

Morpho 0.5

Generated by Doxygen 1.8.20



<b>1 file</b>	<b>1</b>
<b>2 index</b>	<b>3</b>
<b>3 language</b>	<b>5</b>
<b>4 matrix</b>	<b>9</b>
<b>5 sparse</b>	<b>11</b>
<b>6 syntax</b>	<b>13</b>
<b>7 Class Index</b>	<b>15</b>
7.1 Class List . . . . .	15
<b>8 File Index</b>	<b>17</b>
8.1 File List . . . . .	17
<b>9 Class Documentation</b>	<b>19</b>
9.1 _syntactreenode Struct Reference . . . . .	19
9.1.1 Detailed Description . . . . .	19
9.2 builtinclassentry Struct Reference . . . . .	19
9.2.1 Detailed Description . . . . .	20
9.3 callframe Struct Reference . . . . .	20
9.4 codeinfo Struct Reference . . . . .	20
9.5 compilenoderule Struct Reference . . . . .	21
9.5.1 Detailed Description . . . . .	21
9.6 debuginfo Struct Reference . . . . .	21
9.7 dictionary Struct Reference . . . . .	21
9.7.1 Detailed Description . . . . .	22
9.7.2 Member Data Documentation . . . . .	22
9.7.2.1 contents . . . . .	22
9.7.2.2 count . . . . .	22
9.8 dictionaryentry Struct Reference . . . . .	22
9.8.1 Detailed Description . . . . .	22
9.9 error Struct Reference . . . . .	23
9.9.1 Detailed Description . . . . .	23
9.10 errordefinition Struct Reference . . . . .	23
9.10.1 Detailed Description . . . . .	23
9.11 functionstate Struct Reference . . . . .	23
9.11.1 Detailed Description . . . . .	24
9.12 graylist Struct Reference . . . . .	24
9.12.1 Detailed Description . . . . .	24
9.13 keypress Struct Reference . . . . .	24
9.13.1 Detailed Description . . . . .	25

9.14 lexer Struct Reference	25
9.14.1 Detailed Description	25
9.14.2 Member Data Documentation	25
9.14.2.1 current	25
9.14.2.2 line	25
9.14.2.3 posn	26
9.15 linedit_stringlist Struct Reference	26
9.15.1 Detailed Description	26
9.16 linedit_syntaxcolordata Struct Reference	26
9.16.1 Detailed Description	26
9.17 linedit_token Struct Reference	27
9.17.1 Detailed Description	27
9.18 lineditor Struct Reference	27
9.18.1 Detailed Description	27
9.19 objectarray Struct Reference	28
9.20 objectbuiltinfunction Struct Reference	28
9.20.1 Detailed Description	28
9.21 objectclosure Struct Reference	28
9.22 objectdictionary Struct Reference	29
9.23 objectdokkey Struct Reference	29
9.23.1 Detailed Description	29
9.24 objectinstance Struct Reference	29
9.25 objectinvocation Struct Reference	30
9.26 objectmatrix Struct Reference	30
9.26.1 Detailed Description	30
9.27 objectmesh Struct Reference	30
9.28 objectrange Struct Reference	31
9.29 objectsparse Struct Reference	31
9.30 objectstring Struct Reference	31
9.30.1 Detailed Description	31
9.31 parser Struct Reference	32
9.31.1 Detailed Description	32
9.31.2 Member Data Documentation	32
9.31.2.1 err	32
9.31.2.2 left	32
9.31.2.3 nl	32
9.31.2.4 previous	33
9.31.2.5 tree	33
9.32 parserule Struct Reference	33
9.32.1 Detailed Description	33
9.33 registeralloc Struct Reference	33
9.33.1 Detailed Description	34

9.33.2 Member Data Documentation	34
9.33.2.1 iscaptured	34
9.33.2.2 scopedepth	34
9.33.2.3 symbol	34
9.34 scompiler Struct Reference	34
9.34.1 Detailed Description	35
9.35 scompilerlist Struct Reference	35
9.35.1 Detailed Description	35
9.36 slinedit_string Struct Reference	35
9.36.1 Detailed Description	36
9.37 subject Struct Reference	36
9.37.1 Detailed Description	36
9.38 subjectclass Struct Reference	36
9.39 subjectfunction Struct Reference	37
9.39.1 Detailed Description	37
9.40 subjecthelptopic Struct Reference	37
9.41 subjectupvalue Struct Reference	37
9.41.1 Member Data Documentation	38
9.41.1.1 closed	38
9.41.1.2 next	38
9.42 sparseccs Struct Reference	38
9.43 sparsedok Struct Reference	38
9.44 sprogram Struct Reference	39
9.44.1 Detailed Description	39
9.44.2 Member Data Documentation	39
9.44.2.1 global	39
9.44.2.2 info	39
9.44.2.3 nglobals	39
9.44.2.4 symboltable	40
9.45 svm Struct Reference	40
9.45.1 Detailed Description	40
9.45.2 Member Data Documentation	40
9.45.2.1 bound	40
9.45.2.2 frame	41
9.45.2.3 globals	41
9.45.2.4 gray	41
9.45.2.5 nextgc	41
9.45.2.6 objects	41
9.45.2.7 openupvalues	41
9.45.2.8 sp	41
9.45.2.9 stack	42
9.46 syntaxtree Struct Reference	42

9.47 token Struct Reference . . . . .	42
9.47.1 Detailed Description . . . . .	42
9.47.2 Member Data Documentation . . . . .	42
9.47.2.1 length . . . . .	43
9.47.2.2 line . . . . .	43
9.47.2.3 posn . . . . .	43
9.47.2.4 start . . . . .	43
9.48 upvalue Struct Reference . . . . .	43
9.48.1 Detailed Description . . . . .	43
9.48.2 Member Data Documentation . . . . .	43
9.48.2.1 reg . . . . .	44
9.49 value Struct Reference . . . . .	44
9.49.1 Detailed Description . . . . .	44
<b>10 File Documentation . . . . .</b>	<b>45</b>
10.1 build.h File Reference . . . . .	45
10.1.1 Detailed Description . . . . .	46
10.1.2 Macro Definition Documentation . . . . .	46
10.1.2.1 MORPHO_EPS . . . . .	46
10.1.2.2 MORPHO_LINALG_USE_ACCELERATE . . . . .	46
10.1.2.3 MORPHO_LINALG_USE_CSPARSE . . . . .	46
10.2 builtin/builtin.c File Reference . . . . .	47
10.2.1 Detailed Description . . . . .	48
10.2.2 Function Documentation . . . . .	48
10.2.2.1 builtin_addclass() . . . . .	48
10.2.2.2 builtin_addfunction() . . . . .	48
10.2.2.3 builtin_copysymboltable() . . . . .	49
10.2.2.4 builtin_findclass() . . . . .	49
10.2.2.5 builtin_findfunction() . . . . .	49
10.2.2.6 builtin_internsymbol() . . . . .	49
10.2.2.7 builtin_internsymbolascstring() . . . . .	49
10.2.2.8 builtin_printfunction() . . . . .	50
10.2.2.9 Dictionary_getindex() . . . . .	50
10.2.2.10 Dictionary_setindex() . . . . .	50
10.2.2.11 Object_class() . . . . .	50
10.2.2.12 Object_clone() . . . . .	50
10.2.2.13 Object_init() . . . . .	50
10.2.2.14 object_newrange() . . . . .	51
10.2.2.15 Object_serialize() . . . . .	51
10.2.2.16 Object_super() . . . . .	51
10.2.2.17 range_constructor() . . . . .	51
10.2.2.18 range_count() . . . . .	51

10.2.2.19 Range_enumerate()	51
10.2.2.20 range_iterate()	52
10.2.2.21 String_length()	52
10.2.3 Variable Documentation	52
10.2.3.1 objectveneer	52
10.3 builtin/builtin.h File Reference	52
10.3.1 Detailed Description	54
10.3.2 Macro Definition Documentation	54
10.3.2.1 MORPHO_BEGINCLASS	54
10.3.2.2 MORPHO_ENDCLASS	54
10.3.2.3 MORPHO_GETARG	54
10.3.2.4 MORPHO_GETBUILTINFUNCTION	54
10.3.2.5 MORPHO_ISBUILTINFUNCTION	55
10.3.2.6 MORPHO_RAISE	55
10.3.2.7 MORPHO_RAISEVARGS	55
10.3.2.8 MORPHO_SELF	55
10.3.3 Typedef Documentation	55
10.3.3.1 builtinfunction	55
10.3.3.2 builtinfunctionflags	55
10.3.4 Function Documentation	56
10.3.4.1 builtin_addclass()	56
10.3.4.2 builtin_addfunction()	56
10.3.4.3 builtin_copysymboltable()	56
10.3.4.4 builtin_findclass()	57
10.3.4.5 builtin_findfunction()	57
10.3.4.6 builtin_internsymbol()	57
10.3.4.7 builtin_internsymbolascstring()	57
10.3.4.8 builtin_printfunction()	57
10.4 builtin/file.c File Reference	57
10.4.1 Detailed Description	58
10.4.2 Function Documentation	58
10.4.2.1 File_close()	58
10.4.2.2 File_eof()	59
10.4.2.3 File_free()	59
10.4.2.4 File_init()	59
10.4.2.5 File_lines()	59
10.4.2.6 File_readchar()	59
10.4.2.7 file_readintovarray()	60
10.4.2.8 File_readline()	60
10.4.2.9 file_readlineintovarray()	60
10.4.2.10 file_readlineusingvarray()	60
10.4.2.11 File_write()	60

10.5 builtin/file.h File Reference	60
10.5.1 Detailed Description	61
10.5.2 Function Documentation	61
10.5.2.1 file_readintovarray()	61
10.5.2.2 file_readlineintovarray()	62
10.6 builtin/functions.c File Reference	62
10.6.1 Detailed Description	62
10.6.2 Function Documentation	62
10.6.2.1 builtin_clock()	62
10.6.2.2 builtin_exp()	63
10.6.2.3 builtin_random()	63
10.6.2.4 builtin_randomint()	63
10.6.2.5 builtin_system()	63
10.7 builtin/functions.h File Reference	63
10.7.1 Detailed Description	64
10.8 datastructures/dictionary.h File Reference	64
10.8.1 Detailed Description	65
10.8.2 Function Documentation	65
10.8.2.1 dictionary_clear()	65
10.8.2.2 dictionary_freecontents()	66
10.8.2.3 dictionary_get()	66
10.8.2.4 dictionary_getintern()	66
10.8.2.5 dictionary_init()	68
10.8.2.6 dictionary_insert()	68
10.8.2.7 dictionary_insertintern()	69
10.8.2.8 dictionary_intern()	69
10.8.2.9 dictionary_remove()	69
10.9 datastructures/matrix.c File Reference	71
10.9.1 Detailed Description	72
10.9.2 Function Documentation	72
10.9.2.1 matrix_add()	72
10.9.2.2 matrix_constructor()	72
10.9.2.3 matrix_divl()	72
10.9.2.4 matrix_divs()	73
10.9.2.5 matrix_getarraydimensions()	73
10.9.2.6 matrix_getarrayelement()	73
10.9.2.7 matrix_getelement()	73
10.9.2.8 matrix_mul()	74
10.9.2.9 matrix_setelement()	74
10.9.2.10 matrix_sub()	74
10.9.2.11 matrix_trace()	74
10.9.2.12 matrix_transpose()	75



10.9.2.13 <code>object_matrixfromarray()</code> . . . . .	75
10.9.2.14 <code>object_newmatrix()</code> . . . . .	75
10.10 <code>datastructures/matrix.h</code> File Reference . . . . .	75
10.10.1 Detailed Description . . . . .	76
10.10.2 Macro Definition Documentation . . . . .	76
10.10.2.1 <code>MATRIX_ISSMALL</code> . . . . .	77
10.10.2.2 <code>MORPHO_LINALG_USE_LAPACK</code> . . . . .	77
10.10.3 Function Documentation . . . . .	77
10.10.3.1 <code>matrix_add()</code> . . . . .	77
10.10.3.2 <code>matrix_divl()</code> . . . . .	77
10.10.3.3 <code>matrix_divs()</code> . . . . .	77
10.10.3.4 <code>matrix_getarraydimensions()</code> . . . . .	78
10.10.3.5 <code>matrix_getarrayelement()</code> . . . . .	79
10.10.3.6 <code>matrix_getelement()</code> . . . . .	79
10.10.3.7 <code>matrix_mul()</code> . . . . .	79
10.10.3.8 <code>matrix_setelement()</code> . . . . .	80
10.10.3.9 <code>matrix_sub()</code> . . . . .	80
10.10.3.10 <code>matrix_trace()</code> . . . . .	80
10.10.3.11 <code>matrix_transpose()</code> . . . . .	80
10.11 <code>datastructures/object.c</code> File Reference . . . . .	80
10.11.1 Detailed Description . . . . .	82
10.11.2 Function Documentation . . . . .	82
10.11.2.1 <code>object_arrayfromlist()</code> . . . . .	82
10.11.2.2 <code>object_arrayfromvalueindices()</code> . . . . .	82
10.11.2.3 <code>object_arrayfromvarrayvalue()</code> . . . . .	82
10.11.2.4 <code>object_arrayindicestoelement()</code> . . . . .	82
10.11.2.5 <code>object_arrayinit()</code> . . . . .	83
10.11.2.6 <code>object_arrayvaluestoindices()</code> . . . . .	83
10.11.2.7 <code>object_concatenatestring()</code> . . . . .	83
10.11.2.8 <code>object_dictionary()</code> . . . . .	84
10.11.2.9 <code>object_free()</code> . . . . .	84
10.11.2.10 <code>object_functionaddprototype()</code> . . . . .	84
10.11.2.11 <code>object_functionclear()</code> . . . . .	84
10.11.2.12 <code>object_functiongetconstanttable()</code> . . . . .	85
10.11.2.13 <code>object_getarrayelement()</code> . . . . .	85
10.11.2.14 <code>object_getfunctionname()</code> . . . . .	85
10.11.2.15 <code>object_getfunctionparent()</code> . . . . .	85
10.11.2.16 <code>object_init()</code> . . . . .	85
10.11.2.17 <code>object_new()</code> . . . . .	86
10.11.2.18 <code>object_newarray()</code> . . . . .	86
10.11.2.19 <code>object_newclosure()</code> . . . . .	86
10.11.2.20 <code>object_newdictionary()</code> . . . . .	86

10.11.2.21 object_newinvocation()	87
10.11.2.22 object_newupvalue()	87
10.11.2.23 object_print()	87
10.11.2.24 object_printtobuffer()	87
10.11.2.25 object_setarrayelement()	87
10.11.2.26 object_size()	88
10.11.2.27 object_stringfromcstring()	88
10.11.2.28 object_stringfromvarraychar()	88
10.11.2.29 object_upvalueinit()	88
10.12 datastructures/object.h File Reference	89
10.12.1 Detailed Description	92
10.12.2 Macro Definition Documentation	92
10.12.2.1 MORPHO_GETARRAY	92
10.12.2.2 MORPHO_GETCLASS	92
10.12.2.3 MORPHO_GETCLOSURE	92
10.12.2.4 MORPHO_GETCLOSUREFUNCTION	92
10.12.2.5 MORPHO_GETDICTIONARY	92
10.12.2.6 MORPHO_GETDOKKEY	93
10.12.2.7 MORPHO_GETDOKKEYROW	93
10.12.2.8 MORPHO_GETFUNCTION	93
10.12.2.9 MORPHO_GETINSTANCE	93
10.12.2.10 MORPHO_GETINVOCATION	93
10.12.2.11 MORPHO_GETMATRIX	93
10.12.2.12 MORPHO_GETMESH	93
10.12.2.13 MORPHO_GETOBJECTHASH	94
10.12.2.14 MORPHO_GETOBJECTTYPE	94
10.12.2.15 MORPHO_GETRANGE	94
10.12.2.16 MORPHO_GETSPARSE	94
10.12.2.17 MORPHO_GETSUPERCLASS	94
10.12.2.18 MORPHO_GETUPVALUE	94
10.12.2.19 MORPHO_ISARRAY	94
10.12.2.20 MORPHO_ISCLASS	95
10.12.2.21 MORPHO_ISCLOSURE	95
10.12.2.22 MORPHO_ISDICTIONARY	95
10.12.2.23 MORPHO_ISDOKKEY	95
10.12.2.24 MORPHO_ISFUNCTION	95
10.12.2.25 MORPHO_ISINSTANCE	95
10.12.2.26 MORPHO_ISINVOCATION	95
10.12.2.27 MORPHO_ISMATRIX	96
10.12.2.28 MORPHO_ISMESH	96
10.12.2.29 MORPHO_ISRANGE	96
10.12.2.30 MORPHO_ISSPARSE	96

10.12.2.31 MORPHO_ISSTRING . . . . .	96
10.12.2.32 MORPHO_ISUPVALUE . . . . .	96
10.12.2.33 MORPHO_SETOBJECTHASH . . . . .	96
10.12.2.34 MORPHO_STATICDOKKEY . . . . .	97
10.12.2.35 MORPHO_STATICMATRIX . . . . .	97
10.12.2.36 MORPHO_STATICSTRING . . . . .	97
10.12.2.37 MORPHO_STATICSTRINGWITHLENGTH . . . . .	97
10.12.3 Typedef Documentation . . . . .	97
10.12.3.1 objectfunction . . . . .	97
10.12.4 Enumeration Type Documentation . . . . .	98
10.12.4.1 objecttype . . . . .	98
10.12.5 Function Documentation . . . . .	98
10.12.5.1 object_arrayfromlist() . . . . .	98
10.12.5.2 object_arrayfromvalueindices() . . . . .	98
10.12.5.3 object_arrayfromvarrayvalue() . . . . .	98
10.12.5.4 object_arrayindicestoelement() . . . . .	98
10.12.5.5 object_arrayvaluestoindices() . . . . .	99
10.12.5.6 object_concatenatestring() . . . . .	99
10.12.5.7 object_free() . . . . .	100
10.12.5.8 object_functionaddprototype() . . . . .	100
10.12.5.9 object_functionclear() . . . . .	100
10.12.5.10 object_functiongetconstanttable() . . . . .	100
10.12.5.11 object_getarrayelement() . . . . .	101
10.12.5.12 object_getfunctionname() . . . . .	101
10.12.5.13 object_getfunctionparent() . . . . .	101
10.12.5.14 object_init() . . . . .	101
10.12.5.15 object_matrixfromarray() . . . . .	101
10.12.5.16 object_new() . . . . .	102
10.12.5.17 object_newarray() . . . . .	102
10.12.5.18 object_newclosure() . . . . .	102
10.12.5.19 object_newdictionary() . . . . .	102
10.12.5.20 object_newinvocation() . . . . .	103
10.12.5.21 object_newmatrix() . . . . .	103
10.12.5.22 object_newmesh() . . . . .	103
10.12.5.23 object_newrange() . . . . .	103
10.12.5.24 object_newsparse() . . . . .	103
10.12.5.25 object_newupvalue() . . . . .	104
10.12.5.26 object_print() . . . . .	104
10.12.5.27 object_printtobuffer() . . . . .	104
10.12.5.28 object_setarrayelement() . . . . .	104
10.12.5.29 object_size() . . . . .	104
10.12.5.30 object_stringfromcstring() . . . . .	105

10.12.5.31	<a href="#">object_stringfromvarraychar()</a>	105
10.12.5.32	<a href="#">object_upvalueinit()</a>	105
10.13	<a href="#">datastructures/sparse.c File Reference</a>	106
10.13.1	<a href="#">Detailed Description</a>	107
10.13.2	<a href="#">Function Documentation</a>	107
10.13.2.1	<a href="#">DEFINE_VARRAY()</a>	107
10.13.2.2	<a href="#">object_newsparse()</a>	107
10.13.2.3	<a href="#">object_sparsefromarray()</a>	107
10.13.2.4	<a href="#">sparse_add()</a>	108
10.13.2.5	<a href="#">sparse_checkformat()</a>	108
10.13.2.6	<a href="#">sparse_clear()</a>	108
10.13.2.7	<a href="#">sparse_constructor()</a>	109
10.13.2.8	<a href="#">sparse_div()</a>	109
10.13.2.9	<a href="#">sparse_getelement()</a>	109
10.13.2.10	<a href="#">Sparse_getindex()</a>	109
10.13.2.11	<a href="#">sparse_mul()</a>	110
10.13.2.12	<a href="#">sparse_removeformat()</a>	110
10.13.2.13	<a href="#">sparse_setelement()</a>	110
10.13.2.14	<a href="#">Sparse_setindex()</a>	110
10.13.2.15	<a href="#">sparse_size()</a>	111
10.13.2.16	<a href="#">sparseccs_clear()</a>	111
10.13.2.17	<a href="#">sparseccs_doktoocs()</a>	111
10.13.2.18	<a href="#">sparseccs_get()</a>	111
10.13.2.19	<a href="#">sparseccs_getrowindices()</a>	111
10.13.2.20	<a href="#">sparseccs_init()</a>	112
10.13.2.21	<a href="#">sparseccs_print()</a>	112
10.13.2.22	<a href="#">sparseccs_resize()</a>	112
10.13.2.23	<a href="#">sparseccs_set()</a>	112
10.13.2.24	<a href="#">sparsedok_clear()</a>	113
10.13.2.25	<a href="#">sparsedok_get()</a>	113
10.13.2.26	<a href="#">sparsedok_init()</a>	113
10.13.2.27	<a href="#">sparsedok_insert()</a>	113
10.13.2.28	<a href="#">sparsedok_print()</a>	113
10.13.2.29	<a href="#">sparsedok_remove()</a>	114
10.13.2.30	<a href="#">sparsedok_setdimensions()</a>	114
10.14	<a href="#">datastructures/sparse.h File Reference</a>	114
10.14.1	<a href="#">Detailed Description</a>	115
10.14.2	<a href="#">Function Documentation</a>	115
10.14.2.1	<a href="#">sparse_add()</a>	115
10.14.2.2	<a href="#">sparse_clear()</a>	116
10.14.2.3	<a href="#">sparse_getelement()</a>	116
10.14.2.4	<a href="#">sparse_mul()</a>	116

10.14.2.5 <code>sparse_setelement()</code>	117
10.14.2.6 <code>sparse_size()</code>	117
10.14.2.7 <code>sparseccs_clear()</code>	117
10.14.2.8 <code>sparseccs_doktoccs()</code>	117
10.14.2.9 <code>sparseccs_get()</code>	117
10.14.2.10 <code>sparseccs_getrowindices()</code>	118
10.14.2.11 <code>sparseccs_init()</code>	119
10.14.2.12 <code>sparseccs_resize()</code>	119
10.14.2.13 <code>sparsedok_clear()</code>	119
10.14.2.14 <code>sparsedok_get()</code>	119
10.14.2.15 <code>sparsedok_init()</code>	120
10.14.2.16 <code>sparsedok_insert()</code>	120
10.14.2.17 <code>sparsedok_remove()</code>	120
10.14.2.18 <code>sparsedok_setdimensions()</code>	120
10.15 <code>datastructures/syntaxtree.c</code> File Reference	121
10.15.1 Detailed Description	121
10.15.2 Function Documentation	121
10.15.2.1 <code>syntaxtree_addnode()</code>	121
10.15.2.2 <code>syntaxtree_clear()</code>	122
10.15.2.3 <code>syntaxtree_nodefromindx()</code>	122
10.16 <code>datastructures/syntaxtree.h</code> File Reference	122
10.16.1 Detailed Description	123
10.16.2 Function Documentation	123
10.16.2.1 <code>syntaxtree_addnode()</code>	124
10.16.2.2 <code>syntaxtree_clear()</code>	124
10.16.2.3 <code>syntaxtree_nodefromindx()</code>	124
10.17 <code>datastructures/value.c</code> File Reference	124
10.17.1 Detailed Description	125
10.17.2 Function Documentation	125
10.17.2.1 <code>value_promotenumberslist()</code>	125
10.17.2.2 <code>varray_valuefind()</code>	125
10.17.2.3 <code>varray_valuefindsame()</code>	126
10.18 <code>datastructures/value.h</code> File Reference	126
10.18.1 Detailed Description	127
10.18.2 Macro Definition Documentation	128
10.18.2.1 <code>MORPHO_GETINTEGERVALUE</code>	128
10.18.2.2 <code>MORPHO_GETTYPE</code>	128
10.18.2.3 <code>MORPHO_INTEGERTOFLOAT</code>	128
10.18.2.4 <code>MORPHO_ISNIL</code>	128
10.18.2.5 <code>MORPHO_NIL</code>	128
10.18.3 Enumeration Type Documentation	128
10.18.3.1 <code>valuetype</code>	129

10.18.4 Function Documentation	129
10.18.4.1 value_promotenumberslist()	129
10.18.4.2 varray_valuefind()	129
10.18.4.3 varray_valuefindsame()	130
10.19 datastructures/varray.c File Reference	130
10.19.1 Detailed Description	130
10.19.2 Function Documentation	131
10.19.2.1 varray_powerof2ceiling()	131
10.20 datastructures/varray.h File Reference	131
10.20.1 Detailed Description	131
10.20.2 Macro Definition Documentation	132
10.20.2.1 DECLARE_VARRAY	132
10.20.3 Function Documentation	132
10.20.3.1 varray_powerof2ceiling()	132
10.21 geometry/mesh.c File Reference	133
10.21.1 Detailed Description	133
10.21.2 Function Documentation	133
10.21.2.1 mesh_addelementwithvertices()	133
10.21.2.2 mesh_checkconnectivity()	134
10.21.2.3 mesh_constructor()	134
10.21.2.4 mesh_getconnectivityelement()	134
10.21.2.5 mesh_load()	134
10.21.2.6 object_newmesh()	134
10.22 geometry/mesh.h File Reference	135
10.22.1 Detailed Description	135
10.23 interface/cli.c File Reference	135
10.23.1 Detailed Description	136
10.23.2 Function Documentation	136
10.23.2.1 cli_complete()	136
10.23.2.2 cli_disassemblewithsrc()	136
10.23.2.3 cli_help()	137
10.23.2.4 cli_lex()	137
10.23.2.5 cli_reporterror()	137
10.23.2.6 cli_run()	137
10.23.3 Variable Documentation	137
10.23.3.1 cli_tokencolors	137
10.24 interface/cli.h File Reference	138
10.24.1 Detailed Description	138
10.24.2 Function Documentation	138
10.24.2.1 cli_run()	139
10.25 interface/help.c File Reference	139
10.25.1 Detailed Description	139

10.25.2 Function Documentation	140
10.25.2.1 help_cleartopic()	140
10.25.2.2 help_display()	140
10.25.2.3 help_finalize()	140
10.25.2.4 help_initialize()	140
10.25.2.5 help_load()	140
10.25.2.6 help_newtopic()	141
10.25.2.7 help_parsetag()	141
10.25.2.8 help_parsetopiclevel()	141
10.25.2.9 help_parsetopicname()	141
10.25.2.10 help_querylength()	141
10.25.2.11 help_search()	142
10.25.2.12 help_searchpath()	142
10.26 interface/help.h File Reference	142
10.26.1 Detailed Description	143
10.26.2 Function Documentation	143
10.26.2.1 help_display()	143
10.26.2.2 help_finalize()	143
10.26.2.3 help_initialize()	144
10.26.2.4 help_querylength()	144
10.26.2.5 help_search()	144
10.27 interface/linedit.c File Reference	144
10.27.1 Detailed Description	147
10.27.2 Macro Definition Documentation	147
10.27.2.1 LINEDIT_CODESTRINGSIZE	147
10.27.2.2 LINEDIT_DEBUGKEYPRESS	147
10.27.2.3 LINEDIT_UNSUPPORTEDBUFFER	147
10.27.3 Enumeration Type Documentation	147
10.27.3.1 keycodes	147
10.27.3.2 keytype	148
10.27.4 Function Documentation	148
10.27.4.1 linedit()	148
10.27.4.2 linedit_addsuggestion()	148
10.27.4.3 linedit_advanceposition()	148
10.27.4.4 linedit_advancesuggestions()	149
10.27.4.5 linedit_aresuggestionsavailable()	149
10.27.4.6 linedit_autocomplete()	149
10.27.4.7 linedit_checksupport()	149
10.27.4.8 linedit_clear()	149
10.27.4.9 linedit_cstrcasecmp()	150
10.27.4.10 linedit_cstring()	150
10.27.4.11 linedit_currentsuggestion()	150

10.27.4.12	<a href="#">linedit_disablerawmode()</a>	150
10.27.4.13	<a href="#">linedit_displaywithstyle()</a>	150
10.27.4.14	<a href="#">linedit_displaywithsyntaxcoloring()</a>	151
10.27.4.15	<a href="#">linedit_enablerawmode()</a>	151
10.27.4.16	<a href="#">linedit_generatesuggestions()</a>	151
10.27.4.17	<a href="#">linedit_getmode()</a>	151
10.27.4.18	<a href="#">linedit_historyadd()</a>	152
10.27.4.19	<a href="#">linedit_historyadvance()</a>	152
10.27.4.20	<a href="#">linedit_historyclear()</a>	152
10.27.4.21	<a href="#">linedit_historyselect()</a>	152
10.27.4.22	<a href="#">linedit_init()</a>	152
10.27.4.23	<a href="#">linedit_newstring()</a>	152
10.27.4.24	<a href="#">linedit_noterminal()</a>	153
10.27.4.25	<a href="#">linedit_processkeypress()</a>	153
10.27.4.26	<a href="#">linedit_readkey()</a>	153
10.27.4.27	<a href="#">linedit_refreshline()</a>	153
10.27.4.28	<a href="#">linedit_setmode()</a>	153
10.27.4.29	<a href="#">linedit_setposition()</a>	153
10.27.4.30	<a href="#">linedit_setprompt()</a>	154
10.27.4.31	<a href="#">linedit_showstring()</a>	154
10.27.4.32	<a href="#">linedit_stringaddcharacter()</a>	154
10.27.4.33	<a href="#">linedit_stringaddcstring()</a>	154
10.27.4.34	<a href="#">linedit_stringclear()</a>	155
10.27.4.35	<a href="#">linedit_stringinit()</a>	155
10.27.4.36	<a href="#">linedit_stringinsert()</a>	155
10.27.4.37	<a href="#">linedit_stringlistadd()</a>	155
10.27.4.38	<a href="#">linedit_stringlistclear()</a>	155
10.27.4.39	<a href="#">linedit_stringlistinit()</a>	155
10.27.4.40	<a href="#">linedit_stringlistremove()</a>	156
10.27.4.41	<a href="#">linedit_stringlistselect()</a>	156
10.27.4.42	<a href="#">linedit_stringresize()</a>	156
10.27.4.43	<a href="#">linedit_supported()</a>	157
10.27.4.44	<a href="#">linedit_syntaxcolor()</a>	157
10.27.4.45	<a href="#">linedit_syntaxcolorstring()</a>	157
10.27.4.46	<a href="#">linedit_atendoffline()</a>	157
10.27.5	<a href="#">Variable Documentation</a>	157
10.27.5.1	<a href="#">terminit</a>	158
10.28	<a href="#">interface/linedit.h File Reference</a>	158
10.28.1	<a href="#">Detailed Description</a>	159
10.28.2	<a href="#">Typedef Documentation</a>	159
10.28.2.1	<a href="#">linedit_completer</a>	159
10.28.2.2	<a href="#">linedit_string</a>	159



10.28.2.3 <code>linedit_tokenizer</code>	159
10.28.3 Enumeration Type Documentation	160
10.28.3.1 <code>linedit_color</code>	160
10.28.3.2 <code>lineditormode</code>	160
10.28.4 Function Documentation	160
10.28.4.1 <code>linedit()</code>	160
10.28.4.2 <code>linedit_addsuggestion()</code>	161
10.28.4.3 <code>linedit_autocomplete()</code>	161
10.28.4.4 <code>linedit_clear()</code>	161
10.28.4.5 <code>linedit_displaywithstyle()</code>	162
10.28.4.6 <code>linedit_displaywithsyntaxcoloring()</code>	162
10.28.4.7 <code>linedit_init()</code>	162
10.28.4.8 <code>linedit_setprompt()</code>	162
10.28.4.9 <code>linedit_syntaxcolor()</code>	163
10.29 <code>main.c</code> File Reference	163
10.29.1 Detailed Description	163
10.30 <code>morpho.h</code> File Reference	164
10.30.1 Detailed Description	165
10.30.2 Function Documentation	165
10.30.2.1 <code>morpho_compile()</code>	165
10.30.2.2 <code>morpho_defineerror()</code>	165
10.30.2.3 <code>morpho_disassemble()</code>	166
10.30.2.4 <code>morpho_finalize()</code>	166
10.30.2.5 <code>morpho_freecompiler()</code>	166
10.30.2.6 <code>morpho_freevm()</code>	166
10.30.2.7 <code>morpho_geterrorid()</code>	167
10.30.2.8 <code>morpho_initialize()</code>	167
10.30.2.9 <code>morpho_interpret()</code>	167
10.30.2.10 <code>morpho_newcompiler()</code>	167
10.30.2.11 <code>morpho_newvm()</code>	167
10.30.2.12 <code>morpho_runtimeerror()</code>	168
10.30.2.13 <code>morpho_stacktrace()</code>	168
10.30.2.14 <code>morpho_writeerrorwithid()</code>	168
10.31 <code>utils/common.c</code> File Reference	168
10.31.1 Detailed Description	169
10.31.2 Function Documentation	169
10.31.2.1 <code>morpho_powerof2ceiling()</code>	169
10.31.2.2 <code>morpho_printvalue()</code>	170
10.31.2.3 <code>morpho_strdup()</code>	170
10.32 <code>utils/common.h</code> File Reference	170
10.32.1 Detailed Description	171
10.32.2 Macro Definition Documentation	171

10.32.2.1 EQUAL	171
10.32.2.2 MORPHO_ISEQUAL	172
10.32.2.3 MORPHO_ISSAME	172
10.32.3 Function Documentation	172
10.32.3.1 morpho_powerof2ceiling()	172
10.32.3.2 morpho_printvalue()	172
10.32.3.3 morpho_strdup()	173
10.33 utils/error.c File Reference	173
10.33.1 Detailed Description	174
10.33.2 Function Documentation	174
10.33.2.1 error_clear()	174
10.33.2.2 error_finalize()	174
10.33.2.3 error_init()	174
10.33.2.4 error_initialize()	175
10.33.2.5 morpho_defineerror()	175
10.33.2.6 morpho_getdefinitionfromid()	175
10.33.2.7 morpho_geterrorid()	175
10.33.2.8 morpho_writeerrorwithid()	176
10.33.2.9 morpho_writeerrorwithidvalist()	176
10.34 utils/error.h File Reference	176
10.34.1 Detailed Description	180
10.34.2 Macro Definition Documentation	180
10.34.2.1 BSD_EX_SOFTWARE	180
10.34.2.2 ERROR_ISRUNTIMEERROR	180
10.34.2.3 ERROR_POSNUNIDENTIFIABLE	180
10.34.2.4 ERROR_SHOULDCONTINUE	181
10.34.2.5 ERROR_SUCCEEDED	181
10.34.2.6 UNREACHABLE	181
10.34.3 Enumeration Type Documentation	181
10.34.3.1 errorcategory	181
10.34.4 Function Documentation	181
10.34.4.1 error_clear()	182
10.34.4.2 error_finalize()	182
10.34.4.3 error_init()	182
10.34.4.4 error_initialize()	182
10.34.4.5 morpho_defineerror()	182
10.34.4.6 morpho_geterrorid()	183
10.34.4.7 morpho_writeerrorwithid()	183
10.34.4.8 morpho_writeerrorwithidvalist()	183
10.35 utils/memory.h File Reference	184
10.35.1 Detailed Description	184
10.35.2 Macro Definition Documentation	184

10.35.2.1 MORPHO_FREE . . . . .	184
10.35.2.2 MORPHO_MALLOC . . . . .	185
10.35.2.3 MORPHO_REALLOC . . . . .	185
10.35.3 Function Documentation . . . . .	185
10.35.3.1 morpho_allocate() . . . . .	185
10.36 utils/parse.c File Reference . . . . .	185
10.36.1 Detailed Description . . . . .	186
10.36.2 Macro Definition Documentation . . . . .	186
10.36.2.1 UNUSED . . . . .	186
10.36.3 Function Documentation . . . . .	187
10.36.3.1 lex() . . . . .	187
10.36.3.2 lex_init() . . . . .	187
10.36.3.3 lex_recordtoken() . . . . .	187
10.36.3.4 parse() . . . . .	189
10.36.3.5 parse_init() . . . . .	189
10.36.3.6 parse_stringtovaluearray() . . . . .	189
10.36.3.7 parse_synchronize() . . . . .	190
10.36.3.8 syntaxtree_addnode() . . . . .	190
10.37 utils/parse.h File Reference . . . . .	190
10.37.1 Detailed Description . . . . .	192
10.37.2 Macro Definition Documentation . . . . .	192
10.37.2.1 TOKEN_BLANK . . . . .	192
10.37.3 Enumeration Type Documentation . . . . .	192
10.37.3.1 tokentype . . . . .	192
10.37.4 Function Documentation . . . . .	192
10.37.4.1 lex() . . . . .	192
10.37.4.2 lex_init() . . . . .	193
10.37.4.3 parse() . . . . .	193
10.37.4.4 parse_init() . . . . .	193
10.37.4.5 parse_stringtovaluearray() . . . . .	194
10.38 utils/random.c File Reference . . . . .	194
10.38.1 Detailed Description . . . . .	195
10.38.2 Function Documentation . . . . .	195
10.38.2.1 random_double() . . . . .	195
10.38.2.2 random_initialize() . . . . .	195
10.38.2.3 random_int() . . . . .	195
10.38.2.4 splitmix64_seed() . . . . .	195
10.39 vm/compile.c File Reference . . . . .	196
10.39.1 Detailed Description . . . . .	197
10.39.2 Function Documentation . . . . .	197
10.39.2.1 compile_finalize() . . . . .	197
10.39.2.2 compile_initialize() . . . . .	197

10.39.2.3 compiler_addupvalue()	197
10.39.2.4 compiler_beginscope()	197
10.39.2.5 compiler_clear()	197
10.39.2.6 compiler_closure()	198
10.39.2.7 compiler_copyglobals()	198
10.39.2.8 compiler_currentscope()	198
10.39.2.9 compiler_endscope()	198
10.39.2.10 compiler_findclass()	198
10.39.2.11 compiler_fstackclear()	199
10.39.2.12 compiler_fstackinit()	199
10.39.2.13 compiler_init()	199
10.39.2.14 compiler_stripend()	199
10.39.2.15 morpho_compile()	199
10.39.2.16 morpho_freecompile()	200
10.39.2.17 morpho_newcompiler()	200
10.39.3 Variable Documentation	200
10.39.3.1 noderules	200
10.40 vm/compile.h File Reference	200
10.40.1 Detailed Description	202
10.40.2 Typedef Documentation	202
10.40.2.1 compiler_nodefn	202
10.40.2.2 compilerlist	202
10.40.3 Enumeration Type Documentation	202
10.40.3.1 functiontype	202
10.40.4 Function Documentation	202
10.40.4.1 compiler_clear()	202
10.40.4.2 compiler_init()	203
10.41 vm/core.h File Reference	203
10.41.1 Detailed Description	205
10.41.2 Macro Definition Documentation	205
10.41.2.1 DECODE_A	205
10.41.2.2 DECODE_B	205
10.41.2.3 DECODE_Bx	205
10.41.2.4 DECODE_C	205
10.41.2.5 DECODE_F	205
10.41.2.6 DECODE_ISBCONSTANT	206
10.41.2.7 DECODE_ISCCONSTANT	206
10.41.2.8 DECODE_OP	206
10.41.2.9 DECODE_sBx	206
10.41.2.10 ENCODE	206
10.41.2.11 ENCODE_BYTE	206
10.41.2.12 ENCODE_DOUBLE	207

10.41.2.13 ENCODE_EMPTYOPERAND	207
10.41.2.14 ENCODE_LONG	207
10.41.2.15 ENCODE_LONGFLAGS	207
10.41.2.16 ENCODE_SINGLE	207
10.41.2.17 ENCODEC	208
10.41.3 Typedef Documentation	208
10.41.3.1 instruction	208
10.41.4 Function Documentation	208
10.41.4.1 compile_finalize()	208
10.41.4.2 compile_initialize()	209
10.42 vm/opcodes.h File Reference	209
10.42.1 Detailed Description	209
10.43 vm/vm.c File Reference	209
10.43.1 Detailed Description	211
10.43.2 Macro Definition Documentation	211
10.43.2.1 CHECKCMPTYPE	211
10.43.2.2 DISASSEMBLE_OPAB	212
10.43.2.3 DISASSEMBLE_OPABC	212
10.43.2.4 DISASSEMBLE_OPACB	212
10.43.2.5 DISASSEMBLE_OPB	212
10.43.2.6 DISASSEMBLE_SHOWA [1/2]	212
10.43.2.7 DISASSEMBLE_SHOWA [2/2]	213
10.43.2.8 INTERPRET_LOOP	213
10.43.3 Function Documentation	213
10.43.3.1 morpho_disassemble()	214
10.43.3.2 morpho_finalize()	214
10.43.3.3 morpho_freemv()	214
10.43.3.4 morpho_getdebugfromindx()	214
10.43.3.5 morpho_initialize()	214
10.43.3.6 morpho_interpret()	214
10.43.3.7 morpho_newvm()	215
10.43.3.8 morpho_runtimeerror()	215
10.43.3.9 morpho_stacktrace()	215
10.43.3.10 program_bindobject()	215
10.43.3.11 program_getentry()	216
10.43.3.12 program_internsymbol()	216
10.43.3.13 program_setentry()	216
10.43.3.14 vm_bindobject()	216
10.43.3.15 vm_collectgarbage()	216
10.43.3.16 vm_disassembleinstruction()	217
10.43.3.17 vm_freeobjects()	217
10.43.3.18 vm_gcmarray()	217

---

10.43.3.19 vm_gcmarkdictionary()	217
10.43.3.20 vm_gcmarkobject()	217
10.43.3.21 vm_gcmarkroots()	218
10.43.3.22 vm_gcmarkvalue()	218
10.43.3.23 vm_gcsweep()	218
10.43.3.24 vm_gctrace()	218
10.43.3.25 vm_runtimeerror()	218
10.44 vm/vm.h File Reference	219
10.44.1 Detailed Description	219
10.44.2 Function Documentation	219
10.44.2.1 morpho_finalize()	219
10.44.2.2 morpho_initialize()	220
10.44.2.3 program_bindobject()	220
10.44.2.4 program_getentry()	220
10.44.2.5 program_internsymbol()	220
10.44.2.6 program_setentry()	220
10.44.2.7 vm_disassembleinstruction()	220
10.44.2.8 vm_freeobjects()	221
<b>Index</b>	<b>223</b>

# Chapter 1

## file

[comment]: # File class help [version]: # 0.5

#File [tag]: # File

The File class provides the capability to read from and write to files, or to obtain the contents of a file in convenient formats.

To open a file, create a File object with the filename as the argument

```
var f = File("myfile.txt")
```

which opens "myfile.txt" for *reading*. To open a file for writing or appending, you need to provide a mode selector

```
var g = File("myfile.txt", "write")
```

or

```
var g = File("myfile.txt", "append")
```

Once the file is open, you can then read or write by calling appropriate methods:

```
f.lines()           // reads the contents of the file into an array of lines.
f.readline()        // reads a single line
f.readchar()         // reads a single character.
f.write(string)      // writes the arguments to the file.
```

After you're done with the file, close it with

```
f.close()
```

#File.lines [tag]: # lines

Returns the contents of a file as an array of strings; each element corresponds to a single line.

#File.readline [tag]: # readline

Reads a single line from a file; returns the result as a string.

#File.readchar [tag]: # readchar

Reads a single character from a file; returns the result as a string.

#File.write [tag]: # write

Writes to a file.

#File.close [tag]: # close

Closes an open file





## Chapter 2

# index

[comment]: # Morpho language help file [version]: # 0.5

#Help [tag]: # help

Welcome to the morpho interactive help system. To get help about a topic called `topicname`, type

```
help topicname
```

Possible topics include language keywords like `class`, `fn` and `for`, built in classes like `Graphics` and `File` or information about functions like `exp` and `random`.

Some topics have additional subtopics: to access these type

```
help topic subtopic
```

For example, to get help on a method for a particular class, you could type

```
help Classname methodname
```

You can also use `?` as a shorthand synonym for `help`

```
? topic
```



## Chapter 3

# language

[comment]: # Morpho language help file [version]: # 0.5

#Functions [tag]: # fn [tag]: # func

A function in Morpho is defined with the `fn` keyword, followed by the function's name, a list of parameters enclosed in parentheses, and the body of the function in curly braces. This example computes the square of a number:

```
fn sqr(x) {  
  return x*x  
}
```

#Return [tag]: # return

The `return` keyword is used to exit from a function, optionally passing a given value back to the caller. `return` can be used anywhere within a function. The below example calculates the `n` th Fibonacci number,

```
fn fib(n) {  
  if (n<2) return n  
  return fib(n-1) + fib(n-2)  
}
```

by returning early if `n<2`, otherwise returning the result by recursively calling itself.

#Variables [tag]: # var

Variables are defined using the `var` keyword followed by the variable name:

```
var a
```

Optionally, an initial assignment may be given:

```
var a = 1
```

Variables defined in a block of code are visible only within that block, so

```
var greeting = "Hello"  
{  
  var greeting = "Goodbye"  
  print greeting  
}  
print greeting
```

will print

*Goodbye Hello*

Multiple variables can be defined at once by separating them with commas

```
var a, b=2, c[2]=[1,2]
```

where each can have its own initializer (or not).

**#Classes [tag]: # class**

Classes are defined using the `class` keyword followed by the name of the class. The definition includes methods that the class responds to. The special `init` method is called whenever an object is created.

```
class Cake {
  init(type) {
    self.type = type
  }

  eat() {
    print "A delicious "+type+" cake"
  }
}
```

Objects are created by calling the class as if it was a function:

```
var c = Cake("carrot")
```

Methods are called using the `.` operator:

```
c.eat()
```

**#If [tag]: # if**

If allows you to selectively execute a section of code depending on whether a condition is met. The simplest version looks like this:

```
if (x<1) print x
```

where the body of the loop, `print x`, is only executed if `x` is less than 1. The body can be a code block to accomodate longer sections of code:

```
if (x<1) {
  ... // do something
}
```

If you want to choose between two alternatives, use `else`:

```
if (a==b) {
  // do something
} else {
  // this code is executed only if the condition is false
}
```

You can even chain multiple tests together like this:

---

```

if (a==b) {
    // option 1
} else if (a==c) {
    // option 2
} else {
    // something else
}

```

#While [tag]: # while

While loops

#For [tag]: # for

For loops allow you to repeatedly execute a section of code. They come in two versions: the simpler version looks like this,

```
for (i in 1..5) print i
```

which prints the numbers 1 to 5 in turn. The variable `i` is the *loop variable*, which takes on a different value each iteration. `1..5` is a range, which denotes a sequence of numbers. The *body* of the loop, `print i`, is the code to be repeatedly executed.

