

## SParse

Generated by Doxygen 1.9.5



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Class Documentation</b>	<b>7</b>
3.1 BTree Struct Reference	7
3.2 Class Struct Reference	7
3.3 Declarator Struct Reference	8
3.4 FileDesc Class Reference	8
3.5 FileIo Struct Reference	9
3.6 FileMgr Class Reference	9
3.6.1 Member Function Documentation	9
3.6.1.1 FileMgr_addDirectory()	10
3.6.1.2 FileMgr_addFile()	10
3.6.1.3 FileMgr_copy()	10
3.6.1.4 FileMgr_filterFiles()	10
3.6.1.5 FileMgr_getRef()	11
3.6.1.6 FileMgr_getRootLocation()	11
3.6.1.7 FileMgr_load()	11
3.6.1.8 FileMgr_setRootLocation()	11
3.7 FileReader Class Reference	12
3.7.1 Member Function Documentation	12
3.7.1.1 FileReader_addFile()	12
3.7.1.2 FileReader_copy()	13
3.7.1.3 FileReader_getBuffer()	13
3.7.1.4 FileReader_getName()	13
3.7.1.5 FileReader_new()	13
3.8 Grammar2 Class Reference	14
3.8.1 Member Function Documentation	14
3.8.1.1 Grammar2_copy()	14
3.8.1.2 Grammar2_new()	14
3.9 GrammarContext Struct Reference	15
3.10 IncludeInfo Struct Reference	15
3.11 List Class Reference	15
3.11.1 Member Function Documentation	16
3.11.1.1 List_compare()	16
3.11.1.2 List_copy()	16
3.11.1.3 List_forEach()	16
3.11.1.4 List_getSize()	17
3.11.1.5 List_insertHead()	17
3.11.1.6 List_insertTail()	17

3.11.1.7 List_merge()	18
3.11.1.8 List_new()	18
3.12 Map Class Reference	18
3.12.1 Member Function Documentation	19
3.12.1.1 Map_copy()	19
3.12.1.2 Map_getAll()	19
3.12.1.3 Map_insert()	19
3.13 MapEntry Struct Reference	20
3.14 Node Struct Reference	20
3.15 Object Struct Reference	20
3.15.1 Member Function Documentation	21
3.15.1.1 Object_comp()	21
3.15.1.2 Object_copy()	21
3.15.1.3 Object_getRef()	21
3.15.1.4 Object_new()	21
3.15.1.5 Object_print()	22
3.16 ObjectInfo Struct Reference	22
3.17 ObjectMgr Class Reference	22
3.17.1 Member Function Documentation	23
3.17.1.1 ObjectMgr_allocate()	23
3.17.1.2 ObjectMgr_copy()	23
3.17.1.3 ObjectMgr_deallocate()	23
3.17.1.4 ObjectMgr_getRef()	24
3.17.2 Member Data Documentation	24
3.17.2.1 maxNbObjectAllocated	24
3.18 OptionDefault Struct Reference	24
3.19 OptionMgr Class Reference	25
3.19.1 Member Function Documentation	25
3.19.1.1 OptionMgr_getRef()	25
3.19.1.2 OptionMgr_readFromCmdLine()	25
3.20 PoolCache Struct Reference	26
3.21 SdbMgr Class Reference	26
3.21.1 Member Function Documentation	26
3.21.1.1 SdbMgr_copy()	27
3.21.1.2 SdbMgr_execute()	27
3.21.1.3 SdbMgr_getRef()	27
3.22 SdbRequest Class Reference	27
3.22.1 Member Function Documentation	28
3.22.1.1 SdbRequest_delete()	28
3.22.1.2 SdbRequest_execute()	28
3.22.1.3 SdbRequest_new()	29
3.23 SkipNode Struct Reference	29

3.24 SParse Class Reference . . . . .	29
3.24.1 Member Function Documentation . . . . .	29
3.24.1.1 SParse_delete() . . . . .	29
3.24.1.2 SParse_new() . . . . .	30
3.24.1.3 SParse_parse() . . . . .	30
3.25 String Struct Reference . . . . .	30
3.25.1 Detailed Description . . . . .	31
3.25.2 Member Function Documentation . . . . .	31
3.25.2.1 String_compare() . . . . .	31
3.25.2.2 String_copy() . . . . .	32
3.25.2.3 String_getRef() . . . . .	32
3.26 TestFileMgr Struct Reference . . . . .	32
3.27 TestItem Struct Reference . . . . .	33
3.28 testOptionMgr Struct Reference . . . . .	33
3.29 TestSdbMgr Struct Reference . . . . .	33
3.30 TestTimeMgr Struct Reference . . . . .	33
3.31 TimeMgr Class Reference . . . . .	34
3.31.1 Member Function Documentation . . . . .	34
3.31.1.1 TimeMgr_copy() . . . . .	34
3.31.1.2 TimeMgr_delete() . . . . .	34
3.31.1.3 TimeMgr_getRef() . . . . .	35
3.31.1.4 TimeMgr_latchTime() . . . . .	35
3.32 Timer Class Reference . . . . .	35
3.32.1 Member Function Documentation . . . . .	36
3.32.1.1 Timer_copy() . . . . .	36
3.32.1.2 Timer_new() . . . . .	36
3.33 yy_buffer_state Struct Reference . . . . .	37
3.33.1 Member Data Documentation . . . . .	37
3.33.1.1 yy_bs_column . . . . .	37
3.33.1.2 yy_bs_lineno . . . . .	37
3.34 yy_trans_info Struct Reference . . . . .	37
3.35 yyallocc Union Reference . . . . .	38
3.36 yyguts_t Struct Reference . . . . .	38
3.36.1 Member Data Documentation . . . . .	38
3.36.1.1 yy_buffer_stack . . . . .	38
3.36.1.2 yy_buffer_stack_max . . . . .	39
3.36.1.3 yy_buffer_stack_top . . . . .	39
3.37 YYSTYPE Union Reference . . . . .	39
<b>4 File Documentation . . . . .</b>	<b>41</b>
4.1 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/FileDesc.c File Reference . . . . .	41
4.1.1 Detailed Description . . . . .	41

4.2 FileDesc.h . . . . .	42
4.3 FileDesc.h . . . . .	42
4.4 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/FileMgr/FileMgr.c File Reference . . . . .	42
4.4.1 Detailed Description . . . . .	43
4.5 FileMgr.h . . . . .	43
4.6 FileMgr.h . . . . .	43
4.7 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/OptionMgr/OptionMgr.c File Reference . . . . .	44
4.7.1 Detailed Description . . . . .	44
4.8 OptionMgr.h . . . . .	44
4.9 OptionMgr.h . . . . .	45
4.10 /home/thomas/Projects/SParse-master/SParse/src/main.c File Reference . . . . .	45
4.10.1 Detailed Description . . . . .	45
4.10.2 Function Documentation . . . . .	46
4.10.2.1 main() . . . . .	46
4.10.2.2 print_usage() . . . . .	46
4.10.2.3 sighandler() . . . . .	46
4.10.2.4 start_application() . . . . .	46
4.11 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/SdbMgr/SdbMgr.c File Reference . . . . .	47
4.11.1 Detailed Description . . . . .	48
4.12 SdbMgr.h . . . . .	48
4.13 SdbMgr.h . . . . .	48
4.14 SdbRequest.h . . . . .	48
4.15 SdbRequest.h . . . . .	49
4.16 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/TimeMgr/TimeMgr.c File Reference . . . . .	49
4.16.1 Detailed Description . . . . .	49
4.17 TimeMgr.h . . . . .	50
4.18 TimeMgr.h . . . . .	50
4.19 Timer.h . . . . .	50
4.20 Timer.h . . . . .	50
4.21 Array.h . . . . .	51
4.22 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/BTree.c File Reference . . . . .	51
4.22.1 Detailed Description . . . . .	51
4.23 BTree.h . . . . .	51
4.24 BTree.h . . . . .	52
4.25 CommonTypes.h . . . . .	52
4.26 Node.h . . . . .	53
4.27 Node.h . . . . .	53
4.28 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Debug/Debug.c File Reference . . . . .	53
4.28.1 Detailed Description . . . . .	54
4.29 Debug.h . . . . .	54
4.30 Debug.h . . . . .	54
4.31 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Error/Error.c File Reference . . . . .	54

4.31.1 Detailed Description . . . . .	55
4.31.2 Function Documentation . . . . .	55
4.31.2.1 Error_new() . . . . .	55
4.32 Error.h . . . . .	55
4.33 Error.h . . . . .	55
4.34 FileIo.h . . . . .	56
4.35 FileIo.h . . . . .	56
4.36 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/List.c File Reference . . . . .	57
4.36.1 Detailed Description . . . . .	57
4.37 List.h . . . . .	57
4.38 List.h . . . . .	58
4.39 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/Map.c File Reference . . . . .	58
4.39.1 Detailed Description . . . . .	59
4.40 Map.h . . . . .	59
4.41 Map.h . . . . .	59
4.42 MapEntry.h . . . . .	59
4.43 MapEntry.h . . . . .	60
4.44 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Memory/Memory.c File Reference . . . . .	60
4.44.1 Detailed Description . . . . .	60
4.45 Memory.h . . . . .	61
4.46 Memory.h . . . . .	61
4.47 Class.h . . . . .	61
4.48 Class.h . . . . .	61
4.49 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/Object.c File Reference . . . . .	62
4.49.1 Detailed Description . . . . .	62
4.50 Object.h . . . . .	62
4.51 Object.h . . . . .	63
4.52 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectMgr/ObjectMgr.c File Reference . . . . .	63
4.52.1 Detailed Description . . . . .	64
4.53 ObjectMgr.h . . . . .	64
4.54 ObjectMgr.h . . . . .	64
4.55 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Pool/Pool.c File Reference . . . . .	64
4.55.1 Detailed Description . . . . .	66
4.55.2 Function Documentation . . . . .	66
4.55.2.1 Pool_alloc() . . . . .	66
4.55.2.2 Pool_allocInFile() . . . . .	66
4.55.2.3 Pool_dealloc() . . . . .	67
4.55.2.4 Pool_deallocInFile() . . . . .	67
4.55.2.5 Pool_deallocInMemory() . . . . .	67
4.55.2.6 Pool_free() . . . . .	68
4.55.2.7 Pool_new() . . . . .	68
4.55.2.8 Pool_newFromFile() . . . . .	68

4.55.2.9 Pool_read()	69
4.55.2.10 Pool_readInFile()	69
4.55.2.11 Pool_readInMemory()	69
4.55.2.12 Pool_report()	70
4.55.2.13 Pool_reportInFile()	70
4.55.2.14 Pool_reportInMemory()	71
4.55.2.15 Pool_reportNbNodes()	71
4.55.2.16 Pool_reportSizeInBytes()	71
4.55.2.17 Pool_write()	72
4.55.2.18 Pool_writeInFile()	73
4.55.2.19 Pool_writeInMemory()	73
4.56 Pool.h	73
4.57 Pool.h	74
4.58 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/SkipList.c File Reference	75
4.58.1 Detailed Description	76
4.58.2 Function Documentation	76
4.58.2.1 SkipList_add()	76
4.58.2.2 SkipList_compare()	76
4.58.2.3 SkipList_copy()	77
4.58.2.4 SkipList_delete()	77
4.58.2.5 SkipList_new()	77
4.58.2.6 SkipList_print()	78
4.58.2.7 SkipList_remove()	78
4.59 SkipList.h	78
4.60 SkipList.h	79
4.61 String2.h	79
4.62 String2.h	79
4.63 Times.h	80
4.64 Times.h	80
4.65 Types.h	80
4.66 Types.h	80
4.67 Declarator.h	81
4.68 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/FileReader/FileReader.c File Reference	81
4.68.1 Detailed Description	81
4.69 FileReader.h	82
4.70 FileReader.h	82
4.71 Grammar2.h	82
4.72 Grammar2.h	82
4.73 Grammar2.parse.h	83
4.74 /home/thomas/Projects/SParse-master/SParse/src/ParseLib/SParse/SParse.c File Reference	84
4.74.1 Detailed Description	84
4.75 SParse.h	85



4.76 SParse.h . . . . .	85
<b>Index</b>	<b>87</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

BTree	7
Class	7
Declarator	8
FileDesc	8
Filelo	9
FileMgr	9
FileReader	12
Grammar2	14
GrammarContext	15
IncludeInfo	15
List	15
Map	18
MapEntry	20
Node	20
Object	20
ObjectInfo	22
ObjectMgr	22
OptionDefault	24
OptionMgr	25
PoolCache	26
SdbMgr	26
SdbRequest	27
SkipNode	29
SParse	29
String	30
TestFileMgr	32
TestItem	33
testOptionMgr	33
TestSdbMgr	33
TestTimeMgr	33
TimeMgr	34
Timer	35
yy_buffer_state	37
yy_trans_info	37
yyalloc	38
yyguts_t	38
YYSTYPE	39



