

Arboreal

Generated by Doxygen 1.8.14

Contents

1	README	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Class Documentation	9
5.1	Addition Class Reference	9
5.1.1	Constructor & Destructor Documentation	9
5.1.1.1	Addition()	9
5.1.1.2	~Addition()	10
5.1.2	Member Function Documentation	10
5.1.2.1	write_out()	10
5.2	arboreal_cli_error Class Reference	10
5.2.1	Constructor & Destructor Documentation	10
5.2.1.1	arboreal_cli_error() [1/4]	11
5.2.1.2	arboreal_cli_error() [2/4]	11
5.2.1.3	arboreal_cli_error() [3/4]	11
5.2.1.4	arboreal_cli_error() [4/4]	11
5.2.1.5	~arboreal_cli_error()	11

5.3	arboreal_daemon_error Class Reference	12
5.3.1	Constructor & Destructor Documentation	12
5.3.1.1	arboreal_daemon_error() [1/4]	12
5.3.1.2	arboreal_daemon_error() [2/4]	12
5.3.1.3	arboreal_daemon_error() [3/4]	13
5.3.1.4	arboreal_daemon_error() [4/4]	13
5.3.1.5	~arboreal_daemon_error()	13
5.4	arboreal_exception Class Reference	13
5.4.1	Constructor & Destructor Documentation	14
5.4.1.1	arboreal_exception() [1/4]	14
5.4.1.2	arboreal_exception() [2/4]	14
5.4.1.3	arboreal_exception() [3/4]	14
5.4.1.4	arboreal_exception() [4/4]	14
5.4.1.5	~arboreal_exception()	14
5.4.2	Member Function Documentation	15
5.4.2.1	ecode()	15
5.4.2.2	where()	15
5.4.3	Member Data Documentation	15
5.4.3.1	_ecode	15
5.4.3.2	_where	15
5.5	arboreal_liaison_error Class Reference	15
5.5.1	Constructor & Destructor Documentation	16
5.5.1.1	arboreal_liaison_error() [1/4]	16
5.5.1.2	arboreal_liaison_error() [2/4]	16
5.5.1.3	arboreal_liaison_error() [3/4]	16
5.5.1.4	arboreal_liaison_error() [4/4]	16
5.5.1.5	~arboreal_liaison_error()	17
5.6	arboreal_logic_error Class Reference	17
5.6.1	Constructor & Destructor Documentation	17
5.6.1.1	arboreal_logic_error() [1/4]	17

5.6.1.2	arboreal_logic_error() [2/4]	18
5.6.1.3	arboreal_logic_error() [3/4]	18
5.6.1.4	arboreal_logic_error() [4/4]	18
5.6.1.5	~arboreal_logic_error()	18
5.7	arboreal_runtime_error Class Reference	18
5.7.1	Constructor & Destructor Documentation	19
5.7.1.1	arboreal_runtime_error() [1/4]	19
5.7.1.2	arboreal_runtime_error() [2/4]	19
5.7.1.3	arboreal_runtime_error() [3/4]	19
5.7.1.4	arboreal_runtime_error() [4/4]	20
5.7.1.5	~arboreal_runtime_error()	20
5.7.2	Member Data Documentation	20
5.7.2.1	_ecode	20
5.7.2.2	_where	20
5.8	Attributes Class Reference	20
5.8.1	Constructor & Destructor Documentation	21
5.8.1.1	Attributes()	21
5.8.2	Member Function Documentation	21
5.8.2.1	del()	21
5.8.2.2	get_access()	22
5.8.2.3	get_creation_time()	22
5.8.2.4	get_edit()	22
5.8.2.5	get_file_attributes()	22
5.8.2.6	get_owner()	22
5.8.2.7	get_permissions()	23
5.8.2.8	get_size()	23
5.8.2.9	read_in()	23
5.8.2.10	set_access()	23
5.8.2.11	set_creation_time()	23
5.8.2.12	set_edit()	23

5.8.2.13	set_owner()	24
5.8.2.14	set_permissions()	24
5.8.2.15	update_size()	24
5.8.2.16	write_out()	24
5.9	CLI Class Reference	24
5.9.1	Constructor & Destructor Documentation	25
5.9.1.1	CLI() [1/4]	25
5.9.1.2	CLI() [2/4]	25
5.9.1.3	CLI() [3/4]	26
5.9.1.4	CLI() [4/4]	26
5.9.1.5	~CLI()	26
5.9.2	Member Function Documentation	26
5.9.2.1	await_response()	26
5.9.2.2	build()	27
5.9.2.3	run() [1/2]	27
5.9.2.4	run() [2/2]	28
5.9.2.5	send_cmnd()	28
5.9.2.6	start()	28
5.10	DebugMessages Class Reference	29
5.10.1	Constructor & Destructor Documentation	29
5.10.1.1	DebugMessages() [1/2]	29
5.10.1.2	DebugMessages() [2/2]	29
5.10.1.3	~DebugMessages()	30
5.10.2	Member Function Documentation	30
5.10.2.1	debug()	30
5.10.2.2	display()	30
5.10.2.3	lock()	30
5.10.2.4	log()	31
5.10.2.5	OFF()	31
5.10.2.6	ON()	31

5.10.2.7	<code>unlock()</code>	31
5.11	Deletion Class Reference	32
5.11.1	Constructor & Destructor Documentation	32
5.11.1.1	<code>Deletion()</code>	32
5.11.1.2	<code>~Deletion()</code>	32
5.11.2	Member Function Documentation	32
5.11.2.1	<code>write_out()</code>	33
5.12	Disk Class Reference	33
5.12.1	Constructor & Destructor Documentation	33
5.12.1.1	<code>Disk()</code>	33
5.12.1.2	<code>~Disk()</code>	34
5.12.2	Member Function Documentation	34
5.12.2.1	<code>getBlockCount()</code>	34
5.12.2.2	<code>getBlockSize()</code>	34
5.12.2.3	<code>readDiskBlock()</code>	34
5.12.2.4	<code>writeDiskBlock()</code>	35
5.13	<code>disk_error</code> Class Reference	35
5.13.1	Constructor & Destructor Documentation	36
5.13.1.1	<code>disk_error()</code> [1/4]	36
5.13.1.2	<code>disk_error()</code> [2/4]	36
5.13.1.3	<code>disk_error()</code> [3/4]	36
5.13.1.4	<code>disk_error()</code> [4/4]	36
5.13.1.5	<code>~disk_error()</code>	36
5.14	DiskManager Class Reference	37
5.14.1	Constructor & Destructor Documentation	37
5.14.1.1	<code>DiskManager()</code>	37
5.14.1.2	<code>~DiskManager()</code>	37
5.14.2	Member Function Documentation	37
5.14.2.1	<code>findPart()</code>	37
5.14.2.2	<code>getBlockSize()</code>	38

5.14.2.3	getPartitionSize()	38
5.14.2.4	readDiskBlock()	38
5.14.2.5	writeDiskBlock()	39
5.15	DiskPartition Struct Reference	39
5.15.1	Member Data Documentation	39
5.15.1.1	fileNameSize	40
5.15.1.2	partitionBlkStart	40
5.15.1.3	partitionName	40
5.15.1.4	partitionSize	40
5.16	File Class Reference	40
5.16.1	Constructor & Destructor Documentation	40
5.16.1.1	File()	40
5.16.2	Member Function Documentation	41
5.16.2.1	get_attributes()	41
5.16.2.2	get_name()	41
5.16.2.3	get_tags()	41
5.16.2.4	read_buff()	41
5.17	file_attributes Struct Reference	42
5.17.1	Member Data Documentation	42
5.17.1.1	creationTime	42
5.17.1.2	lastAccess	42
5.17.1.3	lastEdit	43
5.17.1.4	owner	43
5.17.1.5	permissions	43
5.17.1.6	size	43
5.18	file_error Class Reference	43
5.18.1	Constructor & Destructor Documentation	44
5.18.1.1	file_error() [1/4]	44
5.18.1.2	file_error() [2/4]	44
5.18.1.3	file_error() [3/4]	44

5.18.1.4	file_error() [4 / 4]	44
5.18.1.5	~file_error()	44
5.19	FileInfo Class Reference	45
5.19.1	Constructor & Destructor Documentation	46
5.19.1.1	FileInfo()	46
5.19.1.2	~FileInfo()	46
5.19.2	Member Function Documentation	46
5.19.2.1	add_direct_block()	46
5.19.2.2	add_indirect_block()	47
5.19.2.3	del()	47
5.19.2.4	delete_cont_blocks()	47
5.19.2.5	erase()	48
5.19.2.6	get_attributes()	48
5.19.2.7	get_file_attributes()	48
5.19.2.8	get_file_size()	48
5.19.2.9	get_finode()	49
5.19.2.10	get_tags()	49
5.19.2.11	get_vec_tags()	49
5.19.2.12	insert()	49
5.19.2.13	insert_addition()	50
5.19.2.14	insert_deletion()	50
5.19.2.15	mangle() [1 / 3]	50
5.19.2.16	mangle() [2 / 3]	50
5.19.2.17	mangle() [3 / 3]	51
5.19.2.18	read_in()	51
5.19.2.19	serialize()	52
5.19.2.20	set_access()	52
5.19.2.21	set_edit()	52
5.19.2.22	set_permissions()	52
5.19.2.23	update_file_size()	53

5.19.2.24 write_out()	53
5.20 FileOpen Class Reference	53
5.20.1 Constructor & Destructor Documentation	54
5.20.1.1 FileOpen()	54
5.20.2 Member Function Documentation	54
5.20.2.1 byte_to_index()	54
5.20.2.2 decrement_seek()	54
5.20.2.3 get_EOF()	54
5.20.2.4 get_file()	54
5.20.2.5 get_mode()	54
5.20.2.6 get_seek()	55
5.20.2.7 go_past_last_byte()	55
5.20.2.8 increment_index()	55
5.20.2.9 increment_seek()	55
5.20.2.10 refresh()	55
5.20.2.11 reset_seek()	55
5.20.2.12 set_EOF()	55
5.21 FileSystem Class Reference	56
5.21.1 Constructor & Destructor Documentation	56
5.21.1.1 FileSystem()	56
5.21.1.2 ~FileSystem()	57
5.21.2 Member Function Documentation	57
5.21.2.1 append_file()	57
5.21.2.2 close_file()	57
5.21.2.3 create_file()	58
5.21.2.4 create_tag()	58
5.21.2.5 delete_file() [1/2]	58
5.21.2.6 delete_file() [2/2]	59
5.21.2.7 delete_tag()	59
5.21.2.8 file_search()	59