If you want your loop variable to count in increments other than 1, you can specify a stepsize in the range:

```
for (i in 1..5:2) print i
      ^step
```

Ranges need not be integer:

```
for (i in 0.1..0.5:0.1) print i
```

You can also replace the range with other kinds of collection object to loop over their contents:

```
var a = Matrix([1,2,3,4])
for (x in a) print x
```

Morpho also provides a second form of `for` loop similar to that in C:

```
for (var i=0; i<5; i+=1) { print i }
      ^start  ^test ^inc. ^body
```

which is executed as follows: *start*: the variable `i` is initially set to zero. *test*: before each iteration, the test is evaluated. If the test is `false`, the loop terminates. *body*: the body of the loop is executed. *inc*: the variable `i` is increased by 1.

This kind of loop is very flexible as you can include any code that you like in each of the sections.

#Indexing [tag]: # [ [tag]: # ] [tag]: # index [tag]: # subscript

Indexing

#Example

This is *very* important, this is *underlined* and `var` is code.



## Chapter 4

# matrix

[comment]: # Matrix class help [version]: # 0.5

#Matrix [tag]: # Matrix

The Matrix class provides support for matrices. A matrix can be initialized with a given size,

```
var a = Matrix(nrows,ncols)
```

where all elements are initially set to zero. Alternatively, a matrix can be created from an array,

```
var a = Matrix([[1,2], [3,4]])
```

You can create a column vector like this,

```
var v = Matrix([1,2])
```

Once a matrix is created, you can use all the regular arithmetic operators with matrix operands, e.g.

```
a+b  
a*b
```

The division operator is used to solve a linear system, e.g.

```
var a = Matrix([[1,2],[3,4]])  
var b = Matrix([1,2])
```

```
print b/a
```

yields the solution to the system  $a \cdot x = b$ .

#MtrxIncmptbl [tag]: # MtrxIncmptbl

This error occurs when an arithmetic operation is performed on two 'incompatible' matrices. For example, two matrices must have the same dimensions, i.e. the same number of rows and columns, to be added or subtracted,

```
var a = Matrix([[1,2],[3,4]])  
var b = Matrix([[1]])  
print a+b // generates a 'MtrxIncmptbl' error.
```

Or to be multiplied together, the number of columns of the left hand matrix must equal the number of rows of the right hand matrix.

```
var a = Matrix([[1,2],[3,4]])  
var b = Matrix([1,2])  
print a*b // ok  
print b*a // generates a 'MtrxIncmptbl' error.
```





## Chapter 5

### sparse

[comment]: # Sparse class help [version]: # 0.5

#Sparse [tag]: # Sparse

The Sparse class provides support for sparse matrices. An empty sparse matrix can be initialized with a given size,

```
var a = Sparse(nrows,ncols)
```

Alternatively, a matrix can be created from an array of triplets,

```
var a = Matrix([[row, col, value] ...])
```

For example

```
var a = Matrix([[0,0,2], [1,1,-2]])
```

creates the matrix

```
[ 2 0 ]  
[ 0 -2 ]
```

Once a sparse matrix is created, you can use all the regular arithmetic operators with matrix operands, e.g.

```
a+b  
a*b
```



## Chapter 6

# syntax

[comment]: # Morpho language help file [version]: # 0.5

#Syntax [tag]: # syntax

Morpho provides a flexible object oriented language similar to other languages in the C family (like C++, Java and Javascript) with a simplified syntax.

Morpho programs are stored as plain text with the .morpho file extension. A program can be run from the command line by typing

```
morpho program.morpho
```

#Comments [tag]: # comment [tag]: # Comments [tag]: # // [tag]: # /\* [tag]: # \*/ Two types of comment are available. The first type is called a 'line comment' whereby text after // on the same line is ignored by the interpreter.

```
a.dosomething() // A comment
```

Longer 'block' comments can be created by placing text between /\* and \*/. Newlines are ignored

```
/* This
   is
   a longer comment */
```

In contrast to C, these comments can be nested

```
/* A nested /* comment */ */
```

enabling the programmer to quickly comment out a section of code.

#Symbols [tag]: # symbols [tag]: # names

Symbols are used to refer to named entities, including variables, classes, functions etc. Symbols must begin with a letter or underscore \_ as the first character and may include letters or numbers as the remainder. Symbols are case sensitive.

```
asymbol
_alsoasymbol
another_symbol
EvenThis123
YET_ANOTHER_SYMBOL
```

Classes are typically given names with an initial capital letter. Variable names are usually all lower case.

#Newlines [tag]: # newlines

Morpho accepts newlines in place of a semicolon to end a statement.

```
var a = 1; //
```

#Blocks

#Precedence



## Chapter 7

# Class Index

### 7.1 Class List

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

<a href="#">_syntaxtreenode</a>	
A node on the syntax tree is defined by a value and indices of the left and right elements . . .	19
<a href="#">builtinclassentry</a> . . . . .	19
<a href="#">callframe</a> . . . . .	20
<a href="#">codeinfo</a> . . . . .	20
<a href="#">compilenoderule</a>	
A compilenoderule rule will be defined for each syntax tree node type, providing a function to compile the node . . . . .	21
<a href="#">debuginfo</a> . . . . .	21
<a href="#">dictionary</a>	
Dictionary data structure that maps keys to values . . . . .	21
<a href="#">dictionaryentry</a>	
A single dictionary entry . . . . .	22
<a href="#">error</a>	
A static container for error messages . . . . .	23
<a href="#">errordefinition</a>	
Definition of an error message . . . . .	23
<a href="#">functionstate</a> . . . . .	23
<a href="#">graylist</a> . . . . .	24
<a href="#">keypress</a> . . . . .	24
<a href="#">lexer</a>	
Store the current configuration of a lexer . . . . .	25
<a href="#">linedit_stringlist</a> . . . . .	26
<a href="#">linedit_syntaxcolordata</a> . . . . .	26
<a href="#">linedit_token</a> . . . . .	27
<a href="#">lineditor</a> . . . . .	27
<a href="#">objectarray</a> . . . . .	28
<a href="#">objectbuiltinfunction</a> . . . . .	28
<a href="#">objectclosure</a> . . . . .	28
<a href="#">objectdictionary</a> . . . . .	29
<a href="#">objectdokkey</a> . . . . .	29
<a href="#">objectinstance</a> . . . . .	29
<a href="#">objectinvocation</a> . . . . .	30
<a href="#">objectmatrix</a> . . . . .	30
<a href="#">objectmesh</a> . . . . .	30

<a href="#">objectrange</a>	31
<a href="#">objectspare</a>	31
<a href="#">objectstring</a>	31
<a href="#">parser</a>	
A structure that defines the state of a parser	32
<a href="#">parserule</a>	
A parse rule will be defined for each token, providing functions to parse the token if it is encountered in the prefix or infix positions. The parse rule also defines the precedence	33
<a href="#">registeralloc</a>	33
<a href="#">scompiler</a>	
A structure that stores the state of a compiler	34
<a href="#">scompilerlist</a>	35
<a href="#">slinedit_string</a>	35
<a href="#">subject</a>	36
<a href="#">subjectclass</a>	36
<a href="#">subjectfunction</a>	37
<a href="#">subjecthelptopic</a>	37
<a href="#">subjectupvalue</a>	37
<a href="#">sparseccs</a>	38
<a href="#">sparsedok</a>	38
<a href="#">sprogram</a>	
Morpho code program and associated data	39
<a href="#">svm</a>	
A Morpho virtual machine and its current state	40
<a href="#">syntaxtree</a>	42
<a href="#">token</a>	42
<a href="#">upvalue</a>	43
<a href="#">value</a>	
The unboxed value type	44

## Chapter 8

# File Index

### 8.1 File List

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

<a href="#">build.h</a>	Define constants that choose how Morpho is built . . . . .	45
<a href="#">main.c</a>	Main entry point . . . . .	163
<a href="#">morpho.h</a>	Define public interface to Morpho . . . . .	164
<a href="#">builtin/builtin.c</a>	Morpho built in functions and classes . . . . .	47
<a href="#">builtin/builtin.h</a>	Morpho built in functions and classes . . . . .	52
<a href="#">builtin/file.c</a>	Built in class to provide file input and output . . . . .	57
<a href="#">builtin/file.h</a>	Built in class to provide file input and output . . . . .	60
<a href="#">builtin/functions.c</a>	Built in functions . . . . .	62
<a href="#">builtin/functions.h</a>	Built in functions . . . . .	63
<a href="#">datastructures/dictionary.h</a>	Dictionary (hashtable) data structure . . . . .	64
<a href="#">datastructures/matrix.c</a>	Veneer class over the objectmatrix type that interfaces with blas and lapack . . . . .	71
<a href="#">datastructures/matrix.h</a>	Veneer class over the objectmatrix type that interfaces with blas and lapack . . . . .	75
<a href="#">datastructures/object.c</a>	Provide functionality for extended and mutable data types . . . . .	80
<a href="#">datastructures/object.h</a>	Provide functionality for extended and mutable data types . . . . .	89
<a href="#">datastructures/sparse.c</a>	Veneer class over the objectsparse type that provides sparse matrices . . . . .	106
<a href="#">datastructures/sparse.h</a>	Veneer class over the objectsparse type that provides sparse matrices . . . . .	114
<a href="#">datastructures/syntaxtree.c</a>	Syntax tree data structure for morpho . . . . .	121
<a href="#">datastructures/syntaxtree.h</a>	Syntax tree data structure for morpho . . . . .	122

datastructures/ <a href="#">value.c</a>	
Fundamental data type for morpho	124
datastructures/ <a href="#">value.h</a>	
Fundamental data type for morpho	126
datastructures/ <a href="#">varray.c</a>	
Dynamically resizing array (varray) data structure	130
datastructures/ <a href="#">varray.h</a>	
Dynamically resizing array (varray) data structure	131
geometry/ <a href="#">mesh.c</a>	
Mesh class and associated functionality	133
geometry/ <a href="#">mesh.h</a>	
Mesh class and associated functionality	135
interface/ <a href="#">cli.c</a>	
Command line interface	135
interface/ <a href="#">cli.h</a>	
Command line interface	138
interface/ <a href="#">help.c</a>	
Interactive help system	139
interface/ <a href="#">help.h</a>	
Interactive help system	142
interface/ <a href="#">linedit.c</a>	
A line editor with history, autocomplete and syntax highlighting	144
interface/ <a href="#">linedit.h</a>	
A simple line editor with history, prediction and syntax highlighting	158
utils/ <a href="#">common.c</a>	
Common types, data structures and functions for the Morpho VM	168
utils/ <a href="#">common.h</a>	
Morpho virtual machine	170
utils/ <a href="#">error.c</a>	
Morpho error handling	173
utils/ <a href="#">error.h</a>	
Morpho error handling	176
utils/ <a href="#">memory.h</a>	
Morpho memory management	184
utils/ <a href="#">parse.c</a>	
Lexer and parser	185
utils/ <a href="#">parse.h</a>	
Lexer and parser	190
utils/ <a href="#">random.c</a>	
Random number generation	194
utils/ <a href="#">random.h</a>	??
vm/ <a href="#">compile.c</a>	
Compiles raw input to Morpho instructions	196
vm/ <a href="#">compile.h</a>	
Compiles raw input to Morpho instructions	200
vm/ <a href="#">core.h</a>	
Data types for core Morpho components	203
vm/ <a href="#">opcodes.h</a>	
Morpho opcodes	209
vm/ <a href="#">vm.c</a>	
Morpho virtual machine	209
vm/ <a href="#">vm.h</a>	
The Morpho virtual machine	219



## Chapter 9

# Class Documentation

### 9.1 `_syntactreenode` Struct Reference

A node on the syntax tree is defined by a value and indices of the left and right elements.

```
#include <syntactree.h>
```

#### Public Attributes

- `syntactreenodetype` type
- `value` content
- int `line`
- int `posn`
- `syntactreeindx` left
- `syntactreeindx` right

#### 9.1.1 Detailed Description

A node on the syntax tree is defined by a value and indices of the left and right elements.

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

- `datastructures/syntactree.h`

### 9.2 `builtinclassentry` Struct Reference

```
#include <builtin.h>
```

#### Public Types

- enum { `BUILTIN_METHOD`, `BUILTIN_PROPERTY` }

## Public Attributes

- enum builtinclassentry:: { ... } **type**
- char \* **name**
- [builtinfunctionflags](#) **flags**
- [builtinfunction](#) **function**

### 9.2.1 Detailed Description

A type used to store the entries of a built in class

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

- [builtin/builtin.h](#)

## 9.3 callframe Struct Reference

### Public Attributes

- [objectfunction](#) \* **function**
- [objectclosure](#) \* **closure**
- [value](#) \* **reg**
- [instruction](#) \* **pc**
- unsigned int **stackcount**

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

- [vm/core.h](#)

## 9.4 codeinfo Struct Reference

### Public Types

- enum { **REGISTER**, **CONSTANT**, **UPVALUE**, **GLOBAL** }

### Public Attributes

- enum codeinfo:: { ... } **returntype**
- [registerindx](#) **dest**
- unsigned int **ninstructions**

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

- [vm/compile.h](#)

## 9.5 compilenoderule Struct Reference

A compilenoderule rule will be defined for each syntax tree node type, providing a function to compile the node.

```
#include <compile.h>
```

### Public Attributes

- [compiler\\_nodefn](#) **nodefn**

### 9.5.1 Detailed Description

A compilenoderule rule will be defined for each syntax tree node type, providing a function to compile the node.

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

- [vm/compile.h](#)

## 9.6 debuginfo Struct Reference

### Public Attributes

- unsigned int **ninstr**
- [objectfunction](#) \* **function**
- unsigned int **line**
- unsigned int **posn**

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

- [vm/core.h](#)

## 9.7 dictionary Struct Reference

dictionary data structure that maps keys to values

```
#include <dictionary.h>
```

### Public Attributes

- unsigned int **capacity**
- unsigned int **count**
- [dictionaryentry](#) \* **contents**

### 9.7.1 Detailed Description

dictionary data structure that maps keys to values

### 9.7.2 Member Data Documentation

#### 9.7.2.1 contents

```
dictionaryentry* dictionary::contents
```

number of items in the dictionary

#### 9.7.2.2 count

```
unsigned int dictionary::count
```

capacity of the dictionary

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

- datastructures/[dictionary.h](#)

## 9.8 dictionaryentry Struct Reference

A single dictionary entry.

```
#include <dictionary.h>
```

### Public Attributes

- [value](#) key
- [value](#) val

### 9.8.1 Detailed Description

A single dictionary entry.

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

- datastructures/[dictionary.h](#)

## 9.9 error Struct Reference

A static container for error messages.

```
#include <error.h>
```

### Public Attributes

- [errorcategory](#) **cat**
- [errorid](#) **id**
- **int line**
- **int posn**
- **char msg** [[MORPHO\\_ERRORSTRINGSIZE](#)]

### 9.9.1 Detailed Description

A static container for error messages.

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

- [utils/error.h](#)

## 9.10 errordefinition Struct Reference

Definition of an error message.

```
#include <error.h>
```

### Public Attributes

- [errorcategory](#) **cat**
- **char \* msg**

### 9.10.1 Detailed Description

Definition of an error message.

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

- [utils/error.h](#)

## 9.11 functionstate Struct Reference

```
#include <compile.h>
```

## Public Attributes

- [objectfunction](#) \* **func**
- [functiontype](#) **type**
- [varray\\_registeralloc](#) **registers**
- [varray\\_upvalue](#) **upvalues**
- unsigned int **nreg**
- unsigned int **scopeddepth**
- bool **inargs**

### 9.11.1 Detailed Description

This structure tracks compiler information for the current function.

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

- [vm/compile.h](#)

## 9.12 graylist Struct Reference

```
#include <core.h>
```

## Public Attributes

- unsigned int **graycount**
- unsigned int **graycapacity**
- [object](#) \*\* **list**

### 9.12.1 Detailed Description

Gray list for garbage collection

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

- [vm/core.h](#)

## 9.13 keypress Struct Reference

## Public Attributes

- [keytype](#) **type**
- char **c**

### 9.13.1 Detailed Description

A single keypress event obtained and processed by the terminal

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

- interface/[linedit.c](#)

## 9.14 lexer Struct Reference

Store the current configuration of a lexer.

```
#include <parse.h>
```

### Public Attributes

- const char \* **start**
- const char \* [current](#)
- int [line](#)
- int [posn](#)

### 9.14.1 Detailed Description

Store the current configuration of a lexer.

### 9.14.2 Member Data Documentation

#### 9.14.2.1 current

```
const char* lexer::current
```

Starting point to lex

#### 9.14.2.2 line

```
int lexer::line
```

Current point

### 9.14.2.3 posn

```
int lexer::posn
```

Line number

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

- [utils/parse.h](#)

## 9.15 linedit\_stringlist Struct Reference

```
#include <linedit.h>
```

### Public Attributes

- int **posn**
- [linedit\\_string](#) \* **first**

### 9.15.1 Detailed Description

A list of strings

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

- [interface/linedit.h](#)

## 9.16 linedit\_syntaxcolordata Struct Reference

```
#include <linedit.h>
```

### Public Attributes

- [linedit\\_tokenizer](#) **tokenizer**
- unsigned int **ncols**
- bool **lexwarning**
- [linedit\\_color](#) **col** []

### 9.16.1 Detailed Description

Structure to hold all information related to syntax coloring

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

- [interface/linedit.h](#)



## 9.17 `linedit_token` Struct Reference

```
#include <linedit.h>
```

### Public Attributes

- unsigned int **type**
- char \* **start**
- size\_t **length**

#### 9.17.1 Detailed Description

lineditor tokens

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

- interface/[linedit.h](#)

## 9.18 `lineditor` Struct Reference

```
#include <linedit.h>
```

### Public Attributes

- [lineditormode](#) **mode**
- int **posn**
- int **sposn**
- [linedit\\_string](#) **prompt**
- [linedit\\_string](#) **current**
- [linedit\\_string](#) **clipboard**
- [linedit\\_stringlist](#) **history**
- [linedit\\_stringlist](#) **suggestions**
- [linedit\\_syntaxcolordata](#) \* **color**
- [linedit\\_completer](#) **completer**

#### 9.18.1 Detailed Description

Holds all state information needed for a line editor

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

- interface/[linedit.h](#)

## 9.19 objectarray Struct Reference

### Public Attributes

- [object](#) **obj**
- unsigned int **dimensions**
- unsigned int **nelements**
- [value](#) **data** []

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

- datastructures/[object.h](#)

## 9.20 objectbuiltinfunction Struct Reference

```
#include <builtin.h>
```

### Public Attributes

- [object](#) **obj**
- [value](#) **name**
- [builtinfunctionflags](#) **flags**
- [builtinfunction](#) **function**

### 9.20.1 Detailed Description

A built in function object

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

- builtin/[builtin.h](#)

## 9.21 objectclosure Struct Reference

### Public Attributes

- [object](#) **obj**
- [objectfunction](#) \* **func**
- int **nupvalues**
- [objectupvalue](#) \* **upvalues** []

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

- datastructures/[object.h](#)

## 9.22 objectdictionary Struct Reference

### Public Attributes

- [object](#) **obj**
- [dictionary](#) **dict**

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

- [datastructures/object.h](#)

## 9.23 objectdokkey Struct Reference

```
#include <object.h>
```

### Public Attributes

- [object](#) **obj**
- unsigned int **row**
- unsigned int **col**

### 9.23.1 Detailed Description

The dictionary of keys format uses this special object type to store indices, enabling use of the existing dictionary type.

#### Warning

These are for internal use only and should never be returned to user code

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

- [datastructures/object.h](#)

## 9.24 objectinstance Struct Reference

### Public Attributes

- [object](#) **obj**
- [objectclass](#) \* **klass**
- [dictionary](#) **fields**

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

- [datastructures/object.h](#)

## 9.25 objectinvocation Struct Reference

### Public Attributes

- [object](#) **obj**
- [value](#) **receiver**
- [value](#) **method**

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

- [datastructures/object.h](#)

## 9.26 objectmatrix Struct Reference

```
#include <object.h>
```

### Public Attributes

- [object](#) **obj**
- unsigned int **nrows**
- unsigned int **ncols**
- double \* **elements**
- double **matrixdata** []

### 9.26.1 Detailed Description

Matrices are a purely numerical collection type oriented toward linear algebra. Elements are stored in column-major format, i.e.  $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$  is stored ( 1, 3, 2, 4 ) in memory. This is for compatibility with standard linear algebra packages

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

- [datastructures/object.h](#)

## 9.27 objectmesh Struct Reference

### Public Attributes

- [object](#) **obj**
- unsigned int **dim**
- [objectmatrix](#) \* **vert**
- [objectarray](#) \* **conn**

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

- [datastructures/object.h](#)

## 9.28 objectrange Struct Reference

### Public Attributes

- [object](#) **obj**
- unsigned int **nsteps**
- [value](#) **start**
- [value](#) **end**
- [value](#) **step**

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

- [datastructures/object.h](#)

## 9.29 objectsparse Struct Reference

### Public Attributes

- [object](#) **obj**
- [sparsedok](#) **dok**
- [sparseccs](#) **ccs**

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

- [datastructures/object.h](#)

## 9.30 objectstring Struct Reference

```
#include <object.h>
```

### Public Attributes

- [object](#) **obj**
- `size_t` **length**
- `char *` **string**
- `char` **stringdata** []

### 9.30.1 Detailed Description

A string object

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

- [datastructures/object.h](#)

## 9.31 parser Struct Reference

A structure that defines the state of a parser.

```
#include <parse.h>
```

### Public Attributes

- [token](#) **current**
- [token](#) **previous**
- [syntaxtreeindx](#) **left**
- [lexer](#) \* **lex**
- [syntaxtree](#) \* **tree**
- [error](#) \* **err**
- **bool** [nl](#)

### 9.31.1 Detailed Description

A structure that defines the state of a parser.

### 9.31.2 Member Data Documentation

#### 9.31.2.1 [err](#)

```
error* parser::err
```

Syntax tree receiving output

#### 9.31.2.2 [left](#)

```
syntaxtreeindx parser::left
```

The previous token

#### 9.31.2.3 [nl](#)

```
bool parser::nl
```

Error structure to output errors to

#### 9.31.2.4 previous

`token` `parser::previous`

The current token

#### 9.31.2.5 tree

`syntaxtree*` `parser::tree`

Lexer to use

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

- [utils/parse.h](#)

## 9.32 parserule Struct Reference

A parse rule will be defined for each token, providing functions to parse the token if it is encountered in the prefix or infix positions. The parse rule also defines the precedence.

```
#include <parse.h>
```

### Public Attributes

- [parsefunction](#) `prefix`
- [parsefunction](#) `infix`
- [precedence](#) `precedence`

#### 9.32.1 Detailed Description

A parse rule will be defined for each token, providing functions to parse the token if it is encountered in the prefix or infix positions. The parse rule also defines the precedence.

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

- [utils/parse.h](#)

## 9.33 registeralloc Struct Reference

```
#include <compile.h>
```

## Public Attributes

- bool **isallocated**
- bool [iscaptured](#)
- unsigned int [scopeddepth](#)
- [value](#) [symbol](#)

### 9.33.1 Detailed Description

This structure tracks the contents of each register as the function is being compiled.

### 9.33.2 Member Data Documentation

#### 9.33.2.1 iscaptured

```
bool registeralloc::iscaptured
```

Whether the register has been allocated

#### 9.33.2.2 scopeddepth

```
unsigned int registeralloc::scopeddepth
```

Whether the register becomes an upvalue

#### 9.33.2.3 symbol

```
value registeralloc::symbol
```

Scope depth at which the register was allocated

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

- [vm/compile.h](#)

## 9.34 scompiler Struct Reference

A structure that stores the state of a compiler.

```
#include <compile.h>
```



## Public Attributes

- [lexer](#) **lex**
- [parser](#) **parse**
- [syntaxtree](#) **tree**
- [error](#) **err**
- [dictionary](#) **globals**
- [functionstate](#) **fstack** [[MORPHO\\_CALLFRAMESTACKSIZE](#)]
- [indx](#) **fstackp**
- [objectfunction](#) \* **prevfunction**
- [objectclass](#) \* **currentclass**
- [compilerlist](#) \* **currentlist**
- [compilerlist](#) \* **prevlist**
- [syntaxtreenode](#) \* **currentmethod**
- [program](#) \* **out**
- [struct scompiler](#) \* **parent**

### 9.34.1 Detailed Description

A structure that stores the state of a compiler.

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

- [vm/compile.h](#)

## 9.35 scompilerlist Struct Reference

```
#include <compile.h>
```

## Public Attributes

- [varray\\_value](#) **entries**
- [struct scompilerlist](#) \* **next**

### 9.35.1 Detailed Description

This structure holds a list as it is being created

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

- [vm/compile.h](#)

## 9.36 slinedit\_string Struct Reference

```
#include <linedit.h>
```

## Public Attributes

- `size_t capacity`
- `size_t length`
- `char * string`
- `struct slinedit\_string * next`

### 9.36.1 Detailed Description

lineditor strings

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

- `interface/linedit.h`

## 9.37 `object` Struct Reference

```
#include <object.h>
```

## Public Types

- `enum { OBJECT_ISUNMANAGED, OBJECT_ISUNMARKED, OBJECT_ISMARKED }`

## Public Attributes

- `objecttype type`
- `enum object:: { ... } status`
- `hash hsh`
- `struct subject * next`

### 9.37.1 Detailed Description

Simplest object

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

- `datastructures/object.h`

## 9.38 `subjectclass` Struct Reference

## Public Attributes

- `object obj`
- `struct subjectclass * superclass`
- `value name`
- `dictionary methods`

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

- `datastructures/object.h`

## 9.39 subjectfunction Struct Reference

```
#include <object.h>
```

### Public Attributes

- [object](#) **obj**
- int **nargs**
- [value](#) **name**
- int **entry**
- struct [subjectfunction](#) \* **parent**
- int **nupvalues**
- int **nregs**
- [varray\\_value](#) **konst**
- [varray\\_varray\\_upvalue](#) **prototype**

### 9.39.1 Detailed Description

A function object

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

- [datastructures/object.h](#)

## 9.40 subjecthelptopic Struct Reference

### Public Attributes

- [object](#) **obj**
- char \* **topic**
- char \* **file**
- long int **location**
- struct [subjecthelptopic](#) \* **parent**
- struct [subjecthelptopic](#) \* **next**
- [dictionary](#) **subtopics**

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

- [interface/help.h](#)

## 9.41 subjectupvalue Struct Reference

### Public Attributes

- [object](#) **obj**
- [value](#) \* **location**
- [value](#) **closed**
- struct [subjectupvalue](#) \* **next**

### 9.41.1 Member Data Documentation

#### 9.41.1.1 closed

`value` `subjectupvalue::closed`

Pointer to the location of the upvalue

#### 9.41.1.2 next

struct `subjectupvalue*` `subjectupvalue::next`

Closed value of the upvalue

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

- [datastructures/object.h](#)

## 9.42 sparseccs Struct Reference

### Public Attributes

- int **nentries**
- int **nrows**
- int **ncols**
- int \* **cptr**
- int \* **rix**
- double \* **values**

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

- [datastructures/object.h](#)

## 9.43 sparsedok Struct Reference

### Public Attributes

- int **nrows**
- int **ncols**
- [dictionary](#) **dict**
- [objectdokkey](#) \* **keys**

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

- [datastructures/object.h](#)

## 9.44 sprogram Struct Reference

Morpho code program and associated data.

```
#include <core.h>
```

### Public Attributes

- varray\_instruction **code**
- varray\_debuginfo **info**
- [objectfunction](#) \* **global**
- unsigned int **nglobals**
- [object](#) \* **boundlist**
- [dictionary](#) **symboltable**

#### 9.44.1 Detailed Description

Morpho code program and associated data.

#### 9.44.2 Member Data Documentation

##### 9.44.2.1 global

```
objectfunction* sprogram::global
```

Information about how the code connects to the source

##### 9.44.2.2 info

```
varray_debuginfo sprogram::info
```

Compiled instructions

##### 9.44.2.3 nglobals

```
unsigned int sprogram::nglobals
```

Pseudofunction containing global data

#### 9.44.2.4 symboltable

`dictionary` `sprogram::symboltable`

Linked list of static objects bound to this program

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

- `vm/core.h`

### 9.45 svm Struct Reference

A Morpho virtual machine and its current state.

```
#include <core.h>
```

#### Public Attributes

- `program` \* `current`
- `varray_value` `globals`
- `varray_value` `stack`
- `callframe` `frame` [`MORPHO_CALLFRAMESTACKSIZE`]
- `value` \* `sp`
- `callframe` \* `fp`
- `error` `err`
- `object` \* `objects`
- `graylist` `gray`
- `size_t` `bound`
- `size_t` `nextgc`
- `objectupvalue` \* `openupvalues`

#### 9.45.1 Detailed Description

A Morpho virtual machine and its current state.

#### 9.45.2 Member Data Documentation

##### 9.45.2.1 bound

`size_t` `svm::bound`

Graylist for garbage collection

### 9.45.2.2 frame

`callframe` `svm::frame[MORPHO_CALLFRAMESTACKSIZE]`

The stack

### 9.45.2.3 globals

`varray_value` `svm::globals`

The current program being executed

### 9.45.2.4 gray

`graylist` `svm::gray`

Linked list of objects

### 9.45.2.5 nextgc

`size_t` `svm::nextgc`

Estimated size of bound bytes

### 9.45.2.6 objects

`object*` `svm::objects`

An error struct that will be filled out when an error occurs

### 9.45.2.7 openupvalues

`objectupvalue*` `svm::openupvalues`

Next garbage collection threshold

### 9.45.2.8 sp

`value*` `svm::sp`

The call frame stack

### 9.45.2.9 stack

```
varray_value svm::stack
```

Global variables

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

- [vm/core.h](#)

## 9.46 syntaxtree Struct Reference

### Public Attributes

- varray\_syntactreenode **tree**
- syntaxtreeindx **entry**

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

- [datastructures/syntaxtree.h](#)

## 9.47 token Struct Reference

```
#include <parse.h>
```

### Public Attributes

- [tokentype](#) **type**
- const char \* [start](#)
- unsigned int [length](#)
- int [line](#)
- int [posn](#)

### 9.47.1 Detailed Description

A token

### 9.47.2 Member Data Documentation



### 9.47.2.1 length

```
unsigned int token::length
```

Start of the token

### 9.47.2.2 line

```
int token::line
```

Its length

### 9.47.2.3 posn

```
int token::posn
```

Source line

### 9.47.2.4 start

```
const char* token::start
```

Type of the token

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

- [utils/parse.h](#)

## 9.48 upvalue Struct Reference

```
#include <object.h>
```

### Public Attributes

- bool **islocal**
- indx [reg](#)

### 9.48.1 Detailed Description

Each upvalue

### 9.48.2 Member Data Documentation

### 9.48.2.1 reg

```
indx upvalue::reg
```

Set if the upvalue is local to this function

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

- [datastructures/object.h](#)

## 9.49 value Struct Reference

The unboxed value type.

```
#include <value.h>
```

### Public Attributes

- [valuetype](#) type

- 

```
union {  
    int integer  
    double real  
    bool boolean  
    struct object * obj  
} as
```

### 9.49.1 Detailed Description

The unboxed value type.

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

- [datastructures/value.h](#)

## Chapter 10

# File Documentation

### 10.1 build.h File Reference

Define constants that choose how Morpho is built.

#### Macros

- #define **MORPHO\_HELPDIRECTORY** "/usr/local/share/morpho/help"
- #define **MORPHO\_COLORTERMINAL**  
*Use coloring in output.*
- #define **MORPHO\_STRINGINTERPOLATION**  
*Support string interpolation.*
- #define **MORPHO\_NEWLINETERMINATORS**  
*Newlines as statement terminators.*
- #define **MORPHO\_EPS** 1e-16  
*Enable compatibility with Lox language.*
- #define **MORPHO\_ERRORSTRINGSIZE** 255  
*Maximum length of a Morpho error string.*
- #define **MORPHO\_INPUTBUFFERDEFAULTSIZE** 255  
*Default size of input buffer.*
- #define **MORPHO\_MAXIMUMFILENAMELENGTH** 255  
*Maximum file name length.*
- #define **MORPHO\_CALLFRAMESTACKSIZE** 64  
*Size of the call frame stack.*
- #define **MORPHO\_MAXARGS** 255  
*Maximum number of arguments.*
- #define **MORPHO\_COMPUTED\_GOTO**  
*Build Morpho VM with computed gotos.*
- #define **MORPHO\_NAN\_BOXING**  
*Build Morpho VM with small but hacky value type [NaN boxing].*
- #define **MORPHO\_GCINITIAL** 1024\*1024;  
*Number of bytes to bind before GC first runs.*
- #define **MORPHO\_GCGROWTHFACTOR** 2  
*Controls how rapidly the GC tries to collect garbage.*
- #define **MORPHO\_MAXIMUMSTACKALLOC** 256

- *Limits size of statically allocated arrays on the C stack.*
- `#define MORPHO_LINALG_USE_ACCELERATE`  
*Avoid using global variables (suitable for small programs only)*
- `#define MORPHO_LINALG_USE_CSPARSE`
- `#define MORPHO_DEBUG`  
*Include debugging features.*

### 10.1.1 Detailed Description

Define constants that choose how Morpho is built.

Author

T J Atherton

### 10.1.2 Macro Definition Documentation

#### 10.1.2.1 MORPHO\_EPS

```
#define MORPHO_EPS 1e-16
```

Enable compatibility with Lox language.

Turn off features incompatible with lox Value used to detect zero

#### 10.1.2.2 MORPHO\_LINALG\_USE\_ACCELERATE

```
#define MORPHO_LINALG_USE_ACCELERATE
```

Avoid using global variables (suitable for small programs only)

Use Apple's accelerate library for dense linear algebra

#### 10.1.2.3 MORPHO\_LINALG\_USE\_CSPARSE

```
#define MORPHO_LINALG_USE_CSPARSE
```

Use the LAPACK library for dense linear algebra Use CSparse for sparse matrix

## 10.2 builtin/builtin.c File Reference

Morpho built in functions and classes.

```
#include "builtin.h"
#include "common.h"
#include "object.h"
#include "functions.h"
#include "file.h"
#include "matrix.h"
#include "sparse.h"
#include "mesh.h"
```

### Macros

- #define **MORPHO\_CLASS\_METHOD** "class"
- #define **MORPHO\_SUPER\_METHOD** "superclass"
- #define **MORPHO\_SERIALIZE\_METHOD** "serialize"
- #define **MORPHO\_CLONE\_METHOD** "clone"

### Functions

- [value Object\\_init](#) (vm \*v, int nargs, [value](#) \*args)
- [value Object\\_class](#) (vm \*v, int nargs, [value](#) \*args)
- [value Object\\_super](#) (vm \*v, int nargs, [value](#) \*args)
- [value Object\\_serialize](#) (vm \*v, int nargs, [value](#) \*args)
- [value Object\\_clone](#) (vm \*v, int nargs, [value](#) \*args)
- **MORPHO\_METHOD** (MORPHO\_INITIALIZER\_METHOD, [Object\\_init](#), BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MORPHO\_CLASS\_METHOD, [Object\\_class](#), BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MORPHO\_SUPER\_METHOD, [Object\\_super](#), BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MORPHO\_SERIALIZE\_METHOD, [Object\\_serialize](#), BUILTIN\_FLAGSEMPY)
- MORPHO\_ENDCLASS [value String\\_length](#) (vm \*v, int nargs, [value](#) \*args)
- MORPHO\_ENDCLASS [value dictionary\\_constructor](#) (vm \*v, int nargs, [value](#) \*args)
- [value Dictionary\\_getindex](#) (vm \*v, int nargs, [value](#) \*args)
- [value Dictionary\\_setindex](#) (vm \*v, int nargs, [value](#) \*args)
- **MORPHO\_METHOD** (MORPHO\_GETINDEX\_METHOD, [Dictionary\\_getindex](#), BUILTIN\_FLAGSEMPY)
- MORPHO\_ENDCLASS int [range\\_count](#) ([objectrange](#) \*range)
- [value range\\_iterate](#) ([objectrange](#) \*range, unsigned int i)
- [objectrange](#) \* [object\\_newrange](#) ([value](#) start, [value](#) end, [value](#) step)
- [value range\\_constructor](#) (vm \*v, int nargs, [value](#) \*args)
- [value Range\\_enumerate](#) (vm \*v, int nargs, [value](#) \*args)
- void [builtin\\_setveneerclass](#) ([objecttype](#) type, [value](#) class)
  - Sets the veneer class for a particular object type.*
- [objectclass](#) \* [builtin\\_getveneerclass](#) ([objecttype](#) type)
  - Gets the veneer for a particular object type.*
- [value builtin\\_addfunction](#) (char \*name, [builtinfunction](#) func, [builtinfunctionflags](#) flags)
- [value builtin\\_findfunction](#) ([value](#) name)
- void [builtin\\_printfunction](#) ([objectbuiltinfunction](#) \*f)
- [value builtin\\_addclass](#) (char \*name, [builtinclassentry](#) desc[], [value](#) superclass)
- [value builtin\\_findclass](#) ([value](#) name)
- void [builtin\\_copysymboltable](#) ([dictionary](#) \*out)
- [value builtin\\_internsymbol](#) ([value](#) symbol)
- [value builtin\\_internsymbolascstring](#) (char \*symbol)
- void [builtin\\_bindobjects](#) (vm \*v, unsigned int nobj, [value](#) \*obj)
- void [builtin\\_initialize](#) (void)
- void [builtin\\_finalize](#) (void)

## Variables

- `objectclass * objectveneer` [OBJECT\_EXTERN+1]

### 10.2.1 Detailed Description

Morpho built in functions and classes.

#### Author

T J Atherton

### 10.2.2 Function Documentation

#### 10.2.2.1 `builtin_addclass()`

```
value builtin_addclass (
    char * name,
    builtinclassentry desc[],
    value superclass )
```

Defines a built in class

#### Parameters

in	<i>name</i>	the name of the class
in	<i>desc</i>	class description; use MORPHO_GETCLASSDEFINITION(name) to obtain this
in	<i>superclass</i>	the class's superclass

#### Returns

the class object

#### 10.2.2.2 `builtin_addfunction()`

```
value builtin_addfunction (
    char * name,
    builtinfunction func,
    builtinfunctionflags flags )
```

Add a builtin function.

## Parameters

<i>name</i>	name of the function
<i>func</i>	the corresponding C function
<i>flags</i>	flags to define the function

## Returns

value referring to the objectbuiltinfunction

**10.2.2.3 builtin\_copysymboltable()**

```
void builtin_copysymboltable (
    dictionary * out )
```

Copies the built in symbol table into a new dictionary

**10.2.2.4 builtin\_findclass()**

```
value builtin_findclass (
    value name )
```

Finds a builtin class from its name

**10.2.2.5 builtin\_findfunction()**

```
value builtin_findfunction (
    value name )
```

Finds a builtin function from its name

**10.2.2.6 builtin\_internsymbol()**

```
value builtin_internsymbol (
    value symbol )
```

Interns a given symbol.

**10.2.2.7 builtin\_internsymbolascstring()**

```
value builtin_internsymbolascstring (
    char * symbol )
```

Interns a symbo given as a C string.

#### 10.2.2.8 builtin\_printfunction()

```
void builtin_printfunction (
    objectbuiltinfunction * f )
```

Prints a builtin function

#### 10.2.2.9 Dictionary\_getindex()

```
value Dictionary_getindex (
    vm * v,
    int nargs,
    value * args )
```

Find a string's length

#### 10.2.2.10 Dictionary\_setindex()

```
value Dictionary_setindex (
    vm * v,
    int nargs,
    value * args )
```

Find a string's length

#### 10.2.2.11 Object\_class()

```
value Object_class (
    vm * v,
    int nargs,
    value * args )
```

Find the object's class

#### 10.2.2.12 Object\_clone()

```
value Object_clone (
    vm * v,
    int nargs,
    value * args )
```

Generic initializer

#### 10.2.2.13 Object\_init()

```
value Object_init (
    vm * v,
    int nargs,
    value * args )
```

Generic initializer



#### 10.2.2.14 object\_newrange()

```
objectrange* object_newrange (
    value start,
    value end,
    value step )
```

Create a new range. Step may be set to MORPHO\_NIL to use the default value of 1

#### 10.2.2.15 Object\_serialize()

```
value Object_serialize (
    vm * v,
    int nargs,
    value * args )
```

Generic initializer

#### 10.2.2.16 Object\_super()

```
value Object_super (
    vm * v,
    int nargs,
    value * args )
```

Find the object's superclass

#### 10.2.2.17 range\_constructor()

```
value range_constructor (
    vm * v,
    int nargs,
    value * args )
```

Constructor function for ranges

#### 10.2.2.18 range\_count()

```
MORPHO_ENDCLASS int range_count (
    objectrange * range )
```

Calculate the number of steps in a range

#### 10.2.2.19 Range\_enumerate()

```
value Range_enumerate (
    vm * v,
    int nargs,
    value * args )
```

Enumerate members of a range

### 10.2.2.20 range\_iterate()

```
value range_iterate (
    objectrange * range,
    unsigned int i )
```

Find the ith value of a range object

### 10.2.2.21 String\_length()

```
MORPHO_ENDCLASS value String_length (
    vm * v,
    int nargs,
    value * args )
```

Find a string's length

## 10.2.3 Variable Documentation

### 10.2.3.1 objectvener

```
objectclass* objectvener[OBJECT_EXTERN+1]
```

Core object types can be provided with a 'vener' class enabling the user to call methods on it, e.g. <string>.length(). This list provides easy access.

## 10.3 builtin/builtin.h File Reference

Morpho built in functions and classes.

```
#include "object.h"
#include "morpho.h"
```

### Classes

- struct [objectbuiltinfunction](#)
- struct [builtinclassentry](#)

## Macros

- #define **BUILTIN\_FLAGSEMPY** 0
- #define **MORPHO\_GETBUILTINFUNCTION**(val) ((objectbuiltinfunction \*) MORPHO\_GETOBJECT(val))
- #define **MORPHO\_ISBUILTINFUNCTION**(val) object\_istype(val, OBJECT\_BUILTINFUNCTION)
- #define **MORPHO\_BEGINCLASS**(name) builtinclassentry builtinclass\_##name[] = {
- #define **MORPHO\_PROPERTY**(label) ((builtinclassentry) { .type=(BUILTIN\_PROPERTY), .name=(label), .flags=BUILTIN\_FLAGSEMPY, .function=NULL})
- #define **MORPHO\_METHOD**(label, func, flg) ((builtinclassentry) { .type=(BUILTIN\_METHOD), .name=(label), .flags=flg, .function=func})
- #define **MORPHO\_ENDCLASS**
- #define **MORPHO\_GETCLASSDEFINITION**(name) (builtinclass\_##name)
- #define **MORPHO\_GETARG**(args, n) (args[n+1])
- #define **MORPHO\_SELF**(args) (args[0])
- #define **MORPHO\_RAISE**(v, err) { morpho\_runtimeerror(v, err); return **MORPHO\_NIL**; }
- #define **MORPHO\_RAISEVARGS**(v, err, ...)
- #define **OBJECT\_CLASSNAME** "Object"
- #define **STRING\_CLASSNAME** "String"
- #define **DICTIONARY\_CLASSNAME** "Dictionary"
- #define **RANGE\_CLASSNAME** "Range"
- #define **RANGE\_ARGS** "RngArgs"
- #define **RANGE\_ARGS\_MSG** "Range expects numerical arguments: a start, an end and an optional step-size."
- #define **ENUMERATE\_ARGS** "EnmrtArgs"
- #define **ENUMERATE\_ARGS\_MSG** "Enumerate method expects a single integer argument."

## Typedefs

- typedef unsigned int builtinfunctionflags
- typedef value(\* builtinfunction) (vm \*v, int nargs, value \*args)

## Functions

- value builtin\_addfunction (char \*name, builtinfunction func, builtinfunctionflags flags)
- value builtin\_findfunction (value name)
- void builtin\_printfunction (objectbuiltinfunction \*f)
- value builtin\_addclass (char \*name, builtinclassentry desc[], value superclass)
- value builtin\_findclass (value name)
- void builtin\_copysymboltable (dictionary \*out)
- value builtin\_internsymbol (value symbol)
- value builtin\_internsymbolascstring (char \*symbol)
- void builtin\_setveneerclass (objecttype type, value class)
- *Sets the veneer class for a particular object type.*
- objectclass \* builtin\_getveneerclass (objecttype type)
- *Gets the veneer for a particular object type.*
- void builtin\_bindobjects (vm \*v, unsigned int nobj, value \*obj)
- void builtin\_initialize (void)
- void builtin\_finalize (void)

### 10.3.1 Detailed Description

Morpho built in functions and classes.