## 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/ <a href="#">main.c</a>	
Contains the main() function	45
/home/thomas/Projects/SParse-master/SParse/src/AppLib/FileMgr/ <a href="#">FileDesc.c</a>	
The <a href="#">FileDesc</a> class describe a File in the FileMgr	41
/home/thomas/Projects/SParse-master/SParse/src/AppLib/FileMgr/ <a href="#">FileDesc.h</a>	42
/home/thomas/Projects/SParse-master/SParse/src/AppLib/FileMgr/ <a href="#">FileMgr.c</a>	
The <a href="#">FileMgr</a> class manages a list of files contained in a group of locations	42
/home/thomas/Projects/SParse-master/SParse/src/AppLib/FileMgr/ <a href="#">FileMgr.h</a>	43
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/ <a href="#">FileDesc.h</a>	42
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/ <a href="#">FileMgr.h</a>	43
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/ <a href="#">OptionMgr.h</a>	44
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/ <a href="#">SdbMgr.h</a>	48
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/ <a href="#">SdbRequest.h</a>	48
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/ <a href="#">TimeMgr.h</a>	50
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/ <a href="#">Timer.h</a>	50
/home/thomas/Projects/SParse-master/SParse/src/AppLib/OptionMgr/ <a href="#">OptionMgr.c</a>	
The <a href="#">OptionMgr</a> class manages the application configuration	44
/home/thomas/Projects/SParse-master/SParse/src/AppLib/OptionMgr/ <a href="#">OptionMgr.h</a>	45
/home/thomas/Projects/SParse-master/SParse/src/AppLib/SdbMgr/ <a href="#">SdbMgr.c</a>	
TBD	47
/home/thomas/Projects/SParse-master/SParse/src/AppLib/SdbMgr/ <a href="#">SdbMgr.h</a>	48
/home/thomas/Projects/SParse-master/SParse/src/AppLib/SdbMgr/ <a href="#">SdbRequest.h</a>	49
/home/thomas/Projects/SParse-master/SParse/src/AppLib/TimeMgr/ <a href="#">TimeMgr.c</a>	
This file contains the implementation for the class <a href="#">TimeMgr</a>	49
/home/thomas/Projects/SParse-master/SParse/src/AppLib/TimeMgr/ <a href="#">TimeMgr.h</a>	50
/home/thomas/Projects/SParse-master/SParse/src/AppLib/TimeMgr/ <a href="#">Timer.h</a>	50
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Array/ <a href="#">Array.h</a>	51
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/ <a href="#">BTree.c</a>	
This file contains the implementation of the class <a href="#">BTree</a>	51
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/ <a href="#">BTree.h</a>	51
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/ <a href="#">CommonTypes.h</a>	52
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/ <a href="#">Node.h</a>	53
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Debug/ <a href="#">Debug.c</a>	
This file contains debugging functions	53
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Debug/ <a href="#">Debug.h</a>	54

/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Error/ <a href="#">Error.c</a>	
Reports errors	54
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Error/ <a href="#">Error.h</a>	55
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/FileIo/ <a href="#">FileIo.h</a>	56
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">BTree.h</a>	52
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">Class.h</a>	61
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">Debug.h</a>	54
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">Error.h</a>	55
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">FileIo.h</a>	56
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">List.h</a>	57
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">Map.h</a>	59
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">MapEntry.h</a>	59
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">Memory.h</a>	61
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">Node.h</a>	53
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">Object.h</a>	62
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">ObjectMgr.h</a>	64
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">Pool.h</a>	73
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">SkipList.h</a>	78
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">String2.h</a>	79
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">Times.h</a>	80
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/ <a href="#">Types.h</a>	80
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/ <a href="#">List.c</a>	
This file contains the implementation of the class <a href="#">List</a>	57
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/ <a href="#">List.h</a>	58
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/ <a href="#">Map.c</a>	
A <a href="#">Map</a> class. This class provides a container indexed by a string	58
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/ <a href="#">Map.h</a>	59
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/ <a href="#">MapEntry.h</a>	60
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Memory/ <a href="#">Memory.c</a>	
This file provides the implementation of the memory functions	60
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Memory/ <a href="#">Memory.h</a>	61
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/ <a href="#">Class.h</a>	61
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/ <a href="#">Object.c</a>	
This file contains the implementation for the class <a href="#">Object</a>	62
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/ <a href="#">Object.h</a>	63
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectMgr/ <a href="#">ObjectMgr.c</a>	
An object management class	63
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectMgr/ <a href="#">ObjectMgr.h</a>	64
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Pool/ <a href="#">Pool.c</a>	
This file contains the implementation of the class <a href="#">Pool</a>	64
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Pool/ <a href="#">Pool.h</a>	74
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/ <a href="#">SkipList.c</a>	
This file contains the implementation of the class <a href="#">SkipList</a> . The class <a href="#">List</a> implement the <a href="#">SkipList</a> operations	75
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/ <a href="#">SkipList.h</a>	79
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/String/ <a href="#">String2.h</a>	79
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Times/ <a href="#">Times.h</a>	80
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Types/ <a href="#">Types.h</a>	80
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/C89Grammar/ <a href="#">Declarator.h</a>	81
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/FileReader/ <a href="#">FileReader.c</a>	
This file contains the implementation for the class <a href="#">FileReader</a>	81
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/FileReader/ <a href="#">FileReader.h</a>	82
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar2/ <a href="#">Grammar2.h</a>	82
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/Grammar2/ <a href="#">Grammar2.parse.h</a>	83
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/ <a href="#">FileReader.h</a>	82
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/ <a href="#">Grammar2.h</a>	82
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/include/ <a href="#">SParse.h</a>	85

---

/home/thomas/Projects/SParse-master/SParse/src/ParseLib/SParse/SParse.c	
This file contains the implementation for the class <a href="#">SParse</a> . . . . .	84
/home/thomas/Projects/SParse-master/SParse/src/ParseLib/SParse/SParse.h . . . . .	85





## Chapter 3

# Class Documentation

### 3.1 BTree Struct Reference

#### Public Attributes

- void \* **pool**
- [Node](#) \* **root**
- unsigned int **depth**
- unsigned short int **nbObjects**
- unsigned short int **nbNodes**
- unsigned int **order**
- unsigned int **nodeSize**

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

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

### 3.2 Class Struct Reference

#### Public Attributes

- Constructor **f\_new**
- Destructor **f\_delete**
- Copy\_Operator **f\_copy**
- Comp\_Operator **f\_comp**
- Printer **f\_print**

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

### 3.3 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

### 3.4 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](#)

## 3.5 Filelo Struct Reference

### Public Attributes

- FILE \* **f**
- int **status**

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

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

## 3.6 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\_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 [String](#) \* **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 [List](#) \* **FileMgr\_filterFiles** ([FileMgr](#) \*this, const char \*pattern)  
*TBD.*

### Public Attributes

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

### 3.6.1 Member Function Documentation

### 3.6.1.1 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.

### 3.6.1.2 FileMgr\_addFile()

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

Add a files to the list of managed files.

#### Returns

Status.

### 3.6.1.3 FileMgr\_copy()

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

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

#### Returns

New instance

### 3.6.1.4 FileMgr\_filterFiles()

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

TBD.

#### Returns

TBD

### 3.6.1.5 FileMgr\_getRef()

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

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

#### Returns

Reference to the singleton.

### 3.6.1.6 FileMgr\_getRootLocation()

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

TBD.

#### Returns

Status.

### 3.6.1.7 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.

### 3.6.1.8 FileMgr\_setRootLocation()

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

Set the root location.

#### Returns

Status.

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

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

## 3.7 FileReader Class Reference

### Public Member Functions

- PUBLIC [FileReader](#) \* [FileReader\\_new](#) ([FileDesc](#) \*fileDesc)  
*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 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
- [String](#) \* currentBuffer
- [List](#) \* preferredDirs

### 3.7.1 Member Function Documentation

#### 3.7.1.1 [FileReader\\_addFile\(\)](#)

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

Add a new file buffer for filename.

#### Returns

File buffer

### 3.7.1.2 FileReader\_copy()

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

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

#### Returns

New instance

### 3.7.1.3 FileReader\_getBuffer()

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

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

#### Returns

Buffer of characters

### 3.7.1.4 FileReader\_getName()

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

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

#### Returns

File name

### 3.7.1.5 FileReader\_new()

```
PUBLIC FileReader * FileReader_new (
    FileDesc * fileDesc )
```

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](#)

## 3.8 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**
- char \* **buffer**
- int **node\_text\_position**
- [GrammarContext](#) \* **current**
- [List](#) \* **contexts**

### 3.8.1 Member Function Documentation

#### 3.8.1.1 Grammar2\_copy()

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

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

#### Returns

Copied instance.

#### 3.8.1.2 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



## 3.9 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

## 3.10 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](#)

## 3.11 List Class Reference

### Public Member Functions

- PUBLIC [List](#) \* [List\\_new](#) ()  
*Create a new instance of the class [List](#).*
- 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)  
*Insert an item at the head of a list instance.*
- PUBLIC void [List\\_insertTail](#) ([List](#) \*this, void \*item)  
*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\\_getSize](#) ([List](#) \*this)  
*Get the number of item in [List](#) instance.*
- 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](#)*

## Public Attributes

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

### 3.11.1 Member Function Documentation

#### 3.11.1.1 `List_compare()`

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

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

##### Returns

0 if different, 1 if equal.

#### 3.11.1.2 `List_copy()`

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

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

##### Returns

Copy of the given instance.

#### 3.11.1.3 `List_forEach()`

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

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

## Parameters

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

**3.11.1.4 List\_getSize()**

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

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

## Returns

Number of items.

**3.11.1.5 List\_insertHead()**

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

Insert an item at the head of a list instance.

## Parameters

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

**3.11.1.6 List\_insertTail()**

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

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

## Parameters

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

### 3.11.1.7 List\_merge()

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

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

#### Parameters

in	//	Reference to list to merge.
----	----	-----------------------------

### 3.11.1.8 List\_new()

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

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

#### Returns

New instance.

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

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

## 3.12 Map Class Reference

### Public Member Functions

- PUBLIC [Map](#) \* **Map\_new** ()  
*Create a new instance of the class [Map](#).*
- PUBLIC void **Map\_delete** ([Map](#) \*this)  
*TBD.*
- PUBLIC [Map](#) \* **Map\_copy** ([Map](#) \*this)  
*Copy an instance of the class [Map](#).*
- PUBLIC unsigned int **Map\_insert** ([Map](#) \*this, [String](#) \*s, void \*p)  
*TBD.*
- PUBLIC unsigned int **Map\_find** ([Map](#) \*this, [String](#) \*s, void \*\*p)  
*TBD.*
- PUBLIC [List](#) \* **Map\_getAll** ([Map](#) \*this)  
*Get all the entries in an instance of a [Map](#).*

### Public Attributes

- [Object](#) **object**
- [List](#) \* **htable** [HTABLE\_SIZE]

## 3.12.1 Member Function Documentation

### 3.12.1.1 Map\_copy()

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

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

#### Returns

Copy of instance.

### 3.12.1.2 Map\_getAll()

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

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

#### Returns

[List](#) of map

### 3.12.1.3 Map\_insert()

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

TBD.

#### Returns

TBD

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

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

### 3.13 MapEntry Struct Reference

#### Public Attributes

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

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/MapEntry.c

### 3.14 Node Struct Reference

#### Public Attributes

- unsigned short int **nbKeyUsed**
- unsigned short int **isLeaf**
- unsigned int **keys** [ORDER \*2 - 1]
- [Object](#) **leaves** [ORDER \*2]
- [Node](#) \* **children** [ORDER \*2]

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

### 3.15 Object Struct Reference

#### Public Member Functions

- PUBLIC [Object](#) \* [Object\\_new](#) (unsigned int size, [Class](#) \*class)  
*Create an instance of the class [Object](#).*
- PUBLIC void [Object\\_delete](#) ([Object](#) \*this)  
*Delete 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 Attributes

- unsigned int **id**
- [Class](#) \* **class**
- void(\* **delete** )(Object \*this)
- [Object](#) \*(\* **copy** )(Object \*this)
- unsigned int **refCount**
- unsigned int **size**

## 3.15.1 Member Function Documentation

### 3.15.1.1 Object\_comp()

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

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

#### Returns

1 if equal, 0 else.

### 3.15.1.2 Object\_copy()

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

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

#### Returns

New instance

### 3.15.1.3 Object\_getRef()

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

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

#### Returns

Reference to instance

### 3.15.1.4 Object\_new()

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

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

## Parameters

in	<a href="#">Class</a>	to instanciate
----	-----------------------	----------------

**3.15.1.5 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](#)

**3.16 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](#)

**3.17 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 void **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 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**

## 3.17.1 Member Function Documentation

### 3.17.1.1 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](#).

### 3.17.1.2 ObjectMgr\_copy()

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

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

#### Returns

New instance

### 3.17.1.3 ObjectMgr\_deallocate()

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

De Allocate a given object.

## Parameters

in	<i>object</i>	Reference to instance of <a href="#">Object</a> .
----	---------------	---

### 3.17.1.4 ObjectMgr\_getRef()

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

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

## Returns

Reference to the singleton.