5.21.2.9	get_attributes()	60
5.21.2.10	get_file_name_size()	60
5.21.2.11	merge_tags()	60
5.21.2.12	num_of_files()	60
5.21.2.13	num_of_tags()	61
5.21.2.14	open_file()	61
5.21.2.15	path_to_file()	61
5.21.2.16	print_files()	61
5.21.2.17	print_root()	62
5.21.2.18	print_tags()	62
5.21.2.19	read_file()	62
5.21.2.20	rename_file()	62
5.21.2.21	rename_tag()	62
5.21.2.22	seek_file_absolute()	63
5.21.2.23	seek_file_relative()	63
5.21.2.24	set_permissions()	63
5.21.2.25	tag_file() [1/2]	64
5.21.2.26	tag_file() [2/2]	64
5.21.2.27	tag_search()	65
5.21.2.28	untag_file() [1/2]	65
5.21.2.29	untag_file() [2/2]	66
5.21.2.30	write_changes()	66
5.21.2.31	write_file()	66
5.22	finode Struct Reference	67
5.22.1	Member Data Documentation	67
5.22.1.1	attributes	67
5.22.1.2	directBlocks	67
5.22.1.3	level1Indirect	67
5.22.1.4	level2Indirect	67
5.22.1.5	level3Indirect	67

5.23	index Struct Reference	68
5.23.1	Member Data Documentation	68
5.23.1.1	blknum	68
5.23.1.2	offset	68
5.24	invalid_arg Class Reference	68
5.24.1	Constructor & Destructor Documentation	69
5.24.1.1	invalid_arg() [1/4]	69
5.24.1.2	invalid_arg() [2/4]	69
5.24.1.3	invalid_arg() [3/4]	69
5.24.1.4	invalid_arg() [4/4]	69
5.24.1.5	~invalid_arg()	70
5.25	Modification Class Reference	70
5.25.1	Constructor & Destructor Documentation	70
5.25.1.1	Modification()	70
5.25.1.2	~Modification()	71
5.25.2	Member Function Documentation	71
5.25.2.1	write_out()	71
5.25.3	Member Data Documentation	71
5.25.3.1	_mod	71
5.25.3.2	_parent	71
5.26	ParseError Class Reference	71
5.26.1	Constructor & Destructor Documentation	71
5.26.1.1	ParseError()	71
5.26.2	Member Function Documentation	72
5.26.2.1	what()	72
5.26.2.2	where()	72
5.27	Parser Class Reference	72
5.27.1	Constructor & Destructor Documentation	73
5.27.1.1	Parser() [1/4]	73
5.27.1.2	Parser() [2/4]	74

5.27.1.3	Parser() [3/4]	74
5.27.1.4	Parser() [4/4]	74
5.27.1.5	~Parser()	74
5.27.2	Member Function Documentation	75
5.27.2.1	get_cwd_tags()	75
5.27.2.2	parse()	75
5.27.2.3	reset() [1/3]	75
5.27.2.4	reset() [2/3]	76
5.27.2.5	reset() [3/3]	76
5.27.2.6	set_cwd()	77
5.27.2.7	set_max_name_size()	77
5.27.2.8	split_on_delim()	78
5.28	PartitionManager Class Reference	78
5.28.1	Constructor & Destructor Documentation	79
5.28.1.1	PartitionManager()	79
5.28.1.2	~PartitionManager()	79
5.28.2	Member Function Documentation	79
5.28.2.1	get_file_name_size()	79
5.28.2.2	getBlockSize()	79
5.28.2.3	getFreeDiskBlock()	80
5.28.2.4	getPartitionName()	80
5.28.2.5	readDiskBlock()	80
5.28.2.6	returnDiskBlock()	80
5.28.2.7	writeDiskBlock()	81
5.29	rootSuperBlock Struct Reference	81
5.29.1	Member Data Documentation	81
5.29.1.1	lastEntry	81
5.29.1.2	size	81
5.29.1.3	startBlock	82
5.30	RootTree Class Reference	82

5.30.1	Constructor & Destructor Documentation	82
5.30.1.1	RootTree()	82
5.30.1.2	~RootTree()	83
5.30.2	Member Function Documentation	83
5.30.2.1	del()	83
5.30.2.2	read_in()	83
5.30.2.3	write_out()	83
5.31	tag_error Class Reference	84
5.31.1	Constructor & Destructor Documentation	84
5.31.1.1	tag_error() [1/4]	84
5.31.1.2	tag_error() [2/4]	84
5.31.1.3	tag_error() [3/4]	85
5.31.1.4	tag_error() [4/4]	85
5.31.1.5	~tag_error()	85
5.32	TagTree Class Reference	85
5.32.1	Constructor & Destructor Documentation	86
5.32.1.1	TagTree()	86
5.32.1.2	~TagTree()	86
5.32.2	Member Function Documentation	86
5.32.2.1	del()	86
5.32.2.2	read_in()	86
5.32.2.3	write_out()	87
5.33	tagTreeSuperBlock Struct Reference	87
5.33.1	Member Data Documentation	87
5.33.1.1	lastEntry	87
5.33.1.2	size	87
5.33.1.3	startBlock	88
5.34	TreeObject Class Reference	88
5.34.1	Constructor & Destructor Documentation	89
5.34.1.1	~TreeObject()	89

5.34.1.2	TreeObject()	89
5.34.2	Member Function Documentation	90
5.34.2.1	add_index()	90
5.34.2.2	begin()	90
5.34.2.3	del()	90
5.34.2.4	delete_cont_blocks()	90
5.34.2.5	end()	91
5.34.2.6	erase()	91
5.34.2.7	find()	91
5.34.2.8	get_block_number()	92
5.34.2.9	get_free_spots()	92
5.34.2.10	get_index()	92
5.34.2.11	get_last_entry()	93
5.34.2.12	get_name()	93
5.34.2.13	get_start_block()	93
5.34.2.14	increment_allocate()	93
5.34.2.15	increment_follow()	94
5.34.2.16	insert()	94
5.34.2.17	insert_addition()	94
5.34.2.18	insert_deletion()	95
5.34.2.19	read_in()	95
5.34.2.20	set_last_entry()	95
5.34.2.21	set_name()	96
5.34.2.22	size()	96
5.34.2.23	write_out()	96
5.34.3	Member Data Documentation	96
5.34.3.1	_blockNumber	96
5.34.3.2	_freeSpots	97
5.34.3.3	_indeces	97
5.34.3.4	_lastEntry	97
5.34.3.5	_modifications	97
5.34.3.6	_myPartitionManager	97
5.34.3.7	_myTree	97
5.34.3.8	_name	97
5.34.3.9	_startBlock	97

6 File Documentation	99
6.1 CommandLineInterface/CLDependancies/cli_helper.hpp File Reference	99
6.1.1 Macro Definition Documentation	99
6.1.1.1 EXCLUSIVE	100
6.1.1.2 INCLUSIVE	100
6.1.1.3 MERGE_1	100
6.1.1.4 MERGE_2	100
6.1.1.5 NEW_AND_TAG	100
6.1.1.6 NEW_AND_TAG_EXC	100
6.1.1.7 OPEN	100
6.1.1.8 TAG_1	100
6.1.1.9 TAG_2	101
6.1.1.10 TAG_3	101
6.1.2 Function Documentation	101
6.1.2.1 bad_clean()	101
6.1.2.2 clean()	101
6.1.2.3 connect_to_server()	101
6.1.2.4 create_shm_seg()	102
6.1.2.5 delete_shm()	102
6.1.2.6 get_cmnd_id()	102
6.1.2.7 receive_from_server()	103
6.1.2.8 send_to_server()	103
6.1.2.9 set_up_socket()	103
6.2 CommandLineInterface/CLHeaders/Cli.h File Reference	104
6.2.1 Variable Documentation	104
6.2.1.1 Debug	105
6.2.1.2 Flag	105
6.2.1.3 MaxBufferSize	105
6.2.1.4 Permissions	106
6.2.1.5 SharedMemorySize	106

6.3	CommandLineInterface/Cli.cpp File Reference	106
6.3.1	Function Documentation	107
6.3.1.1	main()	107
6.4	diskInfo.d File Reference	107
6.5	exthd.d File Reference	107
6.6	Filesystem/daemon.cpp File Reference	107
6.6.1	Macro Definition Documentation	108
6.6.1.1	STARTTUPDATA	108
6.6.2	Function Documentation	108
6.6.2.1	main()	108
6.7	Filesystem/DaemonDependencies/Disk/Disk.cpp File Reference	108
6.8	Filesystem/DaemonDependencies/Disk/Disk.h File Reference	109
6.9	Filesystem/DaemonDependencies/DiskManager/DiskManager.cpp File Reference	109
6.9.1	Function Documentation	109
6.9.1.1	operator==()	109
6.10	Filesystem/DaemonDependencies/DiskManager/DiskManager.h File Reference	109
6.10.1	Function Documentation	110
6.10.1.1	operator==()	110
6.11	Filesystem/DaemonDependencies/File/File.cpp File Reference	110
6.12	Filesystem/DaemonDependencies/File/File.h File Reference	110
6.13	Filesystem/DaemonDependencies/FileSystem/FileSystem.cpp File Reference	110
6.13.1	Variable Documentation	110
6.13.1.1	EncryptionFlag	110
6.14	Filesystem/DaemonDependencies/FileSystem/FileSystem.h File Reference	111
6.15	Filesystem/DaemonDependencies/PartitionManager/PartitionManager.cpp File Reference	111
6.15.1	Variable Documentation	111
6.15.1.1	DEBUG	111
6.16	Filesystem/DaemonDependencies/PartitionManager/PartitionManager.h File Reference	111
6.17	Filesystem/DaemonDependencies/Trees/Trees.cpp File Reference	112
6.17.1	Function Documentation	112

6.17.1.1	operator"!=()	112
6.17.1.2	operator==()	112
6.18	Filesystem/DaemonDependencies/Trees/Trees.h File Reference	112
6.18.1	Macro Definition Documentation	113
6.18.1.1	DEFAULTOWNER	113
6.18.1.2	DEFAULTPERMISSIONS	113
6.19	Filesystem/DaemonDependencies/Types/types.h File Reference	113
6.19.1	Macro Definition Documentation	114
6.19.1.1	MAXopen_fileS	114
6.19.2	Typedef Documentation	114
6.19.2.1	BlkNumType	114
6.19.2.2	FileAttributes	114
6.19.2.3	Finode	114
6.19.2.4	Index	115
6.19.2.5	RootSuperBlock	115
6.19.2.6	TagTreeSuperBlock	115
6.19.3	Variable Documentation	115
6.19.3.1	DEBUG	115
6.20	Filesystem/DaemonHeaders/daemon.h File Reference	115
6.20.1	Macro Definition Documentation	116
6.20.1.1	CREATEFILEDATA	117
6.20.1.2	CREATETAGDATA	117
6.20.1.3	FILESEARCHDATA	117
6.20.1.4	RENAMETAGDATA	117
6.20.1.5	TAGFILEDATA	117
6.20.1.6	TAGSEARCHDATA	117
6.20.2	Function Documentation	117
6.20.2.1	bind_socket()	117
6.20.2.2	command_to_string()	118
6.20.2.3	create_sock()	118