Author

T J Atherton

### 10.3.2 Macro Definition Documentation

#### 10.3.2.1 MORPHO\_BEGINCLASS

```
#define MORPHO_BEGINCLASS(  
    name ) builtinclassentry builtinclass_##name[] = {
```

The following macros help to define a built in class. They should be used outside of any function declaration. To use: [MORPHO\\_BEGINCLASS\(Object\)](#) - Starts the declaration MORPHO\_PROPERTY("test") - Adds a property called "test" to the definition MORPHO\_METHOD("init", object\_init, BUILTIN\_FLAGSEMPY) - Adds a method called "init" to the definition MORPHO\_ENDCLASS - Ends the declaration

#### 10.3.2.2 MORPHO\_ENDCLASS

```
#define MORPHO_ENDCLASS
```

Value:

```
, MORPHO_PROPERTY(NULL) \  
};
```

#### 10.3.2.3 MORPHO\_GETARG

```
#define MORPHO_GETARG(  
    args,  
    n ) (args[n+1])
```

Macros and functions for built in classes

#### 10.3.2.4 MORPHO\_GETBUILTINFUNCTION

```
#define MORPHO_GETBUILTINFUNCTION(  
    val ) ((objectbuiltinfunction *) MORPHO_GETOBJECT(val))
```

Gets an objectfunction from a value

### 10.3.2.5 MORPHO\_ISBUILTINFUNCTION

```
#define MORPHO_ISBUILTINFUNCTION(  
    val ) object_istype(val, OBJECT_BUILTINFUNCTION)
```

Tests whether an object is a function

### 10.3.2.6 MORPHO\_RAISE

```
#define MORPHO_RAISE(  
    v,  
    err ) { morpho_runtimeerror(v, err ); return MORPHO_NIL; }
```

Raise an error and return nil

### 10.3.2.7 MORPHO\_RAISEVARGS

```
#define MORPHO_RAISEVARGS(  
    v,  
    err,  
    ... )
```

Value:

```
{ morpho_runtimeerror(v, err, __VA_ARGS__); \  
  return MORPHO_NIL; }
```

### 10.3.2.8 MORPHO\_SELF

```
#define MORPHO_SELF(  
    args ) (args[0])
```

This macro gets self

## 10.3.3 Typedef Documentation

### 10.3.3.1 builtinfunction

```
typedef value(* builtinfunction) (vm *v, int nargs, value *args)
```

Type of C function that implements a built in Morpho function

### 10.3.3.2 builtinfunctionflags

```
typedef unsigned int builtinfunctionflags
```

Flags that describe properties of the built in function

## 10.3.4 Function Documentation

### 10.3.4.1 builtin\_addclass()

```
value builtin_addclass (
    char * name,
    builtinclassentry desc[],
    value superclass )
```

Defines a built in class

#### Parameters

in	<i>name</i>	the name of the class
in	<i>desc</i>	class description; use MORPHO_GETCLASSDEFINITION(name) to obtain this
in	<i>superclass</i>	the class's superclass

#### Returns

the class object

### 10.3.4.2 builtin\_addfunction()

```
value builtin_addfunction (
    char * name,
    builtinfunction func,
    builtinfunctionflags flags )
```

Add a builtin function.

#### Parameters

<i>name</i>	name of the function
<i>func</i>	the corresponding C function
<i>flags</i>	flags to define the function

#### Returns

value referring to the objectbuiltinfunction

### 10.3.4.3 builtin\_copysymboltable()

```
void builtin_copysymboltable (
    dictionary * out )
```

Copies the built in symbol table into a new dictionary

#### 10.3.4.4 builtin\_findclass()

```
value builtin_findclass (
    value name )
```

Finds a builtin class from its name

#### 10.3.4.5 builtin\_findfunction()

```
value builtin_findfunction (
    value name )
```

Finds a builtin function from its name

#### 10.3.4.6 builtin\_internsymbol()

```
value builtin_internsymbol (
    value symbol )
```

Interns a given symbol.

#### 10.3.4.7 builtin\_internsymbolascstring()

```
value builtin_internsymbolascstring (
    char * symbol )
```

Interns a symbo given as a C string.

#### 10.3.4.8 builtin\_printfunction()

```
void builtin_printfunction (
    objectbuiltinfunction * f )
```

Prints a builtin function

## 10.4 builtin/file.c File Reference

Built in class to provide file input and output.

```
#include "file.h"
#include "builtin.h"
#include "error.h"
#include "object.h"
#include "morpho.h"
#include "common.h"
#include <stdio.h>
#include <limits.h>
#include <errno.h>
```

## Functions

- bool **file\_getsize** (FILE \*f, size\_t \*s)
- FILE \* **file\_getfile** (value obj)
- void **file\_setfile** (value obj, FILE \*f)
- value **File\_init** (vm \*v, int nargs, value \*args)
- value **File\_close** (vm \*v, int nargs, value \*args)
- int **file\_readlineintovarray** (FILE \*f, varray\_char \*string)
- bool **file\_readintovarray** (FILE \*f, varray\_char \*string)
- value **file\_readlineusingvarray** (FILE \*f, varray\_char \*string)
- value **File\_readline** (vm \*v, int nargs, value \*args)
- value **File\_lines** (vm \*v, int nargs, value \*args)
- value **File\_readchar** (vm \*v, int nargs, value \*args)
- value **File\_write** (vm \*v, int nargs, value \*args)
- value **File\_eof** (vm \*v, int nargs, value \*args)
- value **File\_free** (vm \*v, int nargs, value \*args)
- **MORPHO\_METHOD** (MORPHO\_INITIALIZER\_METHOD, **File\_init**, BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (FILE\_CLOSE, **File\_close**, BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (FILE\_LINES, **File\_lines**, BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (FILE\_READLINE, **File\_readline**, BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (FILE\_READCHAR, **File\_readchar**, BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (FILE\_WRITE, **File\_write**, BUILTIN\_FLAGSEMPY)
- **MORPHO\_ENDCLASS** void **file\_initialize** (void)

### 10.4.1 Detailed Description

Built in class to provide file input and output.

Author

T J Atherton

### 10.4.2 Function Documentation

#### 10.4.2.1 File\_close()

```
value File_close (
    vm * v,
    int nargs,
    value * args )
```

Close a file



### 10.4.2.2 File\_eof()

```
value File_eof (
    vm * v,
    int nargs,
    value * args )
```

Detects whether we're at the end of the file

### 10.4.2.3 File\_free()

```
value File_free (
    vm * v,
    int nargs,
    value * args )
```

Called when the file object is freed

### 10.4.2.4 File\_init()

```
value File_init (
    vm * v,
    int nargs,
    value * args )
```

Initializer In: 1. a file name

1. (optional) a string giving the requested status, e.g. "wr+"

### 10.4.2.5 File\_lines()

```
value File_lines (
    vm * v,
    int nargs,
    value * args )
```

Get the contents of a file as an array

### 10.4.2.6 File\_readchar()

```
value File_readchar (
    vm * v,
    int nargs,
    value * args )
```

Reads a single character

**10.4.2.7 file\_readintovarray()**

```
bool file_readintovarray (
    FILE * f,
    varray_char * string )
```

Reads a whole file into a buffer

**10.4.2.8 File\_readline()**

```
value File_readline (
    vm * v,
    int nargs,
    value * args )
```

Read a line

**10.4.2.9 file\_readlineintovarray()**

```
int file_readlineintovarray (
    FILE * f,
    varray_char * string )
```

Reads a line using a given buffer

**10.4.2.10 file\_readlineusingvarray()**

```
value file_readlineusingvarray (
    FILE * f,
    varray_char * string )
```

Reads a line using a given buffer

**10.4.2.11 File\_write()**

```
value File_write (
    vm * v,
    int nargs,
    value * args )
```

Write to a file

**10.5 builtin/file.h File Reference**

Built in class to provide file input and output.

```
#include <stdio.h>
#include "morpho.h"
```

## Macros

- `#define FILE_CLASSNAME "File"`
- `#define FILE_FILEPROPERTY "@file"`
- `#define FILE_CLOSE "close"`
- `#define FILE_LINES "lines"`
- `#define FILE_READLINE "readline"`
- `#define FILE_READCHAR "readchar"`
- `#define FILE_WRITE "write"`
- `#define FILE_EOF "eof"`
- `#define FILE_READMODE "read"`
- `#define FILE_WRITEMODE "write"`
- `#define FILE_APPENDMODE "append"`
- `#define FILE_OPENFAILED "FIOpnFId"`
- `#define FILE_OPENFAILED_MSG "Couldn't open file '%s'."`
- `#define FILE_FILENAMEARG "FINmArgs"`
- `#define FILE_FILENAMEARG_MSG "First argument to File must be a filename."`
- `#define FILE_NEEDSFILENAME "FINmMssng"`
- `#define FILE_NEEDSFILENAME_MSG "Filename missing."`
- `#define FILE_MODE "FIMode"`
- `#define FILE_MODE_MSG "Second argument to File should be 'read', 'write' or 'append'."`
- `#define FILE_WRITEARGS "FIWrtArgs"`
- `#define FILE_WRITEARGS_MSG "Arguments to File.write must be strings."`
- `#define FILE_WRITEFAIL "FIWrtFId"`
- `#define FILE_WRITEFAIL_MSG "Write to file failed."`

## Functions

- `bool file_getsize (FILE *f, size_t *s)`
- `int file_readlineintovarray (FILE *f, varray_char *string)`
- `bool file_readintovarray (FILE *f, varray_char *string)`
- `void file_initialize (void)`

### 10.5.1 Detailed Description

Built in class to provide file input and output.

Author

T J Atherton

### 10.5.2 Function Documentation

#### 10.5.2.1 file\_readintovarray()

```
bool file_readintovarray (
    FILE * f,
    varray_char * string )
```

Reads a whole file into a buffer

### 10.5.2.2 file\_readlineintovarray()

```
int file_readlineintovarray (
    FILE * f,
    varray_char * string )
```

Reads a line using a given buffer

## 10.6 builtin/functions.c File Reference

Built in functions.

```
#include <time.h>
#include <stdlib.h>
#include "functions.h"
#include "random.h"
#include "builtin.h"
```

### Functions

- [value builtin\\_random](#) (vm \*v, int nargs, [value](#) \*args)
- [value builtin\\_exp](#) (vm \*v, int nargs, [value](#) \*args)
- [value builtin\\_randomint](#) (vm \*v, int nargs, [value](#) \*args)
- [value builtin\\_system](#) (vm \*v, int nargs, [value](#) \*args)
- [value builtin\\_clock](#) (vm \*v, int nargs, [value](#) \*args)
- void [functions\\_initialize](#) (void)

### 10.6.1 Detailed Description

Built in functions.

Author

T J Atherton

### 10.6.2 Function Documentation

#### 10.6.2.1 builtin\_clock()

```
value builtin_clock (
    vm * v,
    int nargs,
    value * args )
```

Clock

### 10.6.2.2 builtin\_exp()

```
value builtin_exp (
    vm * v,
    int nargs,
    value * args )
```

Exponential function

### 10.6.2.3 builtin\_random()

```
value builtin_random (
    vm * v,
    int nargs,
    value * args )
```

Generate a random float between 0 and 1

### 10.6.2.4 builtin\_randomint()

```
value builtin_randomint (
    vm * v,
    int nargs,
    value * args )
```

Generate a random integer with a bound. Efficient and unbiased algorithm from: <https://www.pcg-random.org/posts/bounded-rands.html>

### 10.6.2.5 builtin\_system()

```
value builtin_system (
    vm * v,
    int nargs,
    value * args )
```

Call the operating system

## 10.7 builtin/functions.h File Reference

Built in functions.

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

### Macros

- #define **FUNCTION\_RANDOM** "random"
- #define **FUNCTION\_RANDOMINT** "randomint"
- #define **FUNCTION\_EXP** "exp"
- #define **FUNCTION\_CLOCK** "clock"
- #define **FUNCTION\_SYSTEM** "system"

## Functions

- void **functions\_initialize** (void)

### 10.7.1 Detailed Description

Built in functions.

Author

T J Atherton

## 10.8 datastructures/dictionary.h File Reference

Dictionary (hashtable) data structure.

```
#include "value.h"
```

## Classes

- struct [dictionaryentry](#)  
*A single dictionary entry.*
- struct [dictionary](#)  
*dictionary data structure that maps keys to values*

## Macros

- #define **HASH\_EMPTY** 0

## Typedefs

- typedef uint32\_t **hash**

## Functions

- void `dictionary_init` (`dictionary` \*dict)  
*Initializes a dictionary.*
- void `dictionary_clear` (`dictionary` \*dict)  
*Clears a dictionary structure, freeing attached memory.*
- void `dictionary_freecontents` (`dictionary` \*dict, bool freekeys, bool freevals)  
*Frees a dictionary's contents.*
- bool `dictionary_insert` (`dictionary` \*dict, `value` key, `value` val)  
*Inserts a value in a hashtable given a key.*
- bool `dictionary_insertintern` (`dictionary` \*dict, `value` key, `value` val)  
*Inserts a value in a hashtable given a key, assuming the key has been interned.*
- `value` `dictionary_intern` (`dictionary` \*dict, `value` key)  
*Interns a new key @detail Looks to see if a similar key [one that passes MORPHO\_ISEQUAL] is already in the dictionary, and if so returns it. Otherwise, inserts this key into the dictionary.*
- bool `dictionary_get` (`dictionary` \*dict, `value` key, `value` \*val)  
*Retrieves a value from a dictionary given a key.*
- bool `dictionary_getintern` (`dictionary` \*dict, `value` key, `value` \*val)  
*Retrieves a value from a dictionary given a key assuming that key has been interned.*
- bool `dictionary_remove` (`dictionary` \*dict, `value` key)  
*Removes a key from a dictionary given a key.*
- bool `dictionary_copy` (`dictionary` \*src, `dictionary` \*dest)  
*Copies the entries of one dictionary to another.*

### 10.8.1 Detailed Description

Dictionary (hashtable) data structure.

#### Author

T J Atherton

### 10.8.2 Function Documentation

#### 10.8.2.1 `dictionary_clear()`

```
void dictionary_clear (
    dictionary * dict )
```

Clears a dictionary structure, freeing attached memory.

#### Parameters

<i>dict</i>	the dictionary to clear
-------------	-------------------------

**Warning**

This doesn't free keys or values in the dictionary.

**10.8.2.2 dictionary\_freecontents()**

```
void dictionary_freecontents (
    dictionary * dict,
    bool freekeys,
    bool freevals )
```

Frees a dictionary's contents.

**Parameters**

<i>dict</i>	the dictionary to clear
<i>freekeys</i>	whether to free the keys
<i>freevals</i>	whether to free the vals

**10.8.2.3 dictionary\_get()**

```
bool dictionary_get (
    dictionary * dict,
    value key,
    value * val )
```

Retrieves a value from a dictionary given a key.

**Parameters**

in	<i>dict</i>	the dictionary to get
in	<i>key</i>	key to locate
out	<i>val</i>	Stores the result in this value if found.

**Returns**

true if found, false otherwise

**10.8.2.4 dictionary\_getintern()**

```
bool dictionary_getintern (
    dictionary * dict,
```



```
value key,  
value * val )
```

Retrieves a value from a dictionary given a key assuming that key has been interned.

**Parameters**

in	<i>dict</i>	the dictionary to get
in	<i>key</i>	key to locate
out	<i>val</i>	Stores the result in this value if found.

**Returns**

true if found, false otherwise

**10.8.2.5 dictionary\_init()**

```
void dictionary_init (
    dictionary * dict )
```

Initializes a dictionary.

**Parameters**

<i>dict</i>	the dictionary to initialize
-------------	------------------------------

**10.8.2.6 dictionary\_insert()**

```
bool dictionary_insert (
    dictionary * dict,
    value key,
    value val )
```

Inserts a value in a hashtable given a key.

**Parameters**

in	<i>dict</i>	the dictionary
in	<i>key</i>	key to insert
in	<i>val</i>	value to insert

**Returns**

true if successful, false otherwise

**Warning**

If an entry already exists, it is overwritten. Caller should check for existing keys if this is necessary.

### 10.8.2.7 dictionary\_insertintern()

```
bool dictionary_insertintern (
    dictionary * dict,
    value key,
    value val )
```

Inserts a value in a hashtable given a key, assuming the key has been interned.

#### Parameters

in	<i>dict</i>	the dictionary
in	<i>key</i>	key to insert
in	<i>val</i>	value to insert

#### Returns

true if successful, false otherwise

#### Warning

If an entry already exists, it is overwritten. Caller should check for existing keys if this is necessary.

### 10.8.2.8 dictionary\_intern()

```
value dictionary_intern (
    dictionary * dict,
    value key )
```

Interns a new key @detail Looks to see if a similar key [one that passes MORPHO\_ISEQUAL] is already in the dictionary, and if so returns it. Otherwise, inserts this key into the dictionary.

#### Parameters

in	<i>dict</i>	the dictionary
in	<i>key</i>	a new key to intern

#### Returns

the internalized key, or MORPHO\_NIL on failure.

### 10.8.2.9 dictionary\_remove()

```
bool dictionary_remove (
    dictionary * dict,
    value key )
```

Removes a key from a dictionary given a key.

## Parameters

in	<i>dict</i>	the dictionary to initialize
in	<i>key</i>	key to remove

## Returns

true if the key was found, false otherwise

## 10.9 datastructures/matrix.c File Reference

Veneer class over the objectmatrix type that interfaces with blas and lapack.

```
#include <string.h>
#include "object.h"
#include "matrix.h"
#include "sparse.h"
#include "morpho.h"
#include "builtin.h"
#include "common.h"
```

### Functions

- [objectmatrix](#) \* [object\\_newmatrix](#) (unsigned int nrows, unsigned int ncols, bool zero)
- bool [matrix\\_getarraydimensions](#) ([objectarray](#) \*array, unsigned int dim[], unsigned int maxdim, unsigned int \*ndim)
- [value](#) [matrix\\_getarrayelement](#) ([objectarray](#) \*array, unsigned int ndim, unsigned int \*indx)
- [objectmatrix](#) \* [object\\_matrixfromarray](#) ([objectarray](#) \*array)
- bool [matrix\\_setelement](#) ([objectmatrix](#) \*matrix, unsigned int row, unsigned int col, double [value](#))  
*Sets a matrix element.*
- bool [matrix\\_getelement](#) ([objectmatrix](#) \*matrix, unsigned int row, unsigned int col, double \*[value](#))  
*Gets a matrix element.*
- objectmatrixerror [matrix\\_add](#) ([objectmatrix](#) \*a, [objectmatrix](#) \*b, [objectmatrix](#) \*out)
- objectmatrixerror [matrix\\_sub](#) ([objectmatrix](#) \*a, [objectmatrix](#) \*b, [objectmatrix](#) \*out)
- objectmatrixerror [matrix\\_mul](#) ([objectmatrix](#) \*a, [objectmatrix](#) \*b, [objectmatrix](#) \*out)
- objectmatrixerror [matrix\\_divs](#) ([objectmatrix](#) \*a, [objectmatrix](#) \*b, [objectmatrix](#) \*out)
- objectmatrixerror [matrix\\_divl](#) ([objectmatrix](#) \*a, [objectmatrix](#) \*b, [objectmatrix](#) \*out)
- objectmatrixerror [matrix\\_transpose](#) ([objectmatrix](#) \*a, [objectmatrix](#) \*out)
- objectmatrixerror [matrix\\_trace](#) ([objectmatrix](#) \*a, double \*out)
- [value](#) [matrix\\_constructor](#) (vm \*v, int nargs, [value](#) \*args)
- [value](#) [Matrix\\_getindex](#) (vm \*v, int nargs, [value](#) \*args)
- [value](#) [Matrix\\_setindex](#) (vm \*v, int nargs, [value](#) \*args)
- [value](#) [Matrix\\_print](#) (vm \*v, int nargs, [value](#) \*args)
- [value](#) [Matrix\\_add](#) (vm \*v, int nargs, [value](#) \*args)
- [value](#) [Matrix\\_sub](#) (vm \*v, int nargs, [value](#) \*args)
- [value](#) [Matrix\\_mul](#) (vm \*v, int nargs, [value](#) \*args)
- [value](#) [Matrix\\_div](#) (vm \*v, int nargs, [value](#) \*args)
- [value](#) [Matrix\\_transpose](#) (vm \*v, int nargs, [value](#) \*args)
- [value](#) [Matrix\\_trace](#) (vm \*v, int nargs, [value](#) \*args)
- [value](#) [Matrix\\_enumerate](#) (vm \*v, int nargs, [value](#) \*args)

- **MORPHO\_METHOD** (MORPHO\_GETINDEX\_METHOD, Matrix\_getindex, BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MORPHO\_SETINDEX\_METHOD, Matrix\_setindex, BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MORPHO\_PRINT\_METHOD, Matrix\_print, BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MORPHO\_ADD\_METHOD, Matrix\_add, BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MORPHO\_SUB\_METHOD, Matrix\_sub, BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MORPHO\_MUL\_METHOD, Matrix\_mul, BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MORPHO\_DIV\_METHOD, Matrix\_div, BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MATRIX\_TRANSPOSE\_METHOD, Matrix\_transpose, BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MATRIX\_TRACE\_METHOD, Matrix\_trace, BUILTIN\_FLAGSEMPY)
- MORPHO\_ENDCLASS void **matrix\_initialize** (void)

### 10.9.1 Detailed Description

Veneer class over the objectmatrix type that interfaces with blas and lapack.

Author

T J Atherton

### 10.9.2 Function Documentation

#### 10.9.2.1 matrix\_add()

```
objectmatrixerror matrix_add (
    objectmatrix * a,
    objectmatrix * b,
    objectmatrix * out )
```

Performs  $a + b \rightarrow out$ .

#### 10.9.2.2 matrix\_constructor()

```
value matrix_constructor (
    vm * v,
    int nargs,
    value * args )
```

Constructs a Matrix object

#### 10.9.2.3 matrix\_divl()

```
objectmatrixerror matrix_divl (
    objectmatrix * a,
    objectmatrix * b,
    objectmatrix * out )
```

Solves the system  $a.x = b$  for large matrices (test with MATRIX\_ISSMALL)

### 10.9.2.4 matrix\_divs()

```
objectmatrixerror matrix_divs (
    objectmatrix * a,
    objectmatrix * b,
    objectmatrix * out )
```

Solves the system  $a.x = b$  for small matrices (test with `MATRIX_ISSMALL`)

#### Warning

Uses the C stack for storage, which avoids malloc but can cause stack overflow

### 10.9.2.5 matrix\_getarraydimensions()

```
bool matrix_getarraydimensions (
    objectarray * array,
    unsigned int dim[],
    unsigned int maxdim,
    unsigned int * ndim )
```

Recurses into an objectarray to find the dimensions of the array and all child arrays

#### Parameters

in	<i>array</i>	- to search
out	<i>dim</i>	- array of dimensions to be filled out (must be zero'd before initial call)
in	<i>maxdim</i>	- maximum number of dimensions
out	<i>ndim</i>	- number of dimensions of the array

### 10.9.2.6 matrix\_getarrayelement()

```
value matrix_getarrayelement (
    objectarray * array,
    unsigned int ndim,
    unsigned int * indx )
```

Looks up an array element recursively if necessary

### 10.9.2.7 matrix\_getelement()

```
bool matrix_getelement (
    objectmatrix * matrix,
    unsigned int row,
    unsigned int col,
    double * value )
```

Gets a matrix element.

**Returns**

true if the element is in the range of the matrix, false otherwise

**10.9.2.8 matrix\_mul()**

```
objectmatrixerror matrix_mul (
    objectmatrix * a,
    objectmatrix * b,
    objectmatrix * out )
```

Performs  $a * b \rightarrow out$

**10.9.2.9 matrix\_setelement()**

```
bool matrix_setelement (
    objectmatrix * matrix,
    unsigned int row,
    unsigned int col,
    double value )
```

Sets a matrix element.

**Returns**

true if the element is in the range of the matrix, false otherwise

**10.9.2.10 matrix\_sub()**

```
objectmatrixerror matrix_sub (
    objectmatrix * a,
    objectmatrix * b,
    objectmatrix * out )
```

Performs  $a - b \rightarrow out$

**10.9.2.11 matrix\_trace()**

```
objectmatrixerror matrix_trace (
    objectmatrix * a,
    double * out )
```

Calculate the trace of a matrix



**10.9.2.12 matrix\_transpose()**

```
objectmatrixerror matrix_transpose (
    objectmatrix * a,
    objectmatrix * out )
```

Transpose a matrix

**10.9.2.13 object\_matrixfromarray()**

```
objectmatrix* object_matrixfromarray (
    objectarray * array )
```

Creates a new array from a list of values

**10.9.2.14 object\_newmatrix()**

```
objectmatrix* object_newmatrix (
    unsigned int nrows,
    unsigned int ncols,
    bool zero )
```

Creates a matrix object

**10.10 datastructures/matrix.h File Reference**

Veneer class over the objectmatrix type that interfaces with blas and lapack.

```
#include <stdio.h>
#include <cblas.h>
#include <lapacke.h>
```

**Macros**

- #define MORPHO\_LINALG\_USE\_LAPACK
- #define MATRIX\_LAPACK\_PRESENT
- #define MATRIX\_CLASSNAME "Matrix"
- #define MATRIX\_TRANSPOSE\_METHOD "transpose"
- #define MATRIX\_TRACE\_METHOD "trace"
- #define MATRIX\_DET\_METHOD "det"
- #define MATRIX\_EIGENVALUES\_METHOD "eigenvalues"
- #define MATRIX\_EIGENSYSTEM\_METHOD "eigensystem"
- #define MATRIX\_INDICESOUTSIDEBOUNDS "MtrxBnds"
- #define MATRIX\_INDICESOUTSIDEBOUNDS\_MSG "Matrix index out of bounds."
- #define MATRIX\_INVLDINDICES "MtrxInvldIdx"
- #define MATRIX\_INVLDINDICES\_MSG "Matrix indices must be numerical."
- #define MATRIX\_CONSTRUCTOR "MtrxCns"

- #define **MATRIX\_CONSTRUCTOR\_MSG** "Matrix() should be called either with dimensions or an array initializer."
- #define **MATRIX\_INVLDARRAYINIT** "MtrxInvldInit"
- #define **MATRIX\_INVLDARRAYINIT\_MSG** "Array initializer passed to Matrix() must be a 1 or 2 dimensional array."
- #define **MATRIX\_ARITHARGS** "MtrxInvldArg"
- #define **MATRIX\_ARITHARGS\_MSG** "Matrix arithmetic methods expect a matrix or number as their argument."
- #define **MATRIX\_INCOMPATIBLEMATRICES** "MtrxIncmptbl"
- #define **MATRIX\_INCOMPATIBLEMATRICES\_MSG** "Matrices have incompatible shape."
- #define **MATRIX\_SINGULAR** "MtrxSnglr"
- #define **MATRIX\_SINGULAR\_MSG** "Matrix is singular."
- #define **MATRIX\_NOTSQ** "MtrxNtSq"
- #define **MATRIX\_NOTSQ\_MSG** "Matrix is not square."
- #define **MATRIX\_ISSMALL**(m) (m->nrows\*m->ncols<[MORPHO\\_MAXIMUMSTACKALLOC](#))

## Enumerations

- enum **objectmatrixerror** {  
**MATRIX\_OK**, **MATRIX\_INCMPTBLDIM**, **MATRIX\_SING**, **MATRIX\_INVLD**,  
**MATRIX\_NSQ**, **MATRIX\_ALLOC** }

## Functions

- bool [matrix\\_getarraydimensions](#) ([objectarray](#) \*array, unsigned int dim[], unsigned int maxdim, unsigned int \*ndim)
- [value](#) [matrix\\_getarrayelement](#) ([objectarray](#) \*array, unsigned int ndim, unsigned int \*indx)
- bool [matrix\\_setelement](#) ([objectmatrix](#) \*matrix, unsigned int row, unsigned int col, double [value](#))  
*Sets a matrix element.*
- bool [matrix\\_getelement](#) ([objectmatrix](#) \*matrix, unsigned int row, unsigned int col, double \*[value](#))  
*Gets a matrix element.*
- [objectmatrixerror](#) [matrix\\_add](#) ([objectmatrix](#) \*a, [objectmatrix](#) \*b, [objectmatrix](#) \*out)
- [objectmatrixerror](#) [matrix\\_sub](#) ([objectmatrix](#) \*a, [objectmatrix](#) \*b, [objectmatrix](#) \*out)
- [objectmatrixerror](#) [matrix\\_mul](#) ([objectmatrix](#) \*a, [objectmatrix](#) \*b, [objectmatrix](#) \*out)
- [objectmatrixerror](#) [matrix\\_divs](#) ([objectmatrix](#) \*a, [objectmatrix](#) \*b, [objectmatrix](#) \*out)
- [objectmatrixerror](#) [matrix\\_divl](#) ([objectmatrix](#) \*a, [objectmatrix](#) \*b, [objectmatrix](#) \*out)
- [objectmatrixerror](#) [matrix\\_transpose](#) ([objectmatrix](#) \*a, [objectmatrix](#) \*out)
- [objectmatrixerror](#) [matrix\\_trace](#) ([objectmatrix](#) \*a, double \*out)
- void [matrix\\_initialize](#) (void)

### 10.10.1 Detailed Description

Veneer class over the [objectmatrix](#) type that interfaces with blas and lapack.

Author

T J Atherton

### 10.10.2 Macro Definition Documentation

### 10.10.2.1 MATRIX\_ISSMALL

```
#define MATRIX_ISSMALL(  
    m ) (m->nrows*m->ncols<MORPHO_MAXIMUMSTACKALLOC)
```

Macro to decide if a matrix is 'small' or 'large' and hence static or dynamic allocation should be used.

### 10.10.2.2 MORPHO\_LINALG\_USE\_LAPACKE

```
#define MORPHO_LINALG_USE_LAPACKE
```

Use Apple's Accelerate library for LAPACK and BLAS Otherwise, use LAPACKE

## 10.10.3 Function Documentation

### 10.10.3.1 matrix\_add()

```
objectmatrixerror matrix_add (  
    objectmatrix * a,  
    objectmatrix * b,  
    objectmatrix * out )
```

Performs  $a + b \rightarrow out$ .

### 10.10.3.2 matrix\_divl()

```
objectmatrixerror matrix_divl (  
    objectmatrix * a,  
    objectmatrix * b,  
    objectmatrix * out )
```

Solves the system  $a.x = b$  for large matrices (test with MATRIX\_ISSMALL)

### 10.10.3.3 matrix\_divs()

```
objectmatrixerror matrix_divs (  
    objectmatrix * a,  
    objectmatrix * b,  
    objectmatrix * out )
```

Solves the system  $a.x = b$  for small matrices (test with MATRIX\_ISSMALL)

#### Warning

Uses the C stack for storage, which avoids malloc but can cause stack overflow

#### 10.10.3.4 matrix\_getarraydimensions()

```
bool matrix_getarraydimensions (
    objectarray * array,
    unsigned int dim[],
    unsigned int maxdim,
    unsigned int * ndim )
```

Recurses into an objectarray to find the dimensions of the array and all child arrays

## Parameters

in	<i>array</i>	- to search
out	<i>dim</i>	- array of dimensions to be filled out (must be zero'd before initial call)
in	<i>maxdim</i>	- maximum number of dimensions
out	<i>ndim</i>	- number of dimensions of the array

**10.10.3.5 matrix\_getarrayelement()**

```
value matrix_getarrayelement (
    objectarray * array,
    unsigned int ndim,
    unsigned int * indx )
```

Looks up an array element recursively if necessary

**10.10.3.6 matrix\_getelement()**

```
bool matrix_getelement (
    objectmatrix * matrix,
    unsigned int row,
    unsigned int col,
    double * value )
```

Gets a matrix element.

## Returns

true if the element is in the range of the matrix, false otherwise

**10.10.3.7 matrix\_mul()**

```
objectmatrixerror matrix_mul (
    objectmatrix * a,
    objectmatrix * b,
    objectmatrix * out )
```

Performs  $a * b \rightarrow out$

### 10.10.3.8 matrix\_setelement()

```
bool matrix_setelement (
    objectmatrix * matrix,
    unsigned int row,
    unsigned int col,
    double value )
```

Sets a matrix element.

#### Returns

true if the element is in the range of the matrix, false otherwise

### 10.10.3.9 matrix\_sub()

```
objectmatrixerror matrix_sub (
    objectmatrix * a,
    objectmatrix * b,
    objectmatrix * out )
```

Performs  $a - b \rightarrow out$

### 10.10.3.10 matrix\_trace()

```
objectmatrixerror matrix_trace (
    objectmatrix * a,
    double * out )
```

Calculate the trace of a matrix

### 10.10.3.11 matrix\_transpose()

```
objectmatrixerror matrix_transpose (
    objectmatrix * a,
    objectmatrix * out )
```

Transpose a matrix

## 10.11 datastructures/object.c File Reference

Provide functionality for extended and mutable data types.

```
#include <string.h>
#include <stdio.h>
#include "value.h"
#include "object.h"
#include "builtin.h"
#include "memory.h"
#include "error.h"
#include "sparse.h"
#include "common.h"
```

## Functions

- void `object_init` (`object` \*obj, `objecttype` type)  
*Initializes an object to be a certain type.*
- void `object_free` (`object` \*obj)
- `object` \* `object_new` (`size_t` size, `objecttype` type)  
*Allocates an object.*
- `value` `object_stringfromcstring` (`const char` \*in, `size_t` length)  
*Creates a string from an existing character array with given length.*
- `value` `object_stringfromvarraychar` (`varray_char` \*in)  
*Converts a varray\_char into a string.*
- `value` `object_clonestring` (`value` val)
- `value` `object_concatenatestring` (`value` a, `value` b)  
*Concatenates strings together.*
- **DEFINE\_VARRAY** (`upvalue`, `upvalue`)
- **DEFINE\_VARRAY** (`varray_upvalue`, `varray_upvalue`)
- void `object_functioninit` (`objectfunction` \*func)  
*Initializes a new function.*
- void `object_functionclear` (`objectfunction` \*func)  
*Clears a function.*
- `objectfunction` \* `object_newfunction` (`indx` entry, `value` name, `objectfunction` \*parent, `unsigned int` nargs)  
*Creates a new function.*
- `objectfunction` \* `object_getfunctionparent` (`objectfunction` \*func)
- `value` `object_getfunctionname` (`objectfunction` \*func)
- `varray_value` \* `object_functiongetconstanttable` (`objectfunction` \*func)
- bool `object_functionaddprototype` (`objectfunction` \*func, `varray_upvalue` \*v, `indx` \*ix)
- void **object\_closureinit** (`objectclosure` \*c)
- `objectclosure` \* `object_newclosure` (`objectfunction` \*sf, `objectfunction` \*func, `indx` np)  
*Creates a new closure.*
- void `object_upvalueinit` (`objectupvalue` \*c)
- `objectupvalue` \* `object_newupvalue` (`value` \*reg)
- `objectclass` \* **object\_newclass** (`value` name)
- `objectinstance` \* **object\_newinstance** (`objectclass` \*klass)
- bool **objectinstance\_setproperty** (`objectinstance` \*obj, `value` key, `value` val)
- bool **objectinstance\_getproperty** (`objectinstance` \*obj, `value` key, `value` \*val)
- `objectinvocation` \* `object_newinvocation` (`value` receiver, `value` method)
- `objectdictionary` \* `object_newdictionary` (`void`)
- `dictionary` \* `object_dictionary` (`objectdictionary` \*dict)
- void `object_arrayinit` (`objectarray` \*array, `unsigned int` ndim, `unsigned int` \*dim)
- `objectarray` \* `object_newarray` (`unsigned int` ndim, `unsigned int` \*dim)  
*Creates an array object.*
- `objectarray` \* `object_arrayfromlist` (`unsigned int` n, `value` \*v)
- `objectarray` \* `object_arrayfromvarrayvalue` (`varray_value` \*v)
- bool `object_arrayvaluestoindices` (`unsigned int` ndim, `value` \*indx, `unsigned int` \*iout)
- `objectarray` \* `object_arrayfromvalueindices` (`unsigned int` ndim, `value` \*dim)
- bool `object_arrayindicestoelement` (`objectarray` \*array, `unsigned int` ndim, `unsigned int` \*indx, `unsigned int` \*ixout)  
*Calculates the correct element from a set of array indices.*
- `objectarrayerror` `object_getarrayelement` (`objectarray` \*array, `unsigned int` ndim, `value` \*indx, `value` \*out)
- `objectarrayerror` `object_setarrayelement` (`objectarray` \*array, `unsigned int` ndim, `value` \*indx, `value` set)
- void `object_print` (`value` v)
- void `object_printtobuffer` (`value` v, `varray_char` \*buffer)
- `size_t` `object_size` (`object` \*obj)

### 10.11.1 Detailed Description

Provide functionality for extended and mutable data types.

Author

T J Atherton

### 10.11.2 Function Documentation

#### 10.11.2.1 `object_arrayfromlist()`

```
objectarray* object_arrayfromlist (
    unsigned int n,
    value * v )
```

Creates a new 1D array from a list of values

#### 10.11.2.2 `object_arrayfromvalueindices()`

```
objectarray* object_arrayfromvalueindices (
    unsigned int ndim,
    value * dim )
```

Creates a new array object with the indices given as a list of values

#### 10.11.2.3 `object_arrayfromvarrayvalue()`

```
objectarray* object_arrayfromvarrayvalue (
    varray_value * v )
```

Creates a new 1D array from a list of varray\_value

#### 10.11.2.4 `object_arrayindicestoelement()`

```
bool object_arrayindicestoelement (
    objectarray * array,
    unsigned int ndim,
    unsigned int * indx,
    unsigned int * ixout )
```

Calculates the correct element from a set of array indices.

Parameters

in	<i>array</i>	- the array
in	<i>ndim</i>	- number of dimensions
in	<i>indx</i>	- list of indices
out	<i>ixout</i>	- the element number to use



**Returns**

true on success, false if indices are out of bounds

**10.11.2.5 object\_arrayinit()**

```
void object_arrayinit (
    objectarray * array,
    unsigned int ndim,
    unsigned int * dim )
```

Initializes an array given the size

**10.11.2.6 object\_arrayvaluestoindices()**

```
bool object_arrayvaluestoindices (
    unsigned int ndim,
    value * indx,
    unsigned int * iout )
```

Converts a list of indices into a list of unsigned ints

**Parameters**

<i>ndim</i>	- number of dimensions
<i>indx</i>	- the indices to evaluate
<i>iout</i>	- the indices as integers

**Returns**

true on success, or false if an unexpected type was encountered

**10.11.2.7 object\_concatenatestring()**

```
value object_concatenatestring (
    value a,
    value b )
```

Concatenates strings together.

**Parameters**

<i>a</i>	first string
<i>b</i>	second string

**Returns**

the object (as a value) which will be MORPHO\_NIL on failure

**10.11.2.8 object\_dictionary()**

```
dictionary* object_dictionary (
    objectdictionary * dict )
```

Extracts the dictionary from an objectdictionary.

**10.11.2.9 object\_free()**

```
void object_free (
    object * obj )
```

Frees an object

**10.11.2.10 object\_functionaddprototype()**

```
bool object_functionaddprototype (
    objectfunction * func,
    varray_upvalue * v,
    indx * ix )
```

Adds an upvalue prototype to a function

**Parameters**

in	<i>func</i>	function object to add to
in	<i>v</i>	a varray of upvalues that will be copied into the function definition.
out	<i>ix</i>	index of the closure created

**Returns**

true on success

**10.11.2.11 object\_functionclear()**

```
void object_functionclear (
    objectfunction * func )
```

Clears a function.

Clear the upvalue prototypes

#### 10.11.2.12 object\_functiongetconstanttable()

```
varray_value* object_functiongetconstanttable (
    objectfunction * func )
```

Gets the constant table associated with a function

#### 10.11.2.13 object\_getarrayelement()

```
objectarrayerror object_getarrayelement (
    objectarray * array,
    unsigned int ndim,
    value * indx,
    value * out )
```

Gets an array element

#### 10.11.2.14 object\_getfunctionname()

```
value object_getfunctionname (
    objectfunction * func )
```

Gets the name of a function

#### 10.11.2.15 object\_getfunctionparent()

```
objectfunction* object_getfunctionparent (
    objectfunction * func )
```

Gets the parent of a function

#### 10.11.2.16 object\_init()

```
void object_init (
    object * obj,
    objecttype type )
```

Initializes an object to be a certain type.

##### Parameters

<i>obj</i>	object to initialize
<i>type</i>	type to initialize with

**10.11.2.17 object\_new()**

```
object* object_new (
    size_t size,
    objecttype type )
```

Allocates an object.

**Parameters**

<i>size</i>	size of memory to reserve
<i>type</i>	type to initialize with

**10.11.2.18 object\_newarray()**

```
objectarray* object_newarray (
    unsigned int ndim,
    unsigned int * dim )
```

Creates an array object.

Arrays are stored in memory as follows: objectarray structure with flexible array member value value [0..dim-1] the dimensions of the array value [dim..] array elements in column major order, i.e. the matrix [ [ 1, 2], [ 3, 4] ] is stored as: <structure> // the structure 2, 2, // the dimensions 1, 3, 2, 4 // the elements in column major order

**10.11.2.19 object\_newclosure()**

```
objectclosure* object_newclosure (
    objectfunction * sf,
    objectfunction * func,
    indx np )
```

Creates a new closure.

**Parameters**

<i>sf</i>	the objectfunction of the current environment
<i>func</i>	a function object to enclose
<i>np</i>	the prototype number to use

**10.11.2.20 object\_newdictionary()**

```
objectdictionary* object_newdictionary (
    void )
```

Creates a new dictionary

#### 10.11.2.21 object\_newinvocation()

```
objectinvocation* object_newinvocation (
    value receiver,
    value method )
```

Create a new invocation

#### 10.11.2.22 object\_newupvalue()

```
objectupvalue* object_newupvalue (
    value * reg )
```

Creates a new upvalue for the register pointed to by reg.

#### 10.11.2.23 object\_print()

```
void object_print (
    value v )
```

Prints an object

#### 10.11.2.24 object\_printtobuffer()

```
void object_printtobuffer (
    value v,
    varray_char * buffer )
```

Prints an object to a string buffer

##### Parameters

in	<i>v</i>	Object to convert to a buffer
in	<i>buffer</i>	Buffer to output to

#### 10.11.2.25 object\_setarrayelement()

```
objectarrayerror object_setarrayelement (
    objectarray * array,
    unsigned int ndim,
    value * indx,
    value set )
```

Sets an array element

**10.11.2.26 object\_size()**

```
size_t object_size (
    object * obj )
```

Gets the total size of an object

**10.11.2.27 object\_stringfromcstring()**

```
value object_stringfromcstring (
    const char * in,
    size_t length )
```

Creates a string from an existing character array with given length.

**Parameters**

<i>in</i>	the string to copy
<i>length</i>	length of string to copy

**Returns**

the object (as a value) which will be MORPHO\_NIL on failure

**10.11.2.28 object\_stringfromvarraychar()**

```
value object_stringfromvarraychar (
    varray_char * in )
```

Converts a varray\_char into a string.

**Parameters**

<i>in</i>	the varray to convert
-----------	-----------------------

**Returns**

the object (as a value) which will be MORPHO\_NIL on failure

**10.11.2.29 object\_upvalueinit()**

```
void object_upvalueinit (
    objectupvalue * c )
```

Initializes a new upvalue object.