## 3.17.2 Member Data Documentation

### 3.17.2.1 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](#)

## 3.18 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](#)

## 3.19 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 [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

### 3.19.1 Member Function Documentation

#### 3.19.1.1 OptionMgr\_getRef()

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

*TBD.*

*TBD*

#### 3.19.1.2 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>	<a href="#">List</a> of 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](#)

## 3.20 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](#)

## 3.21 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**

### 3.21.1 Member Function Documentation

### 3.21.1.1 SdbMgr\_copy()

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

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

#### Returns

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

### 3.21.1.2 SdbMgr\_execute()

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

Execute a Sdb request.

#### Returns

status

### 3.21.1.3 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](#)

## 3.22 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**

## 3.22.1 Member Function Documentation

### 3.22.1.1 SdbRequest\_delete()

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

Create a new [SdbRequest](#) instance

@parameter SQL statement template.

#### Returns

Instance of an [SdbRequest](#)

### 3.22.1.2 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](#)

### 3.22.1.3 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

## 3.23 SkipNode Struct Reference

### Public Attributes

- [Object](#) \* **objectInfo**
- unsigned int **key**
- void \* **object**
- unsigned int **level**
- unsigned int **forward** [SKIPLIST\_MAX\_LEVEL]

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

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

## 3.24 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 unsigned int [SParse\\_parse](#) ([SParse](#) \*this, const char \*extension)  
*Parse all files with a given extension.*

### Public Attributes

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

### 3.24.1 Member Function Documentation

#### 3.24.1.1 SParse\_delete()

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

Delete a [SParse](#) object.

## Parameters

<i>Object</i>	to delete.
---------------	------------

### 3.24.1.2 SParse\_new()

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

Create a new [SParse](#) object.

## Returns

New [SParse](#) object.

### 3.24.1.3 SParse\_parse()

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

Parse all files with a given extension.

## Parameters

in	<i>extension</i>	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](#)

## 3.25 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\\_toInt](#) ([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)
- *TBD.*
- PUBLIC unsigned int [String\\_isContained](#) ([String](#) \*this, [String](#) \*s2)
- *TBD.*

## Public Attributes

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

### 3.25.1 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](#)

### 3.25.2 Member Function Documentation

#### 3.25.2.1 [String\\_compare\(\)](#)

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

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

#### Parameters

in	<i>compared</i>	<a href="#">String</a> to compare
----	-----------------	-----------------------------------

**Returns**

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

**3.25.2.2 String\_copy()**

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

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

**Returns**

Copy of instance.

**3.25.2.3 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

## 3.26 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

## 3.27 TestItem Struct Reference

### Public Attributes

- [Object](#) object
- int x
- int y

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

- /home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/tests/UT\_List.c

## 3.28 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/main.c

## 3.29 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/main.c

## 3.30 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/main.c

## 3.31 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 void [TimeMgr\\_latchTime](#) ([TimeMgr](#) \*this, [String](#) \*s)  
*Latch the current time under the specified name.*

### Public Attributes

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

### 3.31.1 Member Function Documentation

#### 3.31.1.1 TimeMgr\_copy()

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

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

#### Returns

New instance

#### 3.31.1.2 TimeMgr\_delete()

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

Delete a [TimeMgr](#) object.

#### Parameters

<a href="#">Object</a>	to delete.
------------------------	------------

### 3.31.1.3 TimeMgr\_getRef()

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

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

#### Returns

Reference to the singleton.

### 3.31.1.4 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](#)

## 3.32 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\\_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**

### 3.32.1 Member Function Documentation

#### 3.32.1.1 `Timer_copy()`

```
PUBLIC Timer * Timer_copy (  
    Timer * this )
```

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

##### Returns

Copied instance.

#### 3.32.1.2 `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`

## 3.33 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**

### 3.33.1 Member Data Documentation

#### 3.33.1.1 yy\_bs\_column

```
int yy_buffer_state::yy_bs_column
```

The column count.

#### 3.33.1.2 yy\_bs\_lineno

```
int yy_buffer_state::yy_bs_lineno
```

The line count.

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

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

## 3.34 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 file:

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

## 3.35 yyalloc Union Reference

### Public Attributes

- `yy_state_t yyss_alloc`
- `YYSTYPE yyvs_alloc`

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

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

## 3.36 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`

### 3.36.1 Member Data Documentation

#### 3.36.1.1 yy\_buffer\_stack

`YY_BUFFER_STATE* yyguts_t::yy_buffer_stack`

Stack as an array.



### 3.36.1.2 yy\_buffer\_stack\_max

`size_t yyguts_t::yy_buffer_stack_max`

capacity of stack.

### 3.36.1.3 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 file:

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

## 3.37 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/AppLib/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"
```

#### Classes

- class `FileDesc`

#### Functions

- PRIVATE `String * FileDesc_getBasename (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

```

1  /* FileDesc.h */
2
3  #ifndef _FILEDESC_H_
4  #define _FILEDESC_H_
5
6  #include "Types.h"
7  #include "String2.h"
8
9  typedef struct FileDesc FileDesc;
10
11 PUBLIC FileDesc * FileDesc_new();
12 PUBLIC void FileDesc_delete(FileDesc * this);
13 PUBLIC FileDesc * FileDesc_copy(FileDesc * this);
14 PUBLIC void FileDesc_setFullName(FileDesc * this, String * fullName);
15 PUBLIC String * FileDesc_getFullName(FileDesc * this);
16 PUBLIC void FileDesc_setName(FileDesc * this, String * name);
17 PUBLIC String * FileDesc_getName(FileDesc * this);
18 PUBLIC String * FileDesc_load(FileDesc * this);
19
20 #endif /* _FILEDESC_H_ */

```

## 4.3 FileDesc.h

```

1  /* FileDesc.h */
2
3  #ifndef _FILEDESC_H_
4  #define _FILEDESC_H_
5
6  #include "Types.h"
7  #include "String2.h"
8
9  typedef struct FileDesc FileDesc;
10
11 PUBLIC FileDesc * FileDesc_new();
12 PUBLIC void FileDesc_delete(FileDesc * this);
13 PUBLIC FileDesc * FileDesc_copy(FileDesc * this);
14 PUBLIC void FileDesc_setFullName(FileDesc * this, String * fullName);
15 PUBLIC String * FileDesc_getFullName(FileDesc * this);
16 PUBLIC void FileDesc_setName(FileDesc * this, String * name);
17 PUBLIC String * FileDesc_getName(FileDesc * this);
18 PUBLIC String * FileDesc_load(FileDesc * this);
19
20 #endif /* _FILEDESC_H_ */

```

## 4.4 /home/thomas/Projects/SParse-master/SParse/src/AppliLib/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 "FileIo.h"

```

### Classes

- class [FileMgr](#)

## Macros

- #define **FILEMGR\_MAX\_PATH** (1024)

## Functions

- PRIVATE void **FileMgr\_listFiles** (**FileMgr** \*this, **String** \*directory)
- PRIVATE **FileDesc** \* **FileMgr\_isManaged** (**FileMgr** \*this, **String** \*fullName)
- PRIVATE unsigned int **FileMgr\_existFS** (**FileMgr** \*this, **String** \*fullName)
- PUBLIC **FileDesc** \* **FileMgr\_searchFile** (**FileMgr** \*this, **String** \*name, **List** \*preferredDir)

### 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

```
1 /* FileMgr.h */
2
3 #ifndef _FILEMGR_H_
4 #define _FILEMGR_H_
5
6 #include "Types.h"
7 #include "List.h"
8 #include "String2.h"
9 #include "FileDesc.h"
10
11 typedef struct FileMgr FileMgr;
12
13 PUBLIC void FileMgr_delete(FileMgr * this);
14 PUBLIC FileMgr * FileMgr_copy(FileMgr * this);
15 PUBLIC String* FileMgr_load(FileMgr* this, const char * fileName);
16 PUBLIC void FileMgr_close(FileMgr* this, String* fileName);
17 PUBLIC unsigned int FileMgr_setRootLocation(FileMgr* this, const char * location);
18 PUBLIC char * FileMgr_getRootLocation(FileMgr* this);
19 PUBLIC FileMgr* FileMgr_getRef();
20 PUBLIC unsigned int FileMgr_addDirectory(FileMgr * this, const char * directoryName);
21 PUBLIC String * FileMgr_addFile(FileMgr * this, const char * fileName);
22 PUBLIC List * FileMgr_filterFiles(FileMgr * this, const char * pattern);
23 PUBLIC FileDesc * FileMgr_searchFile(FileMgr * this, String * name, List * preferredDir);
24 #endif /* _FILEMGR_H_ */
```

## 4.6 FileMgr.h

```
1 /* FileMgr.h */
2
3 #ifndef _FILEMGR_H_
4 #define _FILEMGR_H_
5
6 #include "Types.h"
7 #include "List.h"
8 #include "String2.h"
9 #include "FileDesc.h"
10
11 typedef struct FileMgr FileMgr;
12
13 PUBLIC void FileMgr_delete(FileMgr * this);
14 PUBLIC FileMgr * FileMgr_copy(FileMgr * this);
15 PUBLIC String* FileMgr_load(FileMgr* this, const char * fileName);
16 PUBLIC void FileMgr_close(FileMgr* this, String* fileName);
17 PUBLIC unsigned int FileMgr_setRootLocation(FileMgr* this, const char * location);
18 PUBLIC char * FileMgr_getRootLocation(FileMgr* this);
19 PUBLIC FileMgr* FileMgr_getRef();
20 PUBLIC unsigned int FileMgr_addDirectory(FileMgr * this, const char * directoryName);
21 PUBLIC String * FileMgr_addFile(FileMgr * this, const char * fileName);
22 PUBLIC List * FileMgr_filterFiles(FileMgr * this, const char * pattern);
23 PUBLIC FileDesc * FileMgr_searchFile(FileMgr * this, String * name, List * preferredDir);
24 #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"
```

### 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

```
1 /* OptionMgr.h */
2
3 #ifndef _OPTIONMGR_H_
4 #define _OPTIONMGR_H_
5
6 #include "Types.h"
7 #include "String2.h"
8
9 typedef struct OptionMgr OptionMgr;
10
11 PUBLIC void OptionMgr_delete(OptionMgr * this);
12 PUBLIC OptionMgr * OptionMgr_copy(OptionMgr * this);
13 PUBLIC PUBLIC OptionMgr* OptionMgr_getRef();
14 PUBLIC String * OptionMgr_getOption(OptionMgr * this, const char * name);
15 PUBLIC void OptionMgr_setOption(OptionMgr * this, const char * optionName, String * value);
16 PUBLIC unsigned int OptionMgr_readFromFile(OptionMgr * this);
17 PUBLIC unsigned int OptionMgr_readFromCmdLine(OptionMgr * this, const int argc, const char ** argv);
18 PUBLIC unsigned int OptionMgr_isOptionEnabled(OptionMgr* this, const char * optionName);
19
20 #endif /* _OPTIONMGR_H_ */
```

## 4.9 OptionMgr.h

```

1  /* OptionMgr.h */
2
3  #ifndef _OPTIONMGR_H_
4  #define _OPTIONMGR_H_
5
6  #include "Types.h"
7  #include "String2.h"
8
9  typedef struct OptionMgr OptionMgr;
10
11 PUBLIC void OptionMgr_delete(OptionMgr * this);
12 PUBLIC OptionMgr * OptionMgr_copy(OptionMgr * this);
13 PUBLIC PUBLIC OptionMgr* OptionMgr_getRef();
14 PUBLIC String * OptionMgr_getOption(OptionMgr * this, const char * name);
15 PUBLIC void OptionMgr_setOption(OptionMgr * this, const char * optionName, String * value);
16 PUBLIC unsigned int OptionMgr_readFromFile(OptionMgr * this);
17 PUBLIC unsigned int OptionMgr_readFromCmdLine(OptionMgr * this, const int argc, const char ** argv);
18 PUBLIC unsigned int OptionMgr_isOptionEnabled(OptionMgr* this, const char * optionName);
19
20 #endif /* _OPTIONMGR_H_ */

```

## 4.10 /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 [sighandler](#) (int signum, siginfo\_t \*info, void \*ptr)  
*Display and exit when signal is received.*
- PRIVATE void [start\\_application](#) (String \*inputDir, String \*dbName)  
*Starts the application.*
- PRIVATE void [print\\_usage](#) ()  
*Prints the application help.*
- PUBLIC int [main](#) (const int argc, const char \*\*argv)  
*Initial entry point for the application.*

### 4.10.1 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.

## 4.10.2 Function Documentation

### 4.10.2.1 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.

### 4.10.2.2 print\_usage()

```
PRIVATE void print_usage ( )
```

Prints the application help.

Prints the usage information for the application.

### 4.10.2.3 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.

### 4.10.2.4 start\_application()

```
PRIVATE void start_application (
```



```
String * inputDir,  
String * dbName )
```

Starts the application.

#### Parameters

<i>inputDir</i>	directory containing the files to parse
<i>dbName</i>	DB file to output

#### Returns

This functions starts the main application.