6.20.2.4	execute()	119
6.20.2.5	get_cmnd_id()	119
6.20.2.6	get_file_info() [1/2]	119
6.20.2.7	get_file_info() [2/2]	120
6.20.2.8	get_partition()	120
6.20.2.9	get_set() [1/2]	120
6.20.2.10	get_set() [2/2]	121
6.20.2.11	get_short_file_info()	121
6.20.2.12	is_number()	122
6.20.2.13	listen_on_socket()	122
6.20.2.14	pad_string()	122
6.20.2.15	save_to_disk()	123
6.20.2.16	serialize_fileinfo()	123
6.20.2.17	set_nonblocking()	124
6.20.2.18	set_socket_opt()	124
6.20.2.19	sig_caught()	125
6.20.3	Variable Documentation	125
6.20.3.1	BACKLOG	125
6.20.3.2	current_command_id	125
6.20.3.3	d	125
6.20.3.4	data	125
6.20.3.5	Debug	126
6.20.3.6	dm	126
6.20.3.7	FALSE	126
6.20.3.8	fd_fs_map	126
6.20.3.9	FLAG	126
6.20.3.10	master_set	126
6.20.3.11	MAX_COMMAND_SIZE	126
6.20.3.12	max_fid	126
6.20.3.13	my_fid	127

6.20.3.14	part_fs_map	127
6.20.3.15	path_filedesc_map	127
6.20.3.16	PORT	127
6.20.3.17	TIMEOUT	127
6.20.3.18	TRUE	127
6.20.3.19	verbose	127
6.20.3.20	WILL_TIME	128
6.20.3.21	WRITE_CHANGES_WAIT	128
6.21	Filesystem/driver.cpp File Reference	128
6.21.1	Function Documentation	128
6.21.1.1	main()	128
6.21.2	Variable Documentation	128
6.21.2.1	DEBUG	128
6.22	Filesystem/timing.cpp File Reference	129
6.22.1	Macro Definition Documentation	129
6.22.1.1	CREATEFILEDATA	129
6.22.1.2	CREATETAGDATA	129
6.22.1.3	FILESEARCHDATA	129
6.22.1.4	RENAMETAGDATA	130
6.22.1.5	STARTTUPDATA	130
6.22.1.6	TAGFILEDATA	130
6.22.1.7	TAGSEARCHDATA	130
6.22.2	Function Documentation	130
6.22.2.1	main()	130
6.23	FSFormat/format.cpp File Reference	130
6.23.1	Function Documentation	130
6.23.1.1	main()	131
6.24	LiaisonProcess/liaison.cpp File Reference	131
6.24.1	Function Documentation	131
6.24.1.1	main()	132

6.24.2	Variable Documentation	132
6.24.2.1	Backlog	132
6.24.2.2	DaemonPort	132
6.24.2.3	Debug	132
6.24.2.4	Flag	132
6.24.2.5	MaxBufferSize	132
6.24.2.6	Parser	132
6.24.2.7	Permissions	133
6.24.2.8	SharedMemorySize	133
6.24.2.9	Timeout	133
6.24.2.10	VERBOSE	133
6.25	LiaisonProcess/LiaisonDependancies/liason_helper.hpp File Reference	133
6.25.1	Macro Definition Documentation	134
6.25.1.1	NEW_PLUS	134
6.25.2	Function Documentation	134
6.25.2.1	accept_client()	134
6.25.2.2	bad_clean()	134
6.25.2.3	clean()	135
6.25.2.4	connect_to_daemon()	135
6.25.2.5	create_daemon_sock()	136
6.25.2.6	get_cmnd_id()	136
6.25.2.7	get_command_string()	137
6.25.2.8	get_peername()	137
6.25.2.9	get_shm_seg()	138
6.25.2.10	listen_for_client()	138
6.25.2.11	pad_string()	139
6.25.2.12	recv_msg()	139
6.25.2.13	seg_fault()	140
6.25.2.14	send_response()	140
6.25.2.15	set_up_socket()	141

6.25.2.16 shutdown()	141
6.25.2.17 unat_shm()	142
6.26 README.md File Reference	142
6.27 SharedCPPFiles/Arboreal_Exceptions.cpp File Reference	142
6.28 SharedCPPFiles/Parser.cpp File Reference	143
6.29 SharedHeaders/Arboreal_Exceptions.h File Reference	143
6.30 SharedHeaders/CommandCodes.h File Reference	143
6.30.1 Variable Documentation	144
6.30.1.1 APPND_FCWD	144
6.30.1.2 APPND_FP	145
6.30.1.3 APPND_XFCWDF	145
6.30.1.4 APPND_XFPF	145
6.30.1.5 ATTR_FP	145
6.30.1.6 ATTR_FS	145
6.30.1.7 CD_ABS	145
6.30.1.8 CD_RLP	145
6.30.1.9 CLOSE_F	145
6.30.1.10 CLOSE_FP	146
6.30.1.11 CPY_FCWD	146
6.30.1.12 CPY_FP	146
6.30.1.13 DEL_FP	146
6.30.1.14 DEL_FS	146
6.30.1.15 DEL_TS	146
6.30.1.16 FIND_FS	146
6.30.1.17 FIND_TS	146
6.30.1.18 FTL_ERR	147
6.30.1.19 HANDSHK	147
6.30.1.20 MERG_1_1	147
6.30.1.21 MERG_M_1	147
6.30.1.22 NEW_FP	147

6.30.1.23 NEW_FS	147
6.30.1.24 NEW_TS	147
6.30.1.25 OPEN_F	147
6.30.1.26 OPEN_FP	148
6.30.1.27 QUIT	148
6.30.1.28 READ_FCWD	148
6.30.1.29 READ_FP	148
6.30.1.30 READ_XCWD	148
6.30.1.31 READ_XP	148
6.30.1.32 RNAME_FP	148
6.30.1.33 RNAME_FS	148
6.30.1.34 RNAME_TS	149
6.30.1.35 TAG_FP	149
6.30.1.36 TAG_FS	149
6.30.1.37 UATTR	149
6.30.1.38 UCD	149
6.30.1.39 UCLOSE	149
6.30.1.40 UCOPY	149
6.30.1.41 UDEL	149
6.30.1.42 UFind	150
6.30.1.43 UHELP	150
6.30.1.44 UMERG	150
6.30.1.45 UNEW	150
6.30.1.46 UOPEN	150
6.30.1.47 UQUIT	150
6.30.1.48 UREAD	150
6.30.1.49 URNAME	150
6.30.1.50 UTAG	151
6.30.1.51 UTAG_FP	151
6.30.1.52 UTAG_FS	151

6.30.1.53 UUTAG	151
6.30.1.54 UWRITE	151
6.30.1.55 WRITE_FCWD	151
6.30.1.56 WRITE_FP	151
6.30.1.57 WRITE_XFCWDF	151
6.30.1.58 WRITE_XFPF	152
6.31 SharedHeaders/CommandValidation.h File Reference	152
6.31.1 Function Documentation	153
6.31.1.1 add_tags()	153
6.31.1.2 append_cwd()	154
6.31.1.3 append_path()	154
6.31.1.4 append_x_cwd()	154
6.31.1.5 append_x_path()	154
6.31.1.6 change_dir()	154
6.31.1.7 change_dir_rl()	154
6.31.1.8 check_command()	155
6.31.1.9 check_help()	155
6.31.1.10 check_usage()	155
6.31.1.11 close_file_cd()	155
6.31.1.12 close_files()	155
6.31.1.13 copy_cwd()	155
6.31.1.14 copy_path()	155
6.31.1.15 del_file()	156
6.31.1.16 del_files()	156
6.31.1.17 del_tags()	156
6.31.1.18 find_files()	156
6.31.1.19 find_tags()	156
6.31.1.20 get_attr_cd()	156
6.31.1.21 get_attrs()	156
6.31.1.22 help_1()	157

6.31.1.23 help_10()	157
6.31.1.24 help_11()	157
6.31.1.25 help_12()	157
6.31.1.26 help_13()	157
6.31.1.27 help_14()	157
6.31.1.28 help_15()	157
6.31.1.29 help_16()	158
6.31.1.30 help_2()	158
6.31.1.31 help_3()	158
6.31.1.32 help_4()	158
6.31.1.33 help_5()	158
6.31.1.34 help_6()	158
6.31.1.35 help_7()	158
6.31.1.36 help_8()	159
6.31.1.37 help_9()	159
6.31.1.38 merge_1_1()	159
6.31.1.39 merge_m_1()	159
6.31.1.40 new_file()	159
6.31.1.41 new_files()	159
6.31.1.42 new_tags()	159
6.31.1.43 open_file_cd()	160
6.31.1.44 open_files()	160
6.31.1.45 read_cwd()	160
6.31.1.46 read_path()	160
6.31.1.47 read_x_cwd()	160
6.31.1.48 read_x_path()	160
6.31.1.49 rename_file_cd()	160
6.31.1.50 rename_files()	161
6.31.1.51 rename_tags()	161
6.31.1.52 tag_files()	161

6.31.1.53 <code>untag_file()</code>	161
6.31.1.54 <code>untag_files()</code>	161
6.31.1.55 <code>usage_attr()</code>	161
6.31.1.56 <code>usage_cd()</code>	162
6.31.1.57 <code>usage_close()</code>	162
6.31.1.58 <code>usage_copy()</code>	162
6.31.1.59 <code>usage_delete()</code>	162
6.31.1.60 <code>usage_find()</code>	162
6.31.1.61 <code>usage_help()</code>	162
6.31.1.62 <code>usage_merge()</code>	162
6.31.1.63 <code>usage_new()</code>	163
6.31.1.64 <code>usage_open()</code>	163
6.31.1.65 <code>usage_quit()</code>	163
6.31.1.66 <code>usage_read()</code>	163
6.31.1.67 <code>usage_rename()</code>	163
6.31.1.68 <code>usage_tag()</code>	163
6.31.1.69 <code>usage_untag()</code>	163
6.31.1.70 <code>usage_write()</code>	164
6.31.1.71 <code>write_cwd()</code>	164
6.31.1.72 <code>write_path()</code>	164
6.31.1.73 <code>write_x_cwd()</code>	164
6.31.1.74 <code>write_x_path()</code>	164
6.32 SharedHeaders/DebugMessages.hpp File Reference	164
6.32.1 Function Documentation	165
6.32.1.1 <code>lk()</code>	165
6.32.2 Variable Documentation	165
6.32.2.1 <code>m</code>	165
6.33 SharedHeaders/ErrorsCodes.h File Reference	165
6.34 SharedHeaders/Parser.h File Reference	165
6.34.1 Typedef Documentation	166

