLIC Grammar2 * Grammar2_new(FileReader * fr, SdbMgr * sdbMgr);
00010 PUBLIC void Grammar2_delete(Grammar2 * this);
00011 PUBLIC Grammar2 * Grammar2_copy(Grammar2 * this);
00012 PUBLIC void Grammar2_print(Grammar2 * this);
00013 PUBLIC unsigned int Grammar2_getSize(Grammar2 * this);
00014 PUBLIC void Grammar2_process(Grammar2 * this);
00015 PUBLIC FileReader * Grammar2_getFileReader(Grammar2 * grammar); // Not used
00016 PUBLIC SdbMgr * Grammar2_getSdbMgr(Grammar2 * grammar); // Not used
00017 PUBLIC void Grammar2_addToBuffer(Grammar2 * grammar, char * text); // Used by lex.c
00018 PUBLIC void Grammar2_addComment(Grammar2 * this); // Used by parse.c
00019 PUBLIC void Grammar2_addCodeNode(Grammar2 * this);
00020 PUBLIC void Grammar2_addIncludeNode(Grammar2 * this, char * name);
00021 PUBLIC char * Grammar2_processNewFile(Grammar2 * this, String * fileName); // Used by lex.c
00022 PUBLIC void Grammar2_returnToFile(Grammar2 * this); // Used by lex.c

```

4.113 Grammar2.parse.h

```

00001 /* A Bison parser, made by GNU Bison 3.8.2.  */

```

```

00002
00003 /* Bison interface for Yacc-like parsers in C
00004
00005     Copyright (C) 1984, 1989-1990, 2000-2015, 2018-2021 Free Software Foundation,
00006     Inc.
00007
00008     This program is free software: you can redistribute it and/or modify
00009     it under the terms of the GNU General Public License as published by
00010     the Free Software Foundation, either version 3 of the License, or
00011     (at your option) any later version.
00012
00013     This program is distributed in the hope that it will be useful,
00014     but WITHOUT ANY WARRANTY; without even the implied warranty of
00015     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00016     GNU General Public License for more details.
00017
00018     You should have received a copy of the GNU General Public License
00019     along with this program. If not, see <https://www.gnu.org/licenses/>. */
00020
00021 /* As a special exception, you may create a larger work that contains
00022 part or all of the Bison parser skeleton and distribute that work
00023 under terms of your choice, so long as that work isn't itself a
00024 parser generator using the skeleton or a modified version thereof
00025 as a parser skeleton. Alternatively, if you modify or redistribute
00026 the parser skeleton itself, you may (at your option) remove this
00027 special exception, which will cause the skeleton and the resulting
00028 Bison output files to be licensed under the GNU General Public
00029 License without this special exception.
00030
00031 This special exception was added by the Free Software Foundation in
00032 version 2.2 of Bison. */
00033
00034 /* DO NOT RELY ON FEATURES THAT ARE NOT DOCUMENTED in the manual,
00035 especially those whose name start with YY_ or yy_. They are
00036 private implementation details that can be changed or removed. */
00037
00038 #ifndef YY_GRAMMAR2_GRAMMAR2_PARSE_H_INCLUDED
00039 # define YY_GRAMMAR2_GRAMMAR2_PARSE_H_INCLUDED
00040 /* Debug traces. */
00041 #ifndef YYDEBUG
00042 # define YYDEBUG 0
00043 #endif
00044 #if YYDEBUG
00045 extern int Grammar2_debug;
00046 #endif
00047
00048 /* Token kinds. */
00049 #ifndef YYTOKENTYPE
00050 # define YYTOKENTYPE
00051     enum yytokentype
00052     {
00053         YYEMPTY = -2,
00054         YYEOF = 0, /* "end of file" */
00055         YYerror = 256, /* error */
00056         YYUNDEF = 257, /* "invalid token" */
00057         COMMENT = 258, /* COMMENT */
00058         CODE = 259, /* CODE */
00059         END_OF_UNIT = 260 /* END_OF_UNIT */
00060     };
00061     typedef enum yytokentype yytoken_kind_t;
00062 #endif
00063 /* Token kinds. */
00064 #define YYEMPTY -2
00065 #define YYEOF 0
00066 #define YYerror 256
00067 #define YYUNDEF 257
00068 #define COMMENT 258
00069 #define CODE 259
00070 #define END_OF_UNIT 260
00071
00072 /* Value type. */
00073 #if ! defined YYSTYPE && ! defined YYSTYPE_IS_DECLARED
00074 union YYSTYPE
00075 {
00076     #line 18 "Grammar2.y"
00077
00078     String * text;
00079
00080     #line 81 "Grammar2.parse.h"
00081
00082 };
00083 typedef union YYSTYPE YYSTYPE;
00084 # define YYSTYPE_IS_TRIVIAL 1
00085 # define YYSTYPE_IS_DECLARED 1
00086 #endif
00087
00088

```

```

00089
00090
00091 int Grammar2_parse (void * scanner, Grammar2 * grammar);
00092
00093
00094 #endif /* !YY_GRAMMAR2_GRAMMAR2_PARSE_H_INCLUDED */

```

4.114 GrammarC99.h

```

00001 /* GrammarC99.h */
00002 #ifndef _GRAMMARC99_H_
00003 #define _GRAMMARC99_H_
00004
00005 #include "Types.h"
00006 #include "FileMgr.h"
00007 #include "Grammar.h"
00008
00009 typedef struct GrammarC99 GrammarC99;
00010
00011 PUBLIC Grammar* GrammarC99_new(FileDesc * fileDesc, FileMgr * fm);
00012 PUBLIC void GrammarC99_delete(Grammar* this);
00013 PUBLIC void GrammarC99_print(Grammar* this);
00014 PUBLIC unsigned int GrammarC99_getSize(Grammar* this);
00015 PUBLIC void GrammarC99_process(GrammarC99* this);
00016
00017 #endif /* _GRAMMARC99_H_ */

```

4.115 GrammarC99.h

```

00001 /* GrammarC99.h */
00002 #ifndef _GRAMMARC99_H_
00003 #define _GRAMMARC99_H_
00004
00005 #include "Types.h"
00006 #include "FileMgr.h"
00007 #include "Grammar.h"
00008
00009 typedef struct GrammarC99 GrammarC99;
00010
00011 PUBLIC Grammar* GrammarC99_new(FileDesc * fileDesc, FileMgr * fm);
00012 PUBLIC void GrammarC99_delete(Grammar* this);
00013 PUBLIC void GrammarC99_print(Grammar* this);
00014 PUBLIC unsigned int GrammarC99_getSize(Grammar* this);
00015 PUBLIC void GrammarC99_process(GrammarC99* this);
00016
00017 #endif /* _GRAMMARC99_H_ */

```

4.116 GrammarC99.parse.h

```

00001 /* A Bison parser, made by GNU Bison 3.8.2.  */
00002
00003 /* Bison interface for Yacc-like parsers in C
00004
00005    Copyright (C) 1984, 1989-1990, 2000-2015, 2018-2021 Free Software Foundation,
00006    Inc.
00007
00008    This program is free software: you can redistribute it and/or modify
00009    it under the terms of the GNU General Public License as published by
00010    the Free Software Foundation, either version 3 of the License, or
00011    (at your option) any later version.
00012
00013    This program is distributed in the hope that it will be useful,
00014    but WITHOUT ANY WARRANTY; without even the implied warranty of
00015    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
00016    GNU General Public License for more details.
00017
00018    You should have received a copy of the GNU General Public License
00019    along with this program.  If not, see <https://www.gnu.org/licenses/>.  */
00020
00021 /* As a special exception, you may create a larger work that contains
00022    part or all of the Bison parser skeleton and distribute that work
00023    under terms of your choice, so long as that work isn't itself a
00024    parser generator using the skeleton or a modified version thereof
00025    as a parser skeleton.  Alternatively, if you modify or redistribute
00026    the parser skeleton itself, you may (at your option) remove this

```