- 1) Creates a [FileMgr](#) object for input directory
- 2) Creates a [SParse](#) object
- 3) Measure the execution time

## 4.11 /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)

### Variables

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

### 4.11.1 Detailed Description

TBD.

TBD

## 4.12 SdbMgr.h

```
1 /* SdbMgr.h */
2
3 #ifndef _SDBMGR_H_
4 #define _SDBMGR_H_
5
6 #include "Types.h"
7 #include "String2.h"
8 #include "List.h"
9
10 typedef struct SdbMgr SdbMgr;
11
12 PUBLIC SdbMgr * SdbMgr_new(String * name);
13 PUBLIC void SdbMgr_delete(SdbMgr* this);
14 PUBLIC SdbMgr * SdbMgr_copy(SdbMgr* this);
15 PUBLIC SdbMgr * SdbMgr_getRef();
16 PUBLIC unsigned int SdbMgr_execute(SdbMgr* this, const char* statement, List * result);
17
18 #endif /* _SDBMGR_H_ */
```

## 4.13 SdbMgr.h

```
1 /* SdbMgr.h */
2
3 #ifndef _SDBMGR_H_
4 #define _SDBMGR_H_
5
6 #include "Types.h"
7 #include "String2.h"
8 #include "List.h"
9
10 typedef struct SdbMgr SdbMgr;
11
12 PUBLIC SdbMgr * SdbMgr_new(String * name);
13 PUBLIC void SdbMgr_delete(SdbMgr* this);
14 PUBLIC SdbMgr * SdbMgr_copy(SdbMgr* this);
15 PUBLIC SdbMgr * SdbMgr_getRef();
16 PUBLIC unsigned int SdbMgr_execute(SdbMgr* this, const char* statement, List * result);
17
18 #endif /* _SDBMGR_H_ */
```

## 4.14 SdbRequest.h

```
1 /* SdbRequest.h */
2 #ifndef _SDBREQUEST_H_
3 #define _SDBREQUEST_H_
4
5 #include "Types.h"
6 #include "List.h"
7
8 typedef struct SdbRequest SdbRequest;
9
10 PUBLIC SdbRequest * SdbRequest_new(const char * fmt);
11 PUBLIC void SdbRequest_delete(SdbRequest * this);
12 PUBLIC SdbRequest * SdbRequest_copy(SdbRequest * this);
13 PUBLIC void SdbRequest_execute(SdbRequest * this, ...);
14 PUBLIC unsigned int SdbRequest_getNbResult(SdbRequest * this);
15 PUBLIC List * SdbRequest_getResults(SdbRequest * this);
16
17 #endif /* _SDBREQUEST_H_ */
```

## 4.15 SdbRequest.h

```

1  /* SdbRequest.h */
2  #ifndef _SDBREQUEST_H_
3  #define _SDBREQUEST_H_
4
5  #include "Types.h"
6  #include "List.h"
7
8  typedef struct SdbRequest SdbRequest;
9
10 PUBLIC SdbRequest * SdbRequest_new(const char * fmt);
11 PUBLIC void SdbRequest_delete(SdbRequest * this);
12 PUBLIC SdbRequest * SdbRequest_copy(SdbRequest * this);
13 PUBLIC void SdbRequest_execute(SdbRequest * this, ...);
14 PUBLIC unsigned int SdbRequest_getNbResult(SdbRequest * this);
15 PUBLIC List * SdbRequest_getResults(SdbRequest * this);
16
17 #endif /* _SDBREQUEST_H_ */

```

## 4.16 /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

### 4.16.1 Detailed Description

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

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

## 4.17 TimeMgr.h

```
1  /* TimeMgr.h */
2
3  #ifndef _TIMEMGR_H_
4  #define _TIMEMGR_H_
5
6  #include "Types.h"
7  #include "String2.h"
8
9  typedef struct TimeMgr TimeMgr;
10
11 PUBLIC void TimeMgr_delete(TimeMgr * this);
12 PUBLIC TimeMgr * TimeMgr_copy(TimeMgr * this);
13 PUBLIC TimeMgr * TimeMgr_getRef();
14 PUBLIC void TimeMgr_latchTime(TimeMgr * this, String * s);
15 PUBLIC void TimeMgr_report(TimeMgr * this);
16
17 #endif /* _TIMEMGR_H_ */
```

## 4.18 TimeMgr.h

```
1  /* TimeMgr.h */
2
3  #ifndef _TIMEMGR_H_
4  #define _TIMEMGR_H_
5
6  #include "Types.h"
7  #include "String2.h"
8
9  typedef struct TimeMgr TimeMgr;
10
11 PUBLIC void TimeMgr_delete(TimeMgr * this);
12 PUBLIC TimeMgr * TimeMgr_copy(TimeMgr * this);
13 PUBLIC TimeMgr * TimeMgr_getRef();
14 PUBLIC void TimeMgr_latchTime(TimeMgr * this, String * s);
15 PUBLIC void TimeMgr_report(TimeMgr * this);
16
17 #endif /* _TIMEMGR_H_ */
```

## 4.19 Timer.h

```
1  /* Timer.h */
2
3  #include "Types.h"
4
5  typedef struct Timer Timer;
6
7 PUBLIC Timer * Timer_new();
8 PUBLIC void Timer_delete(Timer * this);
9 PUBLIC Timer * Timer_copy(Timer * this);
10 PUBLIC unsigned int Timer_isEqual(Timer * this, Timer * compared);
11 PUBLIC void Timer_print(Timer * this);
12 PUBLIC void Timer_latchTime(Timer * this);
```

## 4.20 Timer.h

```
1  /* Timer.h */
2
3  #include "Types.h"
4
5  typedef struct Timer Timer;
6
7 PUBLIC Timer * Timer_new();
8 PUBLIC void Timer_delete(Timer * this);
9 PUBLIC Timer * Timer_copy(Timer * this);
10 PUBLIC unsigned int Timer_isEqual(Timer * this, Timer * compared);
11 PUBLIC void Timer_print(Timer * this);
12 PUBLIC void Timer_latchTime(Timer * this);
```

## 4.21 Array.h

```
1 #ifndef _ARRAY_H_
2 #define _ARRAY_H_
3
4 #endif /* _ARRAY_H_ */
```

## 4.22 /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"
```

### Functions

- [BTree](#) \* **BTree\_new** (unsigned int order)
- void **BTree\_add** ([BTree](#) \*tree, Key key, [Object](#) object)
- [Object](#) **BTree\_get** ([BTree](#) \*tree, Key key)
- [Object](#) **BTree\_remove** ([BTree](#) \*tree, Key key)
- void **BTree\_print** ([BTree](#) \*tree)
- PUBLIC [BTree](#) \* **BTree\_newFromFile** (char \*fileName)
- PUBLIC void **BTree\_free** ([BTree](#) \*tree)

### 4.22.1 Detailed Description

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

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

- init
- add
- remove

## 4.23 BTree.h

```
1 #ifndef _BTREE_
2 #define _BTREE_
3 /*****
4  * BTree.h
5  *
6  *****/
7 #include "Types.h"
8 #include "Node.h"
9
10 #define TRUE (1)
11 #define FALSE (0)
12
13 typedef struct BTree
14 {
15     void* pool;
16     Node* root;
17     unsigned int depth;
```

```

18     unsigned short int nbObjects;
19     unsigned short int nbNodes;
20     unsigned int order;
21     unsigned int nodeSize;
22 } BTree;
23
24 PUBLIC BTree * BTree_new();
25 PUBLIC BTree * BTree_newFromFile(char* fileName);
26 PUBLIC void BTree_free(BTree * tree);
27 PUBLIC void BTree_add(BTree * tree, Key key, Object object);
28 PUBLIC Object BTree_get(BTree * tree, Key key);
29 PUBLIC Object BTree_remove(BTree * tree, Key key);
30 PUBLIC void BTree_print(BTree * tree);
31 PUBLIC unsigned int BTree_sizeof(BTree* tree);
32 PUBLIC unsigned int BTree_reportSizeInBytes(BTree * tree);
33
34
35 #endif /* _BTREE_ */
36

```

## 4.24 BTree.h

```

1 #ifndef _BTREE_
2 #define _BTREE_
3 /*****
4  * BTree.h
5  *
6  *****/
7 #include "Types.h"
8 #include "Node.h"
9
10 #define TRUE (1)
11 #define FALSE (0)
12
13 typedef struct BTree
14 {
15     void* pool;
16     Node* root;
17     unsigned int depth;
18     unsigned short int nbObjects;
19     unsigned short int nbNodes;
20     unsigned int order;
21     unsigned int nodeSize;
22 } BTree;
23
24 PUBLIC BTree * BTree_new();
25 PUBLIC BTree * BTree_newFromFile(char* fileName);
26 PUBLIC void BTree_free(BTree * tree);
27 PUBLIC void BTree_add(BTree * tree, Key key, Object object);
28 PUBLIC Object BTree_get(BTree * tree, Key key);
29 PUBLIC Object BTree_remove(BTree * tree, Key key);
30 PUBLIC void BTree_print(BTree * tree);
31 PUBLIC unsigned int BTree_sizeof(BTree* tree);
32 PUBLIC unsigned int BTree_reportSizeInBytes(BTree * tree);
33
34
35 #endif /* _BTREE_ */
36

```

## 4.25 CommonTypes.h

```

1 #ifndef _COMMONTYPES_
2 #define _COMMONTYPES_
3
4 #include <stdio.h>
5 #include <malloc.h>
6
7 #define TRUE (1)
8 #define FALSE (0)
9
10 typedef void * Object;
11 typedef void * Key;
12
13 #endif /* _COMMONTYPES_ */

```

## 4.26 Node.h

```

1  /*
2  * Node.h
3  */
4  #ifndef _NODE_
5  #define _NODE_
6
7  #include "Types.h"
8  #include "CommonTypes.h"
9
10 #define ORDER (3)
11
12 typedef struct Node Node;
13
14 typedef struct Node
15 {
16     unsigned short int nbKeyUsed;
17     unsigned short int isLeaf;
18     unsigned int keys[ORDER * 2 - 1];
19     Object leaves[ORDER * 2];
20     Node* children[ORDER * 2];
21 } Node;
22
23 PUBLIC Node * Node_new(unsigned short int isLeaf);
24 PUBLIC Node* Node_splitNode(Node* node, Node* nodeToSplit, Key key);
25 PUBLIC void Node_insert(Node* node, Key key, Object object);
26 PUBLIC Object Node_remove(Node* node, Key key, unsigned int* keyToUpdate);
27 PUBLIC Object Node_search(Node* node, Key key, unsigned int isFoundAlready);
28 PUBLIC void Node_free(Node* node);
29 PUBLIC void Node_print(Node* node, unsigned int depth);
30
31 #endif /* _NODE_ */
32

```

## 4.27 Node.h

```

1  /*
2  * Node.h
3  */
4  #ifndef _NODE_
5  #define _NODE_
6
7  #include "Types.h"
8  #include "CommonTypes.h"
9
10 #define ORDER (3)
11
12 typedef struct Node Node;
13
14 typedef struct Node
15 {
16     unsigned short int nbKeyUsed;
17     unsigned short int isLeaf;
18     unsigned int keys[ORDER * 2 - 1];
19     Object leaves[ORDER * 2];
20     Node* children[ORDER * 2];
21 } Node;
22
23 PUBLIC Node * Node_new(unsigned short int isLeaf);
24 PUBLIC Node* Node_splitNode(Node* node, Node* nodeToSplit, Key key);
25 PUBLIC void Node_insert(Node* node, Key key, Object object);
26 PUBLIC Object Node_remove(Node* node, Key key, unsigned int* keyToUpdate);
27 PUBLIC Object Node_search(Node* node, Key key, unsigned int isFoundAlready);
28 PUBLIC void Node_free(Node* node);
29 PUBLIC void Node_print(Node* node, unsigned int depth);
30
31 #endif /* _NODE_ */
32

```

## 4.28 /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,...)

### 4.28.1 Detailed Description

This file contains debugging functions.

The debugging function are TBD

## 4.29 Debug.h

```
1 /* Debug.h */
2
3 #include <stdarg.h>
4
5 #define TRACE(x) do { if (DEBUG) dbg_printf x; } while (0)
6
7 #define PRINT(x) do { dbg_printf x; } while (0)
8
9 void dbg_printf(const char *fmt, ...);
```

## 4.30 Debug.h

```
1 /* Debug.h */
2
3 #include <stdarg.h>
4
5 #define TRACE(x) do { if (DEBUG) dbg_printf x; } while (0)
6
7 #define PRINT(x) do { dbg_printf x; } while (0)
8
9 void dbg_printf(const char *fmt, ...);
```

## 4.31 /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.*

### 4.31.1 Detailed Description

Reports errors.

This file contains error reporting functions.

### 4.31.2 Function Documentation

#### 4.31.2.1 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.

## 4.32 Error.h