6.34.1.1	uint	166
6.35	SharedHeaders/Print.h File Reference	166
6.35.1	Function Documentation	166
6.35.1.1	help()	166
6.35.1.2	print_attr()	167
6.35.1.3	print_cd()	167
6.35.1.4	print_close()	167
6.35.1.5	print_cmnd_lst()	167
6.35.1.6	print_command()	167
6.35.1.7	print_copy()	167
6.35.1.8	print_del()	168
6.35.1.9	print_find()	168
6.35.1.10	print_header()	168
6.35.1.11	print_help()	168
6.35.1.12	print_merge()	168
6.35.1.13	print_new()	168
6.35.1.14	print_open()	168
6.35.1.15	print_quit()	168
6.35.1.16	print_read()	169
6.35.1.17	print_rname()	169
6.35.1.18	print_tag()	169
6.35.1.19	print_utag()	169
6.35.1.20	print_vector()	169
6.35.1.21	print_write()	169
	Index	171

Chapter 1

README

This is a file for specific code notes. things to do, consider, etc, that doesn't need to clutter up the main readme file.

Doing TRY-CATCH

tagSearch() returns a vector of structs with (string "filename", int fidentifier) [fidentifier can be FIONODE blknum or unique file identifier that is mapped to a FIONDE blknum] Hand off storage of file tagSearch() return vector to Danny to be stored in a "current" buffer or smoe such/

There should probably be an attributes object to make our lives easier. and thats what real filesystems do. **Attributes** object should be stored in FINODE or another indirect block who's reference is stored in the FINODE. Which one is used should be decided dynamically, if FIONDE is full get empty data block, store address in FIONDE (migrate data)[optional] to new block, add new data to new block, otherwise add data directly to FIONDE. **TAGS ARE ATTRIBUTES**

I think we may need two open functions. One that takes the unique file id,(block number) and one that takes the vector of tags and the file name . similar to a path. **YES**

I removed validName() because we should check for valid input before passing it to our filesystem. as much as possible anyway.

I think we'll be able to get rid of alot of the helper functions actually. because map will be able to do all that for us. the **big helper functions will be reading in a map and writing out a map.** which i think we can just basically write out all the key, value pairs, because a map can do that easily with its iterator. **for reading in, we'll just read in all the key value pairs and add them to the map one by one. Name Length HARD CAPS at size specified in partition info during formatting NEED TREE INODE READING A MAP FROM DISK TO MAIN MEMORY**

****-----****

so we'll have to have a reserved spot at the end of a block for a block number to the next block of continuing data.

We should write everything out in plaintext and have a converter that can change it to byte stuff that we can implement later. also we should have a flag that will zero out blocks (FOR SPEED), mainly for debugging. but can also repurpose to an encrypt flag later.

//LATER: we should try not to write out the whole tag tree everytime. instead we should only write out the parts that changed if we can. I know this is a tough solution, if a tag is deleted in the middle of the tree and we really have no way of knowing where stuff will be in the tree... but it might be possible to keep some sort of secondary data structure, like a vector with all the info because it doesn't matter what order we reconstruct the map in memory, just that all the data is there. this is also somehting we can implement later. **INtermeidary Data structure will store, (in addition to Memory pointer, block pointer) a tuple (int blknum, int pos_in_blknum) of the key_value pair so we can use it later for delete operations.**

A NOTE about speed: right now, in order to do tag search, we have to read in the finode of each file in the smallest tag tree because I am not storing the number of tags associated with a file in the tag tree inodes. This can be changed later, but for now I just want to get it done. If, when we are testing speeds this is something that will surely improve speed.

****Estimated read in time for everything on startup: $O(n^2 \log(n))$ ****

FileInode structure filename - filenameSize Finode struct = sizeof(finode struct) local tag storage = rest of the space possible tag cont. block = sizeof(blknumType)

Restrictions:

1. filename size restricted to no more than 1/2 block size
2. block size should be a power of 2
3. Hard cap on the number of tags that can be associated with a file. = $((\text{blocksize} - \text{filenameSize} - 136) / \text{sizeof}(\text{BlkNumType})) + (\text{blocksize} / \text{sizeof}(\text{BlkNumType}))$. 103 tags for blocksize of 512. and 64b filename
4. max block size = 16k

TODO:

1. Incorporate storing number of tags associated with file in Tag tree on disk, not yet
2. add renameTag function
3. don't allow duplicate tags to be sent to the filesystem when sending a tagset of any kind

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Attributes	20
CLI	24
DebugMessages	29
Disk	33
DiskManager	37
DiskPartition	39
File	40
file_attributes	42
FileOpen	53
FileSystem	56
finode	67
index	68
Modification	70
Addition	9
Deletion	32
ParseError	71
Parser	72
PartitionManager	78
rootSuperBlock	81
runtime_error	
arboreal_exception	13
arboreal_cli_error	10
arboreal_daemon_error	12
arboreal_liaison_error	15
arboreal_logic_error	17
invalid_arg	68
arboreal_runtime_error	18
disk_error	35
file_error	43
tag_error	84
tagTreeSuperBlock	87
TreeObject	88
FileInfo	45
RootTree	82
TagTree	85

Chapter 3

Class Index

3.1 Class List

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

Addition	9
arboreal_cli_error	10
arboreal_daemon_error	12
arboreal_exception	13
arboreal_liaison_error	15
arboreal_logic_error	17
arboreal_runtime_error	18
Attributes	20
CLI	24
DebugMessages	29
Deletion	32
Disk	33
disk_error	35
DiskManager	37
DiskPartition	39
File	40
file_attributes	42
file_error	43
FileInfo	45
FileOpen	53
FileSystem	56
finode	67
index	68
invalid_arg	68
Modification	70
ParseError	71
Parser	72
PartitionManager	78
rootSuperBlock	81
RootTree	82
tag_error	84
TagTree	85
tagTreeSuperBlock	87
TreeObject	88

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

diskInfo.d	107
exthd.d	107
CommandLineInterface/Cli.cpp	106
CommandLineInterface/CLDependancies/cli_helper.hpp	99
CommandLineInterface/CLHeaders/Cli.h	104
Filesystem/daemon.cpp	107
Filesystem/driver.cpp	128
Filesystem/timing.cpp	129
Filesystem/DaemonDependancies/Disk/Disk.cpp	108
Filesystem/DaemonDependancies/Disk/Disk.h	109
Filesystem/DaemonDependancies/DiskManager/DiskManager.cpp	109
Filesystem/DaemonDependancies/DiskManager/DiskManager.h	109
Filesystem/DaemonDependancies/File/File.cpp	110
Filesystem/DaemonDependancies/File/File.h	110
Filesystem/DaemonDependancies/FileSystem/FileSystem.cpp	110
Filesystem/DaemonDependancies/FileSystem/FileSystem.h	111
Filesystem/DaemonDependancies/PartitionManager/PartitionManager.cpp	111
Filesystem/DaemonDependancies/PartitionManager/PartitionManager.h	111
Filesystem/DaemonDependancies/Trees/Trees.cpp	112
Filesystem/DaemonDependancies/Trees/Trees.h	112
Filesystem/DaemonDependancies/Types/types.h	113
Filesystem/DaemonHeaders/daemon.h	115
FSFormat/format.cpp	130
LiaisonProcess/liaison.cpp	131
LiaisonProcess/LiaisonDependancies/liason_helper.hpp	133
SharedCPPFiles/Arboreal_Exceptions.cpp	142
SharedCPPFiles/Parser.cpp	143
SharedHeaders/Arboreal_Exceptions.h	143
SharedHeaders/CommandCodes.h	143
SharedHeaders/CommandValidation.h	152
SharedHeaders/DebugMessages.hpp	164
SharedHeaders/ErrorCodes.h	165
SharedHeaders/Parser.h	165
SharedHeaders/Print.h	166

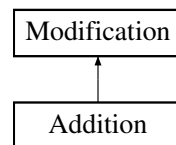
Chapter 5

Class Documentation

5.1 Addition Class Reference

```
#include <Trees.h>
```

Inheritance diagram for Addition:



Public Member Functions

- [Addition](#) ([TreeObject](#) *obj, [TreeObject](#) *parent)
- [~Addition](#) ()
- void [write_out](#) ([PartitionManager](#) *pm)

Additional Inherited Members

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Addition()

```
Addition::Addition (
    TreeObject * obj,
    TreeObject * parent )
```

5.1.1.2 ~Addition()

```
Addition::~~Addition ( )
```

5.1.2 Member Function Documentation

5.1.2.1 write_out()

```
void Addition::write_out (
    PartitionManager * pm ) [virtual]
```

Implements [Modification](#).