## 10.12 datastructures/object.h File Reference

Provide functionality for extended and mutable data types.

```
#include <stddef.h>
#include "value.h"
#include "dictionary.h"
```

### Classes

- struct [object](#)
- struct [objectstring](#)
- struct [upvalue](#)
- struct [objectfunction](#)
- struct [objectupvalue](#)
- struct [objectclosure](#)
- struct [objectclass](#)
- struct [objectinstance](#)
- struct [objectinvocation](#)
- struct [objectrange](#)
- struct [objectdictionary](#)
- struct [objectarray](#)
- struct [objectmatrix](#)
- struct [objectdokkey](#)
- struct [sparsedok](#)
- struct [sparseccs](#)
- struct [objectspare](#)
- struct [objectmesh](#)

### Macros

- #define [MORPHO\\_GETOBJECTTYPE](#)(val) (MORPHO\_GETOBJECT(val)->type)
- #define [MORPHO\\_GETOBJECTHASH](#)(val) (MORPHO\_GETOBJECT(val)->hsh)
- #define [MORPHO\\_SETOBJECTHASH](#)(val, newhash) (MORPHO\_GETOBJECT(val)->hsh = newhash)
- #define [MORPHO\\_GETSTRING](#)(val) ((([objectstring](#) \*) MORPHO\_GETOBJECT(val))
- #define [MORPHO\\_GETCSTRING](#)(val) ((([objectstring](#) \*) MORPHO\_GETOBJECT(val))->string)
- #define [MORPHO\\_GETSTRINGLENGTH](#)(val) ((([objectstring](#) \*) MORPHO\_GETOBJECT(val))->length)
- #define [MORPHO\\_ISSTRING](#)(val) object\_istype(val, OBJECT\_STRING)
- #define [MORPHO\\_STATICSTRING](#)(cstring) { .obj.type=OBJECT\_STRING, .obj.status=OBJECT\_ISUNMANAGED, .obj.next=NULL, .string=cstring, .length=strlen(cstring) }
- #define [MORPHO\\_STATICSTRINGWITHLENGTH](#)(cstring, len) { .obj.type=OBJECT\_STRING, .obj.status=OBJECT\_ISUNMANAGED, .obj.next=NULL, .string=cstring, .length=len }
- #define [OBJECT\\_STRINGLABEL](#) "string"
- #define [OBJECT\\_SYMBOLLABEL](#) "symbol"
- #define [MORPHO\\_GETFUNCTION](#)(val) ((([objectfunction](#) \*) MORPHO\_GETOBJECT(val))
- #define [MORPHO\\_ISFUNCTION](#)(val) object\_istype(val, OBJECT\_FUNCTION)
- #define [MORPHO\\_GETUPVALUE](#)(val) ((([objectupvalue](#) \*) MORPHO\_GETOBJECT(val))
- #define [MORPHO\\_ISUPVALUE](#)(val) object\_istype(val, OBJECT\_UPVALUE)
- #define [MORPHO\\_ISCLOSURE](#)(val) object\_istype(val, OBJECT\_CLOSURE)
- #define [MORPHO\\_GETCLOSURE](#)(val) ((([objectclosure](#) \*) MORPHO\_GETOBJECT(val))
- #define [MORPHO\\_GETCLOSUREFUNCTION](#)(val) ((([objectclosure](#) \*) MORPHO\_GETOBJECT(val))->func)

- #define **MORPHO\_ISCLASS**(val) object\_istype(val, OBJECT\_CLASS)
- #define **MORPHO\_GETCLASS**(val) ((**objectclass** \*) MORPHO\_GETOBJECT(val))
- #define **MORPHO\_GETSUPERCLASS**(val) (MORPHO\_GETCLASS(val)->superclass)
- #define **MORPHO\_ISINSTANCE**(val) object\_istype(val, OBJECT\_INSTANCE)
- #define **MORPHO\_GETINSTANCE**(val) ((**objectinstance** \*) MORPHO\_GETOBJECT(val))
- #define **MORPHO\_ISINVOCATION**(val) object\_istype(val, OBJECT\_INVOCATION)
- #define **MORPHO\_GETINVOCATION**(val) ((**objectinvocation** \*) MORPHO\_GETOBJECT(val))
- #define **MORPHO\_ISRANGE**(val) object\_istype(val, OBJECT\_RANGE)
- #define **MORPHO\_GETRANGE**(val) ((**objectrange** \*) MORPHO\_GETOBJECT(val))
- #define **MORPHO\_ISDICTIONARY**(val) object\_istype(val, OBJECT\_DICTIONARY)
- #define **MORPHO\_GETDICTIONARY**(val) ((**objectdictionary** \*) MORPHO\_GETOBJECT(val))
- #define **MORPHO\_ISARRAY**(val) object\_istype(val, OBJECT\_ARRAY)
- #define **MORPHO\_GETARRAY**(val) ((**objectarray** \*) MORPHO\_GETOBJECT(val))
- #define **MORPHO\_ISMATRIX**(val) object\_istype(val, OBJECT\_MATRIX)
- #define **MORPHO\_GETMATRIX**(val) ((**objectmatrix** \*) MORPHO\_GETOBJECT(val))
- #define **MORPHO\_STATICMATRIX**(darray, nr, nc) { .obj.type=OBJECT\_MATRIX, .obj.status=OBJECT\_ISUNMANAGED, .obj.next=NULL, .elements=darray, .nrows=nr, .ncols=nc }

*Use to create static matrices on the C stack.*

- #define **MORPHO\_STATICDOKKEY**(i, j) { .obj.type=OBJECT\_DOKKEY, .obj.status=OBJECT\_ISUNMANAGED, .obj.next=NULL, .row=i, .col=j }
- #define **MORPHO\_ISDOKKEY**(val) object\_istype(val, OBJECT\_DOKKEY)
- #define **MORPHO\_GETDOKKEY**(val) ((**objectdokkey** \*) MORPHO\_GETOBJECT(val))
- #define **MORPHO\_GETDOKKEYROW**(objptr) ((unsigned int) (objptr)->row)
- #define **MORPHO\_GETDOKKEYCOL**(objptr) ((unsigned int) (objptr)->col)
- #define **MORPHO\_GETDOKROWWVAL**(val) ((unsigned int) (MORPHO\_GETDOKKEY(val)->row))
- #define **MORPHO\_GETDOKCOLWVAL**(val) ((unsigned int) (MORPHO\_GETDOKKEY(val)->col))
- #define **MORPHO\_ISSPARSE**(val) object\_istype(val, OBJECT\_SPARSE)
- #define **MORPHO\_GETSPARSE**(val) ((**objectsparse** \*) MORPHO\_GETOBJECT(val))
- #define **MORPHO\_ISMESH**(val) object\_istype(val, OBJECT\_MESH)
- #define **MORPHO\_GETMESH**(val) ((**objectmesh** \*) MORPHO\_GETOBJECT(val))

## Typedefs

- typedef ptrdiff\_t **indx**
- typedef struct **subjectfunction** **objectfunction**
- typedef struct **subjectupvalue** **objectupvalue**
- typedef struct **subjectclass** **objectclass**

## Enumerations

- enum **objecttype** {  
**OBJECT\_STRING**, **OBJECT\_FUNCTION**, **OBJECT\_BUILTINFUNCTION**, **OBJECT\_CLOSURE**,  
**OBJECT\_UPVALUE**, **OBJECT\_CLASS**, **OBJECT\_INSTANCE**, **OBJECT\_INVOCATION**,  
**OBJECT\_RANGE**, **OBJECT\_DICTIONARY**, **OBJECT\_ARRAY**, **OBJECT\_MATRIX**,  
**OBJECT\_SPARSE**, **OBJECT\_DOKKEY**, **OBJECT\_MESH**, **OBJECT\_EXTERN** }
- enum **objectarrayerror** { **ARRAY\_OK**, **ARRAY\_WRONGDIM**, **ARRAY\_NONNUMERICALINDEX**, **ARRAY\_OUTOFBOUNDS** }



## Functions

- void `object_init` (`object` \*obj, `objecttype` type)  
*Initializes an object to be a certain type.*
- void `object_free` (`object` \*obj)
- void `object_print` (`value` v)
- void `object_printtobuffer` (`value` v, `varray_char` \*buffer)
- `object` \* `object_new` (`size_t` size, `objecttype` type)  
*Allocates an object.*
- `size_t` `object_size` (`object` \*obj)
- `value` `object_stringfromcstring` (`const char` \*in, `size_t` length)  
*Creates a string from an existing character array with given length.*
- `value` `object_stringfromvarraychar` (`varray_char` \*in)  
*Converts a `varray_char` into a string.*
- `value` `object_clonestring` (`value` val)
- `value` `object_concatenatestring` (`value` a, `value` b)  
*Concatenates strings together.*
- void `object_functioninit` (`objectfunction` \*func)  
*Initializes a new function.*
- void `object_functionclear` (`objectfunction` \*func)  
*Clears a function.*
- bool `object_functionaddprototype` (`objectfunction` \*func, `varray_upvalue` \*v, `indx` \*ix)
- `objectfunction` \* `object_getfunctionparent` (`objectfunction` \*func)
- `value` `object_getfunctionname` (`objectfunction` \*func)
- `varray_value` \* `object_functiongetconstanttable` (`objectfunction` \*func)
- `objectfunction` \* `object_newfunction` (`indx` entry, `value` name, `objectfunction` \*parent, `unsigned int` nargs)  
*Creates a new function.*
- void `object_upvalueinit` (`objectupvalue` \*c)
- `objectupvalue` \* `object_newupvalue` (`value` \*reg)
- `objectclosure` \* `object_newclosure` (`objectfunction` \*sf, `objectfunction` \*func, `indx` np)  
*Creates a new closure.*
- `objectclass` \* `object_newclass` (`value` name)
- `objectinstance` \* `object_newinstance` (`objectclass` \*klass)
- bool `objectinstance_setproperty` (`objectinstance` \*obj, `value` key, `value` val)
- bool `objectinstance_getproperty` (`objectinstance` \*obj, `value` key, `value` \*val)
- `objectinvocation` \* `object_newinvocation` (`value` receiver, `value` method)
- bool `objectinstance_insertpropertybycstring` (`objectinstance` \*obj, `char` \*property, `value` val)
- bool `objectinstance_getpropertybycstring` (`objectinstance` \*obj, `char` \*property, `value` \*val)
- `objectrange` \* `object_newrange` (`value` start, `value` end, `value` step)
- `objectdictionary` \* `object_newdictionary` (`void`)
- `objectarray` \* `object_newarray` (`unsigned int` dimension, `unsigned int` \*dim)  
*Creates an array object.*
- `objectarray` \* `object_arrayfromlist` (`unsigned int` n, `value` \*v)
- `objectarray` \* `object_arrayfromvarrayvalue` (`varray_value` \*v)
- `objectarray` \* `object_arrayfromvalueindices` (`unsigned int` ndim, `value` \*dim)
- bool `object_arrayvaluestoindices` (`unsigned int` ndim, `value` \*indx, `unsigned int` \*iout)
- bool `object_arrayindigestoelement` (`objectarray` \*array, `unsigned int` ndim, `unsigned int` \*indx, `unsigned int` \*ixout)  
*Calculates the correct element from a set of array indices.*
- `objectarrayerror` `object_getarrayelement` (`objectarray` \*array, `unsigned int` ndim, `value` \*indx, `value` \*out)
- `objectarrayerror` `object_setarrayelement` (`objectarray` \*array, `unsigned int` ndim, `value` \*indx, `value` set)
- `objectmatrix` \* `object_newmatrix` (`unsigned int` nrows, `unsigned int` ncols, `bool` zero)
- `objectmatrix` \* `object_matrixfromarray` (`objectarray` \*array)
- **DECLARE\_VARRAY** (`dokkey`, `objectdokkey`)
- `objectsparse` \* `object_newsparse` (`int` \*nrows, `int` \*ncols)
- `objectsparse` \* `sparse_sparsefromarray` (`objectarray` \*array)
- `objectmesh` \* `object_newmesh` (`unsigned int` dim, `unsigned int` nv, `double` \*v)

### 10.12.1 Detailed Description

Provide functionality for extended and mutable data types.

Author

T J Atherton

### 10.12.2 Macro Definition Documentation

#### 10.12.2.1 MORPHO\_GETARRAY

```
#define MORPHO_GETARRAY(  
    val ) ((objectarray *) MORPHO_GETOBJECT(val))
```

Gets the object as an array

#### 10.12.2.2 MORPHO\_GETCLASS

```
#define MORPHO_GETCLASS(  
    val ) ((objectclass *) MORPHO_GETOBJECT(val))
```

Gets the object as a class

#### 10.12.2.3 MORPHO\_GETCLOSURE

```
#define MORPHO_GETCLOSURE(  
    val ) ((objectclosure *) MORPHO_GETOBJECT(val))
```

Gets the object as a closure

#### 10.12.2.4 MORPHO\_GETCLOSUREFUNCTION

```
#define MORPHO_GETCLOSUREFUNCTION(  
    val ) (((objectclosure *) MORPHO_GETOBJECT(val))->func)
```

Retrieve the function object from a closure

#### 10.12.2.5 MORPHO\_GETDICTIONARY

```
#define MORPHO_GETDICTIONARY(  
    val ) ((objectdictionary *) MORPHO_GETOBJECT(val))
```

Gets the object as a dictionary

### 10.12.2.6 MORPHO\_GETDOKKEY

```
#define MORPHO_GETDOKKEY(  
    val ) ((objectdokkey *) MORPHO_GETOBJECT(val))
```

Gets the object as a dok key

### 10.12.2.7 MORPHO\_GETDOKKEYROW

```
#define MORPHO_GETDOKKEYROW(  
    objptr ) ((unsigned int) (objptr)->row)
```

Gets the row and column from a objectdokkey

### 10.12.2.8 MORPHO\_GETFUNCTION

```
#define MORPHO_GETFUNCTION(  
    val ) ((objectfunction *) MORPHO_GETOBJECT(val))
```

Gets an objectfunction from a value

### 10.12.2.9 MORPHO\_GETINSTANCE

```
#define MORPHO_GETINSTANCE(  
    val ) ((objectinstance *) MORPHO_GETOBJECT(val))
```

Gets the object as a class

### 10.12.2.10 MORPHO\_GETINVOCATION

```
#define MORPHO_GETINVOCATION(  
    val ) ((objectinvocation *) MORPHO_GETOBJECT(val))
```

Gets the object as an invocation

### 10.12.2.11 MORPHO\_GETMATRIX

```
#define MORPHO_GETMATRIX(  
    val ) ((objectmatrix *) MORPHO_GETOBJECT(val))
```

Gets the object as an matrix

### 10.12.2.12 MORPHO\_GETMESH

```
#define MORPHO_GETMESH(  
    val ) ((objectmesh *) MORPHO_GETOBJECT(val))
```

Gets the object as a mesh

#### 10.12.2.13 MORPHO\_GETOBJECTHASH

```
#define MORPHO_GETOBJECTHASH(  
    val ) (MORPHO_GETOBJECT(val)->hsh)
```

Gets an objects key

#### 10.12.2.14 MORPHO\_GETOBJECTTYPE

```
#define MORPHO_GETOBJECTTYPE(  
    val ) (MORPHO_GETOBJECT(val)->type)
```

Gets the type of the object associated with a value

#### 10.12.2.15 MORPHO\_GETRANGE

```
#define MORPHO_GETRANGE(  
    val ) ((objectrange *) MORPHO_GETOBJECT(val))
```

Gets the object as a range

#### 10.12.2.16 MORPHO\_GETSPARSE

```
#define MORPHO_GETSPARSE(  
    val ) ((objectsparse *) MORPHO_GETOBJECT(val))
```

Gets the object as a sparse matrix

#### 10.12.2.17 MORPHO\_GETSUPERCLASS

```
#define MORPHO_GETSUPERCLASS(  
    val ) (MORPHO_GETCLASS(val)->superclass)
```

Gets the superclass

#### 10.12.2.18 MORPHO\_GETUPVALUE

```
#define MORPHO_GETUPVALUE(  
    val ) ((objectupvalue *) MORPHO_GETOBJECT(val))
```

Gets an objectfunction from a value

#### 10.12.2.19 MORPHO\_ISARRAY

```
#define MORPHO_ISARRAY(  
    val ) object_istype(val, OBJECT_ARRAY)
```

Tests whether an object is an array

### 10.12.2.20 MORPHO\_ISCLASS

```
#define MORPHO_ISCLASS(  
    val ) object_istype(val, OBJECT_CLASS)
```

Tests whether an object is a class

### 10.12.2.21 MORPHO\_ISCLOSURE

```
#define MORPHO_ISCLOSURE(  
    val ) object_istype(val, OBJECT_CLOSURE)
```

Tests whether an object is a closure

### 10.12.2.22 MORPHO\_ISDICTIONARY

```
#define MORPHO_ISDICTIONARY(  
    val ) object_istype(val, OBJECT_DICTIONARY)
```

Tests whether an object is a dictionary

### 10.12.2.23 MORPHO\_ISDOKKEY

```
#define MORPHO_ISDOKKEY(  
    val ) object_istype(val, OBJECT_DOKKEY)
```

Tests whether an object is a dok key

### 10.12.2.24 MORPHO\_ISFUNCTION

```
#define MORPHO_ISFUNCTION(  
    val ) object_istype(val, OBJECT_FUNCTION)
```

Tests whether an object is a function

### 10.12.2.25 MORPHO\_ISINSTANCE

```
#define MORPHO_ISINSTANCE(  
    val ) object_istype(val, OBJECT_INSTANCE)
```

Tests whether an object is a class

### 10.12.2.26 MORPHO\_ISINVOCATION

```
#define MORPHO_ISINVOCATION(  
    val ) object_istype(val, OBJECT_INVOCATION)
```

Tests whether an object is an invocation

#### 10.12.2.27 MORPHO\_ISMATRIX

```
#define MORPHO_ISMATRIX(  
    val ) object_istype(val, OBJECT_MATRIX)
```

Tests whether an object is a matrix

#### 10.12.2.28 MORPHO\_ISMESH

```
#define MORPHO_ISMESH(  
    val ) object_istype(val, OBJECT_MESH)
```

Tests whether an object is a mesh

#### 10.12.2.29 MORPHO\_ISRANGE

```
#define MORPHO_ISRANGE(  
    val ) object_istype(val, OBJECT_RANGE)
```

Tests whether an object is a range

#### 10.12.2.30 MORPHO\_ISSPARSE

```
#define MORPHO_ISSPARSE(  
    val ) object_istype(val, OBJECT_SPARSE)
```

Tests whether an object is a sparse matrix

#### 10.12.2.31 MORPHO\_ISSTRING

```
#define MORPHO_ISSTRING(  
    val ) object_istype(val, OBJECT_STRING)
```

Tests whether an object is a string

#### 10.12.2.32 MORPHO\_ISUPVALUE

```
#define MORPHO_ISUPVALUE(  
    val ) object_istype(val, OBJECT_UPVALUE)
```

Tests whether an object is a function

#### 10.12.2.33 MORPHO\_SETOBJECTHASH

```
#define MORPHO_SETOBJECTHASH(  
    val,  
    newhash ) (MORPHO_GETOBJECT(val)->hsh = newhash)
```

Sets an objects key

### 10.12.2.34 MORPHO\_STATICDOKKEY

```
#define MORPHO_STATICDOKKEY(
    i,
    j ) { .obj.type=OBJECT_DOKKEY, .obj.status=OBJECT_ISUNMANAGED, .obj.next=NULL,
    .row=i, .col=j }
```

Create

### 10.12.2.35 MORPHO\_STATICMATRIX

```
#define MORPHO_STATICMATRIX(
    darray,
    nr,
    nc ) { .obj.type=OBJECT_MATRIX, .obj.status=OBJECT_ISUNMANAGED, .obj.next=NULL,
    .elements=darray, .nrows=nr, .ncols=nc }
```

Use to create static matrices on the C stack.

Intended for small matrices; Caller needs to supply a double array of size nr\*nc.

### 10.12.2.36 MORPHO\_STATICSTRING

```
#define MORPHO_STATICSTRING(
    cstring ) { .obj.type=OBJECT_STRING, .obj.status=OBJECT_ISUNMANAGED, .obj.↵
next=NULL, .string=cstring, .length=strlen(cstring) }
```

Use to create static strings on the C stack

### 10.12.2.37 MORPHO\_STATICSTRINGWITHLENGTH

```
#define MORPHO_STATICSTRINGWITHLENGTH(
    cstring,
    len ) { .obj.type=OBJECT_STRING, .obj.status=OBJECT_ISUNMANAGED, .obj.next=NULL,
    .string=cstring, .length=len }
```

Use to create static strings on the C stack

## 10.12.3 Typedef Documentation

### 10.12.3.1 objectfunction

```
typedef struct sobjectfunction objectfunction
```

A function object

## 10.12.4 Enumeration Type Documentation

### 10.12.4.1 objecttype

```
enum objecttype
```

The type of an object

## 10.12.5 Function Documentation

### 10.12.5.1 object\_arrayfromlist()

```
objectarray* object_arrayfromlist (
    unsigned int n,
    value * v )
```

Creates a new array from a list of values

Creates a new 1D array from a list of values

### 10.12.5.2 object\_arrayfromvalueindices()

```
objectarray* object_arrayfromvalueindices (
    unsigned int ndim,
    value * dim )
```

Creates a new array object with the dimensions given as a list of values

Creates a new array object with the indices given as a list of values

### 10.12.5.3 object\_arrayfromvarrayvalue()

```
objectarray* object_arrayfromvarrayvalue (
    varray_value * v )
```

Creates a new 1D array from a list of varray\_value

### 10.12.5.4 object\_arrayindicestoelement()

```
bool object_arrayindicestoelement (
    objectarray * array,
    unsigned int ndim,
    unsigned int * indx,
    unsigned int * ixout )
```

Calculates the correct element from a set of array indices.



## Parameters

in	<i>array</i>	- the array
in	<i>ndim</i>	- number of dimensions
in	<i>indx</i>	- list of indices
out	<i>ixout</i>	- the element number to use

## Returns

true on success, false if indices are out of bounds

**10.12.5.5 object\_arrayvaluestoindices()**

```
bool object_arrayvaluestoindices (
    unsigned int ndim,
    value * indx,
    unsigned int * iout )
```

Converts a list of indices into a list of unsigned ints

## Parameters

<i>ndim</i>	- number of dimensions
<i>indx</i>	- the indices to evaluate
<i>iout</i>	- the indices as integers

## Returns

true on success, or false if an unexpected type was encountered

**10.12.5.6 object\_concatenatestring()**

```
value object_concatenatestring (
    value a,
    value b )
```

Concatenates strings together.

## Parameters

<i>a</i>	first string
<i>b</i>	second string

**Returns**

the object (as a value) which will be MORPHO\_NIL on failure

**10.12.5.7 object\_free()**

```
void object_free (
    object * obj )
```

Frees an object

**10.12.5.8 object\_functionaddprototype()**

```
bool object_functionaddprototype (
    objectfunction * func,
    varray_upvalue * v,
    indx * ix )
```

Adds an upvalue prototype to a function

**Parameters**

in	<i>func</i>	function object to add to
in	<i>v</i>	a varray of upvalues that will be copied into the function definition.
out	<i>ix</i>	index of the closure created

**Returns**

true on success

**10.12.5.9 object\_functionclear()**

```
void object_functionclear (
    objectfunction * func )
```

Clears a function.

Clear the upvalue prototypes

**10.12.5.10 object\_functiongetconstanttable()**

```
varray_value* object_functiongetconstanttable (
    objectfunction * func )
```

Gets the constant table associated with a function

#### 10.12.5.11 `object_getarrayelement()`

```
objectarrayerror object_getarrayelement (
    objectarray * array,
    unsigned int ndim,
    value * indx,
    value * out )
```

Gets an array element

#### 10.12.5.12 `object_getfunctionname()`

```
value object_getfunctionname (
    objectfunction * func )
```

Gets the name of a function

#### 10.12.5.13 `object_getfunctionparent()`

```
objectfunction* object_getfunctionparent (
    objectfunction * func )
```

Gets the parent of a function

#### 10.12.5.14 `object_init()`

```
void object_init (
    object * obj,
    objecttype type )
```

Initializes an object to be a certain type.

##### Parameters

<i>obj</i>	object to initialize
<i>type</i>	type to initialize with

#### 10.12.5.15 `object_matrixfromarray()`

```
objectmatrix* object_matrixfromarray (
    objectarray * array )
```

Creates a new array from a list of values

**10.12.5.16 object\_new()**

```
object* object_new (
    size_t size,
    objecttype type )
```

Allocates an object.

**Parameters**

<i>size</i>	size of memory to reserve
<i>type</i>	type to initialize with

**10.12.5.17 object\_newarray()**

```
objectarray* object_newarray (
    unsigned int ndim,
    unsigned int * dim )
```

Creates an array object.

Creates an array object

Arrays are stored in memory as follows: objectarray structure with flexible array member value value [0..dim-1] the dimensions of the array value [dim..] array elements in column major order, i.e. the matrix [ [ 1, 2], [ 3, 4] ] is stored as: <structure> // the structure 2, 2, // the dimensions 1, 3, 2, 4 // the elements in column major order

**10.12.5.18 object\_newclosure()**

```
objectclosure* object_newclosure (
    objectfunction * sf,
    objectfunction * func,
    indx np )
```

Creates a new closure.

**Parameters**

<i>sf</i>	the objectfunction of the current environment
<i>func</i>	a function object to enclose
<i>np</i>	the prototype number to use

**10.12.5.19 object\_newdictionary()**

```
objectdictionary* object_newdictionary (
    void )
```

Creates a new dictionary

#### 10.12.5.20 object\_newinvocation()

```
objectinvocation* object_newinvocation (
    value receiver,
    value method )
```

Create a new invocation

#### 10.12.5.21 object\_newmatrix()

```
objectmatrix* object_newmatrix (
    unsigned int nrows,
    unsigned int ncols,
    bool zero )
```

Creates a matrix object

#### 10.12.5.22 object\_newmesh()

```
objectmesh* object_newmesh (
    unsigned int dim,
    unsigned int nv,
    double * v )
```

Creates a mesh object

#### 10.12.5.23 object\_newrange()

```
objectrange* object_newrange (
    value start,
    value end,
    value step )
```

Create a new range. Step may be set to MORPHO\_NIL to use the default value of 1

#### 10.12.5.24 object\_newsparse()

```
objectsparse* object_newsparse (
    int * nrows,
    int * ncols )
```

Creates a sparse matrix object

##### Parameters

in	<i>nrows</i>	} Optional number of rows and columns
in	<i>ncols</i>	}

**10.12.5.25 object\_newupvalue()**

```
objectupvalue* object_newupvalue (
    value * reg )
```

Creates a new upvalue for the register pointed to by reg.

**10.12.5.26 object\_print()**

```
void object_print (
    value v )
```

Prints an object

**10.12.5.27 object\_printtobuffer()**

```
void object_printtobuffer (
    value v,
    varray_char * buffer )
```

Prints an object to a string buffer

**Parameters**

in	<i>v</i>	Object to convert to a buffer
in	<i>buffer</i>	Buffer to output to

**10.12.5.28 object\_setarrayelement()**

```
objectarrayerror object_setarrayelement (
    objectarray * array,
    unsigned int ndim,
    value * indx,
    value set )
```

Sets an array element

**10.12.5.29 object\_size()**

```
size_t object_size (
    object * obj )
```

Gets the total size of an object

### 10.12.5.30 object\_stringfromcstring()

```
value object_stringfromcstring (
    const char * in,
    size_t length )
```

Creates a string from an existing character array with given length.

#### Parameters

<i>in</i>	the string to copy
<i>length</i>	length of string to copy

#### Returns

the object (as a value) which will be MORPHO\_NIL on failure

### 10.12.5.31 object\_stringfromvarraychar()

```
value object_stringfromvarraychar (
    varray_char * in )
```

Converts a varray\_char into a string.

#### Parameters

<i>in</i>	the varray to convert
-----------	-----------------------

#### Returns

the object (as a value) which will be MORPHO\_NIL on failure

### 10.12.5.32 object\_upvalueinit()

```
void object_upvalueinit (
    objectupvalue * c )
```

Initializes a new upvalue object.

## 10.13 datastructures/sparse.c File Reference

Veneer class over the objectsparse type that provides sparse matrices.

```
#include "build.h"
#include "sparse.h"
#include "morpho.h"
#include "dictionary.h"
#include "common.h"
#include "matrix.h"
#include "builtin.h"
#include <limits.h>
#include <stdlib.h>
```

### Functions

- [DEFINE\\_VARRAY](#) (dokkey, objectdokkey)
- void [sparsedok\\_init](#) (sparsedok \*dok)
- void [sparsedok\\_clear](#) (sparsedok \*dok)
- bool [sparsedok\\_insert](#) (sparsedok \*dok, int i, int j, value val)
- bool [sparsedok\\_get](#) (sparsedok \*dok, int i, int j, value \*val)
- bool [sparsedok\\_remove](#) (sparsedok \*dok, int i, int j, value \*val)
- bool [sparsedok\\_setdimensions](#) (sparsedok \*dok, int nrows, int ncols)
- void [sparsedok\\_print](#) (sparsedok \*dok)
- void [sparseccs\\_init](#) (sparseccs \*ccs)
- void [sparseccs\\_clear](#) (sparseccs \*ccs)
- bool [sparseccs\\_resize](#) (sparseccs \*ccs, int nrows, int ncols, unsigned int nentries, bool values)
- bool [sparseccs\\_getrowindices](#) (sparseccs \*ccs, int col, int \*nentries, int \*\*entries)
- bool [sparseccs\\_set](#) (sparseccs \*ccs, int i, int j, double val)
- bool [sparseccs\\_get](#) (sparseccs \*ccs, int i, int j, double \*val)
- bool [sparseccs\\_doktoocs](#) (sparsedok \*in, sparseccs \*out, bool copyvals)
- void [sparseccs\\_print](#) (sparseccs \*ccs)
- bool [sparse\\_checkformat](#) (objectsparse \*sparse, objectsparseformat format, bool force)
- void [sparse\\_removeformat](#) (objectsparse \*s, objectsparseformat format)
- void [sparse\\_test](#) (void)
- objectsparse \* [object\\_newsparse](#) (int \*nrows, int \*ncols)
- objectsparse \* [object\\_sparsefromarray](#) (objectarray \*array)
- bool [sparse\\_setelement](#) (objectsparse \*s, int row, int col, value val)
- bool [sparse\\_getelement](#) (objectsparse \*s, int row, int col, value \*val)
- objectsparseerror [sparse\\_add](#) (objectsparse \*a, objectsparse \*b, double alpha, double beta, objectsparse \*out)
- objectsparseerror [sparse\\_mul](#) (objectsparse \*a, objectsparse \*b, objectsparse \*out)
- objectsparseerror [sparse\\_div](#) (objectsparse \*a, objectmatrix \*b, objectmatrix \*out)
- void [sparse\\_clear](#) (objectsparse \*a)
- size\_t [sparse\\_size](#) (objectsparse \*a)
- void [sparse\\_raiseerror](#) (vm \*v, objectsparseerror err)
- value [sparse\\_constructor](#) (vm \*v, int nargs, value \*args)
- value [Sparse\\_getindex](#) (vm \*v, int nargs, value \*args)
- value [Sparse\\_setindex](#) (vm \*v, int nargs, value \*args)
- value [Sparse\\_print](#) (vm \*v, int nargs, value \*args)
- value [Sparse\\_add](#) (vm \*v, int nargs, value \*args)
- value [Sparse\\_sub](#) (vm \*v, int nargs, value \*args)
- value [Sparse\\_mul](#) (vm \*v, int nargs, value \*args)



- [value](#) **Sparse\_div** (vm \*v, int nargs, [value](#) \*args)
- [value](#) **Sparse\_divr** (vm \*v, int nargs, [value](#) \*args)
- **MORPHO\_METHOD** (MORPHO\_GETINDEX\_METHOD, [Sparse\\_getindex](#), BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MORPHO\_SETINDEX\_METHOD, [Sparse\\_setindex](#), BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MORPHO\_PRINT\_METHOD, [Sparse\\_print](#), BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MORPHO\_ADD\_METHOD, [Sparse\\_add](#), BUILTIN\_FLAGSEMPY)
- **MORPHO\_METHOD** (MORPHO\_SUB\_METHOD, [Sparse\\_sub](#), BUILTIN\_FLAGSEMPY)
- MORPHO\_ENDCLASS void **sparse\_initialize** (void)

### 10.13.1 Detailed Description

Veneer class over the objectsparse type that provides sparse matrices.

Author

T J Atherton

### 10.13.2 Function Documentation

#### 10.13.2.1 DEFINE\_VARRAY()

```
DEFINE_VARRAY (
    dokkey ,
    objectdokkey )
```

— CSparse —

#### 10.13.2.2 object\_newsparse()

```
objectsparse* object_newsparse (
    int * nrows,
    int * ncols )
```

Creates a sparse matrix object

Parameters

in	<i>nrows</i>	} Optional number of rows and columns
in	<i>ncols</i>	}

#### 10.13.2.3 object\_sparsefromarray()

```
objectsparse* object_sparsefromarray (
    objectarray * array )
```

Create a sparse array from a list of lists

#### 10.13.2.4 `sparse_add()`

```
objectsparseerror sparse_add (
    objectsparse * a,
    objectsparse * b,
    double alpha,
    double beta,
    objectsparse * out )
```

Add two matrices

##### Parameters

in	<i>a</i>	- sparse matrix
in	<i>b</i>	- sparse matrix
in	<i>alpha</i>	- scale for a
in	<i>beta</i>	- scale for b
out	<i>out</i>	- alpha*a+beta*b.

#### 10.13.2.5 `sparse_checkformat()`

```
bool sparse_checkformat (
    objectsparse * sparse,
    objectsparseformat format,
    bool force )
```

Checks whether a format is available.

##### Parameters

<i>sparse</i>	the matrix to check
<i>format</i>	format to check
<i>force</i>	if format is unavailable, try to make it available

##### Returns

true if the format is available

#### 10.13.2.6 `sparse_clear()`

```
void sparse_clear (
    objectsparse * a )
```

Clears any data attached to a sparse matrix

**10.13.2.7 sparse\_constructor()**

```
value sparse_constructor (
    vm * v,
    int nargs,
    value * args )
```

Constructs a Matrix object

**10.13.2.8 sparse\_div()**

```
objectsparseerror sparse_div (
    objectsparse * a,
    objectmatrix * b,
    objectmatrix * out )
```

Multiply two matrices

**Parameters**

in	<i>a</i>	- sparse matrix
in	<i>b</i>	- dense rhs (may have more than one column)
out	<i>out</i>	- Solution to $a.x = b$ .

**10.13.2.9 sparse\_getelement()**

```
bool sparse_getelement (
    objectsparse * s,
    int row,
    int col,
    value * val )
```

Get an element

**Parameters**

in	<i>s</i>	the sparse object
in	<i>row</i>	the row
in	<i>col</i>	the column
out	<i>val</i>	the value; pass NULL to check if an element exists

**10.13.2.10 Sparse\_getindex()**

```
value Sparse_getindex (
    vm * v,
```

```
int nargs,
value * args )
```

Retrieve a matrix element

#### 10.13.2.11 `sparse_mul()`

```
objectsparseerror sparse_mul (
    objectsparse * a,
    objectsparse * b,
    objectsparse * out )
```

Multiply two matrices

##### Parameters

in	<i>a</i>	- sparse matrix
in	<i>b</i>	- sparse matrix
out	<i>out</i>	- a*b.

#### 10.13.2.12 `sparse_removeformat()`

```
void sparse_removeformat (
    objectsparse * s,
    objectsparseformat format )
```

Removes data structures for a given format

#### 10.13.2.13 `sparse_setelement()`

```
bool sparse_setelement (
    objectsparse * s,
    int row,
    int col,
    value val )
```

Set an element

#### 10.13.2.14 `Sparse_setindex()`

```
value Sparse_setindex (
    vm * v,
    int nargs,
    value * args )
```

Set a matrix element

### 10.13.2.15 `sparse_size()`

```
size_t sparse_size (
    objectsparse * a )
```

Calculate the size of a sparse matrix structure

### 10.13.2.16 `sparseccs_clear()`

```
void sparseccs_clear (
    sparseccs * ccs )
```

Clears all data structures associated with a sparseccs

### 10.13.2.17 `sparseccs_doktoccs()`

```
bool sparseccs_doktoccs (
    sparsedok * in,
    sparseccs * out,
    bool copyvals )
```

Converts a DOK matrix to a CCS matrix

### 10.13.2.18 `sparseccs_get()`

```
bool sparseccs_get (
    sparseccs * ccs,
    int i,
    int j,
    double * val )
```

Retrieves a matrix element (i,j) from a sparseccs structure

#### Returns

true on success.

### 10.13.2.19 `sparseccs_getrowindices()`

```
bool sparseccs_getrowindices (
    sparseccs * ccs,
    int col,
    int * nentries,
    int ** entries )
```

Retrieves the row indices given a column

## Parameters

in	<i>ccs</i>	the matrix
in	<i>col</i>	column index
out	<i>nentries</i>	the number of entries
out	<i>entries</i>	the entries themselves

**10.13.2.20 sparseccs\_init()**

```
void sparseccs_init (
    sparseccs * ccs )
```

Initializes an empty sparseccs

**10.13.2.21 sparseccs\_print()**

```
void sparseccs_print (
    sparseccs * ccs )
```

Prints a sparsedok matrix

**10.13.2.22 sparseccs\_resize()**

```
bool sparseccs_resize (
    sparseccs * ccs,
    int nrows,
    int ncols,
    unsigned int nentries,
    bool values )
```

Resizes a sparseccs

**10.13.2.23 sparseccs\_set()**

```
bool sparseccs_set (
    sparseccs * ccs,
    int i,
    int j,
    double val )
```

Sets a matrix element (i,j) to be a specified value

## Returns

true if the element exists in the given sparsity structure, false otherwise.

#### 10.13.2.24 sparsedok\_clear()

```
void sparsedok_clear (
    sparsedok * dok )
```

Clears a sparsedok structure

#### 10.13.2.25 sparsedok\_get()

```
bool sparsedok_get (
    sparsedok * dok,
    int i,
    int j,
    value * val )
```

Retrieves a matrix element (i,j) from a sparsedok structure

##### Returns

true on success.

#### 10.13.2.26 sparsedok\_init()

```
void sparsedok_init (
    sparsedok * dok )
```

Initializes a sparsedok structure

#### 10.13.2.27 sparsedok\_insert()

```
bool sparsedok_insert (
    sparsedok * dok,
    int i,
    int j,
    value val )
```

Inserts a matrix element (i,j) -> val into a sparsedok structure

##### Returns

true on success.

#### 10.13.2.28 sparsedok\_print()

```
void sparsedok_print (
    sparsedok * dok )
```

Prints a sparsedok matrix

### 10.13.2.29 sparsedok\_remove()

```
bool sparsedok_remove (
    sparsedok * dok,
    int i,
    int j,
    value * val )
```

Removes a matrix element (i,j) from a sparsedok

#### Returns

true on success.

#### Warning

Use sparingly as the deleted key is not recovered.

### 10.13.2.30 sparsedok\_setdimensions()

```
bool sparsedok_setdimensions (
    sparsedok * dok,
    int nrows,
    int ncols )
```

Sets the dimensions of the matrix

#### Returns

true if successful, or false if the dimensions are incompatible with existing matrix entries This function is intended for use in constructing matrix.

## 10.14 datastructures/sparse.h File Reference

Veneer class over the objectsparse type that provides sparse matrices.

```
#include <stdio.h>
#include "object.h"
#include "morpho.h"
```

### Macros

- #define **SPARSE\_CLASSNAME** "Sparse"
- #define **SPARSE\_CONSTRUCTOR** "SprsCns"
- #define **SPARSE\_CONSTRUCTOR\_MSG** "Sparse() should be called either with dimensions or an array initializer."
- #define **SPARSE\_SETFAILED** "SprsSt"
- #define **SPARSE\_SETFAILED\_MSG** "Attempt to set sparse matrix element failed."
- #define **SPARSE\_INVLDARRAYINIT** "SprsInvldInit"
- #define **SPARSE\_INVLDARRAYINIT\_MSG** "Array initializer passed to Sparse() must be a 1 or 2 dimensional array."
- #define **SPARSE\_CONVFAILEDERR** "SprsCnvFld"
- #define **SPARSE\_CONVFAILEDERR\_MSG** "Sparse format conversion failed."
- #define **SPARSE\_OPFAILEDERR** "SprsOpFld"
- #define **SPARSE\_OPFAILEDERR\_MSG** "Sparse matrix operation failed."



## Enumerations

- enum `objectsparseformat` { `SPARSE_DOK`, `SPARSE_CCS` }
- enum `objectsparseerror` { `SPARSE_OK`, `SPARSE_INCMPTBLDIM`, `SPARSE_CONVFAILED`, `SPARSE_E_FAILED` }

## Functions

- void `sparsedok_init` (`sparsedok` \*dok)
- void `sparsedok_clear` (`sparsedok` \*dok)
- bool `sparsedok_insert` (`sparsedok` \*dok, int i, int j, `value` val)
- bool `sparsedok_get` (`sparsedok` \*dok, int i, int j, `value` \*val)
- bool `sparsedok_remove` (`sparsedok` \*dok, int i, int j, `value` \*val)
- bool `sparsedok_setdimensions` (`sparsedok` \*dok, int nrows, int ncols)
- void `sparseccs_init` (`sparseccs` \*ccs)
- void `sparseccs_clear` (`sparseccs` \*ccs)
- bool `sparseccs_resize` (`sparseccs` \*ccs, int nrows, int ncols, unsigned int nentries, bool values)
- bool `sparseccs_get` (`sparseccs` \*ccs, int i, int j, double \*val)
- bool `sparseccs_getrowindices` (`sparseccs` \*ccs, int col, int \*nentries, int \*\*entries)
- bool `sparseccs_doktoccs` (`sparsedok` \*in, `sparseccs` \*out, bool copyvals)
- bool `sparse_setelement` (`objectsparse` \*matrix, int row, int col, `value` value)
- bool `sparse_getelement` (`objectsparse` \*matrix, int row, int col, `value` \*value)
- `objectsparseerror` `sparse_add` (`objectsparse` \*a, `objectsparse` \*b, double alpha, double beta, `objectsparse` \*out)
- `objectsparseerror` `sparse_mul` (`objectsparse` \*a, `objectsparse` \*b, `objectsparse` \*out)
- void `sparse_clear` (`objectsparse` \*a)
- `size_t` `sparse_size` (`objectsparse` \*a)
- `value` `Sparse_divr` (vm \*v, int nargs, `value` \*args)
- void `sparse_initialize` (void)

### 10.14.1 Detailed Description

Veneer class over the `objectsparse` type that provides sparse matrices.

Author

T J Atherton

### 10.14.2 Function Documentation

#### 10.14.2.1 `sparse_add()`

```
objectsparseerror sparse_add (
    objectsparse * a,
    objectsparse * b,
    double alpha,
    double beta,
    objectsparse * out )
```

Add two matrices

**Parameters**

in	<i>a</i>	- sparse matrix
in	<i>b</i>	- sparse matrix
in	<i>alpha</i>	- scale for a
in	<i>beta</i>	- scale for b
out	<i>out</i>	- alpha*a+beta*b.

**10.14.2.2 sparse\_clear()**

```
void sparse_clear (
    objectsparse * a )
```

Clears any data attached to a sparse matrix

**10.14.2.3 sparse\_getelement()**

```
bool sparse_getelement (
    objectsparse * s,
    int row,
    int col,
    value * val )
```

Get an element

**Parameters**

in	<i>s</i>	the sparse object
in	<i>row</i>	the row
in	<i>col</i>	the column
out	<i>val</i>	the value; pass NULL to check if an element exists

**10.14.2.4 sparse\_mul()**

```
objectsparseerror sparse_mul (
    objectsparse * a,
    objectsparse * b,
    objectsparse * out )
```

Multiply two matrices

**Parameters**

in	<i>a</i>	- sparse matrix
in	<i>b</i>	- sparse matrix
out	<i>out</i>	- a*b.

#### 10.14.2.5 `sparse_setelement()`

```
bool sparse_setelement (
    objectsparse * s,
    int row,
    int col,
    value val )
```

Set an element

#### 10.14.2.6 `sparse_size()`

```
size_t sparse_size (
    objectsparse * a )
```

Calculate the size of a sparse matrix structure

#### 10.14.2.7 `sparseccs_clear()`

```
void sparseccs_clear (
    sparseccs * ccs )
```

Clears all data structures associated with a sparseccs

#### 10.14.2.8 `sparseccs_doktoccs()`

```
bool sparseccs_doktoccs (
    sparsedok * in,
    sparseccs * out,
    bool copyvals )
```

Converts a DOK matrix to a CCS matrix

#### 10.14.2.9 `sparseccs_get()`

```
bool sparseccs_get (
    sparseccs * ccs,
    int i,
    int j,
    double * val )
```

Retrieves a matrix element (i,j) from a sparseccs structure

**Returns**

true on success.

**10.14.2.10 sparseccs\_getrowindices()**

```
bool sparseccs_getrowindices (
    sparseccs * ccs,
    int col,
    int * nentries,
    int ** entries )
```

Retrieves the row indices given a column

## Parameters

in	<i>ccs</i>	the matrix
in	<i>col</i>	column index
out	<i>nentries</i>	the number of entries
out	<i>entries</i>	the entries themselves

**10.14.2.11 sparseccs\_init()**

```
void sparseccs_init (
    sparseccs * ccs )
```

Initializes an empty sparseccs

**10.14.2.12 sparseccs\_resize()**

```
bool sparseccs_resize (
    sparseccs * ccs,
    int nrows,
    int ncols,
    unsigned int nentries,
    bool values )
```

Resizes a sparseccs

**10.14.2.13 sparsedok\_clear()**

```
void sparsedok_clear (
    sparsedok * dok )
```

Clears a sparsedok structure

**10.14.2.14 sparsedok\_get()**

```
bool sparsedok_get (
    sparsedok * dok,
    int i,
    int j,
    value * val )
```

Retrieves a matrix element (i,j) from a sparsedok structure

## Returns

true on success.

#### 10.14.2.15 sparsedok\_init()

```
void sparsedok_init (
    sparsedok * dok )
```

Initializes a sparsedok structure

#### 10.14.2.16 sparsedok\_insert()

```
bool sparsedok_insert (
    sparsedok * dok,
    int i,
    int j,
    value val )
```

Inserts a matrix element (i,j) -> val into a sparsedok structure

##### Returns

true on success.

#### 10.14.2.17 sparsedok\_remove()

```
bool sparsedok_remove (
    sparsedok * dok,
    int i,
    int j,
    value * val )
```

Removes a matrix element (i,j) from a sparsedok

##### Returns

true on success.

##### Warning

Use sparingly as the deleted key is not recovered.

#### 10.14.2.18 sparsedok\_setdimensions()

```
bool sparsedok_setdimensions (
    sparsedok * dok,
    int nrows,
    int ncols )
```

Sets the dimensions of the matrix

##### Returns

true if successful, or false if the dimensions are incompatible with existing matrix entries This function is intended for use in constructing matrix.

## 10.15 datastructures/syntaxtree.c File Reference

Syntax tree data structure for morpho.

```
#include <stdio.h>
#include "syntaxtree.h"
#include "common.h"
```

### Functions

- **DEFINE\_VARRAY** ([syntaxtreenode](#), [syntaxtreenode](#))
- **DEFINE\_VARRAY** ([syntaxtreeindx](#), [syntaxtreeindx](#))
- void [syntaxtree\\_init](#) ([syntaxtree](#) \*tree)  
*Initialize a syntax tree.*
- void [syntaxtree\\_clear](#) ([syntaxtree](#) \*tree)  
*Finalize a syntax tree.*
- [syntaxtreeindx](#) [syntaxtree\\_addnode](#) ([syntaxtree](#) \*tree, [syntaxtreenodetype](#) type, [value](#) content, int line, int posn, [syntaxtreeindx](#) left, [syntaxtreeindx](#) right)  
*Adds a node to the syntax tree.*
- [syntaxtreenode](#) \* [syntaxtree\\_nodefromindx](#) ([syntaxtree](#) \*tree, [syntaxtreeindx](#) indx)
- void **syntaxtree\_flatten** ([syntaxtree](#) \*tree, [syntaxtreeindx](#) indx, unsigned int ntypes, [syntaxtreenodetype](#) \*types, [varray\\_syntaxtreeindx](#) \*list)

### 10.15.1 Detailed Description

Syntax tree data structure for morpho.

Author

T J Atherton

### 10.15.2 Function Documentation

#### 10.15.2.1 [syntaxtree\\_addnode\(\)](#)

```
syntaxtreeindx syntaxtree_addnode (
    syntaxtree * tree,
    syntaxtreenodetype type,
    value content,
    int line,
    int posn,
    syntaxtreeindx left,
    syntaxtreeindx right )
```

Adds a node to the syntax tree.

**Parameters**

<i>tree</i>	tree to add to.
<i>type</i>	type of node to add
<i>content</i>	a value to add
<i>left</i>	} left ...
<i>right</i>	} ...and right branches of the node.

**10.15.2.2 syntaxtree\_clear()**

```
void syntaxtree_clear (
    syntaxtree * tree )
```

Finalize a syntax tree.