```
1  /* Error.h */
2
3  #include "Types.h"
4
5  typedef enum
6  {
7      ERROR_DBG,
8      ERROR_INFO,
9      ERROR_NORMAL,
10     ERROR_FATAL
11 } ErrorSeverity;
12
13 PUBLIC void Error_new(ErrorSeverity severity, char * msg, ...);
```

## 4.33 Error.h

```
1  /* Error.h */
2
3  #include "Types.h"
```

```

4
5 typedef enum
6 {
7     ERROR_DBG,
8     ERROR_INFO,
9     ERROR_NORMAL,
10    ERROR_FATAL
11 } ErrorSeverity;
12
13 PUBLIC void Error_new(ErrorSeverity severity, char * msg, ...);

```

## 4.34 FileIo.h

```

1 #ifndef _FILEIO_H_
2 #define _FILEIO_H_
3 #include "String2.h"
4 #include "Types.h"
5
6 typedef enum FileIoStatus
7 {
8     UNKNOWN=0,
9     FILE_OPEN,
10    DIR_OPEN
11 } FileIoStatus;
12
13 typedef struct FileIo FileIo;
14
15 PUBLIC FileIo * FileIo_new();
16 PUBLIC void FileIo_delete();
17 PUBLIC void FileIo_openFile(FileIo* this, String* fullFileName);
18 PUBLIC void FileIo_createFile(FileIo* this, String* fullFileName);
19 PUBLIC void FileIo_openDir(FileIo* this, String* fullFileName);
20 PUBLIC void FileIo_createDir(FileIo* this, String* fullDirName);
21 PUBLIC void FileIo_write(FileIo* this, char* buffer, int length);
22 PUBLIC void FileIo_read(FileIo* this, char* buffer, int length);
23 PUBLIC void FileIo_remove(FileIo* this, String* fullFileName);
24 PUBLIC String * FileIo_getCwd(FileIo* this);
25 PUBLIC List * FileIo_listDirs(FileIo * this, String * directory);
26 PUBLIC List* FileIo_listFiles(FileIo* this, String * directory);
27 PUBLIC int FileIo_fSeekEnd(FileIo * this, int pos);
28 PUBLIC int FileIo_fSeekSet(FileIo * this, int pos);
29 PUBLIC int FileIo_ftell(FileIo* this);
30 //Opendir
31 //Readdir
32
33 #endif /* _FILEIO_H_ */

```

## 4.35 FileIo.h

```

1 #ifndef _FILEIO_H_
2 #define _FILEIO_H_
3 #include "String2.h"
4 #include "Types.h"
5
6 typedef enum FileIoStatus
7 {
8     UNKNOWN=0,
9     FILE_OPEN,
10    DIR_OPEN
11 } FileIoStatus;
12
13 typedef struct FileIo FileIo;
14
15 PUBLIC FileIo * FileIo_new();
16 PUBLIC void FileIo_delete();
17 PUBLIC void FileIo_openFile(FileIo* this, String* fullFileName);
18 PUBLIC void FileIo_createFile(FileIo* this, String* fullFileName);
19 PUBLIC void FileIo_openDir(FileIo* this, String* fullFileName);
20 PUBLIC void FileIo_createDir(FileIo* this, String* fullDirName);
21 PUBLIC void FileIo_write(FileIo* this, char* buffer, int length);
22 PUBLIC void FileIo_read(FileIo* this, char* buffer, int length);
23 PUBLIC void FileIo_remove(FileIo* this, String* fullFileName);
24 PUBLIC String * FileIo_getCwd(FileIo* this);
25 PUBLIC List * FileIo_listDirs(FileIo * this, String * directory);
26 PUBLIC List* FileIo_listFiles(FileIo* this, String * directory);
27 PUBLIC int FileIo_fSeekEnd(FileIo * this, int pos);
28 PUBLIC int FileIo_fSeekSet(FileIo * this, int pos);
29 PUBLIC int FileIo_ftell(FileIo* this);
30 //Opendir

```

```

31 //Readdir
32
33 #endif /* _FILEIO_H_ */

```

## 4.36 /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"

```

### Classes

- class [List](#)

### Functions

- PUBLIC void \* [List\\_getNext](#) ([List](#) \*this)
- PUBLIC void [List\\_resetIterator](#) ([List](#) \*this)

#### 4.36.1 Detailed Description

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

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

- init
- add

## 4.37 List.h

```

1 /* List.h */
2
3 #ifndef _LIST_H_
4 #define _LIST_H_
5
6 #include "Types.h"
7
8 typedef struct List List;
9
10 PUBLIC List * List_new();
11 PUBLIC void List_delete(List* this);
12 PUBLIC List * List_copy(List* this);
13 PUBLIC int List_compare(List * this, List * compared);
14 PUBLIC void List_print(List * this);
15 PUBLIC void List_insertHead(List* this, void* item);
16 PUBLIC void List_insertTail(List* this, void* item);
17 PUBLIC void List_merge(List* this, List* l1);
18 PUBLIC void List_forEach(List* this, void (*method)(void* o));
19 PUBLIC void * List_getNext(List* this);
20 PUBLIC void * List_removeHead(List * this);
21 PUBLIC void* List_removeTail(List* this);
22 PUBLIC void * List_getHead(List * this);
23 PUBLIC unsigned int List_getSize(List * this);
24 PUBLIC void List_resetIterator(List * this);
25
26 #endif /* _LIST_H_ */

```

## 4.38 List.h

```

1  /* List.h */
2
3  #ifndef _LIST_H_
4  #define _LIST_H_
5
6  #include "Types.h"
7
8  typedef struct List List;
9
10 PUBLIC List * List_new();
11 PUBLIC void List_delete(List* this);
12 PUBLIC List * List_copy(List* this);
13 PUBLIC int List_compare(List * this, List * compared);
14 PUBLIC void List_print(List * this);
15 PUBLIC void List_insertHead(List* this, void* item);
16 PUBLIC void List_insertTail(List* this, void* item);
17 PUBLIC void List_merge(List* this, List* l1);
18 PUBLIC void List_forEach(List* this, void (*method)(void* o));
19 PUBLIC void * List_getNext(List* this);
20 PUBLIC void * List_removeHead(List * this);
21 PUBLIC void* List_removeTail(List* this);
22 PUBLIC void * List_getHead(List * this);
23 PUBLIC unsigned int List_getSize(List * this);
24 PUBLIC void List_resetIterator(List * this);
25
26 #endif /* _LIST_H_ */

```

## 4.39 /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 "Error.h"

```

### Classes

- class [Map](#)

### Macros

- #define **HTABLE\_SIZE** (50)

### Functions

- PRIVATE [MapEntry](#) \* **Map\_findEntry** ([Map](#) \*this, [String](#) \*s)
- PUBLIC void **Map\_print** ([Map](#) \*this)

### 4.39.1 Detailed Description

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

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

## 4.40 Map.h

```
1  /* Map.h */
2
3  #ifndef _MAP_H_
4  #define _MAP_H_
5
6  #include "Types.h"
7  #include "String2.h"
8  #include "List.h"
9
10 typedef struct Map Map;
11
12 PUBLIC Map * Map_new();
13 PUBLIC void Map_delete(Map * this);
14 PUBLIC Map * Map_copy(Map * this);
15 PUBLIC unsigned int Map_insert(Map * this, String* s, void * p);
16 PUBLIC unsigned int Map_find(Map * this, String* s, void ** p);
17 PUBLIC void Map_print(Map * this);
18 PUBLIC List * Map_getAll(Map * this);
19
20 #endif /* _MAP_H_ */
```

## 4.41 Map.h

```
1  /* Map.h */
2
3  #ifndef _MAP_H_
4  #define _MAP_H_
5
6  #include "Types.h"
7  #include "String2.h"
8  #include "List.h"
9
10 typedef struct Map Map;
11
12 PUBLIC Map * Map_new();
13 PUBLIC void Map_delete(Map * this);
14 PUBLIC Map * Map_copy(Map * this);
15 PUBLIC unsigned int Map_insert(Map * this, String* s, void * p);
16 PUBLIC unsigned int Map_find(Map * this, String* s, void ** p);
17 PUBLIC void Map_print(Map * this);
18 PUBLIC List * Map_getAll(Map * this);
19
20 #endif /* _MAP_H_ */
```

## 4.42 MapEntry.h

```
1  /* MapEntry.h */
2
3  #ifndef _MAPENTRY_H_
4  #define _MAPENTRY_H_
5
6  #include "Types.h"
7  #include "String2.h"
8
9  typedef struct MapEntry MapEntry;
10
11 PUBLIC MapEntry * MapEntry_new();
12 PUBLIC void MapEntry_delete(MapEntry * this);
13 PUBLIC MapEntry * MapEntry_copy(MapEntry * this);
14 PUBLIC String * MapEntry_getString(MapEntry * this);
15 PUBLIC void * MapEntry_getItem(MapEntry * this);
16 PUBLIC void MapEntry_setString(MapEntry * this, String * s);
17 PUBLIC void MapEntry_setItem(MapEntry * this, void * item);
18
19 #endif /* _MAPENTRY_H_ */
```

## 4.43 MapEntry.h

```

1  /* MapEntry.h */
2
3  #ifndef _MAPENTRY_H_
4  #define _MAPENTRY_H_
5
6  #include "Types.h"
7  #include "String2.h"
8
9  typedef struct MapEntry MapEntry;
10
11 PUBLIC MapEntry * MapEntry_new();
12 PUBLIC void MapEntry_delete(MapEntry * this);
13 PUBLIC MapEntry * MapEntry_copy(MapEntry * this);
14 PUBLIC String * MapEntry_getString(MapEntry * this);
15 PUBLIC void * MapEntry_getItem(MapEntry * this);
16 PUBLIC void MapEntry_setString(MapEntry * this, String * s);
17 PUBLIC void MapEntry_setItem(MapEntry * this, void * item);
18
19 #endif /* _MAPENTRY_H_ */

```

## 4.44 /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

### 4.44.1 Detailed Description

This file provides the implementation of the memory functions.

TBD

## 4.45 Memory.h

```

1  /* Memory.h */
2
3  #ifndef _MEMORY_H_
4  #define _MEMORY_H_
5
6  #include "Types.h"
7
8  PUBLIC void* Memory_alloc(unsigned int nbBytes);
9  PUBLIC void * Memory_realloc(void * pointer, unsigned int prevSizeBytes, unsigned int newSizeBytes);
10 PUBLIC void Memory_free(void* pointer, unsigned int nbBytes);
11 PUBLIC void Memory_set(void * pointer, unsigned char val, unsigned int nbBytes);
12 PUBLIC void Memory_copy(void * pointer, void * src, unsigned int nbBytes);
13 PUBLIC int Memory_ncmp(void * pointer, void * compared, unsigned int nbBytes);
14 PUBLIC int Memory_cmp(void * pointer, void * compared);
15 PUBLIC unsigned int Memory_len(const void * pointer);
16 PUBLIC void Memory_report();
17 PUBLIC int Memory_getAllocRequestNb();
18 PUBLIC int Memory_getFreeRequestNb();
19
20 #endif /* _MEMORY_H_ */

```

## 4.46 Memory.h

```

1  /* Memory.h */
2
3  #ifndef _MEMORY_H_
4  #define _MEMORY_H_
5
6  #include "Types.h"
7
8  PUBLIC void* Memory_alloc(unsigned int nbBytes);
9  PUBLIC void * Memory_realloc(void * pointer, unsigned int prevSizeBytes, unsigned int newSizeBytes);
10 PUBLIC void Memory_free(void* pointer, unsigned int nbBytes);
11 PUBLIC void Memory_set(void * pointer, unsigned char val, unsigned int nbBytes);
12 PUBLIC void Memory_copy(void * pointer, void * src, unsigned int nbBytes);
13 PUBLIC int Memory_ncmp(void * pointer, void * compared, unsigned int nbBytes);
14 PUBLIC int Memory_cmp(void * pointer, void * compared);
15 PUBLIC unsigned int Memory_len(const void * pointer);
16 PUBLIC void Memory_report();
17 PUBLIC int Memory_getAllocRequestNb();
18 PUBLIC int Memory_getFreeRequestNb();
19
20 #endif /* _MEMORY_H_ */

```

## 4.47 Class.h

```

1  /* Class. h */
2
3  #ifndef _CLASS_H_
4  #define _CLASS_H_
5
6  #include "Object.h"
7
8  struct Class
9  {
10     Constructor f_new;
11     Destructor f_delete;
12     Copy_Operator f_copy;
13     Comp_Operator f_comp;
14     Printer f_print;
15 };
16
17 #endif /* _CLASS_H_ */

```

## 4.48 Class.h

```

1  /* Class. h */
2
3  #ifndef _CLASS_H_
4  #define _CLASS_H_
5
6  #include "Object.h"
7

```

```

8 struct Class
9 {
10     Constructor f_new;
11     Destructor f_delete;
12     Copy_Operator f_copy;
13     Comp_Operator f_comp;
14     Printer f_print;
15 };
16
17 #endif /* _CLASS_H_ */

```

## 4.49 /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"