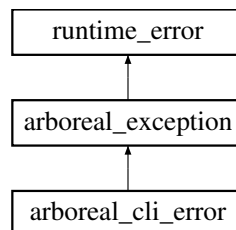
The documentation for this class was generated from the following files:

- [Filesystem/DaemonDependancies/Trees/Trees.h](#)
- [Filesystem/DaemonDependancies/Trees/Trees.cpp](#)

5.2 arboreal_cli_error Class Reference

```
#include <Arboreal_Exceptions.h>
```

Inheritance diagram for `arboreal_cli_error`:



Public Member Functions

- [arboreal_cli_error](#) (const string &[where](#), const string &what, const int [ecode](#)=99)
- [arboreal_cli_error](#) (const char *what, const char *[where](#), const int [ecode](#)=99)
- [arboreal_cli_error](#) (const char *what, const string &[where](#), const int [ecode](#)=99)
- [arboreal_cli_error](#) (const string &what, const char *[where](#), const int [ecode](#)=99)
- [~arboreal_cli_error](#) () throw ()

Additional Inherited Members

5.2.1 Constructor & Destructor Documentation

5.2.1.1 arboreal_cli_error() [1/4]

```
arboreal_cli_error::arboreal_cli_error (
    const string & where,
    const string & what,
    const int ecode = 99 )
```

5.2.1.2 arboreal_cli_error() [2/4]

```
arboreal_cli_error::arboreal_cli_error (
    const char * what,
    const char * where,
    const int ecode = 99 )
```

5.2.1.3 arboreal_cli_error() [3/4]

```
arboreal_cli_error::arboreal_cli_error (
    const char * what,
    const string & where,
    const int ecode = 99 )
```

5.2.1.4 arboreal_cli_error() [4/4]

```
arboreal_cli_error::arboreal_cli_error (
    const string & what,
    const char * where,
    const int ecode = 99 )
```

5.2.1.5 ~arboreal_cli_error()

```
arboreal_cli_error::~~arboreal_cli_error ( ) throw ( )
```

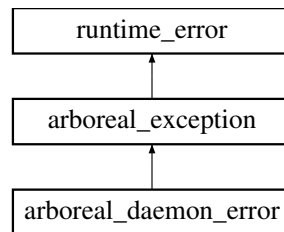
The documentation for this class was generated from the following files:

- SharedHeaders/[Arboreal_Exceptions.h](#)
- SharedCPPFiles/[Arboreal_Exceptions.cpp](#)

5.3 arboreal_daemon_error Class Reference

```
#include <Arboreal_Exceptions.h>
```

Inheritance diagram for `arboreal_daemon_error`:



Public Member Functions

- `arboreal_daemon_error` (const string &*where*, const string &*what*, const int *ecode*=99)
- `arboreal_daemon_error` (const char **what*, const char **where*, const int *ecode*=99)
- `arboreal_daemon_error` (const char **what*, const string &*where*, const int *ecode*=99)
- `arboreal_daemon_error` (const string &*what*, const char **where*, const int *ecode*=99)
- `~arboreal_daemon_error` () throw ()

Additional Inherited Members

5.3.1 Constructor & Destructor Documentation

5.3.1.1 `arboreal_daemon_error()` [1/4]

```
arboreal_daemon_error::arboreal_daemon_error (
    const string & where,
    const string & what,
    const int ecode = 99 )
```

5.3.1.2 `arboreal_daemon_error()` [2/4]

```
arboreal_daemon_error::arboreal_daemon_error (
    const char * what,
    const char * where,
    const int ecode = 99 )
```


5.3.1.3 arboreal_daemon_error() [3/4]

```
arboreal_daemon_error::arboreal_daemon_error (
    const char * what,
    const string & where,
    const int ecode = 99 )
```

5.3.1.4 arboreal_daemon_error() [4/4]

```
arboreal_daemon_error::arboreal_daemon_error (
    const string & what,
    const char * where,
    const int ecode = 99 )
```

5.3.1.5 ~arboreal_daemon_error()

```
arboreal_daemon_error::~arboreal_daemon_error ( ) throw ( )
```

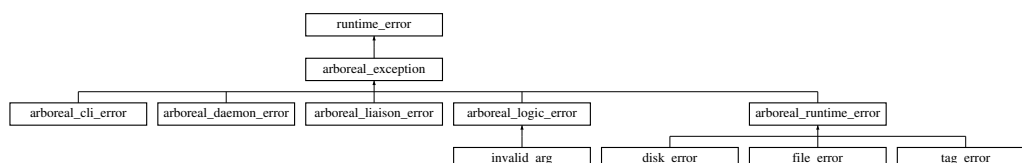
The documentation for this class was generated from the following files:

- SharedHeaders/[Arboreal_Exceptions.h](#)
- SharedCPPFiles/[Arboreal_Exceptions.cpp](#)

5.4 arboreal_exception Class Reference

```
#include <Arboreal_Exceptions.h>
```

Inheritance diagram for arboreal_exception:



Public Member Functions

- [arboreal_exception](#) (const char *what, const char *where, const int ecode=99)
- [arboreal_exception](#) (const char *what, const string &where, const int ecode=99)
- [arboreal_exception](#) (const string &what, const string &where, const int ecode=99)
- [arboreal_exception](#) (const string &what, const char *where, const int ecode=99)
- [~arboreal_exception](#) () throw ()
- virtual const char * where () const
- virtual const int ecode () const

Protected Attributes

- [string _where](#)
- [int _ecode](#)

5.4.1 Constructor & Destructor Documentation

5.4.1.1 `arboreal_exception()` [1/4]

```
arboreal_exception::arboreal_exception (
    const char * what,
    const char * where,
    const int ecode = 99 )
```

5.4.1.2 `arboreal_exception()` [2/4]

```
arboreal_exception::arboreal_exception (
    const char * what,
    const string & where,
    const int ecode = 99 )
```

5.4.1.3 `arboreal_exception()` [3/4]

```
arboreal_exception::arboreal_exception (
    const string & what,
    const string & where,
    const int ecode = 99 )
```

5.4.1.4 `arboreal_exception()` [4/4]

```
arboreal_exception::arboreal_exception (
    const string & what,
    const char * where,
    const int ecode = 99 )
```

5.4.1.5 `~arboreal_exception()`

```
arboreal_exception::~~arboreal_exception ( ) throw ( )
```

5.4.2 Member Function Documentation

5.4.2.1 ecode()

```
const int arboreal_exception::ecode ( ) const [virtual]
```

5.4.2.2 where()

```
const char * arboreal_exception::where ( ) const [virtual]
```

5.4.3 Member Data Documentation

5.4.3.1 _ecode

```
int arboreal_exception::_ecode [protected]
```

5.4.3.2 _where

```
string arboreal_exception::_where [protected]
```

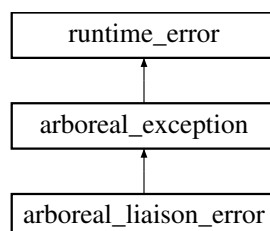
The documentation for this class was generated from the following files:

- SharedHeaders/[Arboreal_Exceptions.h](#)
- SharedCPPFiles/[Arboreal_Exceptions.cpp](#)

5.5 arboreal_liaison_error Class Reference

```
#include <Arboreal_Exceptions.h>
```

Inheritance diagram for arboreal_liaison_error:



Public Member Functions

- [arboreal_liaison_error](#) (const string &[where](#), const string &what, const int [ecode](#)=99)
- [arboreal_liaison_error](#) (const char *what, const char *[where](#), const int [ecode](#)=99)
- [arboreal_liaison_error](#) (const char *what, const string &[where](#), const int [ecode](#)=99)
- [arboreal_liaison_error](#) (const string &what, const char *[where](#), const int [ecode](#)=99)
- [~arboreal_liaison_error](#) () throw ()

Additional Inherited Members

5.5.1 Constructor & Destructor Documentation

5.5.1.1 [arboreal_liaison_error\(\)](#) [1/4]

```
arboreal_liaison_error::arboreal_liaison_error (
    const string & where,
    const string & what,
    const int ecode = 99 )
```

5.5.1.2 [arboreal_liaison_error\(\)](#) [2/4]

```
arboreal_liaison_error::arboreal_liaison_error (
    const char * what,
    const char * where,
    const int ecode = 99 )
```

5.5.1.3 [arboreal_liaison_error\(\)](#) [3/4]

```
arboreal_liaison_error::arboreal_liaison_error (
    const char * what,
    const string & where,
    const int ecode = 99 )
```

5.5.1.4 [arboreal_liaison_error\(\)](#) [4/4]

```
arboreal_liaison_error::arboreal_liaison_error (
    const string & what,
    const char * where,
    const int ecode = 99 )
```

5.5.1.5 ~arboreal_liaison_error()

```
arboreal_liaison_error::~~arboreal_liaison_error ( ) throw ( )
```

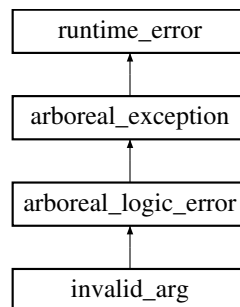
The documentation for this class was generated from the following files:

- SharedHeaders/[Arboreal_Exceptions.h](#)
- SharedCPPFiles/[Arboreal_Exceptions.cpp](#)

5.6 arboreal_logic_error Class Reference

```
#include <Arboreal_Exceptions.h>
```

Inheritance diagram for arboreal_logic_error:



Public Member Functions

- [arboreal_logic_error](#) (const char *what, const char *where, const int ecode=99)
- [arboreal_logic_error](#) (const char *what, const string &where, const int ecode=99)
- [arboreal_logic_error](#) (const string &what, const string &where, const int ecode=99)
- [arboreal_logic_error](#) (const string &what, const char *where, const int ecode=99)
- [~arboreal_logic_error](#) () throw ()

Additional Inherited Members

5.6.1 Constructor & Destructor Documentation

5.6.1.1 arboreal_logic_error() [1/4]

```
arboreal_logic_error::arboreal_logic_error (
    const char * what,
    const char * where,
    const int ecode = 99 )
```

5.6.1.2 arboreal_logic_error() [2/4]

```
arboreal_logic_error::arboreal_logic_error (
    const char * what,
    const string & where,
    const int ecode = 99 )
```

5.6.1.3 arboreal_logic_error() [3/4]

```
arboreal_logic_error::arboreal_logic_error (
    const string & what,
    const string & where,
    const int ecode = 99 )
```

5.6.1.4 arboreal_logic_error() [4/4]

```
arboreal_logic_error::arboreal_logic_error (
    const string & what,
    const char * where,
    const int ecode = 99 )
```

5.6.1.5 ~arboreal_logic_error()

```
arboreal_logic_error::~arboreal_logic_error ( ) throw ( )
```

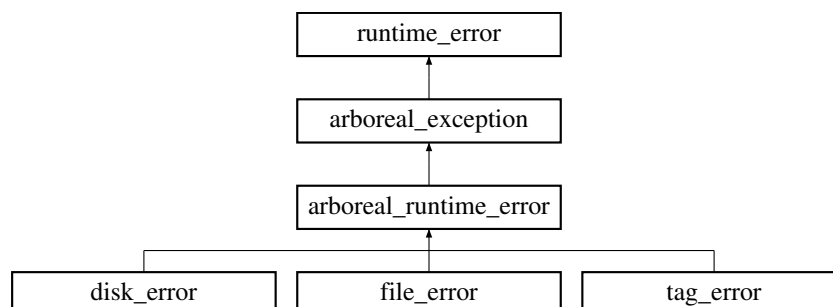
The documentation for this class was generated from the following files:

- [SharedHeaders/Arboreal_Exceptions.h](#)
- [SharedCPPFiles/Arboreal_Exceptions.cpp](#)

5.7 arboreal_runtime_error Class Reference

```
#include <Arboreal_Exceptions.h>
```

Inheritance diagram for arboreal_runtime_error:



Public Member Functions

- [arboreal_runtime_error](#) (const char *what, const char *where, const int [ecode](#)=99)
- [arboreal_runtime_error](#) (const char *what, const string &where, const int [ecode](#)=99)
- [arboreal_runtime_error](#) (const string &what, const string &where, const int [ecode](#)=99)
- [arboreal_runtime_error](#) (const string &what, const char *where, const int [ecode](#)=99)
- [~arboreal_runtime_error](#) () throw ()

Protected Attributes

- string [_where](#)
- int [_ecode](#)

5.7.1 Constructor & Destructor Documentation

5.7.1.1 [arboreal_runtime_error\(\)](#) [1/4]

```
arboreal_runtime_error::arboreal_runtime_error (
    const char * what,
    const char * where,
    const int ecode = 99 )
```

5.7.1.2 [arboreal_runtime_error\(\)](#) [2/4]

```
arboreal_runtime_error::arboreal_runtime_error (
    const char * what,
    const string & where,
    const int ecode = 99 )
```

5.7.1.3 [arboreal_runtime_error\(\)](#) [3/4]

```
arboreal_runtime_error::arboreal_runtime_error (
    const string & what,
    const string & where,
    const int ecode = 99 )
```

5.7.1.4 `arboreal_runtime_error()` [4/4]

```
arboreal_runtime_error::arboreal_runtime_error (
    const string & what,
    const char * where,
    const int ecode = 99 )
```

5.7.1.5 `~arboreal_runtime_error()`

```
arboreal_runtime_error::~~arboreal_runtime_error ( ) throw ( )
```

5.7.2 Member Data Documentation

5.7.2.1 `_ecode`

```
int arboreal_runtime_error::_ecode [protected]
```

5.7.2.2 `_where`

```
string arboreal_runtime_error::_where [protected]
```

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

- SharedHeaders/[Arboreal_Exceptions.h](#)
- SharedCPPFiles/[Arboreal_Exceptions.cpp](#)

5.8 Attributes Class Reference

```
#include <Trees.h>
```


Public Member Functions

- [Attributes](#) ([BlkNumType](#) blknum, [PartitionManager](#) *pm)

Modifier Functions

- void [write_out](#) ()
- void [read_in](#) ()
- void [del](#) ()
- void [set_creation_time](#) ()
- void [set_owner](#) (int owner)
- void [set_permissions](#) (char *perms)
- void [set_access](#) ()
- void [set_edit](#) ()
- void [update_size](#) (size_t size)

Accessor Functions

- time_t [get_creation_time](#) ()
- int [get_owner](#) ()
- char * [get_permissions](#) ()
- time_t [get_access](#) ()
- time_t [get_edit](#) ()
- size_t [get_size](#) ()
- [FileAttributes](#) [get_file_attributes](#) ()

5.8.1 Constructor & Destructor Documentation

5.8.1.1 Attributes()

```
Attributes::Attributes (
    BlkNumType blknum,
    PartitionManager * pm )
```

5.8.2 Member Function Documentation

5.8.2.1 del()

```
void Attributes::del ( )
```

Removes the [Attributes](#) presence on disk

5.8.2.2 `get_access()`

```
time_t Attributes::get_access ( )
```

Returns

the UNIX time the file was last accessed

5.8.2.3 `get_creation_time()`

```
time_t Attributes::get_creation_time ( )
```

Returns

the UNIX time the file was created

5.8.2.4 `get_edit()`

```
time_t Attributes::get_edit ( )
```

Returns

the UNIX time the file was last edited

5.8.2.5 `get_file_attributes()`

```
FileAttributes Attributes::get_file_attributes ( )
```

Returns

the entire FileAttributes struct

5.8.2.6 `get_owner()`

```
int Attributes::get_owner ( )
```

Returns

the UID of the owner of the file

5.8.2.7 `get_permissions()`

```
char * Attributes::get_permissions ( )
```

Returns

the permissions

See also

`FileInfo::get_permissions(char*)`

5.8.2.8 `get_size()`

```
size_t Attributes::get_size ( )
```

Returns

the size of the file in bytes

5.8.2.9 `read_in()`

```
void Attributes::read_in ( )
```

Reads in the [Attributes](#) from disk

5.8.2.10 `set_access()`

```
void Attributes::set_access ( )
```

Marks down the time as accessed time as UNIX timestamp

5.8.2.11 `set_creation_time()`

```
void Attributes::set_creation_time ( )
```

Marks down the creation time of the associated [FileInfo](#) as UNIX timestamp

5.8.2.12 `set_edit()`

```
void Attributes::set_edit ( )
```

Marks down the time as modified time as UNIX timestamp

5.8.2.13 set_owner()

```
void Attributes::set_owner (
    int owner )
```

Marks the owner as their UID

5.8.2.14 set_permissions()

```
void Attributes::set_permissions (
    char * perms )
```

sets the permissions of the file

See also

[FileInfo::set_permissions\(char*\)](#)

5.8.2.15 update_size()

```
void Attributes::update_size (
    size_t size )
```

sets the size to the specified size

5.8.2.16 write_out()

```
void Attributes::write_out ( )
```

Writes out the [Attributes](#) to disk

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

- Filesystem/DaemonDependancies/Trees/[Trees.h](#)
- Filesystem/DaemonDependancies/Trees/[Trees.cpp](#)

5.9 CLI Class Reference

```
#include <Cli.h>
```

Public Member Functions

- [CLI](#) (char **partition)
- [CLI](#) (char **partition, bool debug)
- [CLI](#) (char **partition, char *isScript)
- [CLI](#) (char **partition, char *isScript, bool debug)
- [~CLI](#) ()
- void [start](#) ()
- void [run](#) (std::string input)
- void [run](#) ()
- char * [build](#) (const int id, const std::string input)
- void [send_cmnd](#) (const char *command)
- void [await_response](#) ()

Block while waiting for response from filesystem.

5.9.1 Constructor & Destructor Documentation

5.9.1.1 [CLI\(\)](#) [1/4]

```
CLI::CLI (
    char ** partition )
```

Parameters

<i>partition</i>	A pointer to a character array containing the partition name that this particular command line interface will operate in
------------------	--

Constructor for use in Mode 1 of the Command Line Interface Reads from explicit user input Does NOT print debug data to log

5.9.1.2 [CLI\(\)](#) [2/4]

```
CLI::CLI (
    char ** partition,
    bool debug )
```

Parameters

<i>partition</i>	A pointer to a character array containing the partition name that this particular command line interface will operate in
<i>debug</i>	Whether or not debug messages should be turned on for this interface

Constructor for use in Mode 2 of the Command Line Interface Reads from explicit user input Does PRINTS DEBUG data to log

5.9.1.3 CLI() [3/4]

```
CLI::CLI (
    char ** partition,
    char * isScript )
```

Parameters

<i>partition</i>	A pointer to a character array containing the partition name that this particular command line interface will operate in
<i>isScript</i>	Flag telling whether or not the input for this interface will be coming from a file (The flag value is '-s')

Constructor for use in Mode 3 of the Command Line Interface Reads from file Does NOT print debug data to log

5.9.1.4 CLI() [4/4]

```
CLI::CLI (
    char ** partition,
    char * isScript,
    bool debug )
```

Parameters

<i>partition</i>	A pointer to a character array containing the partition name that this particular command line interface will operate in
<i>debug</i>	Whether or not debug messages should be turned on for this interface
<i>isScript</i>	Flag telling whether or not the input for this interface will be coming from a file (The flag value is '-s')

Constructor for use in Mode 3 of the Command Line Interface Reads from file Does PRINTS DEBUG data to log

5.9.1.5 ~CLI()

```
CLI::~CLI ( )
```

Default Destructor

5.9.2 Member Function Documentation

5.9.2.1 await_response()

```
void CLI::await_response ( )
```

Block while waiting for response from filesystem.

Receive data from the liaison process The data is X number of characters The data can be anything from a list of files returned by the 'find' operation To an error message. This function blocks until it receives data.

The liaison process does all of the outputting to std::out but the Command Line must still wait for the Liaison to output the data before being allowed to request a new command from the user

5.9.2.2 build()

```
char * CLI::build (
    const int id,
    const std::string input )
```

Converts a std::string to a C-Style String, embeds the command id into the C-String, and pads it to length = MaxBuffer↵Size

Parameters

<i>id</i>	File System Command ID
<i>input</i>	File System Command

Returns

A C-Style String of length = MaxBufferSize containing the command ID in the first X Bytes where X is the size of an integer type followed by the command itself followed by as many nullbytes as nescesarry in order to have a length = MaxBufferSize

Format user input for use by Liaison process:

1) Prepend a byte representation of the command ID to the array 2) Copy the user input into the the array (skip the first X indecies were X is the size of an integer (we don't want to overwrite the command ID))

Parameters

<i>id</i>	Comand ID
<i>input</i>	User input string

Returns

A pointer to a charachter array

5.9.2.3 run() [1/2]

```
void CLI::run (
    std::string input )
```

This function operates the same as [run\(\)](#) but takes its input from a filestream rather than a user. Reads in the input data (A [File](#) System Command) and sends it down to the file system.

Some commands that do not need to interact with the [File](#) System code are handled in this function. For example, displaying the 'help' messages is executed from this function since the [File](#) System does not have or need and 'help' command. This function will block until it receives a response from the [File](#) System (provided that the command inputted is intended to go to the [File](#) System) this function will continue reading from the input file until an error occurs or 'end' is read in.

Parameters

<i>input</i>	A std::string value representing a File System command. This value is generally handed to the function by reading an input file. But may also be sent to it from another process such as a UI
--------------	---

5.9.2.4 run() [2/2]

```
void CLI::run ( )
```

Reads in the input data (A [File](#) System Command) and sends it down to the file system.

Some commands that do not need to interact with the [File](#) System code are handled in this function. For example, displaying the 'help' messages is executed from this function since the [File](#) System does not have or need a 'help' command. This function will block until it receives a response from the [File](#) System (provided that the command inputted is intended to go to the [File](#) System) this function will continue reading from user input until an error occurs or the user quits the application.

Reads input from user and sends it to the Liaison Process. Waits for corresponding data from the [File](#) System.

5.9.2.5 send_cmnd()

```
void CLI::send_cmnd (
    const char * cmnd )
```

Sends a command converted to a C-Style String to the Liaison Process for parsing and execution.

Parameters

<i>command</i>	A C-Style String of length = MaxBufferSize containing the command ID in the first X Bytes where X is the size of an integer type followed by the command itself followed by as many nullbytes as necessary in order to have a length = MaxBufferSize
----------------	--

Send user input (A filesystem command) to the Liaison Process

Parameters

<i>cmnd</i>	The input to send
-------------	-------------------

5.9.2.6 start()

```
void CLI::start ( )
```

Performs initial set-up activities such as initiating connections and sending handshakes. Upon the completion of a successful handshake, [run\(\)](#) is called and the interface is ready to use. If the handshake was not successful, the interface notifies the user and quits.