```

00027     special exception, which will cause the skeleton and the resulting
00028     Bison output files to be licensed under the GNU General Public
00029     License without this special exception.
00030
00031     This special exception was added by the Free Software Foundation in
00032     version 2.2 of Bison.  */
00033
00034 /* DO NOT RELY ON FEATURES THAT ARE NOT DOCUMENTED in the manual,
00035    especially those whose name start with YY_ or yy_.  They are
00036    private implementation details that can be changed or removed.  */
00037
00038 #ifndef YY_GRAMMARC99_GRAMMARC99_PARSE_H_INCLUDED
00039 # define YY_GRAMMARC99_GRAMMARC99_PARSE_H_INCLUDED
00040 /* Debug traces.  */
00041 #ifndef GRAMMARC99_DEBUG
00042 # if defined YYDEBUG
00043 #if YYDEBUG
00044 #   define GRAMMARC99_DEBUG 1
00045 #   else
00046 #   define GRAMMARC99_DEBUG 0
00047 #   endif
00048 # else /* ! defined YYDEBUG */
00049 #   define GRAMMARC99_DEBUG 0
00050 #   endif /* ! defined YYDEBUG */
00051 #endif /* ! defined GRAMMARC99_DEBUG */
00052 #if GRAMMARC99_DEBUG
00053 extern int GrammarC99_debug;
00054 #endif
00055 /* "%code requires" blocks.  */
00056 #line 19 "GrammarC99.y"
00057 #include "MyType.h"
00058
00059 #line 60 "GrammarC99.parse.h"
00060
00061 /* Token kinds.  */
00062 #ifndef GRAMMARC99_TOKENTYPE
00063 # define GRAMMARC99_TOKENTYPE
00064     enum GrammarC99_tokentype
00065     {
00066         GRAMMARC99_EMPTY = -2,
00067         GRAMMARC99_EOF = 0,          /* "end of file"  */
00068         GRAMMARC99_error = 256,      /* error          */
00069         GRAMMARC99_UNDEF = 257,      /* "invalid token" */
00070         IDENTIFIER = 258,             /* IDENTIFIER     */
00071         CONSTANT = 259,               /* CONSTANT       */
00072         STRING_LITERAL = 260,         /* STRING_LITERAL  */
00073         SIZEOF = 261,                 /* SIZEOF         */
00074         PTR_OP = 262,                 /* PTR_OP         */
00075         INC_OP = 263,                 /* INC_OP         */
00076         DEC_OP = 264,                 /* DEC_OP         */
00077         LEFT_OP = 265,                /* LEFT_OP        */
00078         RIGHT_OP = 266,               /* RIGHT_OP       */
00079         LE_OP = 267,                  /* LE_OP          */
00080         GE_OP = 268,                  /* GE_OP          */
00081         EQ_OP = 269,                  /* EQ_OP          */
00082         NE_OP = 270,                  /* NE_OP          */
00083         AND_OP = 271,                 /* AND_OP         */
00084         OR_OP = 272,                  /* OR_OP          */
00085         MUL_ASSIGN = 273,              /* MUL_ASSIGN     */
00086         DIV_ASSIGN = 274,              /* DIV_ASSIGN     */
00087         MOD_ASSIGN = 275,              /* MOD_ASSIGN     */
00088         ADD_ASSIGN = 276,              /* ADD_ASSIGN     */
00089         SUB_ASSIGN = 277,              /* SUB_ASSIGN     */
00090         LEFT_ASSIGN = 278,             /* LEFT_ASSIGN    */
00091         RIGHT_ASSIGN = 279,            /* RIGHT_ASSIGN   */
00092         AND_ASSIGN = 280,              /* AND_ASSIGN     */
00093         XOR_ASSIGN = 281,              /* XOR_ASSIGN     */
00094         OR_ASSIGN = 282,               /* OR_ASSIGN      */
00095         TYPE_NAME = 283,               /* TYPE_NAME      */
00096         TYPEDEF = 284,                 /* TYPEDEF        */
00097         EXTERN = 285,                  /* EXTERN         */
00098         STATIC = 286,                  /* STATIC         */
00099         AUTO = 287,                    /* AUTO          */
00100         REGISTER = 288,                /* REGISTER       */
00101         INLINE = 289,                  /* INLINE         */
00102         RESTRICT = 290,                /* RESTRICT       */
00103         CHAR = 291,                    /* CHAR           */
00104         SHORT = 292,                   /* SHORT          */
00105         INT = 293,                     /* INT            */
00106         LONG = 294,                    /* LONG           */
00107         SIGNED = 295,                  /* SIGNED         */
00108         UNSIGNED = 296,                /* UNSIGNED       */
00109         FLOAT = 297,                   /* FLOAT          */
00110         DOUBLE = 298,                  /* DOUBLE         */
00111         CONST = 299,                   /* CONST          */
00112         VOLATILE = 300,                /* VOLATILE       */
00113         VOID = 301,                    /* VOID           */

```

```

00114     BOOL = 302,                                /* BOOL */
00115     COMPLEX = 303,                             /* COMPLEX */
00116     IMAGINARY = 304,                          /* IMAGINARY */
00117     STRUCT = 305,                             /* STRUCT */
00118     UNION = 306,                              /* UNION */
00119     ENUM = 307,                               /* ENUM */
00120     ELLIPSIS = 308,                           /* ELLIPSIS */
00121     CASE = 309,                              /* CASE */
00122     DEFAULT = 310,                           /* DEFAULT */
00123     IF = 311,                                 /* IF */
00124     ELSE = 312,                              /* ELSE */
00125     SWITCH = 313,                            /* SWITCH */
00126     WHILE = 314,                             /* WHILE */
00127     DO = 315,                                /* DO */
00128     FOR = 316,                               /* FOR */
00129     GOTO = 317,                              /* GOTO */
00130     CONTINUE = 318,                          /* CONTINUE */
00131     BREAK = 319,                             /* BREAK */
00132     RETURN = 320                             /* RETURN */
00133 };
00134 typedef enum GrammarC99_tokentype GrammarC99_token_kind_t;
00135 #endif
00136
00137 /* Value type. */
00138 #if ! defined GRAMMARC99_STYPE && ! defined GRAMMARC99_STYPE_IS_DECLARED
00139 typedef MyType GRAMMARC99_STYPE;
00140 # define GRAMMARC99_STYPE_IS_TRIVIAL 1
00141 # define GRAMMARC99_STYPE_IS_DECLARED 1
00142 #endif
00143
00144
00145
00146
00147 int GrammarC99_parse (void * scanner, GrammarC99 * grammar);
00148
00149
00150 #endif /* !YY_GRAMMARC99_GRAMMARC99_PARSE_H_INCLUDED */

```

4.117 MyType.h

```

00001 typedef struct MyType MyType;
00002
00003 struct MyType {
00004     char * sval;
00005 };

```

4.118 HTTPRequest.h

```

00001 /* HTTPRequest.h */
00002 #ifndef _HTTPREQUEST_H_
00003 #define _HTTPREQUEST_H_
00004
00005 #include "Object.h"
00006 #include "Types.h"
00007 #include "Class.h"
00008 #include "String2.h"
00009 #include "Map.h"
00010 #include "Memory.h"
00011 #include "Debug.h"
00012
00013 #include <stdio.h>
00014 #include <stdlib.h>
00015
00016 typedef struct HTTPRequest HTTPRequest;
00017
00018 PRIVATE HTTPRequest * HTTPRequest_new(char * buffer);
00019 PRIVATE void HTTPRequest_delete(HTTPRequest * this);
00020 PRIVATE HTTPRequest * HTTPRequest_copy(HTTPRequest * this);
00021 PRIVATE int HTTPRequest_compare(HTTPRequest* this, HTTPRequest* compared);
00022 PRIVATE void HTTPRequest_print(HTTPRequest * this);
00023 PRIVATE unsigned int HTTPRequest_getSize(HTTPRequest* this);
00024 PRIVATE String* HTTPRequest_getPath(HTTPRequest* this);
00025 PRIVATE enum Method HTTPRequest_getMethod(HTTPRequest* this);
00026 PRIVATE int HTTPRequest_isValid(HTTPRequest* this);
00027 PRIVATE int HTTPRequest_parseBuffer(HTTPRequest* this, char* buffer);
00028
00029 enum Method
00030 {
00031     METHOD_GET=0,

```



```

00032 METHOD_POST,
00033 METHOD_PUT,
00034 METHOD_PATCH,
00035 METHOD_DELETE,
00036 METHOD_INVALID
00037 };
00038
00039 static char* methods_text[] = { "GET", "POST", "PUT", "PATCH", "DELETE" };
00040 /*****
00041 struct HTTPRequest
00042 {
00043     Object object;
00044     enum Method method;
00045     String * path;
00046     int majorVersion;
00047     int minorVersion;
00048     Map* headers;
00049     String * body;
00050     int isValid;
00051 };
00052
00053 /*****
00054 PRIVATE Class httpRequestClass =
00055 {
00056     .f_new = 0,
00057     .f_delete = (Destructor)&HTTPRequest_delete,
00058     .f_copy = (Copy_Operator)&HTTPRequest_copy,
00059     .f_comp = (Comp_Operator)&HTTPRequest_compare,
00060     .f_print = (Printer)&HTTPRequest_print,
00061     .f_size = (Sizer)&HTTPRequest_getSize
00062 };
00063
00064 /*****
00065 PRIVATE HTTPRequest * HTTPRequest_new(char * buffer)
00066 {
00067     HTTPRequest* this = 0;
00068
00069     this = (HTTPRequest*)Object_new(sizeof(HTTPRequest), &httpRequestClass);
00070
00071     if (this == 0) return 0;
00072
00073     this->method = METHOD_INVALID;
00074     this->path = 0;
00075     this->majorVersion = 0;
00076     this->minorVersion = 0;
00077     this->headers = Map_new();
00078     this->body = 0;
00079     this->isValid = HTTPRequest_parseBuffer(this, buffer);
00080
00081     return this;
00082 }
00083
00084 /*****
00085 PRIVATE void HTTPRequest_delete(HTTPRequest * this)
00086 {
00087     if (!Object_isValid((Object*)this)) return;
00088
00089     String_delete(this->path);
00090     Map_delete(this->headers);
00091 }
00092
00093 /*****
00094 PRIVATE HTTPRequest * HTTPRequest_copy(HTTPRequest * this)
00095 {
00096     return 0;
00097 }
00098
00099 /*****
00100 PRIVATE int HTTPRequest_compare(HTTPRequest * this, HTTPRequest * compared)
00101 {
00102     return 0;
00103 }
00104
00105 /*****
00106 PRIVATE void HTTPRequest_print(HTTPRequest * this)
00107 {
00108     if (this->method == METHOD_INVALID) PRINT(("Method: INVALID\n"));
00109     if (this->method == METHOD_GET) PRINT(("Method: GET\n"));
00110     if (this->method == METHOD_POST) PRINT(("Method: POST\n"));
00111     PRINT(("Path: %s\n", String_getBuffer(this->path)));
00112     PRINT(("Version: %d.%d\n", this->majorVersion, this->minorVersion));
00113     //PRINT(("Host: %s\n", host));
00114     //PRINT(("User-Agent: %s\n", userAgent));
00115 }
00116
00117 /*****
00118 PRIVATE unsigned int HTTPRequest_getSize(HTTPRequest * this)

```