```

### 4.49.1 Detailed Description

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

The class [Object](#) is TBD

## 4.50 Object.h

```

1 /* Object.h */
2
3 #ifndef _OBJECT_H_
4 #define _OBJECT_H_
5
6 #include "Types.h"
7
8 typedef struct Class Class;
9 typedef struct Object Object;
10
11 struct Object
12 {
13     unsigned int id;
14     Class * class;
15     void (*delete)(Object * this);
16     Object * (*copy)(Object * this);
17     unsigned int refCount;
18     unsigned int size;
19 };
20
21 typedef struct Object * (*Constructor)();
22 typedef void (*Destructor)(struct Object *);
23 typedef struct Object * (*Copy_Operator)(struct Object *);
24 typedef int (*Comp_Operator)(struct Object *, struct Object *);
25 typedef char * (*Printer)(struct Object *);
26
27
28
29 PUBLIC Object * Object_new(unsigned int size, Class * class);
30 PUBLIC void Object_delete(Object * this);
31 PUBLIC Object * Object_copy(Object * this);
32 PUBLIC unsigned int Object_isEqual(Object * this, Object * compared);
33 PUBLIC char * Object_print(Object * this);
34 PUBLIC Object* Object_getRef(Object* this);
35
36 #endif /* _OBJECT_H_ */

```



## 4.51 Object.h

```

1  /* Object.h */
2
3  #ifndef _OBJECT_H_
4  #define _OBJECT_H_
5
6  #include "Types.h"
7
8  typedef struct Class Class;
9  typedef struct Object Object;
10
11 struct Object
12 {
13     unsigned int id;
14     Class * class;
15     void (*delete)(Object * this);
16     Object * (*copy)(Object * this);
17     unsigned int refCount;
18     unsigned int size;
19 };
20
21 typedef struct Object * (*Constructor)();
22 typedef void (*Destructor)(struct Object *);
23 typedef struct Object * (*Copy_Operator)(struct Object *);
24 typedef int (*Comp_Operator)(struct Object *, struct Object *);
25 typedef char * (*Printer)(struct Object *);
26
27
28
29 PUBLIC Object * Object_new(unsigned int size, Class * class);
30 PUBLIC void Object_delete(Object * this);
31 PUBLIC Object * Object_copy(Object * this);
32 PUBLIC unsigned int Object_isEqual(Object * this, Object * compared);
33 PUBLIC char * Object_print(Object * this);
34 PUBLIC Object* Object_getRef(Object* this);
35
36 #endif /* _OBJECT_H_ */

```

## 4.52 /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 <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](#)

## Functions

- PUBLIC void **ObjectMgr\_reportUnallocated** ([ObjectMgr](#) \*this)

## Variables

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

### 4.52.1 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.

## 4.53 ObjectMgr.h

```
1 /* ObjectMgr.h */
2
3 #ifndef _OBJECTMGR_H_
4 #define _OBJECTMGR_H_
5
6 #include "Object.h"
7 #include "Types.h"
8
9 typedef struct ObjectMgr ObjectMgr;
10
11 PUBLIC void ObjectMgr_delete(ObjectMgr * this);
12 PUBLIC ObjectMgr * ObjectMgr_copy(ObjectMgr * this);
13 PUBLIC ObjectMgr * ObjectMgr_getRef();
14 PUBLIC Object * ObjectMgr_allocate(ObjectMgr * this, unsigned int size);
15 PUBLIC void ObjectMgr_deallocate(ObjectMgr * this, Object * object);
16 PUBLIC void ObjectMgr_reportUnallocated(ObjectMgr* this);
17 PUBLIC void ObjectMgr_report(ObjectMgr * this);
18
19 #endif /* _OBJECTMGR_H_ */
```

## 4.54 ObjectMgr.h

```
1 /* ObjectMgr.h */
2
3 #ifndef _OBJECTMGR_H_
4 #define _OBJECTMGR_H_
5
6 #include "Object.h"
7 #include "Types.h"
8
9 typedef struct ObjectMgr ObjectMgr;
10
11 PUBLIC void ObjectMgr_delete(ObjectMgr * this);
12 PUBLIC ObjectMgr * ObjectMgr_copy(ObjectMgr * this);
13 PUBLIC ObjectMgr * ObjectMgr_getRef();
14 PUBLIC Object * ObjectMgr_allocate(ObjectMgr * this, unsigned int size);
15 PUBLIC void ObjectMgr_deallocate(ObjectMgr * this, Object * object);
16 PUBLIC void ObjectMgr_reportUnallocated(ObjectMgr* this);
17 PUBLIC void ObjectMgr_report(ObjectMgr * this);
18
19 #endif /* _OBJECTMGR_H_ */
```

## 4.55 /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 <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\\_free](#) (Pool \*pool)  
*Pool\_free.*
- 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)

### 4.55.1 Detailed Description

This file contains the implementation of the class Pool.

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

- Alloc
- De-alloc

### 4.55.2 Function Documentation

#### 4.55.2.1 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

#### 4.55.2.2 Pool\_allocInFile()

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

Pool\_allocInFile.

##### Parameters

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

##### Returns

none

### 4.55.2.3 Pool\_dealloc()

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

Pool\_dealloc.

#### Parameters

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

#### Returns

none

### 4.55.2.4 Pool\_deallocInFile()

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

Pool\_deallocInFile.

#### Parameters

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

#### Returns

none

### 4.55.2.5 Pool\_deallocInMemory()

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

Pool\_deallocInMemory.

#### Parameters

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

**Returns**

none

**4.55.2.6 Pool\_free()**

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

Pool\_free.

**Parameters**

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

**Returns**

none

**4.55.2.7 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.

**4.55.2.8 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

**4.55.2.9 Pool\_read()**

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

Pool\_read.

**Parameters**

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

**Returns**

none

**4.55.2.10 Pool\_readInFile()**

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

Pool\_readInFile.

**Parameters**

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

**Returns**

none

**4.55.2.11 Pool\_readInMemory()**

```
PRIVATE void Pool_readInMemory (
    Pool * pool,
```

```
    unsigned int idx,  
    void * p )
```

Pool\_readInMemory.

#### Parameters

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

#### Returns

none

### 4.55.2.12 Pool\_report()

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

Pool\_report.

#### Parameters

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

#### Returns

none

### 4.55.2.13 Pool\_reportInFile()

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

Pool\_reportInFile.

#### Parameters

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

#### Returns

none



#### 4.55.2.14 Pool\_reportInMemory()

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

Pool\_reportInMemory.

##### Parameters

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

##### Returns

none

#### 4.55.2.15 Pool\_reportNbNodes()

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

Pool\_reportNbNodes.

##### Parameters

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

##### Returns

none

#### 4.55.2.16 Pool\_reportSizeInBytes()

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

Pool\_reportSizeInBytes input: none.

##### Returns

none

#### 4.55.2.17 Pool\_write()

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

Pool\_writeCache.

## Parameters

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

## Returns

none none

**4.55.2.18 Pool\_writeInFile()**

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

Pool\_writeInFile.

## Parameters

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

## Returns

none

**4.55.2.19 Pool\_writelnMemory()**

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

Pool\_writelnMemory.

## Parameters

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

## Returns

none

**4.56 Pool.h**

```
1 #ifndef _POOL_
```

```

2 #define _POOL_
3 /*****
4  * Pool.h
5  *
6  *****/
7 #include "Types.h"
8 #include "Pool.h"
9
10 typedef enum AllocStatus
11 {
12     ALLOC_OK = 0,
13     ALLOC_FAIL = 1
14 } AllocStatus;
15
16 typedef struct PoolCache
17 {
18     unsigned int idx;
19     unsigned int isUsed;
20     void* cache;
21 } PoolCache;
22
23 typedef struct Pool Pool;
24
25 PUBLIC Pool* Pool_new(unsigned int nbMemChunks, unsigned int memChunkSize);
26 PUBLIC Pool* Pool_newFromFile(char* fileName, unsigned int nbMemChunks, unsigned int memChunkSize);
27 PUBLIC void Pool_free(Pool* pool);
28 PUBLIC void * Pool_alloc(Pool* pool, unsigned int * ptrIdx);
29 PUBLIC void Pool_dealloc(Pool* pool, unsigned int p);
30 PUBLIC void Pool_write(Pool* pool, unsigned int idx, void* ptrContent);
31 PUBLIC void* Pool_read(Pool* pool, unsigned int idx);
32 PUBLIC unsigned int Pool_addToChunkCache(Pool* pool, void* p, unsigned int length);
33 PUBLIC void Pool_report(Pool* pool);
34 PUBLIC unsigned int Pool_reportSizeInBytes(Pool* pool);
35 PUBLIC unsigned int Pool_reportNbNodes(Pool* pool);
36 PUBLIC void Pool_discardCache(Pool* pool, unsigned int idx);
37 PUBLIC void Pool_discardAllCache(Pool* pool);
38 PUBLIC unsigned int Pool_reportCacheUsed(Pool * pool);
39 #endif /* _POOL_ */

```

## 4.57 Pool.h

```

1 #ifndef _POOL_
2 #define _POOL_
3 /*****
4  * Pool.h
5  *
6  *****/
7 #include "Types.h"
8 #include "Pool.h"
9
10 typedef enum AllocStatus
11 {
12     ALLOC_OK = 0,
13     ALLOC_FAIL = 1
14 } AllocStatus;
15
16 typedef struct PoolCache
17 {
18     unsigned int idx;
19     unsigned int isUsed;
20     void* cache;
21 } PoolCache;
22
23 typedef struct Pool Pool;
24
25 PUBLIC Pool* Pool_new(unsigned int nbMemChunks, unsigned int memChunkSize);
26 PUBLIC Pool* Pool_newFromFile(char* fileName, unsigned int nbMemChunks, unsigned int memChunkSize);
27 PUBLIC void Pool_free(Pool* pool);
28 PUBLIC void * Pool_alloc(Pool* pool, unsigned int * ptrIdx);
29 PUBLIC void Pool_dealloc(Pool* pool, unsigned int p);
30 PUBLIC void Pool_write(Pool* pool, unsigned int idx, void* ptrContent);
31 PUBLIC void* Pool_read(Pool* pool, unsigned int idx);
32 PUBLIC unsigned int Pool_addToChunkCache(Pool* pool, void* p, unsigned int length);
33 PUBLIC void Pool_report(Pool* pool);
34 PUBLIC unsigned int Pool_reportSizeInBytes(Pool* pool);
35 PUBLIC unsigned int Pool_reportNbNodes(Pool* pool);
36 PUBLIC void Pool_discardCache(Pool* pool, unsigned int idx);
37 PUBLIC void Pool_discardAllCache(Pool* pool);
38 PUBLIC unsigned int Pool_reportCacheUsed(Pool * pool);
39 #endif /* _POOL_ */

```

## 4.58 /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 <limits.h>
#include <stdlib.h>
```

### Classes

- struct [SkipNode](#)

### Macros

- #define **SKIPLIST\_MAX\_LEVEL** (6)

### Typedefs

- typedef struct [SkipNode](#) **SkipNode**

### Functions

- PRIVATE unsigned int **SkipList\_randLevel** (SkipList \*this)
- PRIVATE unsigned int **SkipList\_reportCache** (SkipList \*this)
- PRIVATE unsigned int **myGreateOrEqual** (unsigned int k1, unsigned int k2)
- PRIVATE unsigned int **myGreater** (unsigned int k1, unsigned int k2)
- PRIVATE unsigned int **myEqual** (unsigned int k1, unsigned int k2)
- PUBLIC SkipList \* [SkipList\\_new](#) (unsigned int maxObjectNb)  
*SkipList\_new.*
- PUBLIC void [SkipList\\_delete](#) (SkipList \*this)  
*SkipList\_free.*
- PUBLIC SkipList \* [SkipList\\_copy](#) (SkipList \*this)  
*SkipList\_copy.*
- PUBLIC void [SkipList\\_add](#) (SkipList \*this, unsigned int key, void \*object)  
*SkipList\_add.*
- PUBLIC void \* [SkipList\\_remove](#) (SkipList \*this, unsigned int key)  
*SkipList\_delete.*
- PUBLIC void \* **SkipList\_get** (SkipList \*this, unsigned int key)
- PUBLIC int [SkipList\\_compare](#) (SkipList \*this, SkipList \*compared)  
*SkipList\_compare.*
- PUBLIC void [SkipList\\_print](#) (SkipList \*this)  
*SkipList\_print.*

### 4.58.1 Detailed Description

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

- Add
- Remove
- Get

### 4.58.2 Function Documentation

#### 4.58.2.1 SkipList\_add()

```
PUBLIC void SkipList_add (  
    SkipList * this,  
    unsigned int key,  
    void * object )
```

SkipList\_add.

##### Parameters

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

##### Returns

None

#### 4.58.2.2 SkipList\_compare()

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

SkipList\_compare.

##### Parameters

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

**Returns**

None

**4.58.2.3 SkipList\_copy()**

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

SkipList\_copy.

**Parameters**

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

**Returns**

None

**4.58.2.4 SkipList\_delete()**

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

SkipList\_free.

**Parameters**

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

**Returns**

None

**4.58.2.5 SkipList\_new()**