Free attached objects

**10.15.2.3 syntaxtree\_nodefromindx()**

```
syntaxtreenode* syntaxtree_nodefromindx (
    syntaxtree * tree,
    syntaxtreeindx indx )
```

Gets a syntaxtree node from its index

**10.16 datastructures/syntaxtree.h File Reference**

Syntax tree data structure for morpho.

```
#include <stddef.h>
#include "value.h"
#include "varray.h"
```

**Classes**

- struct [\\_syntaxtreenode](#)  
A node on the syntax tree is defined by a value and indices of the left and right elements.
- struct [syntaxtree](#)

**Macros**

- #define **SYNTAXTREE\_ISLEAF**(x) syntaxtree\_istype(x, NODE\_BASE, NODE\_LEAF)
- #define **SYNTAXTREE\_ISUNARY**(x) syntaxtree\_istype(x, NODE\_LEAF, NODE\_UNARY)
- #define **SYNTAXTREE\_ISOPERATOR**(x) syntaxtree\_istype(x, NODE\_UNARY, NODE\_OPERATOR)
- #define **SYNTAXTREE\_ISSTATEMENT**(x) syntaxtree\_istype(x, NODE\_OPERATOR, NODE\_STATEMENT↵  
NT)
- #define **SYNTAXTREE\_UNCONNECTED** -1



## Typedefs

- typedef ptrdiff\_t **syntaxtreeindx**
- typedef struct **\_syntaxtreenode** syntaxtreenode

*A node on the syntax tree is defined by a value and indices of the left and right elements.*

## Enumerations

- enum **syntaxtreenodetype** {  
**NODE\_BASE**, **NODE\_NIL**, **NODE\_BOOL**, **NODE\_FLOAT**,  
**NODE\_INTEGER**, **NODE\_STRING**, **NODE\_SYMBOL**, **NODE\_SELF**,  
**NODE\_SUPER**, **NODE\_LEAF**, **NODE\_NEGATE**, **NODE\_NOT**,  
**NODE\_UNARY**, **NODE\_ADD**, **NODE\_SUBTRACT**, **NODE\_MULTIPLY**,  
**NODE\_DIVIDE**, **NODE\_POW**, **NODE\_ASSIGN**, **NODE\_EQ**,  
**NODE\_NEQ**, **NODE\_LT**, **NODE\_GT**, **NODE\_LTEQ**,  
**NODE\_GTEQ**, **NODE\_AND**, **NODE\_OR**, **NODE\_DOT**,  
**NODE\_RANGE**, **NODE\_OPERATOR**, **NODE\_PRINT**, **NODE\_DECLARATION**,  
**NODE\_FUNCTION**, **NODE\_METHOD**, **NODE\_CLASS**, **NODE\_RETURN**,  
**NODE\_IF**, **NODE\_THEN**, **NODE\_WHILE**, **NODE\_FOR**,  
**NODE\_IN**, **NODE\_STATEMENT**, **NODE\_GROUPING**, **NODE\_SEQUENCE**,  
**NODE\_INTERPOLATION**, **NODE\_ARGLIST**, **NODE\_SCOPE**, **NODE\_CALL**,  
**NODE\_INDEX**, **NODE\_LIST**, **NODE\_IMPORT** }

*Type of node.*

## Functions

- **DECLARE\_VARRAY** (syntaxtreenode, syntaxtreenode)
- **DECLARE\_VARRAY** (syntaxtreeindx, syntaxtreeindx)
- void **syntaxtree\_init** (syntaxtree \*tree)  
*Initialize a syntax tree.*
- void **syntaxtree\_clear** (syntaxtree \*tree)  
*Finalize a syntax tree.*
- void **syntaxtree\_print** (syntaxtree \*tree)
- syntaxtreeindx **syntaxtree\_addnode** (syntaxtree \*tree, syntaxtreenodetype type, value content, int line, int posn, syntaxtreeindx left, syntaxtreeindx right)  
*Adds a node to the syntax tree.*
- syntaxtreenode \* **syntaxtree\_nodefromindx** (syntaxtree \*tree, syntaxtreeindx indx)
- void **syntaxtree\_flatten** (syntaxtree \*tree, syntaxtreeindx indx, unsigned int ntypes, syntaxtreenodetype \*types, varray\_syntaxtreeindx \*list)

### 10.16.1 Detailed Description

Syntax tree data structure for morpho.

Author

T J Atherton

### 10.16.2 Function Documentation

### 10.16.2.1 syntaxtree\_addnode()

```
syntaxtreeindx syntaxtree_addnode (
    syntaxtree * tree,
    syntaxtreenodetype type,
    value content,
    int line,
    int posn,
    syntaxtreeindx left,
    syntaxtreeindx right )
```

Adds a node to the syntax tree.

#### Parameters

<i>tree</i>	tree to add to.
<i>type</i>	type of node to add
<i>content</i>	a value to add
<i>left</i>	} left ...
<i>right</i>	} ...and right branches of the node.

### 10.16.2.2 syntaxtree\_clear()

```
void syntaxtree_clear (
    syntaxtree * tree )
```

Finalize a syntax tree.

Free attached objects

### 10.16.2.3 syntaxtree\_nodefromindx()

```
syntaxtreenode* syntaxtree_nodefromindx (
    syntaxtree * tree,
    syntaxtreeindx indx )
```

Gets a syntaxtree node from its index

## 10.17 datastructures/value.c File Reference

Fundamental data type for morpho.

```
#include "value.h"
#include "common.h"
```

## Functions

- **DEFINE\_VARRAY** ([value](#), [value](#))
- bool [varray\\_valuefind](#) (varray\_value \*varray, [value](#) v, unsigned int \*out)  
*Finds a value in an varray using a loose equality test (MORPHO\_ISEQUAL)*
- bool [varray\\_valuefindsame](#) (varray\_value \*varray, [value](#) v, unsigned int \*out)  
*Finds a value in an varray using strict equality test (MORPHO\_ISSAME)*
- bool [value\\_promotenumberlist](#) (unsigned int nv, [value](#) \*v)

### 10.17.1 Detailed Description

Fundamental data type for morpho.

Author

T J Atherton

### 10.17.2 Function Documentation

#### 10.17.2.1 [value\\_promotenumberlist\(\)](#)

```
bool value_promotenumberlist (
    unsigned int nv,
    value * v )
```

Promotes a list of numbers to floats if any are floating point.

Parameters

in	<i>nv</i>	- number of values
in	<i>v</i>	- list of values

Returns

true if successful, false if any values are not numbers

#### 10.17.2.2 [varray\\_valuefind\(\)](#)

```
bool varray_valuefind (
    varray_value * varray,
    value v,
    unsigned int * out )
```

Finds a value in an varray using a loose equality test (MORPHO\_ISEQUAL)

**Parameters**

in	<i>varray</i>	the array to search
in	<i>v</i>	value to find
out	<i>out</i>	index of the match

**Returns**

whether the value was found or not.

**10.17.2.3 varray\_valuefindsame()**

```
bool varray_valuefindsame (
    varray_value * varray,
    value v,
    unsigned int * out )
```

Finds a value in an varray using strict equality test (MORPHO\_ISSAME)

**Parameters**

in	<i>varray</i>	the array to search
in	<i>v</i>	value to find
out	<i>out</i>	index of the match

**Returns**

whether the value was found or not.

**10.18 datastructures/value.h File Reference**

Fundamental data type for morpho.

```
#include <stdint.h>
#include <stdbool.h>
#include "build.h"
#include "varray.h"
```

**Classes**

- struct [value](#)

*The unboxed value type.*

## Macros

- #define **MORPHO\_GETTYPE**(v) ((v).type)
- #define **MORPHO\_ISNIL**(v) ((v).type==VALUE\_NIL)
- #define **MORPHO\_ISINTEGER**(v) ((v).type==VALUE\_INTEGER)
- #define **MORPHO\_ISFLOAT**(v) ((v).type==VALUE\_DOUBLE)
- #define **MORPHO\_ISBOOL**(v) ((v).type==VALUE\_BOOL)
- #define **MORPHO\_ISOBJECT**(v) ((v).type==VALUE\_OBJECT)
- #define **MORPHO\_NIL** ((value) { VALUE\_NIL, .as.integer = (int) 0 })
- #define **MORPHO\_INTEGER**(x) ((value) { VALUE\_INTEGER, .as.integer = (int) (x) })
- #define **MORPHO\_FLOAT**(x) ((value) { VALUE\_DOUBLE, .as.real = (double) x })
- #define **MORPHO\_BOOL**(x) ((value) { VALUE\_BOOL, .as.boolean = (bool) x })
- #define **MORPHO\_OBJECT**(x) ((value) { VALUE\_OBJECT, .as.obj = (object \*) x })
- #define **MORPHO\_TRUE** MORPHO\_BOOL(true)
- #define **MORPHO\_FALSE** MORPHO\_BOOL(false)
- #define **MORPHO\_GETINTEGERVALUE**(v) ((v).as.integer)
- #define **MORPHO\_GETFLOATVALUE**(v) ((v).as.real)
- #define **MORPHO\_GETBOOLVALUE**(v) ((v).as.boolean)
- #define **MORPHO\_GETOBJECT**(v) ((v).as.obj)
- #define **MORPHO\_ISNUMBER**(v) (morpho\_isnumber(v))
- #define **MORPHO\_INTEGERTOFloat**(x) (MORPHO\_FLOAT((double) MORPHO\_GETINTEGERVALUE((x))))
- #define **MORPHO\_ISFALSE**(x) (morpho\_isfalse(x))
- #define **MORPHO\_ISTRUE**(x) (!morpho\_isfalse(x))

## Typedefs

- typedef struct **object** object

## Enumerations

- enum **valuetype** {  
**VALUE\_NIL**, **VALUE\_INTEGER**, **VALUE\_DOUBLE**, **VALUE\_BOOL**,  
**VALUE\_OBJECT** }

*A Morpho value.*

## Functions

- **DECLARE\_VARRAY** (value, value)
- bool **varray\_valuefind** (varray\_value \*varray, value v, unsigned int \*out)  
*Finds a value in an varray using a loose equality test (MORPHO\_ISEQUAL)*
- bool **varray\_valuefindsame** (varray\_value \*varray, value v, unsigned int \*out)  
*Finds a value in an varray using strict equality test (MORPHO\_ISSAME)*
- bool **value\_promotenumberslist** (unsigned int nv, value \*v)

### 10.18.1 Detailed Description

Fundamental data type for morpho.

Author

T J Atherton

## 10.18.2 Macro Definition Documentation

### 10.18.2.1 MORPHO\_GETINTEGERVALUE

```
#define MORPHO_GETINTEGERVALUE(  
    v ) ((v).as.integer)
```

Get a value

### 10.18.2.2 MORPHO\_GETTYPE

```
#define MORPHO_GETTYPE(  
    v ) ((v).type)
```

This macro gets the type of the value.

#### Warning

Not intended for broad use.

### 10.18.2.3 MORPHO\_INTEGERTOFLOAT

```
#define MORPHO_INTEGERTOFLOAT(  
    x ) (MORPHO_FLOAT((double) MORPHO_GETINTEGERVALUE((x))))
```

Conversion

### 10.18.2.4 MORPHO\_ISNIL

```
#define MORPHO_ISNIL(  
    v ) ((v).type==VALUE_NIL)
```

Test for the type of a value

### 10.18.2.5 MORPHO\_NIL

```
#define MORPHO_NIL ((value) { VALUE_NIL, .as.integer = (int) 0 })
```

Create a literal

## 10.18.3 Enumeration Type Documentation

### 10.18.3.1 valuetype

enum `valuetype`

A Morpho value.

A enumerated type defining the different types available in Morpho.

## 10.18.4 Function Documentation

### 10.18.4.1 value\_promotenumberlist()

```
bool value_promotenumberlist (
    unsigned int nv,
    value * v )
```

Promotes a list of numbers to floats if any are floating point.

#### Parameters

in	<i>nv</i>	- number of values
in	<i>v</i>	- list of values

#### Returns

true if successful, false if any values are not numbers

### 10.18.4.2 varray\_valuefind()

```
bool varray_valuefind (
    varray_value * varray,
    value v,
    unsigned int * out )
```

Finds a value in an varray using a loose equality test (MORPHO\_ISEQUAL)

#### Parameters

in	<i>varray</i>	the array to search
in	<i>v</i>	value to find
out	<i>out</i>	index of the match

**Returns**

whether the value was found or not.

**10.18.4.3 varray\_valuefindsame()**

```
bool varray_valuefindsame (
    varray_value * varray,
    value v,
    unsigned int * out )
```

Finds a value in an varray using strict equality test (MORPHO\_ISSAME)

**Parameters**

in	<i>varray</i>	the array to search
in	<i>v</i>	value to find
out	<i>out</i>	index of the match

**Returns**

whether the value was found or not.

**10.19 datastructures/varray.c File Reference**

Dynamically resizing array (varray) data structure.

```
#include "varray.h"
```

**Functions**

- **DEFINE\_VARRAY** (char, char)
- **DEFINE\_VARRAY** (double, double)
- unsigned int [varray\\_powerof2ceiling](#) (unsigned int n)

*Computes the nearest power of 2 above an integer.*

**10.19.1 Detailed Description**

Dynamically resizing array (varray) data structure.

**Author**

T J Atherton



## 10.19.2 Function Documentation

### 10.19.2.1 varray\_powerof2ceiling()

```
unsigned int varray_powerof2ceiling (  
    unsigned int n )
```

Computes the nearest power of 2 above an integer.

#### Parameters

<i>n</i>	An integer
----------	------------

#### Returns

Nearest power of 2 above n See: <http://graphics.stanford.edu/~seander/bithacks.html#RoundUpPowerOf2Float>

## 10.20 datastructures/varray.h File Reference

Dynamically resizing array (varray) data structure.

```
#include <stdlib.h>  
#include <stdbool.h>  
#include "memory.h"
```

### Macros

- #define **DECLARE\_VARRAY**(name, type)  
*Creates a generic varray containing a specified type.*
- #define **DEFINE\_VARRAY**(name, type)

### Functions

- **DECLARE\_VARRAY** (char, char)
- **DECLARE\_VARRAY** (double, double)
- unsigned int **varray\_powerof2ceiling** (unsigned int n)  
*Computes the nearest power of 2 above an integer.*

### 10.20.1 Detailed Description

Dynamically resizing array (varray) data structure.

#### Author

T J Atherton

## 10.20.2 Macro Definition Documentation

### 10.20.2.1 DECLARE\_VARRAY

```
#define DECLARE_VARRAY (
    name,
    type )
```

#### Value:

```
typedef struct { \
    unsigned int count; \
    unsigned int capacity; \
    type *data; \
} varray_##name; \
\
void varray_##name##_init(varray_##name *v); \
bool varray_##name##_add(varray_##name *v, type *data, int count); \
bool varray_##name##_resize(varray_##name *v, int count); \
int varray_##name##_write(varray_##name *v, type data); \
void varray_##name##_clear(varray_##name *v);
```

Creates a generic varray containing a specified type.

#### Variable array macros

Varrays only differ by their contents, and so we use macros to conveniently define types and functions. To use these: First, call [DECLARE\\_VARRAY\(NAME,TYPE\)](#) in your .h file with a selected name for your varray and the type of thing you want to store in it. This will define:

1. A type called varray\_NAME (where NAME is the name you gave).
2. Functions varray\_NAME\_init(v) - Initializes the varray varray\_NAME\_add(v, data[], count) - Adds elements to the varray varray\_NAME\_write(v, data) - Writes a single element to the varray, returning the index varray\_NAME\_clear(v) - Clears the varray, freeing memory Then, call DEFINE\_VARRAY(NAME,TYPE) in your .c file to define the appropriate functions

## 10.20.3 Function Documentation

### 10.20.3.1 varray\_powerof2ceiling()

```
unsigned int varray_powerof2ceiling (
    unsigned int n )
```

Computes the nearest power of 2 above an integer.

#### Parameters

<i>n</i>	An integer
----------	------------

## Returns

Nearest power of 2 above n See: <http://graphics.stanford.edu/~seander/bithacks.html#RoundUpPowerOf2Float>

## 10.21 geometry/mesh.c File Reference

Mesh class and associated functionality.

```
#include "object.h"
#include "builtin.h"
#include "mesh.h"
#include "file.h"
#include "varray.h"
#include "parse.h"
#include "sparse.h"
```

### Functions

- `objectmesh * object_newmesh` (unsigned int dim, unsigned int nv, double \*v)
- `bool mesh_checkconnectivity` (objectmesh \*mesh)
- `objectsparse * mesh_getconnectivityelement` (objectmesh \*mesh, unsigned int row, unsigned int col, bool create)
- `bool mesh_addelementwithvertices` (objectmesh \*mesh, grade g, vertexid \*v)
- `objectmesh * mesh_load` (char \*file)
- `value mesh_constructor` (vm \*v, int nargs, value \*args)
- `value Mesh_print` (vm \*v, int nargs, value \*args)
- MORPHO\_ENDCLASS void `mesh_initialize` (void)

### 10.21.1 Detailed Description

Mesh class and associated functionality.

## Author

T J Atherton

### 10.21.2 Function Documentation

#### 10.21.2.1 mesh\_addelementwithvertices()

```
bool mesh_addelementwithvertices (
    objectmesh * mesh,
    grade g,
    vertexid * v )
```

Adds an element to a mesh

## Parameters

in	<i>mesh</i>	the mesh
in	<i>g</i>	grade of the element
in	<i>v</i>	vertexids

**10.21.2.2 mesh\_checkconnectivity()**

```
bool mesh_checkconnectivity (
    objectmesh * mesh )
```

Ensures a mesh has a valid connectivity matrix

**10.21.2.3 mesh\_constructor()**

```
value mesh_constructor (
    vm * v,
    int nargs,
    value * args )
```

Constructs a Matrix object

**10.21.2.4 mesh\_getconnectivityelement()**

```
objectsparse* mesh_getconnectivityelement (
    objectmesh * mesh,
    unsigned int row,
    unsigned int col,
    bool create )
```

Gets the connectivity matrix corresponding to (row, col); creates one if create is set

**10.21.2.5 mesh\_load()**

```
objectmesh* mesh_load (
    char * file )
```

Loads a .mesh file.

**10.21.2.6 object\_newmesh()**

```
objectmesh* object_newmesh (
    unsigned int dim,
    unsigned int nv,
    double * v )
```

Creates a mesh object

## 10.22 geometry/mesh.h File Reference

Mesh class and associated functionality.

### Macros

- `#define MESH_CLASSNAME "Mesh"`
- `#define MESH_VERTSECTION "vertices"`
- `#define MESH_EDGESECTION "edges"`
- `#define MESH_FACESECTION "faces"`
- `#define MESH_VOLSECTION "volumes"`

### Typedefs

- `typedef int grade`
- `typedef int vertexid`

### Functions

- `void mesh_initialize (void)`

#### 10.22.1 Detailed Description

Mesh class and associated functionality.

#### Author

T J Atherton

## 10.23 interface/cli.c File Reference

Command line interface.

```
#include <time.h>
#include "cli.h"
#include "parse.h"
```

### Macros

- `#define CLI_BUFFER_SIZE 1024`

## Functions

- void `cli_reporterror` (`error` \*err, vm \*v)
- bool `cli_lex` (char \*in, void \*\*ref, `linedit_token` \*out)
- bool `cli_complete` (char \*in, `linedit_stringlist` \*c)
- void `cli_help` (`lineditor` \*edit, char \*query, `error` \*err, bool avail)
- void `cli` (clioptions opt)
  - Provide a command line interface.*
- void `cli_disassemblewithsrc` (program \*p, char \*src)
- void `cli_run` (const char \*in, clioptions opt)

## Variables

- `linedit_color cli_tokencolors` []

### 10.23.1 Detailed Description

Command line interface.

#### Author

T J Atherton

### 10.23.2 Function Documentation

#### 10.23.2.1 `cli_complete()`

```
bool cli_complete (  
    char * in,  
    linedit_stringlist * c )
```

Autocomplete function

#### 10.23.2.2 `cli_disassemblewithsrc()`

```
void cli_disassemblewithsrc (  
    program * p,  
    char * src )
```

Disassembles the program showing syntax colored lines of source

### 10.23.2.3 cli\_help()

```
void cli_help (
    lineditor * edit,
    char * query,
    error * err,
    bool avail )
```

Interactive help

### 10.23.2.4 cli\_lex()

```
bool cli_lex (
    char * in,
    void ** ref,
    linedit_token * out )
```

A tokenizer for syntax coloring that leverages the parser's lexer

### 10.23.2.5 cli\_reporterror()

```
void cli_reporterror (
    error * err,
    vm * v )
```

Report an error if one has occurred.

### 10.23.2.6 cli\_run()

```
void cli_run (
    const char * in,
    clioptions opt )
```

Loads and runs a file.

## 10.23.3 Variable Documentation

### 10.23.3.1 cli\_tokencolors

```
linedit_color cli_tokencolors[]
```

Define colors for different token types

## 10.24 interface/cli.h File Reference

Command line interface.

```
#include "morpho.h"
#include "varray.h"
#include "error.h"
#include "linedit.h"
#include "help.h"
```

### Macros

- `#define CLI_PROMPT ">"`
- `#define CLI_CONTINUATIONPROMPT "~"`
- `#define CLI_QUIT "quit"`
- `#define CLI_HELP "help"`
- `#define CLI_SHORT_HELP "?"`
- `#define CLI_NORMALCODE "\033[0m"`
- `#define CLI_REDCODE "\033[0;31m"`
- `#define CLI_BLUECODE "\033[0;34m"`
- `#define CLI_ERRORCOLOR ""`
- `#define CLI_NORMALTEXT ""`
- `#define CLI_RUN 0x1`
- `#define CLI_DISASSEMBLE 0x2`
- `#define CLI_DISASSEMBLESHOWSRC 0x4`

### Typedefs

- `typedef unsigned int clioptions`

### Functions

- void [cli\\_run](#) (const char \*in, clioptions opt)
- void [cli](#) (clioptions opt)

*Provide a command line interface.*

#### 10.24.1 Detailed Description

Command line interface.

Author

T J Atherton

#### 10.24.2 Function Documentation



### 10.24.2.1 cli\_run()

```
void cli_run (
    const char * in,
    clioptions opt )
```

Loads and runs a file.

## 10.25 interface/help.c File Reference

Interactive help system.

```
#include <string.h>
#include <ctype.h>
#include <dirent.h>
#include "help.h"
#include "dictionary.h"
#include "parse.h"
#include "common.h"
```

### Macros

- #define **HELP\_LINELENGTH** 2048
- #define **HELP\_MAXLEVEL** 6

### Functions

- [objecthelptopic](#) \* [help\\_newtopic](#) (char \*topic, char \*file, long int location, [objecthelptopic](#) \*parent)
- void [help\\_cleartopic](#) ([objecthelptopic](#) \*topic)
- size\_t [help\\_querylength](#) (char \*query, char \*\*s)
- [objecthelptopic](#) \* [help\\_search](#) (char \*query)
- void [help\\_display](#) ([lineditor](#) \*edit, [objecthelptopic](#) \*topic)
- value [help\\_parsetopicname](#) (char \*line)
- value [help\\_parsetag](#) (char \*line)
- int [help\\_parsetopiclevel](#) (char \*line)
- bool [help\\_load](#) (char \*file)
- bool [help\\_searchpath](#) (char \*path)
- bool [help\\_initialize](#) (void)
- void [help\\_finalize](#) (void)

### 10.25.1 Detailed Description

Interactive help system.

#### Author

T J Atherton

## 10.25.2 Function Documentation

### 10.25.2.1 `help_cleartopic()`

```
void help_cleartopic (
    objecthelptopic * topic )
```

Free attached data from a help topic

### 10.25.2.2 `help_display()`

```
void help_display (
    lineditor * edit,
    objecthelptopic * topic )
```

Displays a help topic

### 10.25.2.3 `help_finalize()`

```
void help_finalize (
    void )
```

Finalizes the help system

### 10.25.2.4 `help_initialize()`

```
bool help_initialize (
    void )
```

Initializes the help system

#### Returns

true if help is available

### 10.25.2.5 `help_load()`

```
bool help_load (
    char * file )
```

Loads a help file

### Parameters

<i>file</i>	file to load
-------------	--------------

### Returns

true if any help entries were successfully loaded

#### 10.25.2.6 help\_newtopic()

```
objecthelptopic* help_newtopic (
    char * topic,
    char * file,
    long int location,
    objecthelptopic * parent )
```

Create a new help topic

#### 10.25.2.7 help\_parsetag()

```
value help_parsetag (
    char * line )
```

Parses a tag, returning it as a Morpho string converted to lower case.

### Warning

the input line is modified in the process

#### 10.25.2.8 help\_parsetopiclevel()

```
int help_parsetopiclevel (
    char * line )
```

Determines the level of topic from the markdown header level

#### 10.25.2.9 help\_parsetopicname()

```
value help_parsetopicname (
    char * line )
```

Parses a topic name, returning it as a Morpho string converted to lower case.

### Warning

the input line is modified in the process

#### 10.25.2.10 help\_querylength()

```
size_t help_querylength (
    char * query,
    char ** s )
```

Determine starting point and length of a query

**Parameters**

in	<i>query</i>	- the query to examine
out	<i>s</i>	- starting character of the query (optional)

**Returns**

length of the query (0 indicates no query present)

**10.25.2.11 help\_search()**

```
objecthelptopic* help_search (
    char * query )
```

Searches for a given query in the help system

**10.25.2.12 help\_searchpath()**

```
bool help_searchpath (
    char * path )
```

Searches for help files

**Parameters**

<i>path</i>	directory to search (recursively)
-------------	-----------------------------------

**Returns**

true if any help files were successfully processed.

## 10.26 interface/help.h File Reference

Interactive help system.

```
#include <stdio.h>
#include "morpho.h"
#include "object.h"
#include "linedit.h"
```

**Classes**

- struct [subjecthelptopic](#)

## Macros

- `#define HELP_INDEXPAGE "help"`

## Typedefs

- `typedef struct subjecthelptopic objecthelptopic`

## Functions

- `size_t help\_querylength (char *query, char **s)`
- `objecthelptopic * help\_search (char *query)`
- `void help\_display (lineditor *edit, objecthelptopic *topic)`
- `bool help\_initialize (void)`
- `void help\_finalize (void)`

### 10.26.1 Detailed Description

Interactive help system.

Author

T J Atherton

### 10.26.2 Function Documentation

#### 10.26.2.1 `help_display()`

```
void help_display (
    lineditor * edit,
    objecthelptopic * topic )
```

Displays a help topic

#### 10.26.2.2 `help_finalize()`

```
void help_finalize (
    void )
```

Finalizes the help system

### 10.26.2.3 help\_initialize()

```
bool help_initialize (
    void )
```

Initializes the help system

#### Returns

true if help is available

### 10.26.2.4 help\_querylength()

```
size_t help_querylength (
    char * query,
    char ** s )
```

Determine starting point and length of a query

#### Parameters

in	<i>query</i>	- the query to examine
out	<i>s</i>	- starting character of the query (optional)

#### Returns

length of the query (0 indicates no query present)

### 10.26.2.5 help\_search()

```
objecthelptopic* help_search (
    char * query )
```

Searches for a given query in the help system

## 10.27 interface/linedit.c File Reference

A line editor with history, autocomplete and syntax highlighting.

```
#include "linedit.h"
```

## Classes

- struct [keypress](#)

## Macros

- #define [lineno\\_MINIMUMSTRINGSIZE](#) 8
- #define [LINEDIT\\_CODESTRINGSIZE](#) 24
- #define [LINEDIT\\_DEBUGKEYPRESS](#)
- #define [LINEDIT\\_UNSUPPORTEDBUFFER](#) 4096

## Enumerations

- enum [keytype](#) { UNKNOWN, CHARACTER, RETURN, TAB, DELETE, UP, DOWN, LEFT, RIGHT, HOME, END, SHIFT\_LEFT, SHIFT\_RIGHT, CTRL }
- enum [lineno\\_terminaltype](#) { LINEDIT\_NOTTTY, LINEDIT\_UNSUPPORTED, LINEDIT\_SUPPORTED }
- enum [keycodes](#) { TAB\_CODE = 9, RETURN\_CODE = 13, ESC\_CODE = 27, DELETE\_CODE = 127 }

## Functions

- void [lineno\\_stringinit](#) (lineno\_string \*string)
- void [lineno\\_stringclear](#) (lineno\_string \*string)
- bool [lineno\\_stringresize](#) (lineno\_string \*string, size\_t size)
- void [lineno\\_stringaddcharacter](#) (lineno\_string \*string, char \*c, size\_t n)
- void [lineno\\_stringinsert](#) (lineno\_string \*string, size\_t posn, char \*c, size\_t n)  
*Inserts characters at a given position.*
- void [lineno\\_stringdelete](#) (lineno\_string \*string, size\_t posn, size\_t n)  
*Deletes characters at a given position.*
- void [lineno\\_stringaddcstring](#) (lineno\_string \*string, char \*s)
- char \* [lineno\\_cstring](#) (lineno\_string \*string)
- lineno\_string \* [lineno\\_newstring](#) (char \*string)
- void [lineno\\_stringlistadd](#) (lineno\_stringlist \*list, char \*string)
- void [lineno\\_stringlistinit](#) (lineno\_stringlist \*list)
- void [lineno\\_stringlistclear](#) (lineno\_stringlist \*list)
- void [lineno\\_stringlistremove](#) (lineno\_stringlist \*list, lineno\_string \*string)
- lineno\_string \* [lineno\\_stringlistselect](#) (lineno\_stringlist \*list, unsigned int n, unsigned int \*m)
- void [lineno\\_historyadd](#) (lineno\_editor \*edit, char \*string)
- void [lineno\\_historyclear](#) (lineno\_editor \*edit)
- unsigned int [lineno\\_historyselect](#) (lineno\_editor \*edit, unsigned int n)
- void [lineno\\_historyadvance](#) (lineno\_editor \*edit, unsigned int n)
- bool [lineno\\_atendoffline](#) (lineno\_editor \*edit)
- void [lineno\\_generatesuggestions](#) (lineno\_editor \*edit)
- bool [lineno\\_aresuggestionsavailable](#) (lineno\_editor \*edit)
- char \* [lineno\\_currentsuggestion](#) (lineno\_editor \*edit)
- void [lineno\\_advancesuggestions](#) (lineno\_editor \*edit, unsigned int n)
- void [lineno\\_enablerawmode](#) (void)  
*Enables 'raw' mode in the terminal.*
- void [lineno\\_disablerawmode](#) (void)

- Restore terminal state to normal.*
- int `linedit_cstrcasecmp` (char \*str1, char \*str2)
  - Compares two c strings independently of case.*
- void `linedit_write` (char \*string)
  - Writes a string to the terminal.*
- void `linedit_writechar` (char c)
  - Writes a character to the terminal.*
- bool `linedit_readkey` (lineditor \*edit, keypress \*out)
  - Read and decode a single keypress from the terminal.*
- void `linedit_move` (linedit\_string \*out, int posn)
  - Writes a control sequence to move to a given position.*
- void `linedit_setdefaulttext` (linedit\_string \*out)
  - Writes a control sequence to reset default text.*
- void `linedit_setcolor` (linedit\_string \*out, linedit\_color col)
  - Writes a control sequence to set a given color.*
- void `linedit_setemphasis` (linedit\_string \*out, linedit\_emphasis emph)
  - Writes a control sequence to set a given emphasis.*
- void `linedit_erasetoendofline` (linedit\_string \*out)
  - Writes a control sequence to erase the rest of the line.*
- void `linedit_eraseline` (linedit\_string \*out)
  - Writes a control sequence to erase the whole line.*
- void `linedit_addcstringwithselection` (lineditor \*edit, char \*in, size\_t offset, size\_t length, linedit\_color \*col, linedit\_string \*out)
- void `linedit_syntaxcolorstring` (lineditor \*edit, linedit\_string \*in, linedit\_string \*out)
- void `linedit_showstring` (lineditor \*edit, linedit\_string \*in, linedit\_string \*out)
- void `linedit_refreshline` (lineditor \*edit)
- void `linedit_setmode` (lineditor \*edit, lineditormode mode)
- lineditormode `linedit_getmode` (lineditor \*edit)
- void `linedit_setposition` (lineditor \*edit, int posn)
- void `linedit_advanceposition` (lineditor \*edit, int delta)
  - Advances the position by delta.*
- bool `linedit_processkeypress` (lineditor \*edit)
- void `linedit_noterminal` (lineditor \*edit)
- void `linedit_unsupported` (lineditor \*edit)
- void `linedit_supported` (lineditor \*edit)
- char \* `linedit` (lineditor \*edit)
- void `linedit_syntaxcolor` (lineditor \*edit, linedit\_tokenizer tokenizer, linedit\_color \*cols, unsigned int ncols)
  - Configures syntax coloring.*
- void `linedit_autocomplete` (lineditor \*edit, linedit\_completer completer)
  - Configures autocomplete.*
- void `linedit_addsuggestion` (linedit\_stringlist \*completion, char \*string)
  - Adds a completion suggestion.*
- void `linedit_setprompt` (lineditor \*edit, char \*prompt)
  - Sets the prompt.*
- void `linedit_displaywithstyle` (lineditor \*edit, char \*string, linedit\_color col, linedit\_emphasis emph)
  - Displays a string with a given color and emphasis.*
- void `linedit_displaywithsyntaxcoloring` (lineditor \*edit, char \*string)
  - Displays a string with syntax coloring.*
- void `linedit_init` (lineditor \*edit)
- void `linedit_clear` (lineditor \*edit)



## Variables

- struct `termios` [termios](#)
- bool `termexitregistered` =false

### 10.27.1 Detailed Description

A line editor with history, autocomplete and syntax highlighting.

#### Author

T J Atherton

### 10.27.2 Macro Definition Documentation

#### 10.27.2.1 `LINEDIT_CODESTRINGSIZE`

```
#define LINEDIT_CODESTRINGSIZE 24
```

Maximum escape code size

#### 10.27.2.2 `LINEDIT_DEBUGKEYPRESS`

```
#define LINEDIT_DEBUGKEYPRESS
```

Enable this macro to get reports on unhandled keypresses

#### 10.27.2.3 `LINEDIT_UNSUPPORTEDBUFFER`

```
#define LINEDIT_UNSUPPORTEDBUFFER 4096
```

If the terminal is unsupported, default to fgets with a fixed buffer

### 10.27.3 Enumeration Type Documentation

#### 10.27.3.1 `keycodes`

```
enum keycodes
```

Raw codes produced by the terminal

### 10.27.3.2 keytype

enum `keytype`

Identifies the type of keypress

## 10.27.4 Function Documentation

### 10.27.4.1 linedit()

```
char* linedit (  
    lineditor * edit )
```

Public interface to the line editor.

#### Parameters

<i>edit</i>	- a line editor that has been initialized with <code>linedit_init</code> .
-------------	--

#### Returns

the string input by the user, or NULL if nothing entered.

Ensure we are not passed a NULL pointer

### 10.27.4.2 linedit\_addsuggestion()

```
void linedit_addsuggestion (  
    linedit_stringlist * completion,  
    char * string )
```

Adds a completion suggestion.

#### Parameters

<i>completion</i>	completion data structure
<i>string</i>	string to add

### 10.27.4.3 linedit\_advanceposition()

```
void linedit_advanceposition (  
    lineditor * edit,  
    int delta )
```

Advances the position by delta.

We ensure that the current position also lies within the string.

#### 10.27.4.4 linenoedit\_advancesuggestions()

```
void linenoedit_advancesuggestions (
    linenoeditor * edit,
    unsigned int n )
```

Advance through the suggestions

#### 10.27.4.5 linenoedit\_aresuggestionsavailable()

```
bool linenoedit_aresuggestionsavailable (
    linenoeditor * edit )
```

Check whether any suggestions are available

#### 10.27.4.6 linenoedit\_autocomplete()

```
void linenoedit_autocomplete (
    linenoeditor * edit,
    linenoedit_completer completer )
```

Configures autocomplete.

##### Parameters

<i>edit</i>	Line editor to configure
<i>completer</i>	a function

#### 10.27.4.7 linenoedit\_checksupport()

```
linenoedit_terminaltype linenoedit_checksupport (
    void )
```

Checks whether the terminal is supported

#### 10.27.4.8 linenoedit\_clear()

```
void linenoedit_clear (
    linenoeditor * edit )
```

Finalize a line editor

#### 10.27.4.9 `linedit_cstrcasecmp()`

```
int linedit_cstrcasecmp (
    char * str1,
    char * str2 )
```

Compares two c strings independently of case.

##### Parameters

in	<i>str1</i>	- } strings to compare
in	<i>str2</i>	- }

##### Returns

0 if the strings are identical, otherwise a positive or negative number indicating their lexicographic order

#### 10.27.4.10 `linedit_cstring()`

```
char* linedit_cstring (
    linedit_string * string )
```

Returns a C string from a string

#### 10.27.4.11 `linedit_currentsuggestion()`

```
char* linedit_currentsuggestion (
    lineditor * edit )
```

Get the current suggestion

#### 10.27.4.12 `linedit_disablerawmode()`

```
void linedit_disablerawmode (
    void )
```

Restore terminal state to normal.

Print a carriage return to ensure we're back on the left hand side

#### 10.27.4.13 `linedit_displaywithstyle()`

```
void linedit_displaywithstyle (
    lineditor * edit,
    char * string,
    linedit_color col,
    linedit_emphasis emph )
```

Displays a string with a given color and emphasis.

## Parameters

<i>edit</i>	Line editor in use
<i>string</i>	String to display

**10.27.4.14 linenoedit\_displaywithsyntaxcoloring()**

```
void linenoedit_displaywithsyntaxcoloring (
    linenoeditor * edit,
    char * string )
```

Displays a string with syntax coloring.

## Parameters

<i>edit</i>	Line editor in use
<i>string</i>	String to display

**10.27.4.15 linenoedit\_enablerawmode()**

```
void linenoedit_enablerawmode (
    void )
```

Enables 'raw' mode in the terminal.

In raw mode key presses are passed directly to us rather than being buffered. Get the original state

**10.27.4.16 linenoedit\_generatesuggestions()**

```
void linenoedit_generatesuggestions (
    linenoeditor * edit )
```

Regenerates the list of autocomplete suggestions

**10.27.4.17 linenoedit\_getmode()**

```
linenoeditormode linenoedit_getmode (
    linenoeditor * edit )
```

Gets the current mode

**10.27.4.18 linedit\_historyadd()**

```
void linedit_historyadd (
    lineditor * edit,
    char * string )
```

Adds an entry to the history list

**10.27.4.19 linedit\_historyadvance()**

```
void linedit_historyadvance (
    lineditor * edit,
    unsigned int n )
```

Advances the history list

**10.27.4.20 linedit\_historyclear()**

```
void linedit_historyclear (
    lineditor * edit )
```

Frees the history list

**10.27.4.21 linedit\_historyselect()**

```
unsigned int linedit_historyselect (
    lineditor * edit,
    unsigned int n )
```

Makes a particular history entry current

**10.27.4.22 linedit\_init()**

```
void linedit_init (
    lineditor * edit )
```

Initialize a line editor

**10.27.4.23 linedit\_newstring()**

```
linedit_string* linedit_newstring (
    char * string )
```

Creates a new string from a C string

#### 10.27.4.24 lineno\_noterminal()

```
void lineno_noterminal (
    lineno * edit )
```

If we're not attached to a terminal, e.g. a pipe, simply read the file in.

#### 10.27.4.25 lineno\_processkeypress()

```
bool lineno_processkeypress (
    lineno * edit )
```

Obtain and process a single keypress

#### 10.27.4.26 lineno\_readkey()

```
bool lineno_readkey (
    lineno * edit,
    keypress * out )
```

Read and decode a single keypress from the terminal.

Decode the escape sequence

#### 10.27.4.27 lineno\_refreshline()

```
void lineno_refreshline (
    lineno * edit )
```

Refreshes a single line

#### 10.27.4.28 lineno\_setmode()

```
void lineno_setmode (
    lineno * edit,
    lineno_mode mode )
```

Sets the current mode, setting/clearing any state dependent data

#### 10.27.4.29 lineno\_setposition()

```
void lineno_setposition (
    lineno * edit,
    int posn )
```

Sets the current position

## Parameters

<i>edit</i>	- the editor
<i>posn</i>	- position to set, or negative to move to end

**10.27.4.30 linedit\_setprompt()**

```
void linedit_setprompt (
    lineditor * edit,
    char * prompt )
```

Sets the prompt.

## Parameters

<i>edit</i>	Line editor to configure
<i>prompt</i>	prompt string to use

**10.27.4.31 linedit\_showstring()**

```
void linedit_showstring (
    lineditor * edit,
    linedit_string * in,
    linedit_string * out )
```

Print a string without syntax coloring

**10.27.4.32 linedit\_stringaddcharacter()**

```
void linedit_stringaddcharacter (
    linedit_string * string,
    char * c,
    size_t n )
```

Adds a character to a string

**10.27.4.33 linedit\_stringaddcstring()**

```
void linedit_stringaddcstring (
    linedit_string * string,
    char * s )
```

Adds a c string to a string



#### 10.27.4.34 lineno\_stringclear()

```
void lineno_stringclear (
    lineno_string * string )
```

Clears a string, deallocating memory if necessary

#### 10.27.4.35 lineno\_stringinit()

```
void lineno_stringinit (
    lineno_string * string )
```

Initializes a string, clearing all fields

#### 10.27.4.36 lineno\_stringinsert()

```
void lineno_stringinsert (
    lineno_string * string,
    size_t posn,
    char * c,
    size_t n )
```

Inserts characters at a given position.

If the position is after the length of the string the new characters are instead appended.

#### 10.27.4.37 lineno\_stringlistadd()

```
void lineno_stringlistadd (
    lineno_stringlist * list,
    char * string )
```

Adds an entry to a string list

#### 10.27.4.38 lineno\_stringlistclear()

```
void lineno_stringlistclear (
    lineno_stringlist * list )
```

Frees the contents of a string list

#### 10.27.4.39 lineno\_stringlistinit()

```
void lineno_stringlistinit (
    lineno_stringlist * list )
```

Initializes a string list

#### 10.27.4.40 linedit\_stringlistremove()

```
void linedit_stringlistremove (
    linedit_stringlist * list,
    linedit_string * string )
```

Removes a string from a list

#### 10.27.4.41 linedit\_stringlistselect()

```
linedit_string* linedit_stringlistselect (
    linedit_stringlist * list,
    unsigned int n,
    unsigned int * m )
```

Chooses an element of a stringlist

##### Parameters

in	<i>list</i>	the list to select from
in	<i>n</i>	entry number to select @parma[out] *m entry number actually selected

##### Returns

the selected element

#### 10.27.4.42 linedit\_stringresize()

```
bool linedit_stringresize (
    linedit_string * string,
    size_t size )
```

Resizes a string

##### Parameters

<i>string</i>	- the string to grow
<i>size</i>	- requested size

##### Returns

true on success, false on failure

#### 10.27.4.43 linenoedit\_supported()

```
void linenoedit_supported (
    linenoeditor * edit )
```

Normal interface used if terminal is present

#### 10.27.4.44 linenoedit\_syntaxcolor()

```
void linenoedit_syntaxcolor (
    linenoeditor * edit,
    linenoedit_tokenizer tokenizer,
    linenoedit_color * cols,
    unsigned int ncols )
```

Configures syntax coloring.

##### Parameters

<i>edit</i>	Line editor to configure
<i>tokenizer</i>	A function to be called that will find the next token from a string
<i>cols</i>	An array of colors, one entry for each token type
<i>ncols</i>	Number of entries in the color array

#### 10.27.4.45 linenoedit\_syntaxcolorstring()

```
void linenoedit_syntaxcolorstring (
    linenoeditor * edit,
    linenoedit_string * in,
    linenoedit_string * out )
```

Print a string with syntax coloring

#### 10.27.4.46 linenoedit\_atendofline()

```
bool linenoedit_atendofline (
    linenoeditor * edit )
```

Checks if we're at the end of the line

### 10.27.5 Variable Documentation

### 10.27.5.1 terminit

```
struct termios terminit
```

Holds the original terminal state

## 10.28 interface/linenoedit.h File Reference

A simple line editor with history, prediction and syntax highlighting.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <ctype.h>
#include <string.h>
#include <termios.h>
#include <unistd.h>
```

### Classes

- struct [slinedit\\_string](#)
- struct [linenoedit\\_stringlist](#)
- struct [linenoedit\\_token](#)
- struct [linenoedit\\_syntaxcolordata](#)
- struct [linenoeditor](#)

### Macros

- `#define LINEDIT_DEFAULTPROMPT ">"`

### Typedefs

- typedef struct [slinedit\\_string](#) [linenoedit\\_string](#)
- typedef bool(\* [linenoedit\\_tokenizer](#)) (char \*in, void \*\*ref, [linenoedit\\_token](#) \*tok)  
*Tokenizer callback.*
- typedef bool(\* [linenoedit\\_completer](#)) (char \*in, [linenoedit\\_stringlist](#) \*completion)  
*Autocompletion callback @params in - a string @params completion - autocompletion structure.*

### Enumerations

- enum [linenoedit\\_color](#) {  
    **LINEDIT\_BLACK**, **LINEDIT\_RED**, **LINEDIT\_GREEN**, **LINEDIT\_YELLOW**,  
    **LINEDIT\_BLUE**, **LINEDIT\_MAGENTA**, **LINEDIT\_CYAN**, **LINEDIT\_WHITE**,  
    **LINEDIT\_DEFAULTCOLOR** }
- enum [linenoedit\\_emphasis](#) { **LINEDIT\_BOLD**, **LINEDIT\_UNDERLINE**, **LINEDIT\_REVERSE**, **LINEDIT\_NONE** }
- enum [linenoeditormode](#) { **LINEDIT\_DEFAULTMODE**, **LINEDIT\_SELECTIONMODE**, **LINEDIT\_HISTORYMODE** }

## Functions

- char \* `linenoedit` (`linenoeditor` \*edit)
- void `linenoedit_syntaxcolor` (`linenoeditor` \*edit, `linenoedit_tokenizer` tokenizer, `linenoedit_color` \*cols, unsigned int ncols)  
*Configures syntax coloring.*
- void `linenoedit_autocomplete` (`linenoeditor` \*edit, `linenoedit_completer` completer)  
*Configures autocomplete.*
- void `linenoedit_addsuggestion` (`linenoedit_stringlist` \*completion, char \*string)  
*Adds a completion suggestion.*
- void `linenoedit_setprompt` (`linenoeditor` \*edit, char \*prompt)  
*Sets the prompt.*
- void `linenoedit_displaywithstyle` (`linenoeditor` \*edit, char \*string, `linenoedit_color` col, `linenoedit_emphasis` emph)  
*Displays a string with a given color and emphasis.*
- void `linenoedit_displaywithsyntaxcoloring` (`linenoeditor` \*edit, char \*string)  
*Displays a string with syntax coloring.*
- void `linenoedit_init` (`linenoeditor` \*edit)
- void `linenoedit_clear` (`linenoeditor` \*edit)

### 10.28.1 Detailed Description

A simple line editor with history, prediction and syntax highlighting.

Author

T J Atherton

### 10.28.2 Typedef Documentation

#### 10.28.2.1 `linenoedit_completer`

```
typedef bool(* linenoedit_completer) (char *in, linenoedit_stringlist *completion)
```

Autocompletion callback @params in - a string @params completion - autocompletion structure.

This user function is called when linenoedit requests autocompletion of a string. The function should identify any possible suggestions and call `linenoedit_addcompletion` to add them one by one.

Only *remaining* characters from the suggestion should be added, e.g. for "hello" if the user has typed "he" the function should add "llo" as a suggestion.

The function should return true if autocompletion was successfully processed or false otherwise.

#### 10.28.2.2 `linenoedit_string`

```
typedef struct slinedit_string linenoedit_string
```

linenoeditor strings

#### 10.28.2.3 `linenoedit_tokenizer`

```
typedef bool(* linenoedit_tokenizer) (char *in, void **ref, linenoedit_token *tok)
```

Tokenizer callback.

## Parameters

<i>in</i>	- a string
<i>ref</i>	- System for storing persistent data between calls to the lexer.
<i>tok</i>	- pointer to a token structure that the caller should fill out. This user function is called when linedit needs to tokenize a string. The function should identify the next token in the string and fill out the following fields: tok->type - should contain the token type. This is used e.g. an index to the color array. tok->start - should point to the first significant character in the token tok->length - should contain the length of the token, in bytes The function should return true if a token was successfully processed or false otherwise.

Storing persistent data over a sequence of calls to implement non-CFG: On the first call, \*ref is NULL. You should malloc your structure, initialize it, set \*ref to point to it and return the first token. After that, \*ref will point to your structure. Once linedit is done tokenizing, it will call free on your pointer.

## Warning

: You must not malloc child structures as these will not be freed.

### 10.28.3 Enumeration Type Documentation

#### 10.28.3.1 linedit\_color

```
enum linedit_color
```

Colors

#### 10.28.3.2 lineditormode

```
enum lineditormode
```

Keep track of what the line editor is doing

### 10.28.4 Function Documentation

#### 10.28.4.1 linedit()

```
char* linedit (
    lineditor * edit )
```

Public interface to the line editor.

## Parameters

<i>edit</i>	- a line editor that has been initialized with linenoedit_init.
-------------	---

## Returns

the string input by the user, or NULL if nothing entered.

Ensure we are not passed a NULL pointer

**10.28.4.2 linenoedit\_addsuggestion()**

```
void linenoedit_addsuggestion (
    linenoedit_stringlist * completion,
    char * string )
```

Adds a completion suggestion.

## Parameters

<i>completion</i>	completion data structure
<i>string</i>	string to add

**10.28.4.3 linenoedit\_autocomplete()**

```
void linenoedit_autocomplete (
    linenoeditor * edit,
    linenoedit_completer completer )
```

Configures autocomplete.

## Parameters

<i>edit</i>	Line editor to configure
<i>completer</i>	a function

**10.28.4.4 linenoedit\_clear()**

```
void linenoedit_clear (
    linenoeditor * edit )
```

Finalize a line editor

#### 10.28.4.5 `linedit_displaywithstyle()`

```
void linedit_displaywithstyle (
    lineditor * edit,
    char * string,
    linedit_color col,
    linedit_emphasis emph )
```

Displays a string with a given color and emphasis.

##### Parameters

<i>edit</i>	Line editor in use
<i>string</i>	String to display
<i>col</i>	Color
<i>emph</i>	Emphasis
<i>edit</i>	Line editor in use
<i>string</i>	String to display

#### 10.28.4.6 `linedit_displaywithsyntaxcoloring()`

```
void linedit_displaywithsyntaxcoloring (
    lineditor * edit,
    char * string )
```

Displays a string with syntax coloring.

##### Parameters

<i>edit</i>	Line editor in use
<i>string</i>	String to display

#### 10.28.4.7 `linedit_init()`

```
void linedit_init (
    lineditor * edit )
```

Initialize a line editor

#### 10.28.4.8 `linedit_setprompt()`

```
void linedit_setprompt (
    lineditor * edit,
    char * prompt )
```

Sets the prompt.



## Parameters

<i>edit</i>	Line editor to configure
<i>prompt</i>	prompt string to use

**10.28.4.9 linedit\_syntaxcolor()**

```
void linedit_syntaxcolor (
    lineditor * edit,
    linedit_tokenizer tokenizer,
    linedit_color * cols,
    unsigned int ncols )
```

Configures syntax coloring.

## Parameters

<i>edit</i>	Line editor to configure
<i>tokenizer</i>	A function to be called that will find the next token from a string
<i>cols</i>	An array of colors, one entry for each token type
<i>ncols</i>	Number of entries in the color array

**10.29 main.c File Reference**

Main entry point.