Run initial Command Line Interface setup operations:

1) Generate Shared Memory Segment For Process Synchronization 2) Fork And Run A Liaison Process 3) Create Sockets For Connection To Liaison 4) Send Handshake Command To [File](#) System 5) Run The Command Line

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

- [CommandLineInterface/CLHeaders/Cli.h](#)
- [CommandLineInterface/Cli.cpp](#)

5.10 DebugMessages Class Reference

```
#include <DebugMessages.hpp>
```

Public Member Functions

- [DebugMessages](#) ()
- [DebugMessages](#) (std::string logfile_name)
- [~DebugMessages](#) ()
- void [ON](#) (void)
- void [OFF](#) (void)
- template<typename T >
void [display](#) (const T [data](#), bool force=false)
- template<typename T >
void [log](#) (const T [data](#), bool force=false)
- template<typename T >
void [debug](#) (const T [data](#), bool force=false)
- void [lock](#) ()
- void [unlock](#) ()

5.10.1 Constructor & Destructor Documentation

5.10.1.1 [DebugMessages\(\)](#) [1/2]

```
DebugMessages::DebugMessages ( ) [inline]
```

Create a new DebugMessage object using default logfile name: 'Arboreal.log' Automatically creates the log if it does not exist and if it does exist it will overwrite all the data in the log with the empty string. Sets the debug flag _DEBUG to FALSE on startup.

5.10.1.2 [DebugMessages\(\)](#) [2/2]

```
DebugMessages::DebugMessages (
    std::string logfile_name ) [inline]
```

Create a new DebugMessage object using a user defined logfile name. Automatically creates the log if it does not exist and if it does exist it will overwrite all the data in the log with the empty string. Sets the debug flag _DEBUG to FALSE on startup.

5.10.1.3 ~DebugMessages()

```
DebugMessages::~~DebugMessages ( ) [inline]
```

Default Destructor

5.10.2 Member Function Documentation

5.10.2.1 debug()

```
template<typename T >
void DebugMessages::debug (
    const T data,
    bool force = false ) [inline]
```

Template function for writing debug information to std::cout AND std::fstream.

Parameters

<i>data</i>	The data to be written to std::cout and a file. If the type of data passed is not supported by std::cout or ostream operators, behavior is undefined.
<i>force</i>	If data needs to be written before debugging officially starts this flag should be set to TRUE. Default value is FALSE.

5.10.2.2 display()

```
template<typename T >
void DebugMessages::display (
    const T data,
    bool force = false ) [inline]
```

Template function for writing debug information to std::cout ONLY.

Parameters

<i>data</i>	The data to be written to std::cout. If the type of data passed is not supported by std::cout, behavior is undefined.
<i>force</i>	If data needs to be written before debugging officially starts this flag should be set to TRUE. Default value is FALSE.

5.10.2.3 lock()

```
void DebugMessages::lock ( ) [inline]
```

5.10.2.4 log()

```
template<typename T >
void DebugMessages::log (
    const T data,
    bool force = false ) [inline]
```

Template function for writing debug information to std::fstream ONLY.

Parameters

<i>data</i>	The data to be written to a file. If the type of data passed is not supported by outstream operators, behavior is undefined.
<i>force</i>	If data needs to be written before debugging officially starts this flag should be set to TRUE. Default value is FALSE.

5.10.2.5 OFF()

```
void DebugMessages::OFF (
    void ) [inline]
```

Turns Debugging OFF Sets _DEBUG to FALSE

5.10.2.6 ON()

```
void DebugMessages::ON (
    void ) [inline]
```

Turns Debugging ON Sets _DEBUG to TRUE

5.10.2.7 unlock()

```
void DebugMessages::unlock ( ) [inline]
```

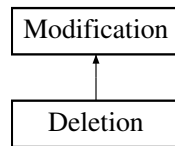
The documentation for this class was generated from the following file:

- SharedHeaders/[DebugMessages.hpp](#)

5.11 Deletion Class Reference

```
#include <Trees.h>
```

Inheritance diagram for Deletion:



Public Member Functions

- [Deletion](#) ([TreeObject](#) *obj, [TreeObject](#) *parent)
- [~Deletion](#) ()
- void [write_out](#) ([PartitionManager](#) *pm)

Additional Inherited Members

5.11.1 Constructor & Destructor Documentation

5.11.1.1 Deletion()

```
Deletion::Deletion (
    TreeObject * obj,
    TreeObject * parent )
```

5.11.1.2 ~Deletion()

```
Deletion::~Deletion ( )
```

5.11.2 Member Function Documentation

5.11.2.1 write_out()

```
void Deletion::write_out (
    PartitionManager * pm ) [virtual]
```

Implements [Modification](#).

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

- Filesystem/DaemonDependancies/Trees/[Trees.h](#)
- Filesystem/DaemonDependancies/Trees/[Trees.cpp](#)

5.12 Disk Class Reference

```
#include <Disk.h>
```

Public Member Functions

- [Disk](#) ([BlkNumType](#) numblocks, size_t blockSize, char *location)
- [~Disk](#) ()

Modifier Functions

- void [writeDiskBlock](#) ([BlkNumType](#) blknum, char *blkdata)

Accessor Functions

- void [readDiskBlock](#) ([BlkNumType](#) blknum, char *blkdata)
- size_t [getBlockSize](#) ()
- int [getBlockCount](#) ()

5.12.1 Constructor & Destructor Documentation

5.12.1.1 Disk()

```
Disk::Disk (
    BlkNumType numblocks,
    size_t blockSize,
    char * location )
```

Parameters

<i>numblocks</i>	the number of blocks on the Disk
<i>blocksize</i>	the block size for Disk blocks
<i>location</i>	the location of the Disk

5.12.1.2 ~Disk()

```
Disk::~~Disk ( )
```

5.12.2 Member Function Documentation

5.12.2.1 getBlockCount()

```
int Disk::getBlockCount ( )
```

Returns

the number of blocks on the entire [Disk](#)

5.12.2.2 getBlockSize()

```
size_t Disk::getBlockSize ( )
```

Returns

the blocksize of the [Disk](#)

5.12.2.3 readDiskBlock()

```
void Disk::readDiskBlock (
    BlkNumType blknum,
    char * blkdata )
```

Reads a block from the [Disk](#).

Parameters

<i>blknum</i>	the blocknumber to be read
<i>blkdata</i>	the buffer to put the read data. must be large enough to contain an entire block of data

See also

PartitionManger::readDiskBlock() PartitionManager::readDiskBlock()

5.12.2.4 writeDiskBlock()

```
void Disk::writeDiskBlock (
    BlkNumType blknum,
    char * blkdata )
```

Writes a block to the [Disk](#).

Parameters

<i>blknum</i>	the blocknumber to be written
<i>blkdata</i>	the buffer to write the data from. It Will write an entire block size of data.

See also

PartitionManger::writeDiskBlock() PartitionManager::writeDiskBlock()

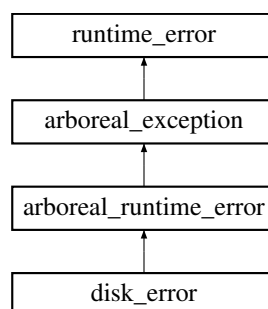
The documentation for this class was generated from the following files:

- Filesystem/DaemonDependancies/Disk/[Disk.h](#)
- Filesystem/DaemonDependancies/Disk/[Disk.cpp](#)

5.13 disk_error Class Reference

```
#include <Arboreal_Exceptions.h>
```

Inheritance diagram for `disk_error`:



Public Member Functions

- [disk_error](#) (const char *what, const char *where, const int ecode=99)
- [disk_error](#) (const char *what, const string &where, const int ecode=99)
- [disk_error](#) (const string &what, const string &where, const int ecode=99)
- [disk_error](#) (const string &what, const char *where, const int ecode=99)
- [~disk_error](#) () throw ()

Additional Inherited Members

5.13.1 Constructor & Destructor Documentation

5.13.1.1 `disk_error()` [1/4]

```
disk_error::disk_error (
    const char * what,
    const char * where,
    const int ecode = 99 )
```

5.13.1.2 `disk_error()` [2/4]

```
disk_error::disk_error (
    const char * what,
    const string & where,
    const int ecode = 99 )
```

5.13.1.3 `disk_error()` [3/4]

```
disk_error::disk_error (
    const string & what,
    const string & where,
    const int ecode = 99 )
```

5.13.1.4 `disk_error()` [4/4]

```
disk_error::disk_error (
    const string & what,
    const char * where,
    const int ecode = 99 )
```

5.13.1.5 `~disk_error()`

```
disk_error::~~disk_error ( ) throw ( )
```

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

- SharedHeaders/[Arboreal_Exceptions.h](#)
- SharedCPPFiles/[Arboreal_Exceptions.cpp](#)

5.14 DiskManager Class Reference

```
#include <DiskManager.h>
```

Public Member Functions

- [DiskManager](#) ([Disk](#) *d)
- [~DiskManager](#) ()

Accessor Functions

- void [readDiskBlock](#) (string partitionName, [BlkNumType](#) blknum, char *blkdata)
- [size_t](#) [getBlockSize](#) ()
- [BlkNumType](#) [getPartitionSize](#) (string partitionName)
- [DiskPartition](#) * [findPart](#) (string partitionName)

Modifier Functions

- void [writeDiskBlock](#) (string partitionName, [BlkNumType](#) blknum, char *blkdata)

5.14.1 Constructor & Destructor Documentation

5.14.1.1 DiskManager()

```
DiskManager::DiskManager (
    Disk * d )
```

Parameters

<i>d</i>	Pointer to the Disk this will manage
----------	--

5.14.1.2 ~DiskManager()

```
DiskManager::~~DiskManager ( )
```

5.14.2 Member Function Documentation

5.14.2.1 findPart()

```
DiskPartition * DiskManager::findPart (
    string partitionName )
```

Parameters

<i>partitionName</i>	the name of the partition
----------------------	---------------------------

Returns

the size of a partition in blocks

5.14.2.2 getBlockSize()

```
size_t DiskManager::getBlockSize ( )
```

Returns

the blocksize of the [Disk](#)

5.14.2.3 getPartitionSize()

```
BlkNumType DiskManager::getPartitionSize (
    string partitionName )
```

Parameters

<i>partitionName</i>	the name of the partition
----------------------	---------------------------

Returns

the size of a partition in blocks

5.14.2.4 readDiskBlock()

```
void DiskManager::readDiskBlock (
    string partitionName,
    BlkNumType blknum,
    char * blkdata )
```

Reads a block from the [Disk](#).