```
PUBLIC SkipList * SkipList_new (
    unsigned int maxObjectNb )
```

SkipList\_new.

**Parameters**

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

**Returns**

New instance of class SkipList.

**4.58.2.6 SkipList\_print()**

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

SkipList\_print.

**Parameters**

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

**Returns**

None

**4.58.2.7 SkipList\_remove()**

```
PUBLIC void * SkipList_remove (
    SkipList * this,
    unsigned int key )
```

SkipList\_delete.

**Parameters**

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

**Returns**

[Object](#) removed from SkipList object.

**4.59 SkipList.h**

```
1 #ifndef _SKIPLIST_
2 #define _SKIPLIST_
3 /* SkipList.h */
```



```

4 #include "Types.h"
5 #include "Pool.h"
6
7 typedef struct SkipList SkipList;
8
9 PUBLIC SkipList* SkipList_new();
10 PUBLIC void SkipList_delete(SkipList* skipList);
11 PUBLIC SkipList * SkipList_copy(SkipList * this);
12 PUBLIC void SkipList_add(SkipList* this, unsigned int key, void* object);
13 PUBLIC void * SkipList_remove(SkipList* this, unsigned int key);
14 PUBLIC void * SkipList_get(SkipList* this, unsigned int key);
15 PUBLIC int SkipList_compare(SkipList * this, SkipList * compared);
16 PUBLIC void SkipList_print(SkipList* this);
17 #endif /* _SKIPLIST_ */

```

## 4.60 SkipList.h

```

1 #ifndef _SKIPLIST_
2 #define _SKIPLIST_
3 /* SkipList.h */
4 #include "Types.h"
5 #include "Pool.h"
6
7 typedef struct SkipList SkipList;
8
9 PUBLIC SkipList* SkipList_new();
10 PUBLIC void SkipList_delete(SkipList* skipList);
11 PUBLIC SkipList * SkipList_copy(SkipList * this);
12 PUBLIC void SkipList_add(SkipList* this, unsigned int key, void* object);
13 PUBLIC void * SkipList_remove(SkipList* this, unsigned int key);
14 PUBLIC void * SkipList_get(SkipList* this, unsigned int key);
15 PUBLIC int SkipList_compare(SkipList * this, SkipList * compared);
16 PUBLIC void SkipList_print(SkipList* this);
17 #endif /* _SKIPLIST_ */

```

## 4.61 String2.h

```

1 /* String2.h */
2
3 #ifndef _STRING2_H_
4 #define _STRING2_H_
5
6 #include "Types.h"
7 #include "List.h"
8
9 typedef struct String String;
10
11 PUBLIC String * String_new(const char * constString);
12 PUBLIC void String_delete(String * this);
13 PUBLIC String * String_copy(String * this);
14 PUBLIC String * String_getRef(String * this);
15 PUBLIC unsigned int String_getLength(String * this);
16 PUBLIC char * String_getBuffer(String * this);
17 PUBLIC void String_setBuffer(String * this, char * buffer);
18 PUBLIC unsigned int String_isContained(String * this, String * s2);
19 PUBLIC unsigned int String_prepend(String * this, const char * prefix);
20 PUBLIC unsigned int String_append(String * this, const char* postfix);
21 PUBLIC int String_compare(String * this, String * compared);
22 PUBLIC String * String_subString(String * this, unsigned int idx, unsigned int length);
23 PUBLIC unsigned int String_matchWildcard(String * this, const char * wildcard);
24 PUBLIC int String_toInt(String* this);
25 PUBLIC List* String_splitToken(String* this, const char* separator);
26 PUBLIC void String_stealBuffer(String* this, String* s);
27 #endif /* _STRING2_H_ */

```

## 4.62 String2.h

```

1 /* String2.h */
2
3 #ifndef _STRING2_H_
4 #define _STRING2_H_
5
6 #include "Types.h"
7 #include "List.h"
8

```

```

9 typedef struct String String;
10
11 PUBLIC String * String_new(const char * constString);
12 PUBLIC void String_delete(String * this);
13 PUBLIC String * String_copy(String * this);
14 PUBLIC String * String_getRef(String * this);
15 PUBLIC unsigned int String_getLength(String * this);
16 PUBLIC char * String_getBuffer(String * this);
17 PUBLIC void String_setBuffer(String * this, char * buffer);
18 PUBLIC unsigned int String_isContained(String * this, String * s2);
19 PUBLIC unsigned int String_prepend(String * this, const char * prefix);
20 PUBLIC unsigned int String_append(String* this, const char* postfix);
21 PUBLIC int String_compare(String * this, String * compared);
22 PUBLIC String * String_subString(String * this, unsigned int idx, unsigned int length);
23 PUBLIC unsigned int String_matchWildcard(String * this, const char * wildcard);
24 PUBLIC int String_toInt(String* this);
25 PUBLIC List* String_splitToken(String* this, const char* separator);
26 PUBLIC void String_stealBuffer(String* this, String* s);
27 #endif /* _STRING2_H_ */

```

## 4.63 Times.h

```

1 /* Time.h */
2
3 long double get_wall_time();
4 long double get_cpu_time();
5

```

## 4.64 Times.h

```

1 /* Time.h */
2
3 long double get_wall_time();
4 long double get_cpu_time();
5

```

## 4.65 Types.h

```

1 /* Types.h */
2
3 #ifndef _TYPES_H_
4 #define _TYPES_H_
5
6 #define PUBLIC
7
8 #ifndef UNIT_TEST
9     #define PRIVATE static
10 #else
11     #define PRIVATE
12 #endif
13 #endif /* _TYPES_H_ */

```

## 4.66 Types.h

```

1 /* Types.h */
2
3 #ifndef _TYPES_H_
4 #define _TYPES_H_
5
6 #define PUBLIC
7
8 #ifndef UNIT_TEST
9     #define PRIVATE static
10 #else
11     #define PRIVATE
12 #endif
13 #endif /* _TYPES_H_ */

```

## 4.67 Declarator.h

```

1  /* Declarator.h */
2
3  #ifndef _DECLARATOR_H_
4  #define _DECLARATOR_H_
5
6  typedef enum
7  {
8      E_DEC_FUNCTION,
9      E_DEC_VAR,
10     E_DEC_TYPE
11 } DeclaratorType;
12
13 typedef struct Declarator Declarator;
14
15 Declarator * Declarator_new(DeclaratorType * t)
16 {
17 }
18
19 void Declarator_delete(Declarator * this)
20 {
21 }
22
23 #endif /* #ifndef _DECLARATOR_H_

```

## 4.68 /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](#)

### Functions

- PRIVATE void [FileReader\\_getListPreferredDir](#) ([FileReader](#) \*this)
- PRIVATE void [FileReader\\_deleteListPreferredDir](#) ([FileReader](#) \*this)
- PRIVATE void [FileReader\\_printListPreferredDir](#) ([FileReader](#) \*this)

### 4.68.1 Detailed Description

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

The class [FileReader](#) is TBD

## 4.69 FileReader.h

```

1  /* FileReader.h */
2
3  #ifndef _FILEREADER_H_
4  #define _FILEREADER_H_
5
6  #include "Types.h"
7  #include "String2.h"
8
9  typedef struct FileReader FileReader;
10
11 PUBLIC FileReader * FileReader_new();
12 PUBLIC void FileReader_delete(FileReader * this);
13 PUBLIC FileReader * FileReader_copy(FileReader * this);
14 PUBLIC char * FileReader_getBuffer(FileReader * this);
15 PUBLIC String * FileReader_getName(FileReader * this);
16 PUBLIC char * FileReader_addFile(FileReader * this, String * fileName);
17 #endif /* _FILEREADER_H_ */

```

## 4.70 FileReader.h

```

1  /* FileReader.h */
2
3  #ifndef _FILEREADER_H_
4  #define _FILEREADER_H_
5
6  #include "Types.h"
7  #include "String2.h"
8
9  typedef struct FileReader FileReader;
10
11 PUBLIC FileReader * FileReader_new();
12 PUBLIC void FileReader_delete(FileReader * this);
13 PUBLIC FileReader * FileReader_copy(FileReader * this);
14 PUBLIC char * FileReader_getBuffer(FileReader * this);
15 PUBLIC String * FileReader_getName(FileReader * this);
16 PUBLIC char * FileReader_addFile(FileReader * this, String * fileName);
17 #endif /* _FILEREADER_H_ */

```

## 4.71 Grammar2.h

```

1  /* Grammar2.h */
2
3  #include "Types.h"
4  #include "SdbMgr.h"
5  #include "FileReader.h"
6
7  typedef struct Grammar2 Grammar2;
8
9 PUBLIC Grammar2 * Grammar2_new(FileReader * fr, SdbMgr * sdbMgr);
10 PUBLIC void Grammar2_delete(Grammar2 * this);
11 PUBLIC Grammar2 * Grammar2_copy(Grammar2 * this);
12 PUBLIC void Grammar2_process(Grammar2 * this);
13 PUBLIC FileReader * Grammar2_getFileReader(Grammar2 * grammar);
14 PUBLIC SdbMgr * Grammar2_getSdbMgr(Grammar2 * grammar);
15 PUBLIC void Grammar2_addToBuffer(Grammar2 * grammar, char * text);
16 PUBLIC void Grammar2_addComment(Grammar2 * this);
17 PUBLIC void Grammar2_addCodeNode(Grammar2 * this);
18 PUBLIC void Grammar2_addIncludeNode(Grammar2 * this, char * name);
19 PUBLIC char * Grammar2_processNewFile(Grammar2 * this, String * fileName);
20 PUBLIC void Grammar2_returnToFile(Grammar2 * this);

```

## 4.72 Grammar2.h

```

1  /* Grammar2.h */
2
3  #include "Types.h"
4  #include "SdbMgr.h"
5  #include "FileReader.h"
6
7  typedef struct Grammar2 Grammar2;
8
9 PUBLIC Grammar2 * Grammar2_new(FileReader * fr, SdbMgr * sdbMgr);
10 PUBLIC void Grammar2_delete(Grammar2 * this);

```

```

11 PUBLIC Grammar2 * Grammar2_copy(Grammar2 * this);
12 PUBLIC void Grammar2_process(Grammar2 * this);
13 PUBLIC FileReader * Grammar2_getFileReader(Grammar2 * grammar);
14 PUBLIC SdbMgr * Grammar2_getSdbMgr(Grammar2 * grammar);
15 PUBLIC void Grammar2_addToBuffer(Grammar2 * grammar, char * text);
16 PUBLIC void Grammar2_addComment(Grammar2 * this);
17 PUBLIC void Grammar2_addCodeNode(Grammar2 * this);
18 PUBLIC void Grammar2_addIncludeNode(Grammar2 * this, char * name);
19 PUBLIC char * Grammar2_processNewFile(Grammar2 * this, String * fileName);
20 PUBLIC void Grammar2_returnToFile(Grammar2 * this);

```

## 4.73 Grammar2.parse.h

```

1  /* A Bison parser, made by GNU Bison 3.8.2.  */
2
3  /* Bison interface for Yacc-like parsers in C
4
5     Copyright (C) 1984, 1989-1990, 2000-2015, 2018-2021 Free Software Foundation,
6     Inc.
7
8     This program is free software: you can redistribute it and/or modify
9     it under the terms of the GNU General Public License as published by
10    the Free Software Foundation, either version 3 of the License, or
11    (at your option) any later version.
12
13    This program is distributed in the hope that it will be useful,
14    but WITHOUT ANY WARRANTY; without even the implied warranty of
15    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16    GNU General Public License for more details.
17
18    You should have received a copy of the GNU General Public License
19    along with this program. If not, see <https://www.gnu.org/licenses/>.  */
20
21 /* As a special exception, you may create a larger work that contains
22    part or all of the Bison parser skeleton and distribute that work
23    under terms of your choice, so long as that work isn't itself a
24    parser generator using the skeleton or a modified version thereof
25    as a parser skeleton. Alternatively, if you modify or redistribute
26    the parser skeleton itself, you may (at your option) remove this
27    special exception, which will cause the skeleton and the resulting
28    Bison output files to be licensed under the GNU General Public
29    License without this special exception.
30
31    This special exception was added by the Free Software Foundation in
32    version 2.2 of Bison.  */
33
34 /* DO NOT RELY ON FEATURES THAT ARE NOT DOCUMENTED in the manual,
35    especially those whose name start with YY_ or yy_. They are
36    private implementation details that can be changed or removed.  */
37
38 #ifndef YY_GRAMMAR2_GRAMMAR2_PARSE_H_INCLUDED
39 # define YY_GRAMMAR2_GRAMMAR2_PARSE_H_INCLUDED
40 /* Debug traces.  */
41 #ifndef YYDEBUG
42 # define YYDEBUG 0
43 #endif
44 #if YYDEBUG
45 extern int Grammar2_debug;
46 #endif
47
48 /* Token kinds.  */
49 #ifndef YYTOKENTYPE
50 # define YYTOKENTYPE
51     enum yytokentype
52     {
53         YYEMPTY = -2,
54         YYEOF = 0, /* "end of file" */
55         YYerror = 256, /* error */
56         YYUNDEF = 257, /* "invalid token" */
57         COMMENT = 258, /* COMMENT */
58         CODE = 259, /* CODE */
59         END_OF_UNIT = 260 /* END_OF_UNIT */
60     };
61     typedef enum yytokentype yytoken_kind_t;
62 #endif
63 /* Token kinds.  */
64 #define YYEMPTY -2
65 #define YYEOF 0
66 #define YYerror 256
67 #define YYUNDEF 257
68 #define COMMENT 258
69 #define CODE 259
70 #define END_OF_UNIT 260

```