```
#include <stdio.h>
#include <stdarg.h>
#include "cli.h"
#include "value.h"
#include "object.h"
#include "random.h"
#include "sparse.h"
```

**Functions**

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

**10.29.1 Detailed Description**

Main entry point.

## Author

T J Atherton

## 10.30 morpho.h File Reference

Define public interface to Morpho.

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "build.h"
#include "value.h"
#include "error.h"
```

### Macros

- #define **MORPHO\_INITIALIZER\_METHOD** "init"
- #define **MORPHO\_GETINDEX\_METHOD** "index"
- #define **MORPHO\_SETINDEX\_METHOD** "setindex"
- #define **MORPHO\_ADD\_METHOD** "add"
- #define **MORPHO\_SUB\_METHOD** "sub"
- #define **MORPHO\_MUL\_METHOD** "mul"
- #define **MORPHO\_DIV\_METHOD** "div"
- #define **MORPHO\_ENUMERATE\_METHOD** "enumerate"
- #define **MORPHO\_PRINT\_METHOD** "prnt"

### Typedefs

- typedef void **vm**
- typedef void **program**
- typedef void **compiler**

### Functions

- void [morpho\\_writeerrorwithid](#) ([error](#) \*err, [errorid](#) id, int line, int position,...)  
*Writes an error message to an error structure.*
- void [morpho\\_defineerror](#) ([errorid](#) id, [errorcategory](#) cat, char \*message)
- [errorid](#) [morpho\\_geterrorid](#) ([error](#) \*err)
- program \* [morpho\\_newprogram](#) (void)  
*Creates and initializes a new program.*
- void [morpho\\_freeprogram](#) (program \*p)  
*Frees a program.*
- vm \* [morpho\\_newvm](#) (void)
- void [morpho\\_freemv](#) (vm \*v)
- void [morpho\\_runtimeerror](#) (vm \*v, [errorid](#) id,...)  
*Public interface to raise a runtime error.*
- compiler \* [morpho\\_newcompiler](#) (program \*out)
- void [morpho\\_freecompile](#) (compiler \*c)
- bool [morpho\\_compile](#) (char \*in, compiler \*c, [error](#) \*err)
- const char \* [morpho\\_compilerrestartpoint](#) (compiler \*c)
- void [morpho\\_resetentry](#) (program \*p)
- bool [morpho\\_interpret](#) (vm \*v, program \*code, [error](#) \*err)  
*Executes a sequence of code.*
- void [morpho\\_disassemble](#) (program \*code, unsigned int \*matchline)
- void [morpho\\_stacktrace](#) (vm \*v)
- void [morpho\\_initialize](#) (void)
- void [morpho\\_finalize](#) (void)

## Variables

- [value](#) `initselector`
- [value](#) `indexselector`
- [value](#) `setindexselector`
- [value](#) `addselector`
- [value](#) `subselector`
- [value](#) `mulselector`
- [value](#) `divselector`
- [value](#) `printselector`
- [value](#) `enumerateselector`

### 10.30.1 Detailed Description

Define public interface to Morpho.

#### Author

T J Atherton

### 10.30.2 Function Documentation

#### 10.30.2.1 `morpho_compile()`

```
bool morpho_compile (  
    char * in,  
    compiler * c,  
    error * err )
```

Interface to the compiler

#### Parameters

in	<i>in</i>	A string to compile
in	<i>c</i>	The compiler
out	<i>err</i>	Pointer to error block on failure

#### Returns

A bool indicating success or failure

#### 10.30.2.2 `morpho_defineerror()`

```
void morpho_defineerror (  
    errorid id,
```

```
    errorcategory cat,  
    char * message )
```

Defines an error

#### Parameters

<i>id</i>	Error struct to fill out
<i>cat</i>	The category of error
<i>message</i>	The message string

### 10.30.2.3 morpho\_disassemble()

```
void morpho_disassemble (  
    program * code,  
    unsigned int * matchline )
```

Disassembles a program

#### Parameters

<i>code</i>	- program to disassemble
<i>matchline</i>	- optional line number to match

### 10.30.2.4 morpho\_finalize()

```
void morpho_finalize (  
    void )
```

Finalizes morpho

### 10.30.2.5 morpho\_freecompiler()

```
void morpho_freecompiler (  
    compiler * c )
```

Frees a compiler

### 10.30.2.6 morpho\_freemv()

```
void morpho_freemv (  
    vm * v )
```

Frees a virtual machine

### 10.30.2.7 morpho\_geterrorid()

```
errorid morpho_geterrorid (
    error * err )
```

Gets the id of an error

### 10.30.2.8 morpho\_initialize()

```
void morpho_initialize (
    void )
```

Initializes morpho

### 10.30.2.9 morpho\_interpret()

```
bool morpho_interpret (
    vm * v,
    program * code,
    error * err )
```

Executes a sequence of code.

#### Parameters

<i>v</i>	The virtual machine to use
<i>code</i>	The program to execute

#### Returns

A morpho error

### 10.30.2.10 morpho\_newcompiler()

```
compiler* morpho_newcompiler (
    program * out )
```

Creates a new compiler

### 10.30.2.11 morpho\_newvm()

```
vm* morpho_newvm (
    void )
```

Creates a new virtual machine

### 10.30.2.12 morpho\_runtimeerror()

```
void morpho_runtimeerror (
    vm * v,
    errorid id,
    ... )
```

Public interface to raise a runtime error.

#### Parameters

<i>v</i>	the virtual machine
<i>id</i>	error id
...	additional data for sprintf.

### 10.30.2.13 morpho\_stacktrace()

```
void morpho_stacktrace (
    vm * v )
```

Prints a stacktrace

### 10.30.2.14 morpho\_writeerrorwithid()

```
void morpho_writeerrorwithid (
    error * err,
    errorid id,
    int line,
    int posn,
    ... )
```

Writes an error message to an error structure.

#### Parameters

<i>err</i>	The error structure
<i>id</i>	The error id.
<i>line</i>	The line at which the error occurred, if identifiable.
<i>posn</i>	The position in the line at which the error occurred, if identifiable.
...	Additional parameters (the data for the printf commands in the message)

## 10.31 utils/common.c File Reference

Common types, data structures and functions for the Morpho VM.

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <ctype.h>
#include <sys/stat.h>
#include "common.h"
#include "object.h"
```

## Macros

- #define [MORPHO\\_TOSTRINGTMPBUFFERSIZE](#) 64  
*Concatenates a sequence of values as a string.*

## Functions

- void [morpho\\_printvalue](#) (value v)  
*Prints a value.*
- value [morpho\\_concatenatestringvalues](#) (int nval, value \*v)
- char \* [morpho\\_strdup](#) (char \*string)  
*Duplicates a string.*
- unsigned int [morpho\\_powerof2ceiling](#) (unsigned int n)  
*Computes the nearest power of 2 above an integer.*
- bool [morpho\\_isdirectory](#) (const char \*path)

### 10.31.1 Detailed Description

Common types, data structures and functions for the Morpho VM.

#### Author

T J Atherton

### 10.31.2 Function Documentation

#### 10.31.2.1 [morpho\\_powerof2ceiling\(\)](#)

```
unsigned int morpho_powerof2ceiling (
    unsigned int n )
```

Computes the nearest power of 2 above an integer.

#### Parameters

<i>n</i>	An integer
----------	------------

**Returns**

Nearest power of 2 above n See: <http://graphics.stanford.edu/~seander/bithacks.html#RoundUpPowerOf2Float>

**10.31.2.2 morpho\_printvalue()**

```
void morpho_printvalue (
    value v )
```

Prints a value.

**Parameters**

<i>v</i>	The value to print
----------	--------------------

**10.31.2.3 morpho\_strdup()**

```
char* morpho_strdup (
    char * string )
```

Duplicates a string.

**Parameters**

<i>string</i>	String to duplicate
---------------	---------------------

**Warning**

Caller should call MALLOC\_FREE on the allocated string

**10.32 utils/common.h File Reference**

Morpho virtual machine.

```
#include <stddef.h>
#include <stdarg.h>
#include <math.h>
#include <float.h>
#include <string.h>
#include "value.h"
#include "object.h"
#include "error.h"
```



## Macros

- #define **COMMON\_NILSTRING** "nil"
- #define **COMMON\_TRUESTRING** "true"
- #define **COMMON\_FALSESTRING** "false"
- #define **EQUAL** 0  
*Compares two values.*
- #define **NOTEQUAL** 1
- #define **BIGGER** 1
- #define **SMALLER** -1
- #define **MORPHO\_ISEQUAL**(a, b) (!morpho\_comparevalue(a,b))
- #define **MORPHO\_ISSAME**(a, b) (morpho\_comparevaluesame(a,b))

## Functions

- void **morpho\_printvalue** (value v)  
*Prints a value.*
- value **morpho\_concatenatestringvalues** (int nval, value \*v)
- char \* **morpho\_strdup** (char \*string)  
*Duplicates a string.*
- unsigned int **morpho\_powerof2ceiling** (unsigned int n)  
*Computes the nearest power of 2 above an integer.*
- bool **morpho\_isdirectory** (const char \*path)

### 10.32.1 Detailed Description

Morpho virtual machine.

#### Author

T J Atherton

### 10.32.2 Macro Definition Documentation

#### 10.32.2.1 EQUAL

```
#define EQUAL 0
```

Compares two values.

#### Parameters

<i>a</i>	value to compare
<i>b</i>	value to compare

**Returns**

0 if a and b are equal, a positive number if  $b > a$  and a negative number if  $a < b$

**10.32.2.2 MORPHO\_ISEQUAL**

```
#define MORPHO_ISEQUAL(
    a,
    b ) (!morpho_comparevalue(a,b))
```

Macros to compare values

Use this one to carefully compare the values in each object

**10.32.2.3 MORPHO\_ISSAME**

```
#define MORPHO_ISSAME(
    a,
    b ) (morpho_comparevaluesame(a,b))
```

Use this one where we want to check the values refer to the same object

**10.32.3 Function Documentation****10.32.3.1 morpho\_powerof2ceiling()**

```
unsigned int morpho_powerof2ceiling (
    unsigned int n )
```

Computes the nearest power of 2 above an integer.

**Parameters**

<i>n</i>	An integer
----------	------------

**Returns**

Nearest power of 2 above n See: <http://graphics.stanford.edu/~seander/bithacks.html#RoundUpPowerOf2Float> ↩

**10.32.3.2 morpho\_printvalue()**

```
void morpho_printvalue (
    value v )
```

Prints a value.

#### Parameters

<i>v</i>	The value to print
----------	--------------------

### 10.32.3.3 morpho\_strdup()

```
char* morpho_strdup (
    char * string )
```

Duplicates a string.

#### Parameters

<i>string</i>	String to duplicate
---------------	---------------------

#### Warning

Caller should call MALLOC\_FREE on the allocated string

## 10.33 utils/error.c File Reference

Morpho error handling.

```
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include "error.h"
#include "common.h"
#include "dictionary.h"
```

### Functions

- void [error\\_clear](#) ([error](#) \*err)
- void [error\\_init](#) ([error](#) \*err)
- bool [morpho\\_getdefinitionfromid](#) ([errorid](#) id, [errordefinition](#) \*\*def)
- void [morpho\\_writeerrorwithidvalist](#) ([error](#) \*err, [errorid](#) id, int line, int posn, va\_list args)  
*Writes an error message to an error structure.*
- void [morpho\\_writeerrorwithid](#) ([error](#) \*err, [errorid](#) id, int line, int posn,...)  
*Writes an error message to an error structure.*
- void [morpho\\_defineerror](#) ([errorid](#) id, [errorcategory](#) cat, char \*message)
- [errorid](#) [morpho\\_geterrorid](#) ([error](#) \*err)
- void [error\\_initialize](#) (void)
- void [error\\_finalize](#) (void)

### 10.33.1 Detailed Description

Morpho error handling.

Morpho memory management.

Author

T J Atherton

### 10.33.2 Function Documentation

#### 10.33.2.1 `error_clear()`

```
void error_clear (
    error * err )
```

Clears an error structure

Parameters

<code>err</code>	Error struct to fill out
------------------	--------------------------

#### 10.33.2.2 `error_finalize()`

```
void error_finalize (
    void )
```

Finalizes the error handling system

#### 10.33.2.3 `error_init()`

```
void error_init (
    error * err )
```

Clears an error structure

Parameters

<code>err</code>	Error struct to fill out
------------------	--------------------------

#### 10.33.2.4 error\_initialize()

```
void error_initialize (
    void )
```

Initializes the error handling system

#### 10.33.2.5 morpho\_defineerror()

```
void morpho_defineerror (
    errorid id,
    errorcategory cat,
    char * message )
```

Defines an error

##### Parameters

<i>id</i>	Error struct to fill out
<i>cat</i>	The category of error
<i>message</i>	The message string

#### 10.33.2.6 morpho\_getdefinitionfromid()

```
bool morpho_getdefinitionfromid (
    errorid id,
    errordefinition ** def )
```

Gets an error definition given an errorid

##### Parameters

in	<i>id</i>	Error to retrieve
out	<i>def</i>	The error definition

##### Returns

true on success

#### 10.33.2.7 morpho\_geterrorid()

```
errorid morpho_geterrorid (
    error * err )
```

Gets the id of an error

### 10.33.2.8 morpho\_writeerrorwithid()

```
void morpho_writeerrorwithid (
    error * err,
    errorid id,
    int line,
    int posn,
    ... )
```

Writes an error message to an error structure.

#### Parameters

<i>err</i>	The error structure
<i>id</i>	The error id.
<i>line</i>	The line at which the error occurred, if identifiable.
<i>posn</i>	The position in the line at which the error occurred, if identifiable.
...	Additional parameters (the data for the printf commands in the message)

### 10.33.2.9 morpho\_writeerrorwithidvalist()

```
void morpho_writeerrorwithidvalist (
    error * err,
    errorid id,
    int line,
    int posn,
    va_list args )
```

Writes an error message to an error structure.

#### Parameters

<i>err</i>	The error structure
<i>id</i>	The error id.
<i>line</i>	The line at which the error occurred, if identifiable.
<i>posn</i>	The position in the line at which the error occurred, if identifiable.
<i>args</i>	Additional parameters (the data for the printf commands in the message)

## 10.34 utils/error.h File Reference

Morpho error handling.

```
#include <stdarg.h>
#include "build.h"
#include "varray.h"
```

## Classes

- struct [error](#)  
*A static container for error messages.*
- struct [errordefinition](#)  
*Definition of an error message.*

## Macros

- #define [ERROR\\_SHOULDCONTINUE](#)(cat) (cat < [ERROR\\_HALT](#))
- #define [ERROR\\_SUCCEEDED](#)(err) ((err).cat == ERROR\_NONE)
- #define [ERROR\\_ISRUNTIMEERROR](#)(err) ((err).cat <= [ERROR\\_EXIT](#))
- #define [ERROR\\_POSNUNIDENTIFIABLE](#) -1
- #define [UNREACHABLE](#)(x)
- #define [BSD\\_EX\\_SOFTWARE](#) 70
- #define [ERROR\\_ALLOCATIONFAILED](#) "Alloc"
- #define [ERROR\\_ALLOCATIONFAILED\\_MSG](#) "%s allocation failed."
- #define [ERROR\\_INTERNALERROR](#) "Intrnl"
- #define [ERROR\\_INTERNALERROR\\_MSG](#) "Internal [error](#) (contact developer)."
- #define [COMPILE\\_UNTERMINATEDCOMMENT](#) "UntrmComm"
- #define [COMPILE\\_UNTERMINATEDCOMMENT\\_MSG](#) "Unterminated multiline comment '/\*'."
- #define [COMPILE\\_UNTERMINATEDSTRING](#) "UntrmStrng"
- #define [COMPILE\\_UNTERMINATEDSTRING\\_MSG](#) "Unterminated string."
- #define [COMPILE\\_INCOMPLETEEXPRESSION](#) "IncExp"
- #define [COMPILE\\_INCOMPLETEEXPRESSION\\_MSG](#) "Incomplete expression."
- #define [COMPILE\\_MISSINGPARENTHESIS](#) "MssngParen"
- #define [COMPILE\\_MISSINGPARENTHESIS\\_MSG](#) "Expect ')' after expression."
- #define [COMPILE\\_EXPECTEXPRESSION](#) "ExpExpr"
- #define [COMPILE\\_EXPECTEXPRESSION\\_MSG](#) "Expected expression."
- #define [COMPILE\\_MISSINGSEMICOLON](#) "MssngSemiVal"
- #define [COMPILE\\_MISSINGSEMICOLON\\_MSG](#) "Expect ; after value."
- #define [COMPILE\\_MISSINGSEMICOLONEXP](#) "MssngExpTerm"
- #define [COMPILE\\_MISSINGSEMICOLONEXP\\_MSG](#) "Expect expression terminator (; or newline) after expression."
- #define [COMPILE\\_MISSINGSEMICOLONVAR](#) "MssngSemiVar"
- #define [COMPILE\\_MISSINGSEMICOLONVAR\\_MSG](#) "Expect ; after variable declaration."
- #define [COMPILE\\_VAREXPECTED](#) "VarExpct"
- #define [COMPILE\\_VAREXPECTED\\_MSG](#) "Variable name expected after var."
- #define [COMPILE\\_BLOCKTERMINATOREXP](#) "MssngBrc"
- #define [COMPILE\\_BLOCKTERMINATOREXP\\_MSG](#) "Expected '}' to finish block."
- #define [COMPILE\\_IFLFTPARENMISSING](#) "IfMssngLftPrn"
- #define [COMPILE\\_IFLFTPARENMISSING\\_MSG](#) "Expected '(' after if."
- #define [COMPILE\\_IFRGHTPARENMISSING](#) "IfMssngRgtPrn"
- #define [COMPILE\\_IFRGHTPARENMISSING\\_MSG](#) "Expected ')' after condition."
- #define [COMPILE\\_WHILELFTPARENMISSING](#) "WhlMssngLftPrn"
- #define [COMPILE\\_WHILELFTPARENMISSING\\_MSG](#) "Expected '(' after while."
- #define [COMPILE\\_FORLFTPARENMISSING](#) "ForMssngLftPrn"
- #define [COMPILE\\_FORLFTPARENMISSING\\_MSG](#) "Expected '(' after for."
- #define [COMPILE\\_FORSEMICOLONMISSING](#) "ForMssngSemi"
- #define [COMPILE\\_FORSEMICOLONMISSING\\_MSG](#) "Expected ';'."
- #define [COMPILE\\_FORRGHTPARENMISSING](#) "ForMssngRgtPrn"
- #define [COMPILE\\_FORRGHTPARENMISSING\\_MSG](#) "Expected ')' after for clauses."
- #define [COMPILE\\_FNNAMEMISSING](#) "FnNoName"

- `#define COMPILE_FNNAMEMISSING_MSG` "Expected function or method name."
- `#define COMPILE_FNLEFTPARENMISSING` "FnMssngLftPrn"
- `#define COMPILE_FNLEFTPARENMISSING_MSG` "Expect '(' after name."
- `#define COMPILE_FNRGHTPARENMISSING` "FnMssngRgtPrn"
- `#define COMPILE_FNRGHTPARENMISSING_MSG` "Expect ')' after parameters."
- `#define COMPILE_FNLEFTCURLYMISSING` "FnMssngLftBrC"
- `#define COMPILE_FNLEFTCURLYMISSING_MSG` "Expect '{' before body."
- `#define COMPILE_CALLRGHTPARENMISSING` "ClMssngRgtPrn"
- `#define COMPILE_CALLRGHTPARENMISSING_MSG` "Expect ')' after arguments."
- `#define COMPILE_EXPECTCLASSNAME` "ClsNmMssng"
- `#define COMPILE_EXPECTCLASSNAME_MSG` "Expect class name."
- `#define COMPILE_CLASSLEFTCURLYMISSING` "ClsMssngLftBrC"
- `#define COMPILE_CLASSLEFTCURLYMISSING_MSG` "Expect '{' before class body."
- `#define COMPILE_CLASSRGHTCURLYMISSING` "ClsMssngRgtBrC"
- `#define COMPILE_CLASSRGHTCURLYMISSING_MSG` "Expect '}' after class body."
- `#define COMPILE_EXPECTDOTAFTERSUPER` "ExpctDtSpr"
- `#define COMPILE_EXPECTDOTAFTERSUPER_MSG` "Expect '.' after 'super'"
- `#define COMPILE_INCOMPLETESTRINGINT` "Intrplncmp"
- `#define COMPILE_INCOMPLETESTRINGINT_MSG` "Incomplete string after interpolation."
- `#define COMPILE_VARBLANKINDEX` "EmptyIndx"
- `#define COMPILE_VARBLANKINDEX_MSG` "Empty capacity in variable declaration."
- `#define COMPILE_IMPORTMISSINGNAME` "ImprtMssngNm"
- `#define COMPILE_IMPORTMISSINGNAME_MSG` "Import expects a module or file name."
- `#define COMPILE_IMPORTUNEXPCTDTOK` "ImprtExpctFrAs"
- `#define COMPILE_IMPORTUNEXPCTDTOK_MSG` "Import only expects for or as after module or file name."
- `#define COMPILE_IMPORTASSYMBL` "ExpctSymblAfrAs"
- `#define COMPILE_IMPORTASSYMBL_MSG` "Expect symbol after as in import."
- `#define COMPILE_IMPORTFORSYMBL` "ExpctSymblAfrFr"
- `#define COMPILE_IMPORTFORSYMBL_MSG` "Expect symbol(s) after for in import."
- `#define PARSE_UNRECGNZEDTOK` "UnrcgnzdTok"
- `#define PARSE_UNRECGNZEDTOK_MSG` "Encountered an unrecognized token."
- `#define COMPILE_SYMBOLNOTDEFINED` "SymblUndf"
- `#define COMPILE_SYMBOLNOTDEFINED_MSG` "Symbol '%s' not defined."
- `#define COMPILE_TOOMANYCONSTANTS` "TooMnyCnst"
- `#define COMPILE_TOOMANYCONSTANTS_MSG` "Too many constants."
- `#define COMPILE_ARGSNOTSYMBOLS` "FnPrmSymb"
- `#define COMPILE_ARGSNOTSYMBOLS_MSG` "Function parameters must be symbols."
- `#define COMPILE_PROPERTYNAMERQD` "PptyNmRqd"
- `#define COMPILE_PROPERTYNAMERQD_MSG` "Property name required."
- `#define COMPILE_SELFOUTSIDECLASS` "SlfOtsdClss"
- `#define COMPILE_SELFOUTSIDECLASS_MSG` "Cannot use 'self' outside of a class."
- `#define COMPILE_RETURNINIALIZER` "InitRtn"
- `#define COMPILE_RETURNINIALIZER_MSG` "Cannot return a [value](#) in an initializer."
- `#define COMPILE_EXPECTSUPER` "SprNmMssng"
- `#define COMPILE_EXPECTSUPER_MSG` "Expect superclass name."
- `#define COMPILE_SUPERCLASSNOTFOUND` "SptNtFnd"
- `#define COMPILE_SUPERCLASSNOTFOUND_MSG` "Superclass '%s' not found."
- `#define COMPILE_SUPEROUTSIDECLASS` "SptOtsdClss"
- `#define COMPILE_SUPEROUTSIDECLASS_MSG` "Cannot use 'super' outside of a class."
- `#define COMPILE_NOSUPER` "SprSelMthd"
- `#define COMPILE_NOSUPER_MSG` "Can only use 'super' to select a method."
- `#define COMPILE_INVALIDASSIGNMENT` "InvlidAssgn"
- `#define COMPILE_INVALIDASSIGNMENT_MSG` "Invalid assignment target."
- `#define COMPILE_CLASSINHERITSELF` "ClsCrcRf"
- `#define COMPILE_CLASSINHERITSELF_MSG` "A class cannot inherit from itself."



- `#define COMPILE_TOOMANYARGS "TooMnyArg"`
- `#define COMPILE_TOOMANYARGS_MSG "Too many arguments."`
- `#define COMPILE_TOOMANYPARAMS "TooMnyPrm"`
- `#define COMPILE_TOOMANYPARAMS_MSG "Too many parameters."`
- `#define COMPILE_ISOLATEDSUPER "IsoSpr"`
- `#define COMPILE_ISOLATEDSUPER_MSG "Expect '.' after 'super'."`
- `#define COMPILE_VARALREADYDECLARED "VblDcl"`
- `#define COMPILE_VARALREADYDECLARED_MSG "Variable with this name already declared in this scope."`
- `#define VM_NONNUMERICALOPERAND "NonNumOp"`
- `#define VM_NONNUMERICALOPERAND_MSG "Operands must be numbers."`
- `#define VM_CONCATENATIONFAILED "Cnct"`
- `#define VM_CONCATENATIONFAILED_MSG "Concatenation failed."`
- `#define VM_UNCALLABLE "Uncallable"`
- `#define VM_UNCALLABLE_MSG "Can only call a function or method."`
- `#define VM_GLOBALRETURN "GblRtrn"`
- `#define VM_GLOBALRETURN_MSG "Return encountered outside a function or method."`
- `#define VM_INSTANTIATEFAILED "InstFail"`
- `#define VM_INSTANTIATEFAILED_MSG "Could not instantiate object."`
- `#define VM_NOTANOBJECT "NotAnObj"`
- `#define VM_NOTANOBJECT_MSG "Not an object."`
- `#define VM_OBJECTLACKSPROPERTY "ObjLcksPrp"`
- `#define VM_OBJECTLACKSPROPERTY_MSG "Object lacks property or method '%s'."`
- `#define VM_NOINITIALIZER "NoInit"`
- `#define VM_NOINITIALIZER_MSG "Cannot instantiate with arguments because class '%s' does not provide an initializer."`
- `#define VM_NOTANINSTANCE "NotAnInst"`
- `#define VM_NOTANINSTANCE_MSG "Can only invoke methods on objects."`
- `#define VM_CLASSLACKSPROPERTY "ClsLcksMthd"`
- `#define VM_CLASSLACKSPROPERTY_MSG "Class lacks method '%s'."`
- `#define VM_INVALIDARGS "InvldArgs"`
- `#define VM_INVALIDARGS_MSG "Expected %u arguments but got %u."`
- `#define VM_INVALIDOPERANDS "InvldOp"`
- `#define VM_INVALIDOPERANDS_MSG "Cannot add these operands."`
- `#define VM_NOTINDEXABLE "NotIndxbl"`
- `#define VM_NOTINDEXABLE_MSG "Value or object not indexable."`
- `#define VM_OUTOFBOUNDS "ArrayBnds"`
- `#define VM_OUTOFBOUNDS_MSG "Array index out of bounds."`
- `#define VM_NONNUMINDEX "NonNmIndx"`
- `#define VM_NONNUMINDEX_MSG "Non-numerical array index."`
- `#define VM_ARRAYWRONGDIM "ArrayDim"`
- `#define VM_ARRAYWRONGDIM_MSG "Incorrect number of dimensions for array."`

## Typedefs

- `typedef char * errorid`  
*Identifier for errors.*
- `typedef errorcategory morphoerror`  
*A type used by public-facing morpho functions.*

## Enumerations

- enum `errorcategory` {  
**ERROR\_NONE**, **ERROR\_INFO**, **ERROR\_WARNING**, **ERROR\_HALT**,  
**ERROR\_EXIT**, **ERROR\_LEX**, **ERROR\_PARSE**, **ERROR\_COMPILE** }

## Functions

- void `error_init` (`error` \*err)
- void `error_clear` (`error` \*err)
- void `morpho_writeerrorwithid` (`error` \*err, `errorid` id, int line, int posn,...)  
*Writes an error message to an error structure.*
- void `morpho_writeerrorwithidvalist` (`error` \*err, `errorid` id, int line, int posn, va\_list args)  
*Writes an error message to an error structure.*
- void `morpho_defineerror` (`errorid` id, `errorcategory` cat, char \*message)
- `errorid` `morpho_geterrorid` (`error` \*err)
- void `error_initialize` (void)
- void `error_finalize` (void)

### 10.34.1 Detailed Description

Morpho error handling.

Author

T J Atherton

### 10.34.2 Macro Definition Documentation

#### 10.34.2.1 BSD\_EX\_SOFTWARE

```
#define BSD_EX_SOFTWARE 70
```

Exit codes from BSD standard

#### 10.34.2.2 ERROR\_ISRUNTIMEERROR

```
#define ERROR_ISRUNTIMEERROR(  

    err ) ((err).cat <= ERROR_EXIT)
```

Is this a runtime error?

#### 10.34.2.3 ERROR\_POSNUNIDENTIFIABLE

```
#define ERROR_POSNUNIDENTIFIABLE -1
```

Set line and posn to this value if they can't be determined

#### 10.34.2.4 ERROR\_SHOULDCONTINUE

```
#define ERROR_SHOULDCONTINUE(  
    cat ) (cat < ERROR_HALT)
```

Checks if execution should continue after the error

#### 10.34.2.5 ERROR\_SUCCEEDED

```
#define ERROR_SUCCEEDED(  
    err ) ((err).cat == ERROR_NONE)
```

Did an operation succeed without errors?

#### 10.34.2.6 UNREACHABLE

```
#define UNREACHABLE(  
    x )
```

A varray of errordefinitions Macro to place in code that should be unreachable

### 10.34.3 Enumeration Type Documentation

#### 10.34.3.1 errorcategory

```
enum errorcategory
```

Identifies the category of error that has occurred

Enumerator

ERROR_INFO	No error.
ERROR_WARNING	Informational messages generated.
ERROR_HALT	Warnings generated.
ERROR_EXIT	or has occurred. Should return to the user as fast as possible.
ERROR_LEX	Unrecoverable error has occurred; Morpho will exit quickly.
ERROR_PARSE	or generated by the lexer
ERROR_COMPILE	or generated by the parser or generated by the compiler

#### 10.34.4 Function Documentation

#### 10.34.4.1 `error_clear()`

```
void error_clear (
    error * err )
```

Clears an error structure

##### Parameters

<i>err</i>	Error struct to fill out
------------	--------------------------

#### 10.34.4.2 `error_finalize()`

```
void error_finalize (
    void )
```

Finalizes the error handling system

#### 10.34.4.3 `error_init()`

```
void error_init (
    error * err )
```

Clears an error structure

##### Parameters

<i>err</i>	Error struct to fill out
------------	--------------------------

#### 10.34.4.4 `error_initialize()`

```
void error_initialize (
    void )
```

Initializes the error handling system

#### 10.34.4.5 `morpho_defineerror()`

```
void morpho_defineerror (
    errorid id,
    errorcategory cat,
    char * message )
```

Defines an error

## Parameters

<i>id</i>	Error struct to fill out
<i>cat</i>	The category of error
<i>message</i>	The message string

**10.34.4.6 morpho\_geterrorid()**

```
errorid morpho_geterrorid (  
    error * err )
```

Gets the id of an error

**10.34.4.7 morpho\_writeerrorwithid()**

```
void morpho_writeerrorwithid (  
    error * err,  
    errorid id,  
    int line,  
    int posn,  
    ... )
```

Writes an error message to an error structure.

## Parameters

<i>err</i>	The error structure
<i>id</i>	The error id.
<i>line</i>	The line at which the error occurred, if identifiable.
<i>posn</i>	The position in the line at which the error occurred, if identifiable.
<i>...</i>	Additional parameters (the data for the printf commands in the message)

**10.34.4.8 morpho\_writeerrorwithidvalist()**

```
void morpho_writeerrorwithidvalist (  
    error * err,  
    errorid id,  
    int line,  
    int posn,  
    va_list args )
```

Writes an error message to an error structure.

## Parameters

<i>err</i>	The error structure
<i>id</i>	The error id.
<i>line</i>	The line at which the error occurred, if identifiable.
<i>posn</i>	The position in the line at which the error occurred, if identifiable.
<i>args</i>	Additional parameters (the data for the printf commands in the message)

## 10.35 utils/memory.h File Reference

Morpho memory management.

```
#include <stdlib.h>
#include "build.h"
```

### Macros

- #define `MORPHO_MALLOC(size)` `morpho_allocate(NULL, 0, size)`
- #define `MORPHO_FREE(x)` `morpho_allocate(x, 0, 0)`
- #define `MORPHO_REALLOC(x, size)` `morpho_allocate(x, 0, size)`

### Functions

- void \* `morpho_allocate` (void \*old, size\_t oldsize, size\_t newsize)  
*Generic allocator function.*

### 10.35.1 Detailed Description

Morpho memory management.

Author

T J Atherton

### 10.35.2 Macro Definition Documentation

#### 10.35.2.1 MORPHO\_FREE

```
#define MORPHO_FREE(  
    x ) morpho_allocate(x, 0, 0)
```

Macro to redirect free through our memory management

### 10.35.2.2 MORPHO\_MALLOC

```
#define MORPHO_MALLOC(  
    size ) morpho_allocate(NULL, 0, size)
```

Macro to redirect malloc through our memory management

### 10.35.2.3 MORPHO\_REALLOC

```
#define MORPHO_REALLOC(  
    x,  
    size ) morpho_allocate(x, 0, size)
```

Macro to redirect realloc through our memory management

## 10.35.3 Function Documentation

### 10.35.3.1 morpho\_allocate()

```
void* morpho_allocate (  
    void * old,  
    size_t oldsize,  
    size_t newsize )
```

Generic allocator function.

#### Parameters

<i>old</i>	A previously allocated pointer, or NULL to allocate new memory
<i>oldsize</i>	The previously allocated size
<i>newsize</i>	New size to allocate

#### Returns

A pointer to allocated memory, or NULL on failure.

## 10.36 utils/parse.c File Reference

Lexer and parser.

```
#include <string.h>  
#include "parse.h"  
#include "object.h"  
#include "common.h"
```

## Macros

- `#define UNUSED { NULL, NULL, PREC_NONE }`  
*Parse table. Each line in the table defines the [parserule\(s\)](#) for a specific token type.*
- `#define PREFIX(fn) { fn, NULL, PREC_NONE }`
- `#define INFIX(fn, prec) { NULL, fn, prec }`
- `#define MIXFIX(unaryfn, infixfn, prec) { unaryfn, infixfn, prec }`

## Functions

- void `lex_init` (lexer \*l, const char \*start, int line)  
*Initializes a lexer with a given starting point.*
- void `lex_recordtoken` (lexer \*l, tokentype type, token \*tok)  
*Records a token.*
- tokentype `lex_symboltype` (lexer \*l)  
*Determines if a token matches any of the reserved words.*
- bool `lex` (lexer \*l, token \*tok, error \*err)  
*Identifies the next token.*
- void `parse_init` (parser \*p, lexer \*lex, error \*err, syntaxtree \*tree)  
*Initialize a parser.*
- void `parse_synchronize` (parser \*p)
- syntaxtreeindx `syntaxtree_addnode` (syntaxtree \*tree, syntaxtreenodetype type, value content, int line, int posn, syntaxtreeindx left, syntaxtreeindx right)  
*Adds a node to the syntax tree.*
- bool `parse` (parser \*p)
- bool `parse_stringtovaluearray` (char \*string, unsigned int nmax, value \*v, unsigned int \*n, error \*err)

## Variables

- `parserule rules []`

### 10.36.1 Detailed Description

Lexer and parser.

Author

T J Atherton

### 10.36.2 Macro Definition Documentation

#### 10.36.2.1 UNUSED

```
#define UNUSED { NULL, NULL, PREC_NONE }
```

Parse table. Each line in the table defines the [parserule\(s\)](#) for a specific token type.

#### Warning

It is imperative that this table be in the same order as the tokentype enum



## 10.36.3 Function Documentation

### 10.36.3.1 lex()

```
bool lex (
    lexer * l,
    token * tok,
    error * err )
```

Identifies the next token.

#### Parameters

in	<i>l</i>	The lexer in use
out	<i>err</i>	An error block to fill out on an error
out	<i>tok</i>	Token structure to fill out

#### Returns

true on success or false on failure

### 10.36.3.2 lex\_init()

```
void lex_init (
    lexer * l,
    const char * start,
    int line )
```

Initializes a lexer with a given starting point.

#### Parameters

<i>l</i>	The lexer to initialize
<i>start</i>	Starting point to lex from
<i>line</i>	The current line number

### 10.36.3.3 lex\_recordtoken()

```
void lex_recordtoken (
    lexer * l,
```

```
tokentype type,  
token * tok )
```

Records a token.

## Parameters

in	<i>l</i>	The lexer in use
in	<i>type</i>	Type of token to record
out	<i>tok</i>	Token structure to fill out

**10.36.3.4 parse()**

```
bool parse (  
    parser * p )
```

Entry point into the parser

**10.36.3.5 parse\_init()**

```
void parse_init (  
    parser * p,  
    lexer * lex,  
    error * err,  
    syntaxtree * tree )
```

Initialize a parser.

## Parameters

<i>p</i>	the parser to initialize
<i>lex</i>	lexer to use
<i>err</i>	an error structure to fill out if necessary
<i>tree</i>	syntaxtree to fill out

**10.36.3.6 parse\_stringtovaluearray()**

```
bool parse_stringtovaluearray (  
    char * string,  
    unsigned int nmax,  
    value * v,  
    unsigned int * n,  
    error * err )
```

Convenience function to parse a string into an array of values

## Parameters

in	<i>string</i>	- string to parse
in	<i>nmax</i>	- maximum number of values to read

**Parameters**

in	<i>v</i>	- value array, filled out on return
out	<i>n</i>	- number of values read
out	<i>err</i>	- error structure filled out if an error occurs

**Returns**

true if successful, false otherwise.

**10.36.3.7 parse\_synchronize()**

```
void parse_synchronize (
    parser * p )
```

Keep parsing til the end of a statement boundary. Align

**10.36.3.8 syntaxtree\_addnode()**

```
syntaxtreeindx syntaxtree_addnode (
    syntaxtree * tree,
    syntaxtreenodetype type,
    value content,
    int line,
    int posn,
    syntaxtreeindx left,
    syntaxtreeindx right )
```

Adds a node to the syntax tree.

**Parameters**

<i>tree</i>	tree to add to.
<i>type</i>	type of node to add
<i>content</i>	a value to add
<i>left</i>	} left ...
<i>right</i>	} ...and right branches of the node.

**10.37 utils/parse.h File Reference**

Lexer and parser.

```
#include <stdio.h>
#include "error.h"
#include "syntaxtree.h"
```

## Classes

- struct [token](#)
- struct [lexer](#)

*Store the current configuration of a lexer.*

- struct [parser](#)

*A structure that defines the state of a parser.*

- struct [parserule](#)

*A parse rule will be defined for each token, providing functions to parse the token if it is encountered in the prefix or infix positions. The parse rule also defines the precedence.*

## Macros

- #define [TOKEN\\_BLANK](#) (([token](#)) { .type=TOKEN\_NONE, .start=NULL, .length=0, .line=0, .posn=0} )

## Typedefs

- typedef syntaxtreeindx(\* [parsefunction](#)) ([parser](#) \*c)

*Definition of a parse function.*

## Enumerations

- enum [tokentype](#) {  
[TOKEN\\_NONE](#), [TOKEN\\_NEWLINE](#), [TOKEN\\_HELP](#), [TOKEN\\_STRING](#),  
[TOKEN\\_INTERPOLATION](#), [TOKEN\\_INTEGER](#), [TOKEN\\_NUMBER](#), [TOKEN\\_SYMBOL](#),  
[TOKEN\\_TRUE](#), [TOKEN\\_FALSE](#), [TOKEN\\_NIL](#), [TOKEN\\_SELF](#),  
[TOKEN\\_SUPER](#), [TOKEN\\_LEFTPAREN](#), [TOKEN\\_RIGHTPAREN](#), [TOKEN\\_LEFTSQBRACKET](#),  
[TOKEN\\_RIGHTSQBRACKET](#), [TOKEN\\_LEFTCURLYBRACKET](#), [TOKEN\\_RIGHTCURLYBRACKET](#), [TOKEN\\_COLON](#),  
[TOKEN\\_SEMICOLON](#), [TOKEN\\_COMMA](#), [TOKEN\\_PLUS](#), [TOKEN\\_MINUS](#),  
[TOKEN\\_STAR](#), [TOKEN\\_SLASH](#), [TOKEN\\_CIRCUMFLEX](#), [TOKEN\\_PLUSPLUS](#),  
[TOKEN\\_MINUSMINUS](#), [TOKEN\\_PLUSEQ](#), [TOKEN\\_MINUSEQ](#), [TOKEN\\_STAREQ](#),  
[TOKEN\\_SLASHEQ](#), [TOKEN\\_DOT](#), [TOKEN\\_DOTDOT](#), [TOKEN\\_EXCLAMATION](#),  
[TOKEN\\_AMP](#), [TOKEN\\_VBAR](#), [TOKEN\\_DBLAMP](#), [TOKEN\\_DBLVBAR](#),  
[TOKEN\\_EQUAL](#), [TOKEN\\_EQ](#), [TOKEN\\_NEQ](#), [TOKEN\\_LT](#),  
[TOKEN\\_GT](#), [TOKEN\\_LTEQ](#), [TOKEN\\_GTEQ](#), [TOKEN\\_PRINT](#),  
[TOKEN\\_VAR](#), [TOKEN\\_IF](#), [TOKEN\\_ELSE](#), [TOKEN\\_IN](#),  
[TOKEN\\_WHILE](#), [TOKEN\\_FOR](#), [TOKEN\\_FUNCTION](#), [TOKEN\\_RETURN](#),  
[TOKEN\\_CLASS](#), [TOKEN\\_IMPORT](#), [TOKEN\\_AS](#), [TOKEN\\_INCOMPLETE](#),  
[TOKEN\\_ERROR](#), [TOKEN\\_EOF](#) }
- enum [precedence](#) {  
[PREC\\_NONE](#), [PREC\\_LOWEST](#), [PREC\\_ASSIGN](#), [PREC\\_OR](#),  
[PREC\\_AND](#), [PREC\\_EQUALITY](#), [PREC\\_COMPARISON](#), [PREC\\_RANGE](#),  
[PREC\\_TERM](#), [PREC\\_FACTOR](#), [PREC\\_POW](#), [PREC\\_UNARY](#),  
[PREC\\_CALL](#), [PREC\\_HIGHEST](#) }

*an enumerated type that defines precedence order in Morpho.*

## Functions

- void `lex_init` (`lexer` \*l, const char \*start, int line)  
*Initializes a lexer with a given starting point.*
- bool `lex` (`lexer` \*l, `token` \*tok, `error` \*err)  
*Identifies the next token.*
- void `parse_init` (`parser` \*p, `lexer` \*lex, `error` \*err, `syntaxtree` \*tree)  
*Initialize a parser.*
- bool `parse` (`parser` \*p)
- bool `parse_stringtovaluearray` (char \*string, unsigned int nmax, `value` \*v, unsigned int \*n, `error` \*err)

### 10.37.1 Detailed Description

Lexer and parser.

Author

T J Atherton and others (see below)

### 10.37.2 Macro Definition Documentation

#### 10.37.2.1 TOKEN\_BLANK

```
#define TOKEN_BLANK ((token) {.type=TOKEN_NONE, .start=NULL, .length=0, .line=0, .posn=0} )
```

Literal for a blank token

### 10.37.3 Enumeration Type Documentation

#### 10.37.3.1 tokentype

```
enum tokentype
```

The type of token. N.B. Each token will have a parserule in the parser

### 10.37.4 Function Documentation

#### 10.37.4.1 lex()

```
bool lex (  
    lexer * l,  
    token * tok,  
    error * err )
```

Identifies the next token.

**Parameters**

<i>in</i>	<i>l</i>	The lexer in use
<i>out</i>	<i>err</i>	An error block to fill out on an error
<i>out</i>	<i>tok</i>	Token structure to fill out

**Returns**

true on success or false on failure

**10.37.4.2 lex\_init()**

```
void lex_init (
    lexer * l,
    const char * start,
    int line )
```

Initializes a lexer with a given starting point.

**Parameters**

<i>l</i>	The lexer to initialize
<i>start</i>	Starting point to lex from
<i>line</i>	The current line number

**10.37.4.3 parse()**

```
bool parse (
    parser * p )
```

Entry point into the parser

**10.37.4.4 parse\_init()**

```
void parse_init (
    parser * p,
    lexer * lex,
    error * err,
    syntaxtree * tree )
```

Initialize a parser.

**Parameters**

<i>p</i>	the parser to initialize
<i>lex</i>	lexer to use
<i>err</i>	an error structure to fill out if necessary
<i>tree</i>	syntaxtree to fill out

**10.37.4.5 parse\_stringtovaluearray()**

```
bool parse_stringtovaluearray (
    char * string,
    unsigned int nmax,
    value * v,
    unsigned int * n,
    error * err )
```

Convenience function to parse a string into an array of values

**Parameters**

in	<i>string</i>	- string to parse
in	<i>nmax</i>	- maximum number of values to read
in	<i>v</i>	- value array, filled out on return
out	<i>n</i>	- number of values read
out	<i>err</i>	- error structure filled out if an error occurs

**Returns**

true if successful, false otherwise.

**10.38 utils/random.c File Reference**

Random number generation.