```

00153 {
00154     return sizeof(HTTPRequest);
00155 }
00156
00157 PRIVATE String * HTTPRequest_getPath(HTTPRequest* this)
00158 {
00159     return this->path;
00160 }
00161
00162 PRIVATE enum Method HTTPRequest_getMethod(HTTPRequest* this)
00163 {
00164     return this->method;
00165 }
00166
00167 PRIVATE int HTTPRequest_isValid(HTTPRequest* this)
00168 {
00169     return this->isValid;
00170 }
00171
00172 PRIVATE int HTTPRequest_parseBuffer(HTTPRequest* this, char* buffer)
00173 {
00174     int isValid = 0;
00175     char * path_start = buffer;
00176     int path_length = 0;
00177
00178     /* Read method */
00179     for (enum Method i = METHOD_GET; i < METHOD_INVALID; i++)
00180     {
00181         if (Memory_ncmp(buffer, methods_text[i], sizeof(methods_text[i]) - 1))
00182         {
00183             this->method = i;
00184             path_start = buffer + sizeof(methods_text[i]);
00185             isValid = 1;
00186             break;
00187         }
00188     }
00189
00190     /* Read path */
00191     while ((path_length < (int)Memory_len(buffer)) && (*(path_start + path_length) != ' '))
00192     {
00193         path_length++;
00194     }
00195
00196     char* path_buffer = Memory_alloc(path_length + 1);
00197     Memory_copy(path_buffer, path_start, path_length + 1);
00198     path_buffer[path_length + 1] = 0;
00199
00200     this->path = String_newByRef(path_buffer);
00201     char * version_start = path_start + path_length + 1;
00202
00203     /* Read version */
00204     if (Memory_ncmp(version_start, "HTTP/1.1", sizeof("HTTP/1.1") - 1))
00205     {
00206         this->majorVersion = 1;
00207         this->minorVersion = 1;
00208         isValid = isValid && 1;
00209     }
00210
00211     return isValid;
00212 }
00213 #endif /* _HTTPREQUEST_H_ */

```

4.119 HTTPResponse.h

```

00001 /* HTTPResponse.h */
00002 #ifndef _HTTPRESPONSE_H_
00003 #define _HTTPRESPONSE_H_
00004
00005 #include "Object.h"
00006 #include "Types.h"
00007 #include "Class.h"
00008 #include "String2.h"
00009 #include "Map.h"
00010 #include <stdio.h>
00011
00012 typedef struct HTTPResponse HTTPResponse;
00013
00014 enum Reason
00015 {
00016     REASON_OK,
00017     REASON_INVALID
00018 };
00019

```

```

00020 PRIVATE HTTPResponse * HTTPResponse_new();
00021 PRIVATE void HTTPResponse_delete(HTTPResponse * this);
00022 PRIVATE HTTPResponse* HTTPResponse_copy(HTTPResponse* this);
00023 PRIVATE int HTTPResponse_compare(HTTPResponse* this, HTTPResponse* compared);
00024 PRIVATE void HTTPResponse_print(HTTPResponse* this);
00025 PRIVATE unsigned int HTTPResponse_getSize(HTTPResponse* this);
00026 PRIVATE void HTTPResponse_setVersion(HTTPResponse* this, int majorVersion, int minorVersion);
00027 PRIVATE void HTTPResponse_setStatusCode(HTTPResponse* this, int code);
00028 PRIVATE void HTTPResponse_setReason(HTTPResponse* this, enum Reason);
00029 PRIVATE void HTTPResponse_addHeader(HTTPResponse* this, char* key, char* value);
00030 PRIVATE void HTTPResponse_setBody(HTTPResponse* this, char* body);
00031 PRIVATE int HTTPResponse_generate(HTTPResponse* this, char* buffer, int size);
00032
00033 /*****
00036 struct HTTPResponse
00037 {
00038     Object object;
00039     int statusCode;
00040     enum Reason reason;
00041     int majorVersion;
00042     int minorVersion;
00043     Map* headers;
00044     String* body;
00045     int isValid;
00046 };
00047
00048 /*****
00051 PRIVATE Class httpResponseClass =
00052 {
00053     .f_new = 0,
00054     .f_delete = (Destructor)&HTTPResponse_delete,
00055     .f_copy = (Copy_Operator)&HTTPResponse_copy,
00056     .f_comp = (Comp_Operator)&HTTPResponse_compare,
00057     .f_print = (Printer)&HTTPResponse_print,
00058     .f_size = (Sizer)&HTTPResponse_getSize
00059 };
00060
00061 /*****
00068 PRIVATE HTTPResponse* HTTPResponse_new()
00069 {
00070     HTTPResponse* this = 0;
00071
00072     this = (HTTPResponse*)Object_new(sizeof(HTTPResponse), &httpResponseClass);
00073
00074     if (this == 0) return 0;
00075
00076     this->statusCode = REASON_INVALID;
00077     this->majorVersion = 0;
00078     this->minorVersion = 0;
00079     this->headers = Map_new();
00080     this->body = 0;
00081     this->isValid = 0;
00082
00083     return this;
00084 }
00085
00086 /*****
00091 PRIVATE void HTTPResponse_delete(HTTPResponse* this)
00092 {
00093     if (!Object_isValid((Object*)this)) return;
00094
00095     String_delete(this->body);
00096     Map_delete(this->headers);
00097 }
00098
00099 /*****
00105 PRIVATE HTTPResponse* HTTPResponse_copy(HTTPResponse* this)
00106 {
00107     return 0;
00108 }
00109
00110 /*****
00116 PRIVATE int HTTPResponse_compare(HTTPResponse* this, HTTPResponse* compared)
00117 {
00118     return 0;
00119 }
00120
00121 /*****
00126 PRIVATE void HTTPResponse_print(HTTPResponse* this)
00127 {
00128
00129 }
00130
00131 /*****
00138 PRIVATE unsigned int HTTPResponse_getSize(HTTPResponse* this)
00139 {
00140     return sizeof(HTTPResponse);

```

```

00141 }
00142
00143 PRIVATE void HTTPResponse_setReason(HTTPResponse* this, enum Reason reason)
00144 {
00145     this->reason = REASON_OK;
00146 }
00147
00148 PRIVATE void HTTPResponse_setStatusCode(HTTPResponse* this, int statusCode)
00149 {
00150     this->statusCode = statusCode;
00151 }
00152
00153 PRIVATE void HTTPResponse_setVersion(HTTPResponse* this, int majorVersion, int minorVersion)
00154 {
00155     this->majorVersion = majorVersion;
00156     this->minorVersion = minorVersion;
00157 }
00158
00159 PRIVATE void HTTPResponse_addHeader(HTTPResponse* this, char* key, char* value)
00160 {
00161
00162 }
00163
00164 PRIVATE void HTTPResponse_setBody(HTTPResponse* this, char* body)
00165 {
00166     this->body = String_newByRef(body);
00167 }
00168
00169 PRIVATE int HTTPResponse_generate(HTTPResponse* this, char * buffer, int size)
00170 {
00171     char test_response[] = "HTTP/1.1 200 OK\r\n"
00172         "Content-Type: text/html; charset=UTF-8\r\n\r\n"
00173         "<doctype !html><html><head><title>Hello World</title></head>"
00174         "<body><h1>Hello world!</h1></body></html>\r\n";
00175
00176     int nbCharToWrite = snprintf(buffer, size, "HTTP/%d.%d %d OK\r\nContent-Type: text/html;
    charset=UTF-8\r\n\r\n%s", this->majorVersion, this->minorVersion, this->statusCode,
    String_getBuffer(this->body));
00177
00178     return nbCharToWrite;
00179 }
00180 #endif /* _HTTPRESPONSE_H_ */

```

4.120 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/↵ HTTPServer/HTTPServer.c File Reference

A HTTP Server class. This class provides server function to create, start HTML pages.