```

71
72 /* Value type. */
73 #if ! defined YYSTYPE && ! defined YYSTYPE_IS_DECLARED
74 union YYSTYPE
75 {
76 #line 18 "Grammar2.y"
77
78     String * text;
79
80 #line 81 "Grammar2.parse.h"
81
82 };
83 typedef union YYSTYPE YYSTYPE;
84 # define YYSTYPE_IS_TRIVIAL 1
85 # define YYSTYPE_IS_DECLARED 1
86 #endif
87
88
89
90
91 int Grammar2_parse (void * scanner, Grammar2 * grammar);
92
93
94 #endif /* !YY_GRAMMAR2_GRAMMAR2_PARSE_H_INCLUDED */

```

## 4.74 /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 "SdbMgr.h"
#include "Error.h"
#include "Grammar2.h"
#include "FileMgr.h"
#include "FileDesc.h"
#include "List.h"

```

### Classes

- class [SParse](#)

### Functions

- PRIVATE unsigned int **SParse\_parseFile** ([SParse](#) \*this, [FileDesc](#) \*fileDesc)
- PRIVATE void **SParse\_buildPreferredDirList** ([SParse](#) \*this, const char \*extension)
- PUBLIC [SParse](#) \* **SParse\_copy** ([SParse](#) \*this)

#### 4.74.1 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.

## 4.75 SParse.h

```
1  /* SParse.h */
2
3  #ifndef _SPARSE_H_
4  #define _SPARSE_H_
5
6  #include "Types.h"
7
8  typedef struct SParse SParse;
9
10 PUBLIC SParse *SParse_new(/* Sdb name */);
11 PUBLIC void SParse_delete(SParse * this);
12 PUBLIC SParse * SParse_copy(SParse * this);
13 PUBLIC unsigned int SParse_parse(SParse * this, const char * extension);
14
15 #endif /* _SPARSE_H_ */
```

## 4.76 SParse.h

```
1  /* SParse.h */
2
3  #ifndef _SPARSE_H_
4  #define _SPARSE_H_
5
6  #include "Types.h"
7
8  typedef struct SParse SParse;
9
10 PUBLIC SParse *SParse_new(/* Sdb name */);
11 PUBLIC void SParse_delete(SParse * this);
12 PUBLIC SParse * SParse_copy(SParse * this);
13 PUBLIC unsigned int SParse_parse(SParse * this, const char * extension);
14
15 #endif /* _SPARSE_H_ */
```





# Index

/home/thomas/Projects/SParse-master/SParse/src/AppLib/FileMgr/51bDesc.c,  
41 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Debug/De  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/FileMgr/51bDesc.h,  
42 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Error/Erro  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/FileMgr/51bMgr.c,  
42 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Error/Erro  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/FileMgr/51bMgr.h,  
43 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Filelo/File  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/OptionMgr/57OptionMgr.c,  
44 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/List.c,  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/OptionMgr/57OptionMgr.h,  
45 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/List/List.h,  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/SdbMgr/58bMgr.c,  
47 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/Map.  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/SdbMgr/58bMgr.h,  
48 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/Map.  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/SdbMgr/58bRequest.h,  
49 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Map/Map.  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/TimeMgr/60TimeMgr.c,  
49 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Memory/M  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/TimeMgr/60TimeMgr.h,  
50 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Memory/M  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/TimeMgr/60Timer.h,  
50 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/Cl  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/61bDesc.h,  
42 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/Ob  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/61bMgr.h,  
43 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Object/Ob  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/61bOptionMgr.h,  
44 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectMg  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/61bSdbMgr.h,  
48 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/ObjectMg  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/61bRequest.h,  
48 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Pool/Pool.  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/61bTimeMgr.h,  
50 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Pool/Pool.  
/home/thomas/Projects/SParse-master/SParse/src/AppLib/include/61bTimer.h,  
50 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/S  
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Array/78Array.h,  
51 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/SkipList/S  
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/79BTree.c,  
51 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/String/Stri  
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/79BTree.h,  
51 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Times/Tim  
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/80CommonTypes.h,  
52 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/Types/Typ  
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/BTree/80Node.h,  
53 /home/thomas/Projects/SParse-master/SParse/src/CommonLib/include/BT  
/home/thomas/Projects/SParse-master/SParse/src/CommonLib/Debug/59Debug.c,



- IncludeInfo, [15](#)
- List, [15](#)
  - List\_compare, [16](#)
  - List\_copy, [16](#)
  - List\_forEach, [16](#)
  - List\_getSize, [17](#)
  - List\_insertHead, [17](#)
  - List\_insertTail, [17](#)
  - List\_merge, [17](#)
  - List\_new, [18](#)
- List\_compare
  - List, [16](#)
- List\_copy
  - List, [16](#)
- List\_forEach
  - List, [16](#)
- List\_getSize
  - List, [17](#)
- List\_insertHead
  - List, [17](#)
- List\_insertTail
  - List, [17](#)
- List\_merge
  - List, [17](#)
- List\_new
  - List, [18](#)
- main
  - main.c, [46](#)
- main.c
  - main, [46](#)
  - print\_usage, [46](#)
  - sighandler, [46](#)
  - start\_application, [46](#)
- Map, [18](#)
  - Map\_copy, [19](#)
  - Map\_getAll, [19](#)
  - Map\_insert, [19](#)
- Map\_copy
  - Map, [19](#)
- Map\_getAll
  - Map, [19](#)
- Map\_insert
  - Map, [19](#)
- MapEntry, [20](#)
- maxNbObjectAllocated
  - ObjectMgr, [24](#)
- Node, [20](#)
- Object, [20](#)
  - Object\_comp, [21](#)
  - Object\_copy, [21](#)
  - Object\_getRef, [21](#)
  - Object\_new, [21](#)
  - Object\_print, [22](#)
- Object\_comp
  - Object, [21](#)
- Object\_copy
  - Object, [21](#)
- Object\_getRef
  - Object, [21](#)
- Object\_new
  - Object, [21](#)
- Object\_print
  - Object, [22](#)
- ObjectInfo, [22](#)
- ObjectMgr, [22](#)
  - maxNbObjectAllocated, [24](#)
  - ObjectMgr\_allocate, [23](#)
  - ObjectMgr\_copy, [23](#)
  - ObjectMgr\_deallocate, [23](#)
  - ObjectMgr\_getRef, [24](#)
- ObjectMgr\_allocate
  - ObjectMgr, [23](#)
- ObjectMgr\_copy
  - ObjectMgr, [23](#)
- ObjectMgr\_deallocate
  - ObjectMgr, [23](#)
- ObjectMgr\_getRef
  - ObjectMgr, [24](#)
- OptionDefault, [24](#)
- OptionMgr, [25](#)
  - OptionMgr\_getRef, [25](#)
  - OptionMgr\_readFromCmdLine, [25](#)
- OptionMgr\_getRef
  - OptionMgr, [25](#)
- OptionMgr\_readFromCmdLine
  - OptionMgr, [25](#)
- Pool.c
  - Pool\_alloc, [66](#)
  - Pool\_allocInFile, [66](#)
  - Pool\_dealloc, [66](#)
  - Pool\_deallocInFile, [67](#)
  - Pool\_deallocInMemory, [67](#)
  - Pool\_free, [68](#)
  - Pool\_new, [68](#)
  - Pool\_newFromFile, [68](#)
  - Pool\_read, [69](#)
  - Pool\_readInFile, [69](#)
  - Pool\_readInMemory, [69](#)
  - Pool\_report, [70](#)
  - Pool\_reportInFile, [70](#)
  - Pool\_reportInMemory, [70](#)
  - Pool\_reportNbNodes, [71](#)
  - Pool\_reportSizeInBytes, [71](#)
  - Pool\_write, [71](#)
  - Pool\_writeInFile, [73](#)
  - Pool\_writeInMemory, [73](#)
- Pool\_alloc
  - Pool.c, [66](#)
- Pool\_allocInFile
  - Pool.c, [66](#)
- Pool\_dealloc
  - Pool.c, [66](#)
- Pool\_deallocInFile

- Pool.c, 67
- Pool\_deallocInMemory
  - Pool.c, 67
- Pool\_free
  - Pool.c, 68
- Pool\_new
  - Pool.c, 68
- Pool\_newFromFile
  - Pool.c, 68
- Pool\_read
  - Pool.c, 69
- Pool\_readInFile
  - Pool.c, 69
- Pool\_readInMemory
  - Pool.c, 69
- Pool\_report
  - Pool.c, 70
- Pool\_reportInFile
  - Pool.c, 70
- Pool\_reportInMemory
  - Pool.c, 70
- Pool\_reportNbNodes
  - Pool.c, 71
- Pool\_reportSizeInBytes
  - Pool.c, 71
- Pool\_write
  - Pool.c, 71
- Pool\_writeInFile
  - Pool.c, 73
- Pool\_writeInMemory
  - Pool.c, 73
- PoolCache, 26
- print\_usage
  - main.c, 46
- SdbMgr, 26
  - SdbMgr\_copy, 26
  - SdbMgr\_execute, 27
  - SdbMgr\_getRef, 27
- SdbMgr\_copy
  - SdbMgr, 26
- SdbMgr\_execute
  - SdbMgr, 27
- SdbMgr\_getRef
  - SdbMgr, 27
- SdbRequest, 27
  - SdbRequest\_delete, 28
  - SdbRequest\_execute, 28
  - SdbRequest\_new, 28
- SdbRequest\_delete
  - SdbRequest, 28
- SdbRequest\_execute
  - SdbRequest, 28
- SdbRequest\_new
  - SdbRequest, 28
- sighandler
  - main.c, 46
- SkipList.c
  - SkipList\_add, 76
  - SkipList\_compare, 76
  - SkipList\_copy, 77
  - SkipList\_delete, 77
  - SkipList\_new, 77
  - SkipList\_print, 78
  - SkipList\_remove, 78
- SkipList\_add
  - SkipList.c, 76
- SkipList\_compare
  - SkipList.c, 76
- SkipList\_copy
  - SkipList.c, 77
- SkipList\_delete
  - SkipList.c, 77
- SkipList\_new
  - SkipList.c, 77
- SkipList\_print
  - SkipList.c, 78
- SkipList\_remove
  - SkipList.c, 78
- SkipNode, 29
- SParse, 29
  - SParse\_delete, 29
  - SParse\_new, 30
  - SParse\_parse, 30
- SParse\_delete
  - SParse, 29
- SParse\_new
  - SParse, 30
- SParse\_parse
  - SParse, 30
- start\_application
  - main.c, 46
- String, 30
  - String\_compare, 31
  - String\_copy, 32
  - String\_getRef, 32
- String\_compare
  - String, 31
- String\_copy
  - String, 32
- String\_getRef
  - String, 32
- TestFileMgr, 32
- TestItem, 33
- testOptionMgr, 33
- TestSdbMgr, 33
- TestTimeMgr, 33
- TimeMgr, 34
  - TimeMgr\_copy, 34
  - TimeMgr\_delete, 34
  - TimeMgr\_getRef, 35
  - TimeMgr\_latchTime, 35
- TimeMgr\_copy
  - TimeMgr, 34
- TimeMgr\_delete
  - TimeMgr, 34
- TimeMgr\_getRef

- TimeMgr, [35](#)
- TimeMgr\_latchTime
  - TimeMgr, [35](#)
- Timer, [35](#)
  - Timer\_copy, [36](#)
  - Timer\_new, [36](#)
- Timer\_copy
  - Timer, [36](#)
- Timer\_new
  - Timer, [36](#)
- yy\_bs\_column
  - yy\_buffer\_state, [37](#)
- yy\_bs\_lineno
  - yy\_buffer\_state, [37](#)
- yy\_buffer\_stack
  - yyguts\_t, [38](#)
- yy\_buffer\_stack\_max
  - yyguts\_t, [38](#)
- yy\_buffer\_stack\_top
  - yyguts\_t, [39](#)
- yy\_buffer\_state, [37](#)
  - yy\_bs\_column, [37](#)
  - yy\_bs\_lineno, [37](#)
- yy\_trans\_info, [37](#)
- yyalloc, [38](#)
- yyguts\_t, [38](#)
  - yy\_buffer\_stack, [38](#)
  - yy\_buffer\_stack\_max, [38](#)
  - yy\_buffer\_stack\_top, [39](#)
- YYSTYPE, [39](#)