```
#include "random.h"
```

**Functions**

- void [splitmix64\\_seed](#) (uint64\_t seed)
- void [xoshiro256pp\\_jump](#) (void)
- void [xoshiro256pp\\_longjump](#) (void)
- void [xoshiro256p\\_jump](#) (void)
- void [xoshiro256p\\_longjump](#) (void)
- double [random\\_double](#) (void)
- unsigned int [random\\_int](#) (void)
- void [random\\_initialize](#) (void)



## Variables

- `uint64_t splitmix64_state`

### 10.38.1 Detailed Description

Random number generation.

#### Author

T J Atherton and others (see below)

### 10.38.2 Function Documentation

#### 10.38.2.1 `random_double()`

```
double random_double (  
    void )
```

Generate a random double on the interval [0.0,1.0]

#### 10.38.2.2 `random_initialize()`

```
void random_initialize (  
    void )
```

Initialize the random number generator

#### 10.38.2.3 `random_int()`

```
unsigned int random_int (  
    void )
```

Generate a random 32 bit unsigned int

#### 10.38.2.4 `splitmix64_seed()`

```
void splitmix64_seed (  
    uint64_t seed )
```

Set the seed for splitmix64

## 10.39 vm/compile.c File Reference

Compiles raw input to Morpho instructions.

```
#include <stdarg.h>
#include <string.h>
#include "compile.h"
#include "error.h"
#include "vm.h"
#include "morpho.h"
#include "file.h"
```

### Macros

- `#define NODE_NORULE NULL`
- `#define NODE_UNDEFINED { NODE_NORULE }`

### Functions

- void `compiler_adddebuginfo` (`compiler *c`, `syntaxtreenode *node`)  
*Adds source info to the program Debugging info is stored in a simple list with run length encoding, i.e. Each entry corresponds to ninstr instructions.*
- `DEFINE_VARRAY` (`registeralloc`, `registeralloc`)
- void `compiler_fstackinit` (`compiler *c`)
- void `compiler_fstackclear` (`compiler *c`)
- void `compiler_beginscope` (`compiler *c`)
- void `compiler_endscope` (`compiler *c`)
- unsigned int `compiler_currentscope` (`compiler *c`)
- `registeridx` `compiler_addupvalue` (`functionstate *f`, bool islocal, `indx ix`)
- `indx` `compiler_closure` (`compiler *c`, `syntaxtreenode *node`, `registeridx reg`)
- `codeinfo` `compiler_movefromregister` (`compiler *c`, `syntaxtreenode *node`, `codeinfo dest`, `registeridx reg`)
- bool `compiler_listentries` (`compiler *c`, `syntaxtreenode *node`)
- `objectclass *` `compiler_findclass` (`objectfunction *f`, `value name`)
- void `compiler_stripend` (`compiler *c`)
- void `compiler_copyglobals` (`compiler *src`, `compiler *dest`)
- void `compiler_init` (const char \*source, program \*out, `compiler *c`)  
*Initialize a compiler.*
- void `compiler_clear` (`compiler *c`)  
*Clear attached data structures from a compiler.*
- bool `morpho_compile` (char \*in, `compiler *c`, `error *err`)
- `compiler *` `morpho_newcompiler` (program \*out)
- void `morpho_freecompiler` (`compiler *c`)
- void `compile_initialize` (void)
- void `compile_finalize` (void)

### Variables

- `compilenoderule noderules []`

## 10.39.1 Detailed Description

Compiles raw input to Morpho instructions.

Author

T J Atherton

## 10.39.2 Function Documentation

### 10.39.2.1 compile\_finalize()

```
void compile_finalize (
    void )
```

Finalizes the compiler

### 10.39.2.2 compile\_initialize()

```
void compile_initialize (
    void )
```

Initializes the error handling system

### 10.39.2.3 compiler\_addupvalue()

```
registerindx compiler_addupvalue (
    functionstate * f,
    bool islocal,
    indx ix )
```

Adds an upvalue to a functionstate

### 10.39.2.4 compiler\_beginscope()

```
void compiler_beginscope (
    compiler * c )
```

Increments the scope counter in the current functionstate

### 10.39.2.5 compiler\_clear()

```
void compiler_clear (
    compiler * c )
```

Clear attached data structures from a compiler.

## Parameters

in	c	compiler to clear
----	---	-------------------

**10.39.2.6 compiler\_closure()**

```
indx compiler_closure (
    compiler * c,
    syntaxtreenode * node,
    registerindx reg )
```

Creates code to generate a closure for the current environment.

**10.39.2.7 compiler\_copyglobals()**

```
void compiler_copyglobals (
    compiler * src,
    compiler * dest )
```

Copies the globals across from one compiler to another

**10.39.2.8 compiler\_currentscope()**

```
unsigned int compiler_currentscope (
    compiler * c )
```

Decrements the scope counter in the current functionstate

**10.39.2.9 compiler\_endscope()**

```
void compiler_endscope (
    compiler * c )
```

Decrements the scope counter in the current functionstate

**10.39.2.10 compiler\_findclass()**

```
objectclass* compiler_findclass (
    objectfunction * f,
    value name )
```

Finds a class in the constant table of a function

### 10.39.2.11 compiler\_fstackclear()

```
void compiler_fstackclear (
    compiler * c )
```

Clears the function stack

### 10.39.2.12 compiler\_fstackinit()

```
void compiler_fstackinit (
    compiler * c )
```

Initializes the function stack

### 10.39.2.13 compiler\_init()

```
void compiler_init (
    const char * source,
    program * out,
    compiler * c )
```

Initialize a compiler.

#### Parameters

in	<i>source</i>	source code to compile
in	<i>out</i>	destination program to compile to
out	<i>c</i>	compiler structure is filled out

### 10.39.2.14 compiler\_stripend()

```
void compiler_stripend (
    compiler * c )
```

Strip an 'end' instruction at the end of the program

### 10.39.2.15 morpho\_compile()

```
bool morpho_compile (
    char * in,
    compiler * c,
    error * err )
```

Interface to the compiler

**Parameters**

in	<i>in</i>	A string to compile
in	<i>c</i>	The compiler
out	<i>err</i>	Pointer to error block on failure

**Returns**

A bool indicating success or failure

**10.39.2.16 morpho\_freecompiler()**

```
void morpho_freecompiler (
    compiler * c )
```

Frees a compiler

**10.39.2.17 morpho\_newcompiler()**

```
compiler* morpho_newcompiler (
    program * out )
```

Creates a new compiler

**10.39.3 Variable Documentation****10.39.3.1 noderules**

```
compilenoderule noderules[ ]
```

Compiler definition table. This specifies a compiler function that handles each node type

**10.40 vm/compile.h File Reference**

Compiles raw input to Morpho instructions.

```
#include "core.h"
#include "syntaxtree.h"
#include "parse.h"
```

## Classes

- struct [registeralloc](#)
- struct [functionstate](#)
- struct [scompilerlist](#)
- struct [scompiler](#)

*A structure that stores the state of a compiler.*

- struct [codeinfo](#)
- struct [compilenoderule](#)

*A compilenoderule rule will be defined for each syntax tree node type, providing a function to compile the node.*

## Macros

- #define **MORPHO\_CORE**
- #define [GLOBAL\\_UNALLOCATED](#) -1  
*Value to indicate a global has not been allocated.*
- #define [REGISTER\\_UNALLOCATED](#) -1  
*Value to indicate a register has not been allocated.*
- #define **FUNCTIONTYPE\_ISMETHOD**(f) (f==FUNCTION ? false : true)
- #define **FUNCTIONTYPE\_ISINITIALIZER**(f) (f==INITIALIZER)
- #define **CODEINFO\_ISREGISTER**(info) (info.returntype==REGISTER)
- #define **CODEINFO\_ISCONSTANT**(info) (info.returntype==CONSTANT)
- #define **CODEINFO\_ISUPVALUE**(info) (info.returntype==UPVALUE)
- #define **CODEINFO\_ISGLOBAL**(info) (info.returntype==GLOBAL)
- #define **CODEINFO**(c, d, n) (([codeinfo](#)) { .returntype=(c), .dest=(d), .ninstructions=(n)})
- #define **CODEINFO\_EMPTY** CODEINFO(REGISTER, [REGISTER\\_UNALLOCATED](#), 0)

## Typedefs

- typedef int [globalindx](#)  
*Index of a global.*
- typedef int [registerindx](#)  
*Index of a register.*
- typedef struct [scompilerlist](#) [compilerlist](#)
- typedef struct [scompiler](#) [compiler](#)  
*A structure that stores the state of a compiler.*
- typedef [codeinfo](#)(\* [compiler\\_nodfn](#)) ([compiler](#) \*c, [syntaxtreenode](#) \*node, [registerindx](#) reg)

## Enumerations

- enum [functiontype](#) { **FUNCTION**, **METHOD**, **INITIALIZER** }

## Functions

- void [compiler\\_init](#) (const char \*source, program \*out, [compiler](#) \*c)  
*Initialize a compiler.*
- void [compiler\\_clear](#) ([compiler](#) \*c)  
*Clear attached data structures from a compiler.*

### 10.40.1 Detailed Description

Compiles raw input to Morpho instructions.

Author

T J Atherton

### 10.40.2 Typedef Documentation

#### 10.40.2.1 compiler\_nodfn

```
typedef codeinfo(* compiler_nodfn) (compiler *c, syntactreenode *node, registerindx reg)
```

A compiler\_nodfn takes a syntax tree node and compiles it to bytecode

#### 10.40.2.2 compilerlist

```
typedef struct scompilerlist compilerlist
```

This structure holds a list as it is being created

### 10.40.3 Enumeration Type Documentation

#### 10.40.3.1 functiontype

```
enum functiontype
```

The type of the current function

### 10.40.4 Function Documentation

#### 10.40.4.1 compiler\_clear()

```
void compiler_clear (  
    compiler * c )
```

Clear attached data structures from a compiler.



## Parameters

in	<i>c</i>	compiler to clear
----	----------	-------------------

## 10.40.4.2 compiler\_init()

```
void compiler_init (
    const char * source,
    program * out,
    compiler * c )
```

Initialize a compiler.

## Parameters

in	<i>source</i>	source code to compile
in	<i>out</i>	destination program to compile to
out	<i>c</i>	compiler structure is filled out

## 10.41 vm/core.h File Reference

Data types for core Morpho components.

```
#include <stdio.h>
#include <stddef.h>
#include "error.h"
#include "random.h"
#include "varray.h"
#include "value.h"
#include "object.h"
#include "common.h"
#include "dictionary.h"
#include "builtin.h"
#include "opcodes.h"
```

## Classes

- struct [callframe](#)
- struct [debuginfo](#)
- struct [sprogram](#)

*Morpho code program and associated data.*

- struct [graylist](#)
- struct [svm](#)

*A Morpho virtual machine and its current state.*

## Macros

- `#define ENCODE(op, A, B, C) ( (((unsigned) op) & 0xff) << 26) | ((A & 0xff) << 18) | ((B & 0xff) << 9) | (C & 0xff) )`
- `#define ENCODEC(op, A, Bc, B, Cc, C) ( (((unsigned) op) & 0xff) << 26) | ((A & 0xff) << 18) | ((Bc & 0x1) << 17) | ((B & 0xff) << 9) | ((Cc & 0x1) << 8) | (C & 0xff) )`
- `#define ENCODE_LONG(op, A, Bx) ( (((unsigned) op) & 0xff) << 26) | ((A & 0xff) << 18) | (Bx & 0xffff) )`
- `#define ENCODE_LONGFLAGS(op, A, f, g, Bx) ( (((unsigned) op) & 0xff) << 26) | ((A & 0xff) << 18) | ((f & 0x1) << 17) | ((g & 0x1) << 16) | (Bx & 0xffff) )`
- `#define ENCODE_SINGLE(op, A) ( (((unsigned) op) & 0xff) << 26) | ((A & 0xff) << 18))`
- `#define ENCODE_DOUBLE(op, A, Bc, B) ( (((unsigned) op) & 0xff) << 26) | ((A & 0xff) << 18) | ((Bc & 0x1) << 17) | ((B & 0xff) << 9) )`
- `#define ENCODE_BYTE(op) (((unsigned) op) & 0xff) << 26)`
- `#define ENCODE_EMPTYOPERAND 0`
- `#define DECODE_OP(x) (x >> 26)`
- `#define DECODE_A(x) ((x >> 18) & (0xff))`
- `#define DECODE_B(x) ((x >> 9) & (0xff))`
- `#define DECODE_C(x) (x & 0xff)`
- `#define DECODE_ISBCONSTANT(x) ((x >> 17) & (0x1))`
- `#define DECODE_ISCCONSTANT(x) ((x >> 8) & (0x1))`
- `#define DECODE_Bx(x) (x & 0xffff)`
- `#define DECODE_sBx(x) ((short) (x & 0xffff))`
- `#define DECODE_F(x) ((bool) ((x >> 17) & (0x1)))`
- `#define DECODE_G(x) ((bool) ((x >> 16) & (0x1)))`
- `#define OPCODE(name) OP_##name,`
- `#define MORPHO_MAXREGISTERS 255`  
*Maximum number of registers per call frame.*
- `#define VM_MAXIMUMREGISTERNUMBER 255`  
*Highest register addressable in a window.*

## Typedefs

- `typedef struct sprogram program`
- `typedef struct svm vm`
- `typedef unsigned int instruction`  
*A Morpho instruction.*
- `typedef indx instructionindx`  
*Index into instructions.*

## Enumerations

- `enum opcode`

## Functions

- `DECLARE_VARRAY (instruction, instruction)`
- `void compile_initialize (void)`
- `void compile_finalize (void)`

### 10.41.1 Detailed Description

Data types for core Morpho components.

Author

T J Atherton

### 10.41.2 Macro Definition Documentation

#### 10.41.2.1 DECODE\_A

```
#define DECODE_A(  
    x ) ((x>>18) & (0xff))
```

Decode operand A

#### 10.41.2.2 DECODE\_B

```
#define DECODE_B(  
    x ) ((x>>9) & (0xff))
```

Decode operand B

#### 10.41.2.3 DECODE\_Bx

```
#define DECODE_Bx(  
    x ) (x & 0xffff)
```

Decode long operand Bx

#### 10.41.2.4 DECODE\_C

```
#define DECODE_C(  
    x ) (x & 0xff)
```

Decode operand C

#### 10.41.2.5 DECODE\_F

```
#define DECODE_F(  
    x ) ((bool) ((x>>17) & (0x1)))
```

Decode flags F and G

**10.41.2.6 DECODE\_ISBCONSTANT**

```
#define DECODE_ISBCONSTANT(
    x ) ((x>>17) & (0x1))
```

Is Operand B a constant or a register?

**10.41.2.7 DECODE\_ISCCONSTANT**

```
#define DECODE_ISCCONSTANT(
    x ) ((x>>8) & (0x1))
```

Is Operand C a constant or a register?

**10.41.2.8 DECODE\_OP**

```
#define DECODE_OP(
    x ) (x >> 26)
```

Decode the opcode

**10.41.2.9 DECODE\_sBx**

```
#define DECODE_sBx(
    x ) ((short) (x & 0xffff))
```

Decode signed long operand Bx

**10.41.2.10 ENCODE**

```
#define ENCODE(
    op,
    A,
    B,
    C ) ( (((unsigned) op) & 0xff) << 26 | ((A & 0xff) << 18) | ((B & 0xff) << 9)
    | (C & 0xff) )
```

Encodes an instruction with operands A, B and C.

**10.41.2.11 ENCODE\_BYTE**

```
#define ENCODE_BYTE(
    op ) (((unsigned) op) & 0xff) << 26)
```

Encodes an instruction with no operands

**10.41.2.12 ENCODE\_DOUBLE**

```
#define ENCODE_DOUBLE(
    op,
    A,
    Bc,
    B ) ( (((unsigned) op) & 0xff) << 26) | ((A & 0xff) << 18) | ((Bc & 0x1) <<
17) | ((B & 0xff) << 9) )
```

Encodes an instruction with operands A and B

**10.41.2.13 ENCODE\_EMPTYOPERAND**

```
#define ENCODE_EMPTYOPERAND 0
```

Encodes an empty operand

**10.41.2.14 ENCODE\_LONG**

```
#define ENCODE_LONG(
    op,
    A,
    Bx ) ( (((unsigned) op) & 0xff) << 26) | ((A & 0xff) << 18) | (Bx & 0xffff) )
```

Encodes an instruction with operand A and long operand Bx

**10.41.2.15 ENCODE\_LONGFLAGS**

```
#define ENCODE_LONGFLAGS(
    op,
    A,
    f,
    g,
    Bx ) ( (((unsigned) op) & 0xff) << 26) | ((A & 0xff) << 18) | ((f & 0x1) <<
17) | ((g & 0x1) << 16) | (Bx & 0xffff) )
```

Encodes an instruction with operand A and long operand Bx

**10.41.2.16 ENCODE\_SINGLE**

```
#define ENCODE_SINGLE(
    op,
    A ) ( (((unsigned) op) & 0xff) << 26) | ((A & 0xff) << 18))
```

Encodes an instruction with operand A

### 10.41.2.17 ENCODEC

```
#define ENCODEC(
    op,
    A,
    Bc,
    B,
    Cc,
    C ) ( (((unsigned) op) & 0xff) << 26) | ((A & 0xff) << 18) | ((Bc & 0x1) <<
17) | ((B & 0xff) << 9) | ((Cc & 0x1) << 8) | (C & 0xff) )
```

Encodes an instruction with operands A, B and C also setting the constant flags Bc and Cc

## 10.41.3 Typedef Documentation

### 10.41.3.1 instruction

```
typedef unsigned int instruction
```

A Morpho instruction.

Each instruction fits into a 32 bit unsigned int with the following arrangement

24	16	8	0	
.....	.....	.....	.....	
<b>**op**</b>				Opcode (gives 64 instructions)
<b>***A***</b>				Operand A
*                  *				Select Constant (set) or Register for operands B & C
<b>***B***    ***C***</b>				} Operands B & C
<b>fg**Bx*****</b>				} Operand Bx with two boolean flags f & g
<b>fgsBx*****</b>				} Signed operand Bx with two boolean flags f & g

Morpho instructions

## 10.41.4 Function Documentation

### 10.41.4.1 compile\_finalize()

```
void compile_finalize (
    void )
```

Finalizes the compiler

#### 10.41.4.2 compile\_initialize()

```
void compile_initialize (
    void )
```

Initializes the error handling system

## 10.42 vm/opcodes.h File Reference

Morpho opcodes.

### 10.42.1 Detailed Description

Morpho opcodes.

Author

T J Atherton

## 10.43 vm/vm.c File Reference

Morpho virtual machine.

```
#include <stdarg.h>
#include <time.h>
#include "vm.h"
#include "compile.h"
#include "morpho.h"
#include "object.h"
```

### Macros

- #define [DISASSEMBLE\\_SHOWA](#) 1  
*Disassembles a single ABC format instruction.*
- #define **DISASSEMBLE\_SHOWB** 2
- #define **DISASSEMBLE\_SHOWC** 4
- #define **DISASSEMBLE\_LITC** 8
- #define [DISASSEMBLE\\_SHOWA](#) 1  
*Disassembles a single ABC format instruction.*
- #define **DISASSEMBLE\_SHOWBx** 2
- #define **DISASSEMBLE\_SHOWsBx** 4
- #define **DISASSEMBLE\_SHOWF** 8
- #define **DISASSEMBLE\_SHOWG** 16
- #define **DISASSEMBLE\_OP**(op) printf("%s", op)
- #define **DISASSEMBLE\_OPA**(op, i) disassemble\_ABx(op, i, [DISASSEMBLE\\_SHOWA](#), reg)
- #define **DISASSEMBLE\_OPAB**(op, i)
- #define **DISASSEMBLE\_OPAcB**(op, i)

- #define **DISASSEMBLE\_OPABC**(op, i)
- #define **DISASSEMBLE\_OPB**(op, i)
- #define **DISASSEMBLE\_OPABx**(op, i) disassemble\_ABx(op, i, [DISASSEMBLE\\_SHOWA](#) | [DISASSEMBLE\\_SHOWBx](#), reg)
- #define **DISASSEMBLE\_OPBx**(op, i) disassemble\_ABx(op, i, [DISASSEMBLE\\_SHOWBx](#), reg)
- #define **DISASSEMBLE\_OPAsBx**(op, i) disassemble\_ABx(op, i, [DISASSEMBLE\\_SHOWA](#) | [DISASSEMBLE\\_SHOWBx](#), reg)
- #define **DISASSEMBLE\_OPsBx**(op, i) disassemble\_ABx(op, i, [DISASSEMBLE\\_SHOWsBx](#), reg)
- #define **MORPHO\_DISASSEMBLE\_INSTRUCTION**(bc, pc, k, r) ;
- #define **OPCODECNT**(p)
- #define **OPOPCODECNT**(p, bc)
- #define **INTERPRET\_LOOP**
- #define **CASE\_CODE**(name) case OP\_##name
- #define **DISPATCH**() goto loop;
- #define **ERROR**(id) { [vm\\_runtimeerror](#)(v, pc-instructions, id); goto vm\_error; }
- #define **VError**(id, ...) { [vm\\_runtimeerror](#)(v, pc-instructions, id, \_\_VA\_ARGS\_\_); goto vm\_error; }
- #define **ERRORCHK**() if (v->err.cat!=ERROR\_NONE) goto vm\_error;
- #define **CHECKCMPTYPE**(l, r)

## Functions

- **DEFINE\_VARRAY** ([instruction](#), [instruction](#))
- **DEFINE\_VARRAY** ([debuginfo](#), [debuginfo](#))
- program \* [morpho\\_newprogram](#) (void)  
*Creates and initializes a new program.*
- void [morpho\\_freeprogram](#) (program \*p)  
*Frees a program.*
- void [program\\_setentry](#) (program \*p, [instructionindx](#) entry)
- [instructionindx](#) [program\\_getentry](#) (program \*p)
- [debuginfo](#) \* [morpho\\_getdebugfromindx](#) (program \*p, [instructionindx](#) indx)
- void [program\\_bindobject](#) (program \*p, [object](#) \*obj)  
*Binds an object to a program.*
- [value](#) [program\\_internsymbol](#) (program \*p, [value](#) symbol)  
*Interns a symbol into the programs symbol table.*
- void [vm\\_graylistinit](#) ([graylist](#) \*g)
- void [vm\\_graylistclear](#) ([graylist](#) \*g)
- void [vm\\_graylistadd](#) ([graylist](#) \*g, [object](#) \*obj)
- vm \* [morpho\\_newvm](#) (void)
- void [morpho\\_freemv](#) (vm \*v)
- void [vm\\_freeobjects](#) (vm \*v)
- void [vm\\_collectgarbage](#) (vm \*v)
- void [vm\\_bindobject](#) (vm \*v, [value](#) obj)  
*Binds an object to a Virtual Machine.*
- void [vm\\_disassembleinstruction](#) ([instruction](#) [instruction](#), ptrdiff\_t indx, [value](#) \*konst, [value](#) \*reg)  
*Disassembles a single instruction, writing the output to the console.*
- void [morpho\\_disassemble](#) (program \*code, unsigned int \*matchline)
- void [morpho\\_stacktrace](#) (vm \*v)
- void [vm\\_gcmarkobject](#) (vm \*v, [object](#) \*obj)
- void [vm\\_gcmarkvalue](#) (vm \*v, [value](#) val)
- void [vm\\_gcmarkdictionary](#) (vm \*v, [dictionary](#) \*dict)
- void [vm\\_gcmarkarray](#) (vm \*v, [varray\\_value](#) \*array)
- void [vm\\_gcmarkroots](#) (vm \*v)
- void [vm\\_gcmarkretainobject](#) (vm \*v, [object](#) \*obj)



- void `vm_gctrace` (vm \*v)
- void `vm_gcsweep` (vm \*v)
- void `vm_runtimeerror` (vm \*v, ptrdiff\_t iidx, `errorid` id,...)  
*Raises a runtime error.*
- void `morpho_runtimeerror` (vm \*v, `errorid` id,...)  
*Public interface to raise a runtime error.*
- bool `morpho_interpret` (vm \*v, program \*code, `error` \*err)  
*Executes a sequence of code.*
- void `morpho_initialize` (void)
- void `morpho_finalize` (void)

## Variables

- value `initselector` = MORPHO\_NIL
- value `indexselector` = MORPHO\_NIL
- value `setindexselector` = MORPHO\_NIL
- value `addselector` = MORPHO\_NIL
- value `subselector` = MORPHO\_NIL
- value `mulselector` = MORPHO\_NIL
- value `divselector` = MORPHO\_NIL
- value `printselector` = MORPHO\_NIL
- value `enumerateselector` = MORPHO\_NIL
- vm \* `globalvm` = NULL

### 10.43.1 Detailed Description

Morpho virtual machine.

Author

T J Atherton

### 10.43.2 Macro Definition Documentation

#### 10.43.2.1 CHECKCMPTYPE

```
#define CHECKCMPTYPE(
    l,
    r )
```

Value:

```
if (!morpho_ofsametype(l, r)) { \
    if (MORPHO_ISINTEGER(l) && MORPHO_ISFLOAT(r)) { \
        l = MORPHO_INTEGERTOFloat(l); \
    } else if (MORPHO_ISFLOAT(l) && MORPHO_ISINTEGER(r)) { \
        r = MORPHO_INTEGERTOFloat(r); \
    } \
}
```

### 10.43.2.2 DISASSEMBLE\_OPAB

```
#define DISASSEMBLE_OPAB(  
    op,  
    i )
```

**Value:**

```
disassemble_ABC(op, i, DISASSEMBLE_SHOWA | \  
DISASSEMBLE_SHOWB, konst, reg)
```

### 10.43.2.3 DISASSEMBLE\_OPABC

```
#define DISASSEMBLE_OPABC(  
    op,  
    i )
```

**Value:**

```
disassemble_ABC(op, i, DISASSEMBLE_SHOWA | \  
DISASSEMBLE_SHOWB | DISASSEMBLE_SHOWC, \  
konst, reg)
```

### 10.43.2.4 DISASSEMBLE\_OPAcB

```
#define DISASSEMBLE_OPAcB(  
    op,  
    i )
```

**Value:**

```
disassemble_ABC(op, i, DISASSEMBLE_SHOWA | \  
DISASSEMBLE_SHOWB | DISASSEMBLE_SHOWAc, konst, reg)
```

### 10.43.2.5 DISASSEMBLE\_OPB

```
#define DISASSEMBLE_OPB(  
    op,  
    i )
```

**Value:**

```
disassemble_ABC(op, i, DISASSEMBLE_SHOWB, \  
konst, reg)
```

### 10.43.2.6 DISASSEMBLE\_SHOWA [1/2]

```
#define DISASSEMBLE_SHOWA 1
```

Disassembles a single ABC format instruction.

Disassembles a single ABx format instruction.

## Parameters

<i>op</i>	string to print to describe the op
<i>i</i>	the instruction
<i>show</i>	which operands to show
<i>konst</i>	current constant table
<i>reg</i>	current registers
<i>op</i>	string to print to describe the op
<i>i</i>	the instruction
<i>show</i>	which operands to show
<i>reg</i>	current registers

### 10.43.2.7 DISASSEMBLE\_SHOWA [2/2]

```
#define DISASSEMBLE_SHOWA 1
```

Disassembles a single ABC format instruction.

Disassembles a single ABx format instruction.

## Parameters

<i>op</i>	string to print to describe the op
<i>i</i>	the instruction
<i>show</i>	which operands to show
<i>konst</i>	current constant table
<i>reg</i>	current registers
<i>op</i>	string to print to describe the op
<i>i</i>	the instruction
<i>show</i>	which operands to show
<i>reg</i>	current registers

### 10.43.2.8 INTERPRET\_LOOP

```
#define INTERPRET_LOOP
```

**Value:**

```
loop:
bc=pc++;
OPOPCODECNT(pp, bc)
op=DECODE_OP(bc);
OPCODECNT(op)
MORPHO_DISASSEMBLE_INSTRUCTION(bc,pc-instructions,konst, reg)
switch (op)
```

### 10.43.3 Function Documentation

### 10.43.3.1 morpho\_disassemble()

```
void morpho_disassemble (
    program * code,
    unsigned int * matchline )
```

Disassembles a program

#### Parameters

<i>code</i>	- program to disassemble
<i>matchline</i>	- optional line number to match

### 10.43.3.2 morpho\_finalize()

```
void morpho_finalize (
    void )
```

Finalizes morpho

### 10.43.3.3 morpho\_freemv()

```
void morpho_freemv (
    vm * v )
```

Frees a virtual machine

### 10.43.3.4 morpho\_getdebugfromindx()

```
debuginfo* morpho_getdebugfromindx (
    program * p,
    instructionindx indx )
```

Finds the debugging info associated with instruction at indx

### 10.43.3.5 morpho\_initialize()

```
void morpho_initialize (
    void )
```

Initializes morpho

### 10.43.3.6 morpho\_interpret()

```
bool morpho_interpret (
    vm * v,
    program * code,
    error * err )
```

Executes a sequence of code.

## Parameters

<i>v</i>	The virtual machine to use
<i>code</i>	The program to execute

## Returns

A morpho error

**10.43.3.7 morpho\_newvm()**

```
vm* morpho_newvm (
    void )
```

Creates a new virtual machine

**10.43.3.8 morpho\_runtimeerror()**

```
void morpho_runtimeerror (
    vm * v,
    errorid id,
    ... )
```

Public interface to raise a runtime error.

## Parameters

<i>v</i>	the virtual machine
<i>id</i>	error id
...	additional data for sprintf.

**10.43.3.9 morpho\_stacktrace()**

```
void morpho_stacktrace (
    vm * v )
```

Prints a stacktrace

**10.43.3.10 program\_bindobject()**

```
void program_bindobject (
    program * p,
    object * obj )
```

Binds an object to a program.

Objects bound to the program are freed with the program; use for static data (e.g. held in constant tables)

#### 10.43.3.11 program\_getentry()

```
instructionindx program_getentry (
    program * p )
```

Gets the entry point of a program

#### 10.43.3.12 program\_internsymbol()

```
value program_internsymbol (
    program * p,
    value symbol )
```

Interns a symbol into the programs symbol table.

Note that the string is cloned if it does not exist already. Interning is used to accelerate dynamic lookups as the same string for a symbol will be used universally

#### 10.43.3.13 program\_setentry()

```
void program_setentry (
    program * p,
    instructionindx entry )
```

Sets the entry point of a program

#### 10.43.3.14 vm\_bindobject()

```
void vm_bindobject (
    vm * v,
    value obj )
```

Binds an object to a Virtual Machine.

Any object created during execution should be bound to a VM; this object is then managed by the garbage collector.

##### Parameters

<i>v</i>	the virtual machine
<i>obj</i>	object to bind

#### 10.43.3.15 vm\_collectgarbage()

```
void vm_collectgarbage (
    vm * v )
```

Collects garbage

### 10.43.3.16 vm\_disassembleinstruction()

```
void vm_disassembleinstruction (
    instruction instruction,
    ptrdiff_t indx,
    value * konst,
    value * reg )
```

Disassembles a single instruction, writing the output to the console.

#### Parameters

<i>instruction</i>	The instruction to disassemble
<i>indx</i>	Instruction index to display
<i>konst</i>	current constant table
<i>reg</i>	current registers

### 10.43.3.17 vm\_freeobjects()

```
void vm_freeobjects (
    vm * v )
```

Frees all objects bound to a virtual machine

### 10.43.3.18 vm\_gcmarkarray()

```
void vm_gcmarkarray (
    vm * v,
    varray_value * array )
```

Marks all entries in an array

### 10.43.3.19 vm\_gcmarkdictionary()

```
void vm_gcmarkdictionary (
    vm * v,
    dictionary * dict )
```

Marks all entries in a dictionary

### 10.43.3.20 vm\_gcmarkobject()

```
void vm_gcmarkobject (
    vm * v,
    object * obj )
```

Marks an object as reachable

#### 10.43.3.21 vm\_gcmarkroots()

```
void vm_gcmarkroots (
    vm * v )
```

Searches a vm for all reachable objects Mark anything on the stack

Mark closure objects in use

#### 10.43.3.22 vm\_gcmarkvalue()

```
void vm_gcmarkvalue (
    vm * v,
    value val )
```

Marks a value as reachable

#### 10.43.3.23 vm\_gcsweep()

```
void vm_gcsweep (
    vm * v )
```

Go through the VM's object list and free all unmarked objects

#### 10.43.3.24 vm\_gctrace()

```
void vm_gctrace (
    vm * v )
```

Trace all objects on the graylist

#### 10.43.3.25 vm\_runtimeerror()

```
void vm_runtimeerror (
    vm * v,
    ptrdiff_t iindx,
    errorid id,
    ... )
```

Raises a runtime error.

##### Parameters

<i>v</i>	the virtual machine
<i>id</i>	error id
...	additional data for sprintf.



## 10.44 vm/vm.h File Reference

The Morpho virtual machine.

```
#include "core.h"
```

### Macros

- `#define MORPHO_CORE`
- `#define MORPHO_PROGRAMSTART 0`

### Functions

- void `program_setentry` (program \*p, `instructionindx` entry)
- `instructionindx` `program_getentry` (program \*p)
- `varray_value` \* `program_getconstanttable` (program \*p)
- void `program_bindobject` (program \*p, `object` \*obj)  
*Binds an object to a program.*
- `value` `program_internsymbol` (program \*p, `value` symbol)  
*Interns a symbol into the programs symbol table.*
- void `vm_disassembleinstruction` (`instruction` bc, `ptrdiff_t` pc, `value` \*konst, `value` \*reg)  
*Disassembles a single instruction, writing the output to the console.*
- void `morpho_bindobject` (vm \*v, `value` obj)
- void `vm_freeobjects` (vm \*v)
- void `morpho_collectgarbage` (void)
- void `morpho_initialize` (void)
- void `morpho_finalize` (void)

### 10.44.1 Detailed Description

The Morpho virtual machine.

Author

T J Atherton

### 10.44.2 Function Documentation

#### 10.44.2.1 `morpho_finalize()`

```
void morpho_finalize (
    void )
```

Finalizes morpho

#### 10.44.2.2 morpho\_initialize()

```
void morpho_initialize (
    void )
```

Initializes morpho

#### 10.44.2.3 program\_bindobject()

```
void program_bindobject (
    program * p,
    object * obj )
```

Binds an object to a program.

Objects bound to the program are freed with the program; use for static data (e.g. held in constant tables)

#### 10.44.2.4 program\_getentry()

```
instructionidx program_getentry (
    program * p )
```

Gets the entry point of a program

#### 10.44.2.5 program\_internsymbol()

```
value program_internsymbol (
    program * p,
    value symbol )
```

Interns a symbol into the programs symbol table.

Note that the string is cloned if it does not exist already. Interning is used to accelerate dynamic lookups as the same string for a symbol will be used universally

#### 10.44.2.6 program\_setentry()

```
void program_setentry (
    program * p,
    instructionidx entry )
```

Sets the entry point of a program

#### 10.44.2.7 vm\_disassembleinstruction()

```
void vm_disassembleinstruction (
    instruction instruction,
    ptrdiff_t indx,
    value * konst,
    value * reg )
```

Disassembles a single instruction, writing the output to the console.

## Parameters

<i>instruction</i>	The instruction to disassemble
<i>indx</i>	Instruction index to display
<i>konst</i>	current constant table
<i>reg</i>	current registers

**10.44.2.8 vm\_freeobjects()**