```

#include "HTTPServer.h"
#include "HTTPRequest.h"
#include "HTTPResponse.h"
#include "TaskMgr.h"
#include "Task.h"
#include "Object.h"
#include "Memory.h"
#include "Debug.h"
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <netinet/ip.h>
#include <unistd.h>
#include <pthread.h>
#include <time.h>
#include <errno.h>

```

Classes

- class [HTTPServer](#)
- struct [ConnectionParam](#)

Macros

- `#define REQUEST_BUFFER_SIZE (4096)`
- `#define RESPONSE_BUFFER_SIZE (4096)`

Functions

- `int msleep` (long msec)
- `PRIVATE void * HTTPServer_listenTaskBody` (void *params)
- `PUBLIC void HTTPServer_delete` (`HTTPServer` *this)
- `PUBLIC HTTPServer * HTTPServer_copy` (`HTTPServer` *this)
- `PUBLIC int HTTPServer_compare` (`HTTPServer` *this, `HTTPServer` *compared)
- `PUBLIC void HTTPServer_print` (`HTTPServer` *this)
- `PUBLIC unsigned int HTTPServer_getSize` (`HTTPServer` *this)
- `PUBLIC void HTTPServer_start` (`HTTPServer` *this)

4.120.1 Detailed Description

A HTTP Server class. This class provides server function to create, start HTML pages.

4.121 HTTPServer.h

```
00001 /* HTTPServer.h */
00002 #ifndef _HTTPSERVER_H_
00003 #define _HTTPSERVER_H_
00004
00005 #include "Types.h"
00006
00007 typedef struct HTTPServer HTTPServer;
00008
00009 PUBLIC HTTPServer * HTTPServer_new();
00010 PUBLIC void HTTPServer_delete(HTTPServer * this);
00011 PUBLIC HTTPServer* HTTPServer_copy(HTTPServer* this);
00012 PUBLIC int HTTPServer_compare(HTTPServer* this, HTTPServer* compared);
00013 PUBLIC void HTTPServer_print(HTTPServer* this);
00014 PUBLIC void HTTPServer_start(HTTPServer* this);
00015 PUBLIC unsigned int HTTPServer_getSize(HTTPServer* this);
00016 //PUBLIC void HTTPServer_start(HTTPServer* this);
00017 #endif /* _HTTPSERVER_H_ */
```

4.122 HTTPServer.h

```
00001 /* HTTPServer.h */
00002 #ifndef _HTTPSERVER_H_
00003 #define _HTTPSERVER_H_
00004
00005 #include "Types.h"
00006
00007 typedef struct HTTPServer HTTPServer;
00008
00009 PUBLIC HTTPServer * HTTPServer_new();
00010 PUBLIC void HTTPServer_delete(HTTPServer * this);
00011 PUBLIC HTTPServer* HTTPServer_copy(HTTPServer* this);
00012 PUBLIC int HTTPServer_compare(HTTPServer* this, HTTPServer* compared);
00013 PUBLIC void HTTPServer_print(HTTPServer* this);
00014 PUBLIC void HTTPServer_start(HTTPServer* this);
00015 PUBLIC unsigned int HTTPServer_getSize(HTTPServer* this);
00016 //PUBLIC void HTTPServer_start(HTTPServer* this);
00017 #endif /* _HTTPSERVER_H_ */
```

4.123 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/↵ SParse/SParse.c File Reference

This file contains the implementation for the class [SParse](#).

```
#include "SParse.h"
#include "Class.h"
#include "Object.h"
#include "FileReader.h"
#include "TransUnit.h"
#include "Configuration.h"
#include "Product.h"
#include "SdbMgr.h"
#include "Error.h"
#include "Grammar2.h"
#include "FileMgr.h"
#include "FileDesc.h"
#include "List.h"
#include "OptionMgr.h"
#include "Debug.h"
```

Classes

- class [SParse](#)

Functions

- PRIVATE unsigned int **SParse_parseFile** ([SParse](#) *this, [FileDesc](#) *fileDesc, [FileMgr](#) *fileMgr)
- PUBLIC unsigned int **SParse_getSize** ([SParse](#) *this)

4.123.1 Detailed Description

This file contains the implementation for the class [SParse](#).

The class [SParse](#) parses all files with extension .X and stores the result of the parsing in the SQLite DB name.

4.124 SParse.h

```
00001 /* SParse.h */
00002
00003 #ifndef _SPARSE_H_
00004 #define _SPARSE_H_
00005
00006 #include "Types.h"
00007
00008 typedef struct SParse SParse;
00009
00010 PUBLIC SParse *SParse_new(/* Sdb name */);
00011 PUBLIC void SParse_delete(SParse * this);
00012 PUBLIC SParse * SParse_copy(SParse * this);
00013 PUBLIC void SParse_print(SParse * this);
00014 PUBLIC unsigned int SParse_getSize(SParse * this);
00015 PUBLIC unsigned int SParse_parse(SParse * this, const char * extension);
00016
00017 #endif /* _SPARSE_H_ */
```

4.125 SParse.h

```

00001  /* SParse.h */
00002
00003  #ifndef _SPARSE_H_
00004  #define _SPARSE_H_
00005
00006  #include "Types.h"
00007
00008  typedef struct SParse SParse;
00009
00010  PUBLIC SParse *SParse_new(/* Sdb name */);
00011  PUBLIC void SParse_delete(SParse * this);
00012  PUBLIC SParse * SParse_copy(SParse * this);
00013  PUBLIC void SParse_print(SParse * this);
00014  PUBLIC unsigned int SParse_getSize(SParse * this);
00015  PUBLIC unsigned int SParse_parse(SParse * this, const char * extension);
00016
00017  #endif /* _SPARSE_H_ */

```

4.126 Buffer.h

```

00001  /* Buffer.h */
00002  #ifndef _BUFFER_H_
00003  #define _BUFFER_H_
00004
00005  #include "Types.h"
00006  #include "Class.h"
00007  #include "Object.h"
00008  #include "Debug.h"
00009
00010  typedef struct Buffer Buffer;
00011
00012  PRIVATE Buffer * Buffer_new();
00013  PRIVATE void Buffer_delete(Buffer * this);
00014  PRIVATE void Buffer_print(Buffer * this);
00015  PRIVATE unsigned int Buffer_getSize(Buffer * this);
00016
00017  struct Buffer
00018  {
00019      Object object;
00020      String* string;
00021      char* currentPtr;
00022      char* startPtr;
00023      int nbCharRead;
00024  };
00025
00026  PRIVATE Class bufferClass =
00027  {
00028      .f_new = (Constructor)0,
00029      .f_delete = (Destructor)&Buffer_delete,
00030      .f_copy = (Copy_Operator)0,
00031      .f_comp = (Comp_Operator)0,
00032      .f_print = (Printer)&Buffer_print,
00033      .f_size = (Sizer)&Buffer_getSize
00034  };
00035
00036  PRIVATE Buffer * Buffer_new(String * content)
00037  {
00038      Buffer * this = 0;
00039
00040      this = (Buffer*)Object_new(sizeof(Buffer), &bufferClass);
00041
00042      this->string = content;
00043      this->startPtr = String_getBuffer(this->string);
00044      this->currentPtr = this->startPtr;
00045      this->nbCharRead = 0;
00046
00047      return this;
00048  }
00049
00050  PRIVATE void Buffer_delete(Buffer * this)
00051  {
00052      if (this == 0) return;
00053
00054      /* De-allocate the specific members */
00055      String_delete(this->string);
00056      this->startPtr = 0;
00057      this->currentPtr = 0;
00058      this->nbCharRead = 0;
00059      /* De-allocate the base object */
00060      Object_deallocate(&this->object);
00061  }

```

```

00062
00063 PRIVATE void Buffer_print(Buffer * this)
00064 {
00065 }
00066 }
00067
00068 PRIVATE unsigned int Buffer_getSize(Buffer * this)
00069 {
00070     return sizeof(this);
00071 }
00072 #endif /* _BUFFER_H_ */

```

4.127 MacroDefinition.h

```

00001 /* MacroDefinition.h */
00002 #ifndef _MACRODEFINITION_H_
00003 #define _MACRODEFINITION_H_
00004
00005 #include "Types.h"
00006 #include "Class.h"
00007 #include "Object.h"
00008
00009 typedef struct MacroDefinition MacroDefinition;
00010
00011 PRIVATE MacroDefinition * MacroDefinition_new(List * parameters, String * body);
00012 PRIVATE void MacroDefinition_delete(MacroDefinition * this);
00013 PRIVATE void MacroDefinition_print(MacroDefinition * this);
00014 PRIVATE unsigned int MacroDefinition_getSize(MacroDefinition * this);
00015
00016 struct MacroDefinition
00017 {
00018     Object object;
00019     String* body;
00020     List* parameters;
00021 };
00022
00023 PRIVATE Class macroDefinitionClass =
00024 {
00025     .f_new = (Constructor)0,
00026     .f_delete = (Destructor)&MacroDefinition_delete,
00027     .f_copy = (Copy_Operator)0,
00028     .f_comp = (Comp_Operator)0,
00029     .f_print = (Printer)&MacroDefinition_print,
00030     .f_size = (Sizer)&MacroDefinition_getSize
00031 };
00032
00033 PRIVATE MacroDefinition* MacroDefinition_new(List * parameters, String * body)
00034 {
00035     MacroDefinition * this = (MacroDefinition*)Object_new(sizeof(MacroDefinition),
00036 &macroDefinitionClass);
00037     this->parameters = parameters;
00038     this->body = body;
00039     return this;
00040 }
00041
00042 PRIVATE void MacroDefinition_delete(MacroDefinition* this)
00043 {
00044     if (this == 0) return;
00045     /* De-allocate the specific members */
00046     List_delete(this->parameters);
00047     String_delete(this->body);
00048     /* De-allocate the base object */
00049     Object_deallocate(&this->object);
00050 }
00051
00052 PRIVATE void MacroDefinition_print(MacroDefinition* this)
00053 {
00054 }
00055
00056 PRIVATE unsigned int MacroDefinition_getSize(MacroDefinition* this)
00057 {
00058     return sizeof(MacroDefinition);
00059 }
00060 #endif /* _MACRODEFINITION_H_ */

```


4.128 MacroStore.h

```

00001 /* MacroStore.h */
00002 #ifndef _MACROSTORE_H_
00003 #define _MACROSTORE_H_
00004
00005 #include "Types.h"
00006 #include "Class.h"
00007 #include "Object.h"
00008 #include "String2.h"
00009 #include "Memory.h"
00010 #include "MacroDefinition.h"
00011 #include "Debug.h"
00012
00013 #define MAX_CHILDREN (28)
00014
00015 typedef struct MacroStore MacroStore;
00016
00017
00018 PRIVATE char convert[256];
00019 PRIVATE char convertBack[MAX_CHILDREN];
00020
00021 enum MacroEvalName
00022 {
00023     E_NOT_MACRO,
00024     E_POSSIBLE_MACRO,
00025     E_DEFINED_MACRO
00026 };
00027
00028 struct MacroStoreNode
00029 {
00030     int isLeaf;
00031     MacroDefinition * def;
00032     void * children[MAX_CHILDREN];
00033 };
00034
00035 struct MacroStore
00036 {
00037     Object object;
00038     struct MacroStoreNode * root;
00039 };
00040
00041 PRIVATE MacroStore * MacroStore_new();
00042 PRIVATE void MacroStore_delete(MacroStore * this);
00043 PRIVATE void MacroStore_print(MacroStore * this);
00044 PRIVATE unsigned int MacroStore_getSize(MacroStore * this);
00045 PRIVATE int MacroStore_insertName(MacroStore* this, String* name, MacroDefinition* body);
00046 PRIVATE int MacroStore_isDefName(MacroStore* this, String* name);
00047 PRIVATE int MacroStore_removeName(MacroStore* this, String* name);
00048 PRIVATE enum MacroEvalName MacroStore_evalName(MacroStore* this, char* ptr, int length);
00049 PRIVATE void MacroStore_printChildrenNodes(MacroStore* this, struct MacroStoreNode* node, char* name,
    int l);
00050 PRIVATE void MacroStore_deleteChildrenNodes(MacroStore* this, struct MacroStoreNode* node);
00051 PRIVATE String * MacroStore_expandMacro(MacroStore * this, String * inStr);
00052
00053 PRIVATE Class macroStoreClass =
00054 {
00055     .f_new = (Constructor)0,
00056     .f_delete = (Destructor)&MacroStore_delete,
00057     .f_copy = (Copy_Operator)0,
00058     .f_comp = (Comp_Operator)0,
00059     .f_print = (Printer)&MacroStore_print,
00060     .f_size = (Sizer)&MacroStore_getSize
00061 };
00062
00063
00064
00065 PRIVATE MacroStore* MacroStore_new()
00066 {
00067     MacroStore * this = (MacroStore*)Object_new(sizeof(MacroStore), &macroStoreClass);
00068
00069     this->root = (struct MacroStoreNode * )Memory_alloc(sizeof(struct MacroStoreNode));
00070
00071     for (int i = 0; i < MAX_CHILDREN; i++)
00072     {
00073         this->root->children[i] = 0;
00074         this->root->isLeaf = 1;
00075         this->root->def = 0;
00076     }
00077
00078     for (int c = 0; c < 255; c++)
00079     {
00080         if ((c >= 'A') && (c <= 'Z'))
00081         {
00082             convert[c] = c - 'A' + 2;
00083             convertBack[c - 'A' + 2] = c;
00084         }

```

```

00085     else if (c == '_')
00086     {
00087         convert[c] = 1;
00088         convertBack[1] = c;
00089     }
00090     else
00091     {
00092         convert[c] = 0;
00093         convertBack[0] = 0;
00094     }
00095 }
00096 return this;
00097 }
00098
00099 PRIVATE void MacroStore_delete(MacroStore* this)
00100 {
00101     if (this == 0) return;
00102     /* De-allocate the specific members */
00103     for (int i = 0; i < MAX_CHILDREN; i++)
00104     {
00105         if (this->root->children[i])
00106         {
00107             MacroStore_deleteChildrenNodes(this, this->root->children[i]);
00108             Memory_free(this->root->children[i], sizeof(struct MacroStoreNode));
00109             this->root->children[i] = 0;
00110         }
00111     }
00112     Memory_free(this->root, sizeof(struct MacroStoreNode));
00113
00114     /* De-allocate the base object */
00115     Object_deallocate(&this->object);
00116 }
00117
00118 PRIVATE void MacroStore_print(MacroStore* this)
00119 {
00120     struct MacroStoreNode* currentNode = this->root;
00121     char * name = Memory_alloc(256); // MAX Macro name length
00122     int l = 0;
00123     for (int i = 0; i < MAX_CHILDREN; i++)
00124     {
00125         if (currentNode->children[i])
00126         {
00127             name[l] = convertBack[i];
00128             if (!currentNode->isLeaf)
00129                 MacroStore_printChildrenNodes(this, currentNode->children[i], name, l + 1);
00130             else
00131             {
00132                 name[l] = 0;
00133             }
00134             //PRINT(("s\n", name));
00135         }
00136     }
00137     Memory_free(name, 256);
00138 }
00139
00140 PRIVATE unsigned int MacroStore_getSize(MacroStore* this)
00141 {
00142     return sizeof(MacroStore);
00143 }
00144
00145 PRIVATE int MacroStore_insertName(MacroStore* this, String * name, MacroDefinition* body)
00146 {
00147     char* buffer = String_getBuffer(name);
00148     int length = String_getLength(name);
00149     struct MacroStoreNode* currentNode = this->root;
00150
00151     int c = 0;
00152     for (int i = 0; i < length; i++)
00153     {
00154         c = convert[buffer[i]];
00155         if (currentNode->isLeaf)
00156         {
00157             currentNode->isLeaf = 0;
00158             currentNode->children[c] = Memory_alloc(sizeof(struct MacroStoreNode));
00159             currentNode = currentNode->children[c];
00160             for (int c = 0; c < MAX_CHILDREN; c++)
00161                 currentNode->children[c] = 0;
00162             currentNode->isLeaf = 1;
00163             currentNode->def = 0;
00164         }
00165         else if (currentNode->children[c])
00166             currentNode = currentNode->children[c];
00167         else
00168         {
00169             currentNode->children[c] = Memory_alloc(sizeof(struct MacroStoreNode));
00170             currentNode = currentNode->children[c];
00171             for (int c = 0; c < MAX_CHILDREN; c++)

```

```

00172         currentNode->children[c] = 0;
00173         currentNode->isLeaf = 1;
00174         currentNode->def = 0;
00175     }
00176 }
00177 currentNode->isLeaf = 1;
00178 currentNode->def = body;
00179
00180 return 0;
00181 }
00182
00183 PRIVATE int MacroStore_isDefName(MacroStore* this, String* name)
00184 {
00185     char* buffer = String_getBuffer(name);
00186     int length = String_getLength(name);
00187     struct MacroStoreNode* currentNode = this->root;
00188     for (int i = 0; i < length; i++)
00189     {
00190         int c = convert[buffer[i]];
00191         if (currentNode->children[c]) currentNode = currentNode->children[c];
00192         else
00193             return 0; // Not found
00194     }
00195     if (currentNode->def) return 1; // Found
00196     return 0;
00197 }
00198
00199 PRIVATE int MacroStore_removeName(MacroStore* this, String* name)
00200 {
00201     char* buffer = String_getBuffer(name);
00202     int length = String_getLength(name);
00203     struct MacroStoreNode* currentNode = this->root;
00204     if (currentNode->isLeaf) return 0;
00205     for (int i = 0; i < length; i++)
00206     {
00207         int c = convert[buffer[i]];
00208     }
00209     return 0;
00210 }
00211
00212 }
00213
00214 PRIVATE enum MacroEvalName MacroStore_evalName(MacroStore* this, char* buffer, int length)
00215 {
00216     if (this == 0) return E_NOT_MACRO;
00217     if (length <= 0) return E_NOT_MACRO;
00218     struct MacroStoreNode* currentNode = this->root;
00219     int c = 0;
00220     int i;
00221     for (i = 0; i < length; i++)
00222     {
00223         c = convert[buffer[i]];
00224         if (currentNode->children[c])
00225             currentNode = currentNode->children[c];
00226         else
00227             return E_NOT_MACRO;
00228     }
00229     if (currentNode->def != 0) return E_DEFINED_MACRO;
00230     return E_POSSIBLE_MACRO;
00231 }
00232
00233 }
00234
00235 PRIVATE void MacroStore_printChildrenNodes(MacroStore* this, struct MacroStoreNode* node, char* name,
int l)
00236 {
00237     if (node == 0) return;
00238     if (node->isLeaf)
00239     {
00240         name[l] = 0;
00241         PRINT(("s\n", name));
00242         return;
00243     }
00244     else
00245     {
00246         if (node->def)
00247         {
00248             name[l] = 0;
00249             PRINT(("s\n", name));
00250         }
00251         for (int i = 0; i < MAX_CHILDREN; i++)
00252         {
00253             name[l] = convertBack[i];
00254             if (node->children[i]) MacroStore_printChildrenNodes(this, node->children[i], name, l + 1);
00255         }
00256     }
00257 }

```