```
void vm_freeobjects (
    vm * v )
```

Frees all objects bound to a virtual machine



# Index

- [\\_syntaxtreenode](#), [19](#)
- [bound](#)
  - [svm](#), [40](#)
- [BSD\\_EX\\_SOFTWARE](#)
  - [error.h](#), [180](#)
- [build.h](#), [45](#)
  - [MORPHO\\_EPS](#), [46](#)
  - [MORPHO\\_LINALG\\_USE\\_ACCELERATE](#), [46](#)
  - [MORPHO\\_LINALG\\_USE\\_CSPARSE](#), [46](#)
- [builtin.c](#)
  - [builtin\\_addclass](#), [48](#)
  - [builtin\\_addfunction](#), [48](#)
  - [builtin\\_copysymboltable](#), [49](#)
  - [builtin\\_findclass](#), [49](#)
  - [builtin\\_findfunction](#), [49](#)
  - [builtin\\_internsymbol](#), [49](#)
  - [builtin\\_internsymbolascstring](#), [49](#)
  - [builtin\\_printfunction](#), [49](#)
  - [Dictionary\\_getindex](#), [50](#)
  - [Dictionary\\_setindex](#), [50](#)
  - [Object\\_class](#), [50](#)
  - [Object\\_clone](#), [50](#)
  - [Object\\_init](#), [50](#)
  - [object\\_newrange](#), [50](#)
  - [Object\\_serialize](#), [51](#)
  - [Object\\_super](#), [51](#)
  - [objectveneer](#), [52](#)
  - [range\\_constructor](#), [51](#)
  - [range\\_count](#), [51](#)
  - [Range\\_enumerate](#), [51](#)
  - [range\\_iterate](#), [51](#)
  - [String\\_length](#), [52](#)
- [builtin.h](#)
  - [builtin\\_addclass](#), [56](#)
  - [builtin\\_addfunction](#), [56](#)
  - [builtin\\_copysymboltable](#), [56](#)
  - [builtin\\_findclass](#), [57](#)
  - [builtin\\_findfunction](#), [57](#)
  - [builtin\\_internsymbol](#), [57](#)
  - [builtin\\_internsymbolascstring](#), [57](#)
  - [builtin\\_printfunction](#), [57](#)
  - [builtinfunction](#), [55](#)
  - [builtinfunctionflags](#), [55](#)
  - [MORPHO\\_BEGINCLASS](#), [54](#)
  - [MORPHO\\_ENDCLASS](#), [54](#)
  - [MORPHO\\_GETARG](#), [54](#)
  - [MORPHO\\_GETBUILTINFUNCTION](#), [54](#)
  - [MORPHO\\_ISBUILTINFUNCTION](#), [54](#)
  - [MORPHO\\_RAISE](#), [55](#)
  - [MORPHO\\_RAISEVARGS](#), [55](#)
  - [MORPHO\\_SELF](#), [55](#)
- [builtin/builtin.c](#), [47](#)
- [builtin/builtin.h](#), [52](#)
- [builtin/file.c](#), [57](#)
- [builtin/file.h](#), [60](#)
- [builtin/functions.c](#), [62](#)
- [builtin/functions.h](#), [63](#)
- [builtin\\_addclass](#)
  - [builtin.c](#), [48](#)
  - [builtin.h](#), [56](#)
- [builtin\\_addfunction](#)
  - [builtin.c](#), [48](#)
  - [builtin.h](#), [56](#)
- [builtin\\_clock](#)
  - [functions.c](#), [62](#)
- [builtin\\_copysymboltable](#)
  - [builtin.c](#), [49](#)
  - [builtin.h](#), [56](#)
- [builtin\\_exp](#)
  - [functions.c](#), [62](#)
- [builtin\\_findclass](#)
  - [builtin.c](#), [49](#)
  - [builtin.h](#), [57](#)
- [builtin\\_findfunction](#)
  - [builtin.c](#), [49](#)
  - [builtin.h](#), [57](#)
- [builtin\\_internsymbol](#)
  - [builtin.c](#), [49](#)
  - [builtin.h](#), [57](#)
- [builtin\\_internsymbolascstring](#)
  - [builtin.c](#), [49](#)
  - [builtin.h](#), [57](#)
- [builtin\\_printfunction](#)
  - [builtin.c](#), [49](#)
  - [builtin.h](#), [57](#)
- [builtin\\_random](#)
  - [functions.c](#), [63](#)
- [builtin\\_randomint](#)
  - [functions.c](#), [63](#)
- [builtin\\_system](#)
  - [functions.c](#), [63](#)
- [builtinclassentry](#), [19](#)
- [builtinfunction](#)
  - [builtin.h](#), [55](#)
- [builtinfunctionflags](#)
  - [builtin.h](#), [55](#)
- [callframe](#), [20](#)
- [CHECKCMPTYPE](#)

- vm.c, 211
- cli.c
  - cli\_complete, 136
  - cli\_disassemblewithsrc, 136
  - cli\_help, 136
  - cli\_lex, 137
  - cli\_reporterror, 137
  - cli\_run, 137
  - cli\_tokencolors, 137
- cli.h
  - cli\_run, 138
- cli\_complete
  - cli.c, 136
- cli\_disassemblewithsrc
  - cli.c, 136
- cli\_help
  - cli.c, 136
- cli\_lex
  - cli.c, 137
- cli\_reporterror
  - cli.c, 137
- cli\_run
  - cli.c, 137
  - cli.h, 138
- cli\_tokencolors
  - cli.c, 137
- closed
  - sobjectupvalue, 38
- codeinfo, 20
- common.c
  - morpho\_powerof2ceiling, 169
  - morpho\_printvalue, 170
  - morpho\_strdup, 170
- common.h
  - EQUAL, 171
  - MORPHO\_ISEQUAL, 172
  - MORPHO\_ISSAME, 172
  - morpho\_powerof2ceiling, 172
  - morpho\_printvalue, 172
  - morpho\_strdup, 173
- compile.c
  - compile\_finalize, 197
  - compile\_initialize, 197
  - compiler\_addupvalue, 197
  - compiler\_beginscope, 197
  - compiler\_clear, 197
  - compiler\_closure, 198
  - compiler\_copyglobals, 198
  - compiler\_currentscope, 198
  - compiler\_endscope, 198
  - compiler\_findclass, 198
  - compiler\_fstackclear, 198
  - compiler\_fstackinit, 199
  - compiler\_init, 199
  - compiler\_strip, 199
  - morpho\_compile, 199
  - morpho\_freecompiler, 200
  - morpho\_newcompiler, 200
  - noderules, 200
- compile.h
  - compiler\_clear, 202
  - compiler\_init, 203
  - compiler\_nodefn, 202
  - compilerlist, 202
  - functiontype, 202
- compile\_finalize
  - compile.c, 197
  - core.h, 208
- compile\_initialize
  - compile.c, 197
  - core.h, 208
- compilenoderule, 21
- compiler\_addupvalue
  - compile.c, 197
- compiler\_beginscope
  - compile.c, 197
- compiler\_clear
  - compile.c, 197
  - compile.h, 202
- compiler\_closure
  - compile.c, 198
- compiler\_copyglobals
  - compile.c, 198
- compiler\_currentscope
  - compile.c, 198
- compiler\_endscope
  - compile.c, 198
- compiler\_findclass
  - compile.c, 198
- compiler\_fstackclear
  - compile.c, 198
- compiler\_fstackinit
  - compile.c, 199
- compiler\_init
  - compile.c, 199
  - compile.h, 203
- compiler\_nodefn
  - compile.h, 202
- compiler\_strip
  - compile.c, 199
- compilerlist
  - compile.h, 202
- contents
  - dictionary, 22
- core.h
  - compile\_finalize, 208
  - compile\_initialize, 208
  - DECODE\_A, 205
  - DECODE\_B, 205
  - DECODE\_Bx, 205
  - DECODE\_C, 205
  - DECODE\_F, 205
  - DECODE\_ISBCONSTANT, 205
  - DECODE\_ISCCONSTANT, 206
  - DECODE\_OP, 206
  - DECODE\_sBx, 206

- ENCODE, [206](#)
- ENCODE\_BYTE, [206](#)
- ENCODE\_DOUBLE, [206](#)
- ENCODE\_EMPTYOPERAND, [207](#)
- ENCODE\_LONG, [207](#)
- ENCODE\_LONGFLAGS, [207](#)
- ENCODE\_SINGLE, [207](#)
- ENCODEC, [207](#)
- instruction, [208](#)
- count
  - dictionary, [22](#)
- current
  - lexer, [25](#)
- datastructures/dictionary.h, [64](#)
- datastructures/matrix.c, [71](#)
- datastructures/matrix.h, [75](#)
- datastructures/object.c, [80](#)
- datastructures/object.h, [89](#)
- datastructures/sparse.c, [106](#)
- datastructures/sparse.h, [114](#)
- datastructures/syntaxtree.c, [121](#)
- datastructures/syntaxtree.h, [122](#)
- datastructures/value.c, [124](#)
- datastructures/value.h, [126](#)
- datastructures/varray.c, [130](#)
- datastructures/varray.h, [131](#)
- debuginfo, [21](#)
- DECLARE\_VARRAY
  - varray.h, [132](#)
- DECODE\_A
  - core.h, [205](#)
- DECODE\_B
  - core.h, [205](#)
- DECODE\_Bx
  - core.h, [205](#)
- DECODE\_C
  - core.h, [205](#)
- DECODE\_F
  - core.h, [205](#)
- DECODE\_ISBCONSTANT
  - core.h, [205](#)
- DECODE\_ISCCONSTANT
  - core.h, [206](#)
- DECODE\_OP
  - core.h, [206](#)
- DECODE\_sBx
  - core.h, [206](#)
- DEFINE\_VARRAY
  - sparse.c, [107](#)
- dictionary, [21](#)
  - contents, [22](#)
  - count, [22](#)
- dictionary.h
  - dictionary\_clear, [65](#)
  - dictionary\_freecontents, [66](#)
  - dictionary\_get, [66](#)
  - dictionary\_getintern, [66](#)
  - dictionary\_init, [68](#)
  - dictionary\_insert, [68](#)
  - dictionary\_insertintern, [68](#)
  - dictionary\_intern, [69](#)
  - dictionary\_remove, [69](#)
  - dictionary\_clear
    - dictionary.h, [65](#)
  - dictionary\_freecontents
    - dictionary.h, [66](#)
  - dictionary\_get
    - dictionary.h, [66](#)
  - Dictionary\_getindex
    - builtin.c, [50](#)
  - dictionary\_getintern
    - dictionary.h, [66](#)
  - dictionary\_init
    - dictionary.h, [68](#)
  - dictionary\_insert
    - dictionary.h, [68](#)
  - dictionary\_insertintern
    - dictionary.h, [68](#)
  - dictionary\_intern
    - dictionary.h, [69](#)
  - dictionary\_remove
    - dictionary.h, [69](#)
  - Dictionary\_setindex
    - builtin.c, [50](#)
  - dictionaryentry, [22](#)
- DISASSEMBLE\_OPAB
  - vm.c, [211](#)
- DISASSEMBLE\_OPABC
  - vm.c, [212](#)
- DISASSEMBLE\_OPACb
  - vm.c, [212](#)
- DISASSEMBLE\_OPB
  - vm.c, [212](#)
- DISASSEMBLE\_SHOWA
  - vm.c, [212](#), [213](#)
- ENCODE
  - core.h, [206](#)
- ENCODE\_BYTE
  - core.h, [206](#)
- ENCODE\_DOUBLE
  - core.h, [206](#)
- ENCODE\_EMPTYOPERAND
  - core.h, [207](#)
- ENCODE\_LONG
  - core.h, [207](#)
- ENCODE\_LONGFLAGS
  - core.h, [207](#)
- ENCODE\_SINGLE
  - core.h, [207](#)
- ENCODEC
  - core.h, [207](#)
- EQUAL
  - common.h, [171](#)
- err
  - parser, [32](#)
- error, [23](#)

- error.c
  - error\_clear, 174
  - error\_finalize, 174
  - error\_init, 174
  - error\_initialize, 174
  - morpho\_defineerror, 175
  - morpho\_getdefinitionfromid, 175
  - morpho\_geterrorid, 175
  - morpho\_writeerrorwithid, 175
  - morpho\_writeerrorwithidvalist, 176
- error.h
  - BSD\_EX\_SOFTWARE, 180
  - error\_clear, 181
  - ERROR\_COMPILE, 181
  - ERROR\_EXIT, 181
  - error\_finalize, 182
  - ERROR\_HALT, 181
  - ERROR\_INFO, 181
  - error\_init, 182
  - error\_initialize, 182
  - ERROR\_ISRUNTIMEERROR, 180
  - ERROR\_LEX, 181
  - ERROR\_PARSE, 181
  - ERROR\_POSNUNIDENTIFIABLE, 180
  - ERROR\_SHOULDCONTINUE, 180
  - ERROR\_SUCCEEDED, 181
  - ERROR\_WARNING, 181
  - errorcategory, 181
  - morpho\_defineerror, 182
  - morpho\_geterrorid, 183
  - morpho\_writeerrorwithid, 183
  - morpho\_writeerrorwithidvalist, 183
  - UNREACHABLE, 181
- error\_clear
  - error.c, 174
  - error.h, 181
- ERROR\_COMPILE
  - error.h, 181
- ERROR\_EXIT
  - error.h, 181
- error\_finalize
  - error.c, 174
  - error.h, 182
- ERROR\_HALT
  - error.h, 181
- ERROR\_INFO
  - error.h, 181
- error\_init
  - error.c, 174
  - error.h, 182
- error\_initialize
  - error.c, 174
  - error.h, 182
- ERROR\_ISRUNTIMEERROR
  - error.h, 180
- ERROR\_LEX
  - error.h, 181
- ERROR\_PARSE
  - error.h, 181
- ERROR\_POSNUNIDENTIFIABLE
  - error.h, 180
- ERROR\_SHOULDCONTINUE
  - error.h, 180
- ERROR\_SUCCEEDED
  - error.h, 181
- ERROR\_WARNING
  - error.h, 181
- errorcategory
  - error.h, 181
- errordefinition, 23
- file.c
  - File\_close, 58
  - File\_eof, 58
  - File\_free, 59
  - File\_init, 59
  - File\_lines, 59
  - File\_readchar, 59
  - file\_readintovarray, 59
  - File\_readline, 60
  - file\_readlineintovarray, 60
  - file\_readlineusingvarray, 60
  - File\_write, 60
- file.h
  - file\_readintovarray, 61
  - file\_readlineintovarray, 61
- File\_close
  - file.c, 58
- File\_eof
  - file.c, 58
- File\_free
  - file.c, 59
- File\_init
  - file.c, 59
- File\_lines
  - file.c, 59
- File\_readchar
  - file.c, 59
- file\_readintovarray
  - file.c, 59
  - file.h, 61
- File\_readline
  - file.c, 60
- file\_readlineintovarray
  - file.c, 60
  - file.h, 61
- file\_readlineusingvarray
  - file.c, 60
- File\_write
  - file.c, 60
- frame
  - svm, 40
- functions.c
  - builtin\_clock, 62
  - builtin\_exp, 62
  - builtin\_random, 63
  - builtin\_randomint, 63



- [builtin\\_system](#), [63](#)
- [functionstate](#), [23](#)
- [functiontype](#)
  - [compile.h](#), [202](#)
- [geometry/mesh.c](#), [133](#)
- [geometry/mesh.h](#), [135](#)
- [global](#)
  - [sprogram](#), [39](#)
- [globals](#)
  - [svm](#), [41](#)
- [gray](#)
  - [svm](#), [41](#)
- [graylist](#), [24](#)
- [help.c](#)
  - [help\\_cleartopic](#), [140](#)
  - [help\\_display](#), [140](#)
  - [help\\_finalize](#), [140](#)
  - [help\\_initialize](#), [140](#)
  - [help\\_load](#), [140](#)
  - [help\\_newtopic](#), [141](#)
  - [help\\_parsetag](#), [141](#)
  - [help\\_parsetopiclevel](#), [141](#)
  - [help\\_parsetopicname](#), [141](#)
  - [help\\_querylength](#), [141](#)
  - [help\\_search](#), [142](#)
  - [help\\_searchpath](#), [142](#)
- [help.h](#)
  - [help\\_display](#), [143](#)
  - [help\\_finalize](#), [143](#)
  - [help\\_initialize](#), [143](#)
  - [help\\_querylength](#), [144](#)
  - [help\\_search](#), [144](#)
- [help\\_cleartopic](#)
  - [help.c](#), [140](#)
- [help\\_display](#)
  - [help.c](#), [140](#)
  - [help.h](#), [143](#)
- [help\\_finalize](#)
  - [help.c](#), [140](#)
  - [help.h](#), [143](#)
- [help\\_initialize](#)
  - [help.c](#), [140](#)
  - [help.h](#), [143](#)
- [help\\_load](#)
  - [help.c](#), [140](#)
- [help\\_newtopic](#)
  - [help.c](#), [141](#)
- [help\\_parsetag](#)
  - [help.c](#), [141](#)
- [help\\_parsetopiclevel](#)
  - [help.c](#), [141](#)
- [help\\_parsetopicname](#)
  - [help.c](#), [141](#)
- [help\\_querylength](#)
  - [help.c](#), [141](#)
  - [help.h](#), [144](#)
- [help\\_search](#)
  - [help.c](#), [142](#)
  - [help.h](#), [144](#)
- [help\\_searchpath](#)
  - [help.c](#), [142](#)
- [info](#)
  - [sprogram](#), [39](#)
- [instruction](#)
  - [core.h](#), [208](#)
- [interface/cli.c](#), [135](#)
- [interface/cli.h](#), [138](#)
- [interface/help.c](#), [139](#)
- [interface/help.h](#), [142](#)
- [interface/linedit.c](#), [144](#)
- [interface/linedit.h](#), [158](#)
- [INTERPRET\\_LOOP](#)
  - [vm.c](#), [213](#)
- [iscaptured](#)
  - [registeralloc](#), [34](#)
- [keycodes](#)
  - [linedit.c](#), [147](#)
- [keypress](#), [24](#)
- [keytype](#)
  - [linedit.c](#), [147](#)
- [left](#)
  - [parser](#), [32](#)
- [length](#)
  - [token](#), [42](#)
- [lex](#)
  - [parse.c](#), [187](#)
  - [parse.h](#), [192](#)
- [lex\\_init](#)
  - [parse.c](#), [187](#)
  - [parse.h](#), [193](#)
- [lex\\_recordtoken](#)
  - [parse.c](#), [187](#)
- [lexer](#), [25](#)
  - [current](#), [25](#)
  - [line](#), [25](#)
  - [posn](#), [25](#)
- [line](#)
  - [lexer](#), [25](#)
  - [token](#), [43](#)
- [linedit](#)
  - [linedit.c](#), [148](#)
  - [linedit.h](#), [160](#)
- [linedit.c](#)
  - [keycodes](#), [147](#)
  - [keytype](#), [147](#)
  - [linedit](#), [148](#)
  - [linedit\\_addsuggestion](#), [148](#)
  - [linedit\\_advanceposition](#), [148](#)
  - [linedit\\_advancesuggestions](#), [149](#)
  - [linedit\\_aresuggestionsavailable](#), [149](#)
  - [linedit\\_autocomplete](#), [149](#)
  - [linedit\\_checksupport](#), [149](#)
  - [linedit\\_clear](#), [149](#)

- LINEDIT\_CODESTRINGSIZE, 147
- linedit\_cstrcasecmp, 149
- linedit\_cstring, 150
- linedit\_current suggestion, 150
- LINEDIT\_DEBUGKEYPRESS, 147
- linedit\_disablerawmode, 150
- linedit\_displaywithstyle, 150
- linedit\_displaywithsyntaxcoloring, 151
- linedit\_enablerawmode, 151
- linedit\_generatesuggestions, 151
- linedit\_getmode, 151
- linedit\_historyadd, 151
- linedit\_historyadvance, 152
- linedit\_historyclear, 152
- linedit\_historyselect, 152
- linedit\_init, 152
- linedit\_newstring, 152
- linedit\_noterminal, 152
- linedit\_processkeypress, 153
- linedit\_readkey, 153
- linedit\_refreshline, 153
- linedit\_setmode, 153
- linedit\_setposition, 153
- linedit\_setprompt, 154
- linedit\_showstring, 154
- linedit\_stringaddcharacter, 154
- linedit\_stringaddcstring, 154
- linedit\_stringclear, 154
- linedit\_stringinit, 155
- linedit\_stringinsert, 155
- linedit\_stringlistadd, 155
- linedit\_stringlistclear, 155
- linedit\_stringlistinit, 155
- linedit\_stringlistremove, 155
- linedit\_stringlistselect, 156
- linedit\_stringresize, 156
- linedit\_supported, 156
- linedit\_syntaxcolor, 157
- linedit\_syntaxcolorstring, 157
- LINEDIT\_UNSUPPORTEDBUFFER, 147
- linedit\_atendoffline, 157
- terminit, 157
- linedit.h
  - linedit, 160
  - linedit\_addsuggestion, 161
  - linedit\_autocomplete, 161
  - linedit\_clear, 161
  - linedit\_color, 160
  - linedit\_completer, 159
  - linedit\_displaywithstyle, 161
  - linedit\_displaywithsyntaxcoloring, 162
  - linedit\_init, 162
  - linedit\_setprompt, 162
  - linedit\_string, 159
  - linedit\_syntaxcolor, 163
  - linedit\_tokenizer, 159
  - lineditormode, 160
- linedit\_addsuggestion
  - linedit.c, 148
  - linedit.h, 161
- linedit\_advanceposition
  - linedit.c, 148
- linedit\_advancesuggestions
  - linedit.c, 149
- linedit\_aresuggestionsavailable
  - linedit.c, 149
- linedit\_autocomplete
  - linedit.c, 149
  - linedit.h, 161
- linedit\_checksupport
  - linedit.c, 149
- linedit\_clear
  - linedit.c, 149
  - linedit.h, 161
- LINEDIT\_CODESTRINGSIZE
  - linedit.c, 147
- linedit\_color
  - linedit.h, 160
- linedit\_completer
  - linedit.h, 159
- linedit\_cstrcasecmp
  - linedit.c, 149
- linedit\_cstring
  - linedit.c, 150
- linedit\_current suggestion
  - linedit.c, 150
- LINEDIT\_DEBUGKEYPRESS
  - linedit.c, 147
- linedit\_disablerawmode
  - linedit.c, 150
- linedit\_displaywithstyle
  - linedit.c, 150
  - linedit.h, 161
- linedit\_displaywithsyntaxcoloring
  - linedit.c, 151
  - linedit.h, 162
- linedit\_enablerawmode
  - linedit.c, 151
- linedit\_generatesuggestions
  - linedit.c, 151
- linedit\_getmode
  - linedit.c, 151
- linedit\_historyadd
  - linedit.c, 151
- linedit\_historyadvance
  - linedit.c, 152
- linedit\_historyclear
  - linedit.c, 152
- linedit\_historyselect
  - linedit.c, 152
- linedit\_init
  - linedit.c, 152
  - linedit.h, 162
- linedit\_newstring
  - linedit.c, 152
- linedit\_noterminal

- linedit.c, [152](#)
- linedit\_processkeypress
  - linedit.c, [153](#)
- linedit\_readkey
  - linedit.c, [153](#)
- linedit\_refreshline
  - linedit.c, [153](#)
- linedit\_setmode
  - linedit.c, [153](#)
- linedit\_setposition
  - linedit.c, [153](#)
- linedit\_setprompt
  - linedit.c, [154](#)
  - linedit.h, [162](#)
- linedit\_showstring
  - linedit.c, [154](#)
- linedit\_string
  - linedit.h, [159](#)
- linedit\_stringaddcharacter
  - linedit.c, [154](#)
- linedit\_stringaddcstring
  - linedit.c, [154](#)
- linedit\_stringclear
  - linedit.c, [154](#)
- linedit\_stringinit
  - linedit.c, [155](#)
- linedit\_stringinsert
  - linedit.c, [155](#)
- linedit\_stringlist, [26](#)
- linedit\_stringlistadd
  - linedit.c, [155](#)
- linedit\_stringlistclear
  - linedit.c, [155](#)
- linedit\_stringlistinit
  - linedit.c, [155](#)
- linedit\_stringlistremove
  - linedit.c, [155](#)
- linedit\_stringlistselect
  - linedit.c, [156](#)
- linedit\_stringresize
  - linedit.c, [156](#)
- linedit\_supported
  - linedit.c, [156](#)
- linedit\_syntaxcolor
  - linedit.c, [157](#)
  - linedit.h, [163](#)
- linedit\_syntaxcolordata, [26](#)
- linedit\_syntaxcolorstring
  - linedit.c, [157](#)
- linedit\_token, [27](#)
- linedit\_tokenizer
  - linedit.h, [159](#)
- LINEDIT\_UNSUPPORTEDBUFFER
  - linedit.c, [147](#)
- lineditor, [27](#)
- lineditormode
  - linedit.h, [160](#)
- linedit\_atendoffline
  - linedit.c, [157](#)
- main.c, [163](#)
- matrix.c
  - matrix\_add, [72](#)
  - matrix\_constructor, [72](#)
  - matrix\_divl, [72](#)
  - matrix\_divs, [72](#)
  - matrix\_getarraydimensions, [73](#)
  - matrix\_getarrayelement, [73](#)
  - matrix\_getelement, [73](#)
  - matrix\_mul, [74](#)
  - matrix\_setelement, [74](#)
  - matrix\_sub, [74](#)
  - matrix\_trace, [74](#)
  - matrix\_transpose, [74](#)
  - object\_matrixfromarray, [75](#)
  - object\_newmatrix, [75](#)
- matrix.h
  - matrix\_add, [77](#)
  - matrix\_divl, [77](#)
  - matrix\_divs, [77](#)
  - matrix\_getarraydimensions, [77](#)
  - matrix\_getarrayelement, [79](#)
  - matrix\_getelement, [79](#)
  - MATRIX\_ISSMALL, [76](#)
  - matrix\_mul, [79](#)
  - matrix\_setelement, [79](#)
  - matrix\_sub, [80](#)
  - matrix\_trace, [80](#)
  - matrix\_transpose, [80](#)
  - MORPHO\_LINALG\_USE\_LAPACK, [77](#)
- matrix\_add
  - matrix.c, [72](#)
  - matrix.h, [77](#)
- matrix\_constructor
  - matrix.c, [72](#)
- matrix\_divl
  - matrix.c, [72](#)
  - matrix.h, [77](#)
- matrix\_divs
  - matrix.c, [72](#)
  - matrix.h, [77](#)
- matrix\_getarraydimensions
  - matrix.c, [73](#)
  - matrix.h, [77](#)
- matrix\_getarrayelement
  - matrix.c, [73](#)
  - matrix.h, [79](#)
- matrix\_getelement
  - matrix.c, [73](#)
  - matrix.h, [79](#)
- MATRIX\_ISSMALL
  - matrix.h, [76](#)
- matrix\_mul
  - matrix.c, [74](#)
  - matrix.h, [79](#)
- matrix\_setelement
  - matrix.c, [74](#)

- matrix.h, 79
- matrix\_sub
  - matrix.c, 74
  - matrix.h, 80
- matrix\_trace
  - matrix.c, 74
  - matrix.h, 80
- matrix\_transpose
  - matrix.c, 74
  - matrix.h, 80
- memory.h
  - morpho\_allocate, 185
  - MORPHO\_FREE, 184
  - MORPHO\_MALLOC, 184
  - MORPHO\_REALLOC, 185
- mesh.c
  - mesh\_addelementwithvertices, 133
  - mesh\_checkconnectivity, 134
  - mesh\_constructor, 134
  - mesh\_getconnectivityelement, 134
  - mesh\_load, 134
  - object\_newmesh, 134
- mesh\_addelementwithvertices
  - mesh.c, 133
- mesh\_checkconnectivity
  - mesh.c, 134
- mesh\_constructor
  - mesh.c, 134
- mesh\_getconnectivityelement
  - mesh.c, 134
- mesh\_load
  - mesh.c, 134
- morpho.h, 164
  - morpho\_compile, 165
  - morpho\_defineerror, 165
  - morpho\_disassemble, 166
  - morpho\_finalize, 166
  - morpho\_freecompiler, 166
  - morpho\_freevm, 166
  - morpho\_geterrorid, 166
  - morpho\_initialize, 167
  - morpho\_interpret, 167
  - morpho\_newcompiler, 167
  - morpho\_newvm, 167
  - morpho\_runtimeerror, 167
  - morpho\_stacktrace, 168
  - morpho\_writeerrorwithid, 168
- morpho\_allocate
  - memory.h, 185
- MORPHO\_BEGINCLASS
  - builtin.h, 54
- morpho\_compile
  - compile.c, 199
  - morpho.h, 165
- morpho\_defineerror
  - error.c, 175
  - error.h, 182
  - morpho.h, 165
- morpho\_disassemble
  - morpho.h, 166
  - vm.c, 213
- MORPHO\_ENDCLASS
  - builtin.h, 54
- MORPHO\_EPS
  - build.h, 46
- morpho\_finalize
  - morpho.h, 166
  - vm.c, 214
  - vm.h, 219
- MORPHO\_FREE
  - memory.h, 184
- morpho\_freecompiler
  - compile.c, 200
  - morpho.h, 166
- morpho\_freevm
  - morpho.h, 166
  - vm.c, 214
- MORPHO\_GETARG
  - builtin.h, 54
- MORPHO\_GETARRAY
  - object.h, 92
- MORPHO\_GETBUILTINFUNCTION
  - builtin.h, 54
- MORPHO\_GETCLASS
  - object.h, 92
- MORPHO\_GETCLOSURE
  - object.h, 92
- MORPHO\_GETCLOSUREFUNCTION
  - object.h, 92
- morpho\_getdebugfromindx
  - vm.c, 214
- morpho\_getdefinitionfromid
  - error.c, 175
- MORPHO\_GETDICTIONARY
  - object.h, 92
- MORPHO\_GETDOKKEY
  - object.h, 92
- MORPHO\_GETDOKKEYROW
  - object.h, 93
- morpho\_geterrorid
  - error.c, 175
  - error.h, 183
  - morpho.h, 166
- MORPHO\_GETFUNCTION
  - object.h, 93
- MORPHO\_GETINSTANCE
  - object.h, 93
- MORPHO\_GETINTEGERVALUE
  - value.h, 128
- MORPHO\_GETINVOCATION
  - object.h, 93
- MORPHO\_GETMATRIX
  - object.h, 93
- MORPHO\_GETMESH
  - object.h, 93
- MORPHO\_GETOBJECTHASH

- object.h, 93
- MORPHO\_GETOBJECTTYPE
  - object.h, 94
- MORPHO\_GETRANGE
  - object.h, 94
- MORPHO\_GETSPARSE
  - object.h, 94
- MORPHO\_GETSUPERCLASS
  - object.h, 94
- MORPHO\_GETTYPE
  - value.h, 128
- MORPHO\_GETUPVALUE
  - object.h, 94
- morpho\_initialize
  - morpho.h, 167
  - vm.c, 214
  - vm.h, 219
- MORPHO\_INTEGERTOFloat
  - value.h, 128
- morpho\_interpret
  - morpho.h, 167
  - vm.c, 214
- MORPHO\_ISARRAY
  - object.h, 94
- MORPHO\_ISBUILTINFUNCTION
  - builtin.h, 54
- MORPHO\_ISCLASS
  - object.h, 94
- MORPHO\_ISCLOSURE
  - object.h, 95
- MORPHO\_ISDICTIONARY
  - object.h, 95
- MORPHO\_ISDOKKEY
  - object.h, 95
- MORPHO\_ISEQUAL
  - common.h, 172
- MORPHO\_ISFUNCTION
  - object.h, 95
- MORPHO\_ISINSTANCE
  - object.h, 95
- MORPHO\_ISINVOCATION
  - object.h, 95
- MORPHO\_ISMATRIX
  - object.h, 95
- MORPHO\_ISMESH
  - object.h, 96
- MORPHO\_ISNIL
  - value.h, 128
- MORPHO\_ISRANGE
  - object.h, 96
- MORPHO\_ISSAME
  - common.h, 172
- MORPHO\_ISSPARSE
  - object.h, 96
- MORPHO\_ISSTRING
  - object.h, 96
- MORPHO\_ISUPVALUE
  - object.h, 96
- MORPHO\_LINALG\_USE\_ACCELERATE
  - build.h, 46
- MORPHO\_LINALG\_USE\_CSPARSE
  - build.h, 46
- MORPHO\_LINALG\_USE\_LAPACKE
  - matrix.h, 77
- MORPHO\_MALLOC
  - memory.h, 184
- morpho\_newcompiler
  - compile.c, 200
  - morpho.h, 167
- morpho\_newvm
  - morpho.h, 167
  - vm.c, 215
- MORPHO\_NIL
  - value.h, 128
- morpho\_powerof2ceiling
  - common.c, 169
  - common.h, 172
- morpho\_printvalue
  - common.c, 170
  - common.h, 172
- MORPHO\_RAISE
  - builtin.h, 55
- MORPHO\_RAISEVARGS
  - builtin.h, 55
- MORPHO\_REALLOC
  - memory.h, 185
- morpho\_runtimeerror
  - morpho.h, 167
  - vm.c, 215
- MORPHO\_SELF
  - builtin.h, 55
- MORPHO\_SETOBJECTHASH
  - object.h, 96
- morpho\_stacktrace
  - morpho.h, 168
  - vm.c, 215
- MORPHO\_STATICDOKKEY
  - object.h, 96
- MORPHO\_STATICMATRIX
  - object.h, 97
- MORPHO\_STATICSTRING
  - object.h, 97
- MORPHO\_STATICSTRINGWITHLENGTH
  - object.h, 97
- morpho\_strdup
  - common.c, 170
  - common.h, 173
- morpho\_writeerrorwithid
  - error.c, 175
  - error.h, 183
  - morpho.h, 168
- morpho\_writeerrorwithidvalist
  - error.c, 176
  - error.h, 183
- next
  - sobjectupvalue, 38

- nextgc
  - svm, 41
- nglobals
  - sprogram, 39
- nl
  - parser, 32
- noderules
  - compile.c, 200
- object.c
  - object\_arrayfromlist, 82
  - object\_arrayfromvalueindices, 82
  - object\_arrayfromvarrayvalue, 82
  - object\_arrayindicestoelement, 82
  - object\_arrayinit, 83
  - object\_arrayvaluestoindices, 83
  - object\_concatenatestring, 83
  - object\_dictionary, 84
  - object\_free, 84
  - object\_functionaddprototype, 84
  - object\_functionclear, 84
  - object\_functiongetconstanttable, 84
  - object\_getarrayelement, 85
  - object\_getfunctionname, 85
  - object\_getfunctionparent, 85
  - object\_init, 85
  - object\_new, 85
  - object\_newarray, 86
  - object\_newclosure, 86
  - object\_newdictionary, 86
  - object\_newinvocation, 86
  - object\_newupvalue, 87
  - object\_print, 87
  - object\_printtobuffer, 87
  - object\_setarrayelement, 87
  - object\_size, 87
  - object\_stringfromcstring, 88
  - object\_stringfromvarraychar, 88
  - object\_upvalueinit, 88
- object.h
  - MORPHO\_GETARRAY, 92
  - MORPHO\_GETCLASS, 92
  - MORPHO\_GETCLOSURE, 92
  - MORPHO\_GETCLOSUREFUNCTION, 92
  - MORPHO\_GETDICTIONARY, 92
  - MORPHO\_GETDOKKEY, 92
  - MORPHO\_GETDOKKEYROW, 93
  - MORPHO\_GETFUNCTION, 93
  - MORPHO\_GETINSTANCE, 93
  - MORPHO\_GETINVOCATION, 93
  - MORPHO\_GETMATRIX, 93
  - MORPHO\_GETMESH, 93
  - MORPHO\_GETOBJECTHASH, 93
  - MORPHO\_GETOBJECTTYPE, 94
  - MORPHO\_GETRANGE, 94
  - MORPHO\_GETSPARSE, 94
  - MORPHO\_GETSUPERCLASS, 94
  - MORPHO\_GETUPVALUE, 94
  - MORPHO\_ISARRAY, 94
  - MORPHO\_ISCLASS, 94
  - MORPHO\_ISCLOSURE, 95
  - MORPHO\_ISDICTIONARY, 95
  - MORPHO\_ISDOKKEY, 95
  - MORPHO\_ISFUNCTION, 95
  - MORPHO\_ISINSTANCE, 95
  - MORPHO\_ISINVOCATION, 95
  - MORPHO\_ISMATRIX, 95
  - MORPHO\_ISMESH, 96
  - MORPHO\_ISRANGE, 96
  - MORPHO\_ISSPARSE, 96
  - MORPHO\_ISSTRING, 96
  - MORPHO\_ISUPVALUE, 96
  - MORPHO\_SETOBJECTHASH, 96
  - MORPHO\_STATICDOKKEY, 96
  - MORPHO\_STATICMATRIX, 97
  - MORPHO\_STATICSTRING, 97
  - MORPHO\_STATICSTRINGWITHLENGTH, 97
  - object\_arrayfromlist, 98
  - object\_arrayfromvalueindices, 98
  - object\_arrayfromvarrayvalue, 98
  - object\_arrayindicestoelement, 98
  - object\_arrayvaluestoindices, 99
  - object\_concatenatestring, 99
  - object\_free, 100
  - object\_functionaddprototype, 100
  - object\_functionclear, 100
  - object\_functiongetconstanttable, 100
  - object\_getarrayelement, 100
  - object\_getfunctionname, 101
  - object\_getfunctionparent, 101
  - object\_init, 101
  - object\_matrixfromarray, 101
  - object\_new, 101
  - object\_newarray, 102
  - object\_newclosure, 102
  - object\_newdictionary, 102
  - object\_newinvocation, 103
  - object\_newmatrix, 103
  - object\_newmesh, 103
  - object\_newrange, 103
  - object\_newspare, 103
  - object\_newupvalue, 104
  - object\_print, 104
  - object\_printtobuffer, 104
  - object\_setarrayelement, 104
  - object\_size, 104
  - object\_stringfromcstring, 104
  - object\_stringfromvarraychar, 105
  - object\_upvalueinit, 105
  - objectfunction, 97
  - objecttype, 98
- object\_arrayfromlist
  - object.c, 82
  - object.h, 98
- object\_arrayfromvalueindices
  - object.c, 82
  - object.h, 98

- object\_arrayfromvarrayvalue
  - object.c, [82](#)
  - object.h, [98](#)
- object\_arrayindicestoelement
  - object.c, [82](#)
  - object.h, [98](#)
- object\_arrayinit
  - object.c, [83](#)
- object\_arrayvaluestoindices
  - object.c, [83](#)
  - object.h, [99](#)
- Object\_class
  - builtin.c, [50](#)
- Object\_clone
  - builtin.c, [50](#)
- object\_concatenatestring
  - object.c, [83](#)
  - object.h, [99](#)
- object\_dictionary
  - object.c, [84](#)
- object\_free
  - object.c, [84](#)
  - object.h, [100](#)
- object\_functionaddprototype
  - object.c, [84](#)
  - object.h, [100](#)
- object\_functionclear
  - object.c, [84](#)
  - object.h, [100](#)
- object\_functiongetconstanttable
  - object.c, [84](#)
  - object.h, [100](#)
- object\_getarrayelement
  - object.c, [85](#)
  - object.h, [100](#)
- object\_getfunctionname
  - object.c, [85](#)
  - object.h, [101](#)
- object\_getfunctionparent
  - object.c, [85](#)
  - object.h, [101](#)
- Object\_init
  - builtin.c, [50](#)
- object\_init
  - object.c, [85](#)
  - object.h, [101](#)
- object\_matrixfromarray
  - matrix.c, [75](#)
  - object.h, [101](#)
- object\_new
  - object.c, [85](#)
  - object.h, [101](#)
- object\_newarray
  - object.c, [86](#)
  - object.h, [102](#)
- object\_newclosure
  - object.c, [86](#)
  - object.h, [102](#)
- object\_newdictionary
  - object.c, [86](#)
  - object.h, [102](#)
- object\_newinvocation
  - object.c, [86](#)
  - object.h, [103](#)
- object\_newmatrix
  - matrix.c, [75](#)
  - object.h, [103](#)
- object\_newmesh
  - mesh.c, [134](#)
  - object.h, [103](#)
- object\_newrange
  - builtin.c, [50](#)
  - object.h, [103](#)
- object\_newsparse
  - object.h, [103](#)
  - sparse.c, [107](#)
- object\_newupvalue
  - object.c, [87](#)
  - object.h, [104](#)
- object\_print
  - object.c, [87](#)
  - object.h, [104](#)
- object\_printtobuffer
  - object.c, [87](#)
  - object.h, [104](#)
- Object\_serialize
  - builtin.c, [51](#)
- object\_setarrayelement
  - object.c, [87](#)
  - object.h, [104](#)
- object\_size
  - object.c, [87](#)
  - object.h, [104](#)
- object\_sparsefromarray
  - sparse.c, [107](#)
- object\_stringfromcstring
  - object.c, [88](#)
  - object.h, [104](#)
- object\_stringfromvarraychar
  - object.c, [88](#)
  - object.h, [105](#)
- Object\_super
  - builtin.c, [51](#)
- object\_upvalueinit
  - object.c, [88](#)
  - object.h, [105](#)
- objectarray, [28](#)
- objectbuiltinfunction, [28](#)
- objectclosure, [28](#)
- objectdictionary, [29](#)
- objectdokkey, [29](#)
- objectfunction
  - object.h, [97](#)
- objectinstance, [29](#)
- objectinvocation, [30](#)
- objectmatrix, [30](#)

- objectmesh, [30](#)
- objectrange, [31](#)
- objects
  - svm, [41](#)
- objectsparse, [31](#)
- objectstring, [31](#)
- objecttype
  - object.h, [98](#)
- objectveneer
  - builtin.c, [52](#)
- openupvalues
  - svm, [41](#)
- parse
  - parse.c, [189](#)
  - parse.h, [193](#)
- parse.c
  - lex, [187](#)
  - lex\_init, [187](#)
  - lex\_recordtoken, [187](#)
  - parse, [189](#)
  - parse\_init, [189](#)
  - parse\_stringtovaluearray, [189](#)
  - parse\_synchronize, [190](#)
  - syntaxtree\_addnode, [190](#)
  - UNUSED, [186](#)
- parse.h
  - lex, [192](#)
  - lex\_init, [193](#)
  - parse, [193](#)
  - parse\_init, [193](#)
  - parse\_stringtovaluearray, [194](#)
  - TOKEN\_BLANK, [192](#)
  - tokentype, [192](#)
- parse\_init
  - parse.c, [189](#)
  - parse.h, [193](#)
- parse\_stringtovaluearray
  - parse.c, [189](#)
  - parse.h, [194](#)
- parse\_synchronize
  - parse.c, [190](#)
- parser, [32](#)
  - err, [32](#)
  - left, [32](#)
  - nl, [32](#)
  - previous, [32](#)
  - tree, [33](#)
- parserule, [33](#)
- posn
  - lexer, [25](#)
  - token, [43](#)
- previous
  - parser, [32](#)
- program\_bindobject
  - vm.c, [215](#)
  - vm.h, [220](#)
- program\_getentry
  - vm.c, [215](#)
  - vm.h, [220](#)
- program\_internsymbol
  - vm.c, [216](#)
  - vm.h, [220](#)
- program\_setentry
  - vm.c, [216](#)
  - vm.h, [220](#)
- random.c
  - random\_double, [195](#)
  - random\_initialize, [195](#)
  - random\_int, [195](#)
  - splitmix64\_seed, [195](#)
- random\_double
  - random.c, [195](#)
- random\_initialize
  - random.c, [195](#)
- random\_int
  - random.c, [195](#)
- range\_constructor
  - builtin.c, [51](#)
- range\_count
  - builtin.c, [51](#)
- Range\_enumerate
  - builtin.c, [51](#)
- range\_iterate
  - builtin.c, [51](#)
- reg
  - upvalue, [43](#)
- registeralloc, [33](#)
  - iscaptured, [34](#)
  - scopeddepth, [34](#)
  - symbol, [34](#)
- scompiler, [34](#)
- scompilerlist, [35](#)
- scopeddepth
  - registeralloc, [34](#)
- slinedit\_string, [35](#)
- sobject, [36](#)
- sobjectclass, [36](#)
- sobjectfunction, [37](#)
- sobjecthelptopic, [37](#)
- sobjectupvalue, [37](#)
  - closed, [38](#)
  - next, [38](#)
- sp
  - svm, [41](#)
- sparse.c
  - DEFINE\_VARRAY, [107](#)
  - object\_newsparse, [107](#)
  - object\_sparsefromarray, [107](#)
  - sparse\_add, [108](#)
  - sparse\_checkformat, [108](#)
  - sparse\_clear, [108](#)
  - sparse\_constructor, [108](#)
  - sparse\_div, [109](#)
  - sparse\_getelement, [109](#)
  - Sparse\_getindex, [109](#)



- [sparse\\_mul](#), [110](#)
- [sparse\\_removeformat](#), [110](#)
- [sparse\\_setelement](#), [110](#)
- [Sparse\\_setindex](#), [110](#)
- [sparse\\_size](#), [110](#)
- [sparseccs\\_clear](#), [111](#)
- [sparseccs\\_doktoocs](#), [111](#)
- [sparseccs\\_get](#), [111](#)
- [sparseccs\\_getrowindices](#), [111](#)
- [sparseccs\\_init](#), [112](#)
- [sparseccs\\_print](#), [112](#)
- [sparseccs\\_resize](#), [112](#)
- [sparseccs\\_set](#), [112](#)
- [sparsedok\\_clear](#), [112](#)
- [sparsedok\\_get](#), [113](#)
- [sparsedok\\_init](#), [113](#)
- [sparsedok\\_insert](#), [113](#)
- [sparsedok\\_print](#), [113](#)
- [sparsedok\\_remove](#), [113](#)
- [sparsedok\\_setdimensions](#), [114](#)
- [sparse.h](#)
  - [sparse\\_add](#), [115](#)
  - [sparse\\_clear](#), [116](#)
  - [sparse\\_getelement](#), [116](#)
  - [sparse\\_mul](#), [116](#)
  - [sparse\\_setelement](#), [117](#)
  - [sparse\\_size](#), [117](#)
  - [sparseccs\\_clear](#), [117](#)
  - [sparseccs\\_doktoocs](#), [117](#)
  - [sparseccs\\_get](#), [117](#)
  - [sparseccs\\_getrowindices](#), [117](#)
  - [sparseccs\\_init](#), [119](#)
  - [sparseccs\\_resize](#), [119](#)
  - [sparsedok\\_clear](#), [119](#)
  - [sparsedok\\_get](#), [119](#)
  - [sparsedok\\_init](#), [119](#)
  - [sparsedok\\_insert](#), [120](#)
  - [sparsedok\\_remove](#), [120](#)
  - [sparsedok\\_setdimensions](#), [120](#)
- [sparse\\_add](#)
  - [sparse.c](#), [108](#)
  - [sparse.h](#), [115](#)
- [sparse\\_checkformat](#)
  - [sparse.c](#), [108](#)
- [sparse\\_clear](#)
  - [sparse.c](#), [108](#)
  - [sparse.h](#), [116](#)
- [sparse\\_constructor](#)
  - [sparse.c](#), [108](#)
- [sparse\\_div](#)
  - [sparse.c](#), [109](#)
- [sparse\\_getelement](#)
  - [sparse.c](#), [109](#)
  - [sparse.h](#), [116](#)
- [Sparse\\_getindex](#)
  - [sparse.c](#), [109](#)
- [sparse\\_mul](#)
  - [sparse.c](#), [110](#)
- [sparse.h](#), [116](#)
- [sparse\\_removeformat](#)
  - [sparse.c](#), [110](#)
- [sparse\\_setelement](#)
  - [sparse.c](#), [110](#)
  - [sparse.h](#), [117](#)
- [Sparse\\_setindex](#)
  - [sparse.c](#), [110](#)
- [sparse\\_size](#)
  - [sparse.c](#), [110](#)
  - [sparse.h](#), [117](#)
- [sparseccs](#), [38](#)
- [sparseccs\\_clear](#)
  - [sparse.c](#), [111](#)
  - [sparse.h](#), [117](#)
- [sparseccs\\_doktoocs](#)
  - [sparse.c](#), [111](#)
  - [sparse.h](#), [117](#)
- [sparseccs\\_get](#)
  - [sparse.c](#), [111](#)
  - [sparse.h](#), [117](#)
- [sparseccs\\_getrowindices](#)
  - [sparse.c](#), [111](#)
  - [sparse.h](#), [117](#)
- [sparseccs\\_init](#)
  - [sparse.c](#), [112](#)
  - [sparse.h](#), [119](#)
- [sparseccs\\_print](#)
  - [sparse.c](#), [112](#)
- [sparseccs\\_resize](#)
  - [sparse.c](#), [112](#)
  - [sparse.h](#), [119](#)
- [sparseccs\\_set](#)
  - [sparse.c](#), [112](#)
- [sparsedok](#), [38](#)
- [sparsedok\\_clear](#)
  - [sparse.c](#), [112](#)
  - [sparse.h](#), [119](#)
- [sparsedok\\_get](#)
  - [sparse.c](#), [113](#)
  - [sparse.h](#), [119](#)
- [sparsedok\\_init](#)
  - [sparse.c](#), [113](#)
  - [sparse.h](#), [119](#)
- [sparsedok\\_insert](#)
  - [sparse.c](#), [113](#)
  - [sparse.h](#), [120](#)
- [sparsedok\\_print](#)
  - [sparse.c](#), [113](#)
- [sparsedok\\_remove](#)
  - [sparse.c](#), [113](#)
  - [sparse.h](#), [120](#)
- [sparsedok\\_setdimensions](#)
  - [sparse.c](#), [114](#)
  - [sparse.h](#), [120](#)
- [splitmix64\\_seed](#)
  - [random.c](#), [195](#)
- [sprogram](#), [39](#)

- global, 39
- info, 39
- nglobals, 39
- symboltable, 39
- stack
  - svm, 41
- start
  - token, 43
- String\_length
  - builtin.c, 52
- svm, 40
  - bound, 40
  - frame, 40
  - globals, 41
  - gray, 41
  - nextgc, 41
  - objects, 41
  - openupvalues, 41
  - sp, 41
  - stack, 41
- symbol
  - registeralloc, 34
- symboltable
  - sprogram, 39
- syntaxtree, 42
- syntaxtree.c
  - syntaxtree\_addnode, 121
  - syntaxtree\_clear, 122
  - syntaxtree\_nodefromindx, 122
- syntaxtree.h
  - syntaxtree\_addnode, 123
  - syntaxtree\_clear, 124
  - syntaxtree\_nodefromindx, 124
- syntaxtree\_addnode
  - parse.c, 190
  - syntaxtree.c, 121
  - syntaxtree.h, 123
- syntaxtree\_clear
  - syntaxtree.c, 122
  - syntaxtree.h, 124
- syntaxtree\_nodefromindx
  - syntaxtree.c, 122
  - syntaxtree.h, 124
- terminit
  - linedit.c, 157
- token, 42
  - length, 42
  - line, 43
  - posn, 43
  - start, 43
- TOKEN\_BLANK
  - parse.h, 192
- tokentype
  - parse.h, 192
- tree
  - parser, 33
- UNREACHABLE
  - error.h, 181
- UNUSED
  - parse.c, 186
- upvalue, 43
  - reg, 43
- utils/common.c, 168
- utils/common.h, 170
- utils/error.c, 173
- utils/error.h, 176
- utils/memory.h, 184
- utils/parse.c, 185
- utils/parse.h, 190
- utils/random.c, 194
- value, 44
- value.c
  - value\_promotenumberlist, 125
  - varray\_valuefind, 125
  - varray\_valuefindsame, 126
- value.h
  - MORPHO\_GETINTEGERVALUE, 128
  - MORPHO\_GETTYPE, 128
  - MORPHO\_INTEGERTOFLOAT, 128
  - MORPHO\_ISNIL, 128
  - MORPHO\_NIL, 128
  - value\_promotenumberlist, 129
  - valuetype, 128
  - varray\_valuefind, 129
  - varray\_valuefindsame, 130
- value\_promotenumberlist
  - value.c, 125
  - value.h, 129
- valuetype
  - value.h, 128
- varray.c
  - varray\_powerof2ceiling, 131
- varray.h
  - DECLARE\_VARRAY, 132
  - varray\_powerof2ceiling, 132
- varray\_powerof2ceiling
  - varray.c, 131
  - varray.h, 132
- varray\_valuefind
  - value.c, 125
  - value.h, 129
- varray\_valuefindsame
  - value.c, 126
  - value.h, 130
- vm.c
  - CHECKCMPTYPE, 211
  - DISASSEMBLE\_OPAB, 211
  - DISASSEMBLE\_OPABC, 212
  - DISASSEMBLE\_OPACB, 212
  - DISASSEMBLE\_OPB, 212
  - DISASSEMBLE\_SHOWA, 212, 213
  - INTERPRET\_LOOP, 213
  - morpho\_disassemble, 213
  - morpho\_finalize, 214
  - morpho\_freevm, 214

- morpho\_getdebugfromindx, [214](#)
- morpho\_initialize, [214](#)
- morpho\_interpret, [214](#)
- morpho\_newvm, [215](#)
- morpho\_runtimeerror, [215](#)
- morpho\_stacktrace, [215](#)
- program\_bindobject, [215](#)
- program\_getentry, [215](#)
- program\_internsymbol, [216](#)
- program\_setentry, [216](#)
- vm\_bindobject, [216](#)
- vm\_collectgarbage, [216](#)
- vm\_disassembleinstruction, [216](#)
- vm\_freeobjects, [217](#)
- vm\_gcmarkarray, [217](#)
- vm\_gcmarkdictionary, [217](#)
- vm\_gcmarkobject, [217](#)
- vm\_gcmarkroots, [217](#)
- vm\_gcmarkvalue, [218](#)
- vm\_gcsweep, [218](#)
- vm\_gctrace, [218](#)
- vm\_runtimeerror, [218](#)
- vm.h
  - morpho\_finalize, [219](#)
  - morpho\_initialize, [219](#)
  - program\_bindobject, [220](#)
  - program\_getentry, [220](#)
  - program\_internsymbol, [220](#)
  - program\_setentry, [220](#)
  - vm\_disassembleinstruction, [220](#)
  - vm\_freeobjects, [221](#)
- vm/compile.c, [196](#)
- vm/compile.h, [200](#)
- vm/core.h, [203](#)
- vm/opcodes.h, [209](#)
- vm/vm.c, [209](#)
- vm/vm.h, [219](#)
- vm\_bindobject
  - vm.c, [216](#)
- vm\_collectgarbage
  - vm.c, [216](#)
- vm\_disassembleinstruction
  - vm.c, [216](#)
  - vm.h, [220](#)
- vm\_freeobjects
  - vm.c, [217](#)
  - vm.h, [221](#)
- vm\_gcmarkarray
  - vm.c, [217](#)
- vm\_gcmarkdictionary
  - vm.c, [217](#)
- vm\_gcmarkobject
  - vm.c, [217](#)
- vm\_gcmarkroots
  - vm.c, [217](#)
- vm\_gcmarkvalue
  - vm.c, [218](#)
- vm\_gcsweep
  - vm.c, [218](#)
- vm\_gctrace
  - vm.c, [218](#)
- vm\_runtimeerror
  - vm.c, [218](#)