```

00258 }
00259 PRIVATE void MacroStore_deleteChildrenNodes(MacroStore* this, struct MacroStoreNode* node)
00260 {
00261     if (node == 0) return;
00262     MacroDefinition_delete(node->def);
00263     if (node->isLeaf) return;
00264     for (int i = 0; i < MAX_CHILDREN; i++)
00265     {
00266         if (node->children[i])
00267         {
00268             MacroStore_deleteChildrenNodes(this, node->children[i]);
00269             Memory_free(node->children[i], sizeof(struct MacroStoreNode*));
00270             node->children[i] = 0;
00271         }
00272     }
00273 }
00274 PRIVATE String* MacroStore_expandMacro(MacroStore* this, String* inStr)
00275 {
00276     int length = 1;
00277     enum MacroEvalName status;
00278     status = MacroStore_evalName(this, String_getBuffer(inStr), length);
00279     if (status == E_NOT_MACRO) return 0;
00280     while ((status == E_POSSIBLE_MACRO) && (length < String_getLength(inStr)))
00281     {
00282         length++;
00283         status = MacroStore_evalName(this, String_getBuffer(inStr), length);
00284     }
00285     if (status == E_POSSIBLE_MACRO) return 0;
00286     return 0;
00287 }
00288 #endif /* _MACROSTORE_H_ */

```

4.129 test.h

```
00001 void header();
```

4.130 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/TransUnit.c File Reference

This file implements a class that extract C code.

```

#include "TransUnit.h"
#include "MacroDefinition.h"
#include "MacroStore.h"
#include "Buffer.h"
#include "List.h"
#include "Map.h"
#include "String2.h"
#include "Memory.h"
#include "Error.h"
#include "Object.h"
#include "Debug.h"

```

Classes

- class [TransUnit](#)

Macros

- #define **DEBUG** (0)
- #define **IS_MACRO_LETTER**(C) (((C)>='A' && (C)<='Z')) || (((C)>='a' && (C)<='z')) || ((C)=='_')
- #define **OUTPUT_BUFFER_SIZE** (20000)

Functions

- PRIVATE void **TransUnit_consumeLineComment** ([TransUnit](#) *this)
- PRIVATE void **TransUnit_consumeMultilineComment** ([TransUnit](#) *this)
- PRIVATE void **TransUnit_consumeInclude** ([TransUnit](#) *this)
- PRIVATE void **TransUnit_readMacroDefinition** ([TransUnit](#) *this)
- PRIVATE void **TransUnit_checkMacro** ([TransUnit](#) *this, int checkForTrue)
- PRIVATE int **TransUnit_pushNewBuffer** ([TransUnit](#) *this, [String](#) *content)
- PRIVATE int **TransUnit_popBuffer** ([TransUnit](#) *this)
- PRIVATE int **TransUnit_expandMacro** ([TransUnit](#) *this)

4.130.1 Detailed Description

This file implements a class that extract C code.

It removes the comments, expands the macros, parses the include files

4.131 TransUnit.h

```
00001 /* TransUnit.h */
00002 #ifndef _TRANSUNIT_H_
00003 #define _TRANSUNIT_H_
00004
00005 #include "Types.h"
00006 #include "FileMgr.h"
00007
00008 typedef struct TransUnit TransUnit;
00009
00010 PUBLIC TransUnit * TransUnit_new(FileDesc * file, FileMgr * fileMgr);
00011 PUBLIC void TransUnit_delete(TransUnit * this);
00012 PUBLIC void TransUnit_print(TransUnit* this);
00013 PUBLIC unsigned int TransUnit_getSize(TransUnit* this);
00014 PUBLIC char* TransUnit_getName(TransUnit* this);
00015 PUBLIC String * TransUnit_getNextBuffer(TransUnit* this);
00016
00017 #endif /* _CONFIGURATION_H_ */
```

4.132 TransUnit.h

```
00001 /* TransUnit.h */
00002 #ifndef _TRANSUNIT_H_
00003 #define _TRANSUNIT_H_
00004
00005 #include "Types.h"
00006 #include "FileMgr.h"
00007
00008 typedef struct TransUnit TransUnit;
00009
00010 PUBLIC TransUnit * TransUnit_new(FileDesc * file, FileMgr * fileMgr);
00011 PUBLIC void TransUnit_delete(TransUnit * this);
00012 PUBLIC void TransUnit_print(TransUnit* this);
00013 PUBLIC unsigned int TransUnit_getSize(TransUnit* this);
00014 PUBLIC char* TransUnit_getName(TransUnit* this);
00015 PUBLIC String * TransUnit_getNextBuffer(TransUnit* this);
00016
00017 #endif /* _CONFIGURATION_H_ */
```


Index

/home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/[FileMgr.c](#),
[73](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Allocator/[Allocator.c](#),
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/[FileMgr.h](#),
[74](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Allocator/[Allocator.h](#),
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/[FileMgr.c](#),
[74](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Array/Array.c,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/[FileMgr.h](#),
[75](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Array/Array.h,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/OptionMgr/[OptionMgr.c](#),
[76](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/BTree.c,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/OptionMgr/[OptionMgr.h](#),
[77](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/BTree.h,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/SdbMgr/[SdbMgr.c](#),
[77](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/Node.c,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/SdbMgr/[SdbMgr.h](#),
[78](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/Node.h,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/SdbMgr/[SdbMgrRequest.h](#),
[79](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/NodeTest.c,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/Storage/[Storage.h](#),
[79](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Debug/Debug.c,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/TaskMgr/[TaskMgr.c](#),
[80](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Debug/Debug.h,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/TaskMgr/[TaskMgr.h](#),
[81](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Error/Error.c,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/TaskMgr/[TaskMgr.h](#),
[81](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Error/Error.h,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/TimeMgr/[TimeMgr.c](#),
[82](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/FileIo/FileIo.c,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/TimeMgr/[TimeMgr.h](#),
[83](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/FileIo/FileIo.h,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/TimeMgr/[Timer.h](#),
[83](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/List.c,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/[FileMgrDesc.h](#),
[74](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/List.h,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/[FileMgr.h](#),
[76](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/ListNode.c,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/[OptionMgr.h](#),
[77](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/tests/ListTest.c,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/[SdbMgr.h](#),
[78](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/tests/ListTest.h,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/[SdbMgrRequest.h](#),
[79](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/Map.c,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/[TaskMgr.h](#),
[80](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/Map.h,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/[TaskMgr.h](#),
[81](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/MapNode.c,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/[TimeMgr.h](#),
[82](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/MapNode.h,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/[Timer.h](#),
[82](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/MapNodeTest.c,
/home/thomas/Projects/SParse-master/SParse/src/AppliLib/include/[Timer.h](#),
[85](#) /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/MapNodeTest.h,

[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/De](#)
[101](#) [95](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Er](#)
[89](#) [96](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Fil](#)
[91](#) [98](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Lis](#)
[105](#) [99](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Ma](#)
[106](#) [84](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Ma](#)
[107](#) [103](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Ma](#)
[107](#) [105](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Ma](#)
[108](#) [106](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/No](#)
[109](#) [88](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Ob](#)
[110](#) [108](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Ob](#)
[111](#) [110](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Ob](#)
[102](#) [110](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Po](#)
[111](#) [119](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Sk](#)
[119](#) [121](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/St](#)
[120](#) [123](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Tir](#)
[121](#) [124](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Ty](#)
[121](#) [124](#)
[/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/Us](#)
[102](#) [125](#)
[/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Ast/Ast.h,](#)
[89](#) [128](#)
[/home/thomas/Projects/SParse-master/SParse/src/ParseLib/C89Grammar](#)
[92](#) [128](#)
[/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Configuration](#)
[123](#) [128](#)
[/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Configuration](#)
[102](#) [130](#)
[/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Configuration](#)
[124](#) [130](#)
[/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Configuration](#)
[124](#) [131](#)
[/home/thomas/Projects/SParse-master/SParse/src/ParseLib/FileReader/Fi](#)
[125](#) [132](#)
[/home/thomas/Projects/SParse-master/SParse/src/ParseLib/FileReader/Fi](#)
[84](#) [133](#)
[/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar/Gra](#)
[86](#) [133](#)
[/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar2/G](#)
[87](#) [134](#)
[/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar2/G](#)
[107](#) [134](#)

/home/thomas/Projects/SParse-master/SParse/src/ParseLib/GrammarC99/GrammarC99.h,
 136
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/GrammarC99/GrammarC99.parse.h,
 136
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/GrammarC99/MyType.h,
 138
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPRequest.h,
 138
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPResponse.h,
 140
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPServer.c,
 142
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/HTTPServer/HTTPServer.h,
 143
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/SParse/SParse.c,
 144
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/SParse/SParse.h,
 145
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/Buffer.h,
 145
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/MacroDefinition.h,
 146
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/MacroStore.h,
 147
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/TransUnit.c,
 150
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/TransUnit.h,
 151
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/TransUnit/test/test.h,
 150
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/Configuration.h,
 130
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/FileReader.h,
 133
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/Grammar.h,
 133
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/Grammar2.h,
 134
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/GrammarC99.h,
 136
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/HTTPServer.h,
 143
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/Product.h,
 131
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/SParse.h,
 144
 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/TransUnit.h,
 151
 /home/thomas/Projects/SParse-master/SParse/src/main.c, 125
 Allocator, 7
 AllocInfo, 7
 Array, 8
 Array_compare, 8
 Array_copy, 8
 Array_new, 8
 Array_newFromFile, 9
 Array_compare
 Array, 8
 Array_copy, 8
 Array_new, 8
 Array_newFromFile, 9
 Array_param, 9
 BTree, 9
 Buffer, 10
 Class, 10
 Configuration, 10
 Configuration_new, 11
 Configuration.c
 IS_KEY, 129
 IS_STRING, 129
 Configuration_new, 11
 ConnectionParam, 11
 Declarator, 11
 Error.c
 Error_new, 95
 Error_new
 Error.c, 95
 FileBase, 12
 Fileilo, 12
 Fileilo_comp, 13
 Fileilo_createFile, 13
 Fileilo_delete, 14
 Fileilo_getSize, 14
 Fileilo_new, 14
 Fileilo_openDir, 14
 Fileilo_openFile, 15
 Fileilo_print, 15
 Fileilo_comp, 13
 Fileilo, 13
 Fileilo_createFile, 13
 Fileilo_delete, 14
 Fileilo_getSize, 14
 Fileilo_new, 14
 Fileilo_openDir, 14
 Fileilo_openFile, 15
 Fileilo_print, 15
 Fileilo, 14
 Fileilo_new
 Fileilo, 14
 Fileilo_openDir
 Fileilo, 14
 Fileilo_openFile
 Fileilo, 15
 Fileilo_print
 Fileilo, 15

- FileMgr, 15
 - FileMgr_addDirectory, 16
 - FileMgr_addFile, 16
 - FileMgr_copy, 16
 - FileMgr_filterFiles, 17
 - FileMgr_getRef, 17
 - FileMgr_getRootLocation, 17
 - FileMgr_getSize, 17
 - FileMgr_load, 18
 - FileMgr_setRootLocation, 18
 - FileMgr_write, 18
- FileMgr_addDirectory
 - FileMgr, 16
- FileMgr_addFile
 - FileMgr, 16
- FileMgr_copy
 - FileMgr, 16
- FileMgr_filterFiles
 - FileMgr, 17
- FileMgr_getRef
 - FileMgr, 17
- FileMgr_getRootLocation
 - FileMgr, 17
- FileMgr_getSize
 - FileMgr, 17
- FileMgr_load
 - FileMgr, 18
- FileMgr_setRootLocation
 - FileMgr, 18
- FileMgr_write
 - FileMgr, 18
- FileReader, 19
 - FileReader_addFile, 19
 - FileReader_copy, 19
 - FileReader_getBuffer, 20
 - FileReader_getName, 20
 - FileReader_getSize, 20
 - FileReader_new, 20
- FileReader.c
 - includeInfoClass, 132
- FileReader_addFile
 - FileReader, 19
- FileReader_copy
 - FileReader, 19
- FileReader_getBuffer
 - FileReader, 20
- FileReader_getName
 - FileReader, 20
- FileReader_getSize
 - FileReader, 20
- FileReader_new
 - FileReader, 20
- Grammar, 21
- Grammar2, 21
 - Grammar2_copy, 22
 - Grammar2_new, 22
- Grammar2_copy
 - Grammar2, 22
- Grammar2_new
 - Grammar2, 22
- GrammarC99, 22
 - GrammarC99_new, 23
- GrammarC99_new
 - GrammarC99, 23
- GrammarContext, 23
- HttpRequest, 23
 - HttpRequest_compare, 24
 - HttpRequest_copy, 24
 - HttpRequest_getSize, 24
- HttpRequest_compare
 - HttpRequest, 24
- HttpRequest_copy
 - HttpRequest, 24
- HttpRequest_getSize
 - HttpRequest, 24
- HttpRequest_new
 - HTTPServer, 25
- HttpResponse, 25
- HttpResponse_compare
 - HTTPServer, 26
- HttpResponse_getSize
 - HTTPServer, 26
- HTTPServer, 25
 - HttpRequest_new, 25
 - HttpResponse_compare, 26
 - HttpResponse_getSize, 26
 - HTTPServer_new, 26
- HTTPServer_new
 - HTTPServer, 26
- IncludeInfo, 27
- includeInfoClass
 - FileReader.c, 132
- IS_KEY
 - Configuration.c, 129
- IS_STRING
 - Configuration.c, 129
- List, 27
 - List_compare, 28
 - List_copy, 28
 - List_forEach, 28
 - List_getNbNodes, 29
 - List_getSize, 29
 - List_insertHead, 29
 - List_insertTail, 29
 - List_merge, 30
 - List_new, 30
 - List_newFromAllocator, 30
 - ListNode_compare, 30
 - ListNode_copy, 30
 - ListNode_new, 31
 - ListNode_newFromAllocator, 31
- List_compare
 - List, 28
- List_copy

- List, [28](#)
- List_forEach
 - List, [28](#)
- List_getNbNodes
 - List, [29](#)
- List_getSize
 - List, [29](#)
- List_insertHead
 - List, [29](#)
- List_insertTail
 - List, [29](#)
- List_merge
 - List, [30](#)
- List_new
 - List, [30](#)
- List_newFromAllocator
 - List, [30](#)
- ListNode_compare
 - List, [30](#)
- ListNode_copy
 - List, [30](#)
- ListNode_new
 - List, [31](#)
- ListNode_newFromAllocator
 - List, [31](#)
- MacroDefinition, [31](#)
- MacroStore, [32](#)
- MacroStoreNode, [32](#)
- main
 - main.c, [126](#)
- main.c
 - main, [126](#)
 - print_usage, [126](#)
 - sighandler, [126](#)
- Malloc, [32](#)
- Map, [33](#)
 - Map_copy, [33](#)
 - Map_getAll, [33](#)
 - Map_getSize, [33](#)
 - Map_insert, [34](#)
 - Map_new, [34](#)
 - Map_newFromAllocator, [34](#)
 - TaskMgr_new, [34](#)
- Map_copy
 - Map, [33](#)
- Map_getAll
 - Map, [33](#)
- Map_getSize
 - Map, [33](#)
- Map_insert
 - Map, [34](#)
- Map_new
 - Map, [34](#)
- Map_newFromAllocator
 - Map, [34](#)
- MapEntry, [35](#)
- maxNbObjectAllocated
 - ObjectMgr, [41](#)
- mem_align, [35](#)
- Mutex, [36](#)
- MyAllocator, [36](#)
- MyType, [36](#)
- Node, [37](#)
- Object, [37](#)
 - Object_comp, [38](#)
 - Object_copy, [38](#)
 - Object_getRef, [38](#)
 - Object_isValid, [38](#)
 - Object_new, [38](#)
 - Object_newFromAllocator, [39](#)
 - Object_print, [39](#)
- Object_comp
 - Object, [38](#)
- Object_copy
 - Object, [38](#)
- Object_getRef
 - Object, [38](#)
- Object_isValid
 - Object, [38](#)
- Object_new
 - Object, [38](#)
- Object_newFromAllocator
 - Object, [39](#)
- Object_print
 - Object, [39](#)
- ObjectInfo, [39](#)
- ObjectMgr, [40](#)
 - maxNbObjectAllocated, [41](#)
 - ObjectMgr_allocate, [40](#)
 - ObjectMgr_copy, [40](#)
 - ObjectMgr_deallocate, [41](#)
 - ObjectMgr_getRef, [41](#)
- ObjectMgr_allocate
 - ObjectMgr, [40](#)
- ObjectMgr_copy
 - ObjectMgr, [40](#)
- ObjectMgr_deallocate
 - ObjectMgr, [41](#)
- ObjectMgr_getRef
 - ObjectMgr, [41](#)
- ObjectStore, [42](#)
 - ObjectStore_compare, [42](#)
 - ObjectStore_copy, [42](#)
 - ObjectStore_createAllocator, [43](#)
 - ObjectStore_createObject, [43](#)
 - ObjectStore_delete, [43](#)
 - ObjectStore_deleteAllocator, [43](#)
 - ObjectStore_deleteObject, [43](#)
 - ObjectStore_getNbAllocatedObjects, [44](#)
 - ObjectStore_getRef, [44](#)
- ObjectStore_compare
 - ObjectStore, [42](#)
- ObjectStore_copy
 - ObjectStore, [42](#)
- ObjectStore_createAllocator

- ObjectStore, [43](#)
- ObjectStore_createObject
 - ObjectStore, [43](#)
- ObjectStore_delete
 - ObjectStore, [43](#)
- ObjectStore_deleteAllocator
 - ObjectStore, [43](#)
- ObjectStore_deleteObject
 - ObjectStore, [43](#)
- ObjectStore_getNbAllocatedObjects
 - ObjectStore, [44](#)
- ObjectStore_getRef
 - ObjectStore, [44](#)
- OptionDefault, [44](#)
- OptionMgr, [45](#)
 - OptionMgr_getRef, [45](#)
 - OptionMgr_readFromCmdLine, [45](#)
- OptionMgr_getRef
 - OptionMgr, [45](#)
- OptionMgr_readFromCmdLine
 - OptionMgr, [45](#)
- Pool.c
 - Pool_alloc, [113](#)
 - Pool_allocInFile, [113](#)
 - Pool_dealloc, [113](#)
 - Pool_deallocInFile, [114](#)
 - Pool_deallocInMemory, [114](#)
 - Pool_delete, [114](#)
 - Pool_new, [115](#)
 - Pool_newFromFile, [115](#)
 - Pool_read, [115](#)
 - Pool_readInFile, [116](#)
 - Pool_readInMemory, [116](#)
 - Pool_report, [116](#)
 - Pool_reportInFile, [117](#)
 - Pool_reportInMemory, [117](#)
 - Pool_reportNbNodes, [117](#)
 - Pool_reportSizeInBytes, [118](#)
 - Pool_write, [118](#)
 - Pool_writeInFile, [118](#)
 - Pool_writeInMemory, [118](#)
- Pool_alloc
 - Pool.c, [113](#)
- Pool_allocInFile
 - Pool.c, [113](#)
- Pool_dealloc
 - Pool.c, [113](#)
- Pool_deallocInFile
 - Pool.c, [114](#)
- Pool_deallocInMemory
 - Pool.c, [114](#)
- Pool_delete
 - Pool.c, [114](#)
- Pool_new
 - Pool.c, [115](#)
- Pool_newFromFile
 - Pool.c, [115](#)
- Pool_read
 - Pool.c, [115](#)
- Pool.c, [115](#)
- Pool_readInFile
 - Pool.c, [116](#)
- Pool_readInMemory
 - Pool.c, [116](#)
- Pool_report
 - Pool.c, [116](#)
- Pool_reportInFile
 - Pool.c, [117](#)
- Pool_reportInMemory
 - Pool.c, [117](#)
- Pool_reportNbNodes
 - Pool.c, [117](#)
- Pool_reportSizeInBytes
 - Pool.c, [118](#)
- Pool_write
 - Pool.c, [118](#)
- Pool_writeInFile
 - Pool.c, [118](#)
- Pool_writeInMemory
 - Pool.c, [118](#)
- PoolCache, [46](#)
- print_usage
 - main.c, [126](#)
- Product, [46](#)
- SdbMgr, [46](#)
 - SdbMgr_copy, [47](#)
 - SdbMgr_execute, [47](#)
 - SdbMgr_getRef, [47](#)
- SdbMgr_copy
 - SdbMgr, [47](#)
- SdbMgr_execute
 - SdbMgr, [47](#)
- SdbMgr_getRef
 - SdbMgr, [47](#)
- SdbRequest, [48](#)
 - SdbRequest_delete, [48](#)
 - SdbRequest_execute, [48](#)
 - SdbRequest_new, [48](#)
- SdbRequest_delete
 - SdbRequest, [48](#)
- SdbRequest_execute
 - SdbRequest, [48](#)
- SdbRequest_new
 - SdbRequest, [48](#)
- sighandler
 - main.c, [126](#)
- SkipList, [49](#)
 - SkipList_add, [50](#)
 - SkipList_compare, [51](#)
 - SkipList_copy, [51](#)
 - SkipList_delete, [51](#)
 - SkipList_getSize, [52](#)
 - SkipList_new, [52](#)
 - SkipList_newFromAllocator, [52](#)
 - SkipList_print, [53](#)
 - SkipList_remove, [53](#)
- SkipList_add

- SkipList, 50
- SkipList_compare
 - SkipList, 51
- SkipList_copy
 - SkipList, 51
- SkipList_delete
 - SkipList, 51
- SkipList_getSize
 - SkipList, 52
- SkipList_new
 - SkipList, 52
- SkipList_newFromAllocator
 - SkipList, 52
- SkipList_print
 - SkipList, 53
- SkipList_remove
 - SkipList, 53
- SkipNode, 53
- SParse, 54
 - SParse_copy, 54
 - SParse_delete, 54
 - SParse_new, 54
 - SParse_parse, 55
- SParse_copy
 - SParse, 54
- SParse_delete
 - SParse, 54
- SParse_new
 - SParse, 54
- SParse_parse
 - SParse, 55
- String, 55
 - String_compare, 56
 - String_copy, 56
 - String_getRef, 57
- String_compare
 - String, 56
- String_copy
 - String, 56
- String_getRef
 - String, 57
- stub_data, 57
- Task, 57
 - Task_create, 58
 - Task_executeBody, 58
 - Task_isCompleted, 58
 - Task_isReady, 58
 - Task_isRunning, 59
- Task_create
 - Task, 58
- Task_executeBody
 - Task, 58
- Task_isCompleted
 - Task, 58
- Task_isReady
 - Task, 58
- Task_isRunning
 - Task, 59
- TaskMgr, 59
 - TaskMgr_createWorkerThreads, 60
 - TaskMgr_delete, 60
 - TaskMgr_destroySemaphores, 60
 - TaskMgr_findAvailableTask, 61
 - TaskMgr_getRef, 61
 - TaskMgr_getSize, 61
 - TaskMgr_initSemaphores, 61
 - TaskMgr_signalNotEmpty, 62
 - TaskMgr_signalNotFull, 62
 - TaskMgr_start, 62
 - TaskMgr_stop, 63
 - TaskMgr_waitForThread, 63
 - TaskMgr_waitNotEmpty, 63
 - TaskMgr_waitNotFull, 63
- TaskMgr_createWorkerThreads
 - TaskMgr, 60
- TaskMgr_delete
 - TaskMgr, 60
- TaskMgr_destroySemaphores
 - TaskMgr, 60
- TaskMgr_findAvailableTask
 - TaskMgr, 61
- TaskMgr_getRef
 - TaskMgr, 61
- TaskMgr_getSize
 - TaskMgr, 61
- TaskMgr_initSemaphores
 - TaskMgr, 61
- TaskMgr_new
 - Map, 34
- TaskMgr_signalNotEmpty
 - TaskMgr, 62
- TaskMgr_signalNotFull
 - TaskMgr, 62
- TaskMgr_start
 - TaskMgr, 62
- TaskMgr_stop
 - TaskMgr, 63
- TaskMgr_waitForThread
 - TaskMgr, 63
- TaskMgr_waitNotEmpty
 - TaskMgr, 63
- TaskMgr_waitNotFull
 - TaskMgr, 63
- TestClass, 64
- TestFileMgr, 64
- TestObject, 64
- testOptionMgr, 64
- TestSdbMgr, 65
- TestTimeMgr, 65
- TimeMgr, 65
 - TimeMgr_copy, 66
 - TimeMgr_delete, 66
 - TimeMgr_getRef, 66
 - TimeMgr_getSize, 66
 - TimeMgr_latchTime, 66
- TimeMgr_copy

- TimeMgr, [66](#)
- TimeMgr_delete
 - TimeMgr, [66](#)
- TimeMgr_getRef
 - TimeMgr, [66](#)
- TimeMgr_getSize
 - TimeMgr, [66](#)
- TimeMgr_latchTime
 - TimeMgr, [66](#)
- Timer, [67](#)
 - Timer_copy, [67](#)
 - Timer_new, [67](#)
- Timer_copy
 - Timer, [67](#)
- Timer_new
 - Timer, [67](#)
- TransUnit, [68](#)
 - TransUnit_getName, [69](#)
 - TransUnit_getNextBuffer, [69](#)
 - TransUnit_getSize, [69](#)
 - TransUnit_new, [69](#)
- TransUnit_getName
 - TransUnit, [69](#)
- TransUnit_getNextBuffer
 - TransUnit, [69](#)
- TransUnit_getSize
 - TransUnit, [69](#)
- TransUnit_new
 - TransUnit, [69](#)

- yy_bs_column
 - yy_buffer_state, [70](#)
- yy_bs_lineno
 - yy_buffer_state, [70](#)
- yy_buffer_stack
 - yyguts_t, [71](#)
- yy_buffer_stack_max
 - yyguts_t, [71](#)
- yy_buffer_stack_top
 - yyguts_t, [72](#)
- yy_buffer_state, [70](#)
 - yy_bs_column, [70](#)
 - yy_bs_lineno, [70](#)
- yy_trans_info, [70](#)
- yyalloc, [71](#)
- yyguts_t, [71](#)
 - yy_buffer_stack, [71](#)
 - yy_buffer_stack_max, [71](#)
 - yy_buffer_stack_top, [72](#)
- YYSTYPE, [72](#)