Parameters

<i>partitionName</i>	the name of the partition to write the block to
<i>blknum</i>	the blocknumber to be read
<i>blkdata</i>	the buffer to put the read data. must be large enough to contain an entire block of data

See also

PartitionManger::readDiskBlock() PartitionManager::readDiskBlock()

5.14.2.5 writeDiskBlock()

```
void DiskManager::writeDiskBlock (
    string partitionName,
    BlkNumType blknum,
    char * blkdata )
```

Writes a block to the [Disk](#).

Parameters

<i>partitionName</i>	the name of the partition to write the block to
<i>blknum</i>	the blocknumber to be written
<i>blkdata</i>	the buffer to write the data from. It Will write an entire block size of data.

See also

PartitionManger::writeDiskBlock() PartitionManager::writeDiskBlock()

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

- Filesystem/DaemonDependancies/DiskManager/[DiskManager.h](#)
- Filesystem/DaemonDependancies/DiskManager/[DiskManager.cpp](#)

5.15 DiskPartition Struct Reference

```
#include <DiskManager.h>
```

Public Attributes

- string [partitionName](#)
- [BlkNumType](#) [partitionSize](#)
- [BlkNumType](#) [partitionBlkStart](#)
- int [fileNameSize](#)

5.15.1 Member Data Documentation

5.15.1.1 fileNameSize

```
int DiskPartition::fileNameSize
```

5.15.1.2 partitionBlkStart

```
BlkNumType DiskPartition::partitionBlkStart
```

5.15.1.3 partitionName

```
string DiskPartition::partitionName
```

5.15.1.4 partitionSize

```
BlkNumType DiskPartition::partitionSize
```

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

- Filesystem/DaemonDependancies/DiskManager/[DiskManager.h](#)

5.16 File Class Reference

```
#include <File.h>
```

Public Member Functions

- [File](#) (string name, const vector< string > &tags, [FileAttributes](#) attributes)

Accessor Functions

- string [get_name](#) ()
- vector< string > & [get_tags](#) ()
- [FileAttributes](#) [get_attributes](#) ()

Static Public Member Functions

- static [File](#) * [read_buff](#) (const char *serializedFile)

5.16.1 Constructor & Destructor Documentation

5.16.1.1 File()

```
File::File (  
    string name,  
    const vector< string > & tags,  
    FileAttributes attributes )
```

Parameters

<i>name</i>	the name of the File
<i>tags</i>	the tags to be associated with the File
<i>attributes</i>	the File attributes

5.16.2 Member Function Documentation

5.16.2.1 `get_attributes()`

```
FileAttributes File::get_attributes ( )
```

Returns

the attributes associated with this [File](#)

5.16.2.2 `get_name()`

```
string File::get_name ( )
```

Returns

The name of the [File](#)

5.16.2.3 `get_tags()`

```
vector< string > & File::get_tags ( )
```

Returns

The tags associated with this [File](#)

5.16.2.4 `read_buff()`

```
File * File::read_buff (
    const char * serializedFile ) [static]
```

Will take a `char*` buffer and create a [File](#) object from it. The buffer must have been serialized in the correct format

Parameters

<i>serializedFile</i>	the serializedFile object
-----------------------	---------------------------

Returns

a File* to the created [File](#)

See also

[FileInfo::serialize\(\)](#)

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

- Filesystem/DaemonDependancies/File/[File.h](#)
- Filesystem/DaemonDependancies/File/[File.cpp](#)

5.17 file_attributes Struct Reference

```
#include <types.h>
```

Public Attributes

- time_t [creationTime](#)
- time_t [lastAccess](#)
- time_t [lastEdit](#)
- size_t [size](#)
- char [permissions](#) [12]
- int [owner](#)

5.17.1 Member Data Documentation

5.17.1.1 creationTime

```
time_t file_attributes::creationTime
```

5.17.1.2 lastAccess

```
time_t file_attributes::lastAccess
```

5.17.1.3 lastEdit

```
time_t file_attributes::lastEdit
```

5.17.1.4 owner

```
int file_attributes::owner
```

5.17.1.5 permissions

```
char file_attributes::permissions[12]
```

5.17.1.6 size

```
size_t file_attributes::size
```

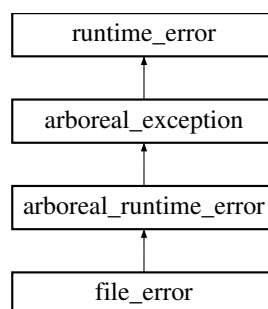
The documentation for this struct was generated from the following file:

- Filesystem/DaemonDependencies/Types/[types.h](#)

5.18 file_error Class Reference

```
#include <Arboreal_Exceptions.h>
```

Inheritance diagram for file_error:



Public Member Functions

- [file_error](#) (const char *what, const char *[where](#), const int [ecode](#)=99)
- [file_error](#) (const char *what, const string &[where](#), const int [ecode](#)=99)
- [file_error](#) (const string &what, const string &[where](#), const int [ecode](#)=99)
- [file_error](#) (const string &what, const char *[where](#), const int [ecode](#)=99)
- [~file_error](#) () throw ()

Additional Inherited Members

5.18.1 Constructor & Destructor Documentation

5.18.1.1 `file_error()` [1/4]

```
file_error::file_error (
    const char * what,
    const char * where,
    const int ecode = 99 )
```

5.18.1.2 `file_error()` [2/4]

```
file_error::file_error (
    const char * what,
    const string & where,
    const int ecode = 99 )
```

5.18.1.3 `file_error()` [3/4]

```
file_error::file_error (
    const string & what,
    const string & where,
    const int ecode = 99 )
```

5.18.1.4 `file_error()` [4/4]

```
file_error::file_error (
    const string & what,
    const char * where,
    const int ecode = 99 )
```

5.18.1.5 `~file_error()`

```
file_error::~file_error ( ) throw ( )
```

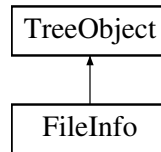
The documentation for this class was generated from the following files:

- SharedHeaders/[Arboreal_Exceptions.h](#)
- SharedCPPFiles/[Arboreal_Exceptions.cpp](#)

5.19 FileInfo Class Reference

```
#include <Trees.h>
```

Inheritance diagram for FileInfo:



Public Member Functions

- [FileInfo](#) (string filename, [BlkNumType](#) blknum, [PartitionManager](#) *pm)
- [~FileInfo](#) ()
- void [write_out](#) ()
- void [read_in](#) (unordered_multimap< string, [FileInfo](#) *> *allFiles, [RootTree](#) *rootTree)
- void [erase](#) (string name)
- void [insert](#) (string name, [TreeObject](#) *ptr)
- void [del](#) ()
- void [delete_cont_blocks](#) ([BlkNumType](#) blknum)
- void [insert_addition](#) ([TreeObject](#) *add)
- void [insert_deletion](#) ([TreeObject](#) *del)

Accessor Functions

- string [mangle](#) ()
mangles the filename with its tags
- string [mangle](#) (vector< string > &tags)
mangles the filename with the specified tags
- string [mangle](#) (unordered_set< string > &tags)
mangles the filename with the specified tags
- [Finode](#) [get_finode](#) ()
- [size_t](#) [get_file_size](#) ()
- [Attributes](#) * [get_attributes](#) ()
- [FileAttributes](#) [get_file_attributes](#) ()
- unordered_set< string > [get_tags](#) ()
- vector< string > [get_vec_tags](#) ()

Modifier Functions

- void [add_direct_block](#) ([BlkNumType](#) blknum, int index)
- void [add_indirect_block](#) ([BlkNumType](#) blknum, short level)
- void [update_file_size](#) ([size_t](#) bytes)
- void [set_access](#) ()
- void [set_edit](#) ()
- void [set_permissions](#) (char *perms)
sets the permississions for this file

Static Public Member Functions

- static string * [serialize](#) ([FileInfo](#) *file)

Additional Inherited Members

5.19.1 Constructor & Destructor Documentation

5.19.1.1 FileInfo()

```
FileInfo::FileInfo (
    string filename,
    BlkNumType blknum,
    PartitionManager * pm )
```

Parameters

<i>filename</i>	Name of the File
<i>blknum</i>	the blocknumber of the associated Finode on disk

5.19.1.2 ~FileInfo()

```
FileInfo::~~FileInfo ( )
```

5.19.2 Member Function Documentation

5.19.2.1 add_direct_block()

```
void FileInfo::add_direct_block (
    BlkNumType blknum,
    int index )
```

adds the specified blocknumber to the array of direct blocks in this file's Finode

Parameters

<i>blknum</i>	the block number of the direct block that has already been allocated
<i>index</i>	the index of the blknum in the array, must be less than 12 and at least 0.

Exceptions

arboreal_logic_error	index out of bounds
--------------------------------------	---------------------

See also

[add_indirect_block](#)

5.19.2.2 add_indirect_block()

```
void FileInfo::add_indirect_block (
    BlkNumType blknum,
    short level )
```

adds the specified blocknumber to the Finode as the start of the specified level of indirect blocks

Parameters

<i>blknum</i>	the block number of the indirect block that has already been allocated
<i>level</i>	the level that the block number is associated with. must be 1, 2 or 3.

Exceptions

arboreal_logic_error	Invalid level
--------------------------------------	---------------

See also

[add_direct_block](#)

5.19.2.3 del()

```
void FileInfo::del ( ) [virtual]
```

Will completely remove the [TreeObject](#)'s presence on disk

Implements [TreeObject](#).

5.19.2.4 delete_cont_blocks()

```
void FileInfo::delete_cont_blocks (
    BlkNumType blknum ) [virtual]
```

Will follow the chain of continuation blocks and free all of them

Parameters

<i>blknum</i>	will free the blknum and use it to follow the chain of continuation blocks
---------------	--

Reimplemented from [TreeObject](#).

5.19.2.5 erase()

```
void FileInfo::erase (
    string name ) [virtual]
```

Disassociate the given name from this object

Parameters

<i>name</i>	the name of the object to be erased.
-------------	--------------------------------------

Exceptions

arboreal_logic_error	
--------------------------------------	--

Reimplemented from [TreeObject](#).

5.19.2.6 get_attributes()

```
Attributes * FileInfo::get_attributes ( )
```

Returns

the [Attributes](#) accociated with this file

5.19.2.7 get_file_attributes()

```
FileAttributes FileInfo::get_file_attributes ( )
```

5.19.2.8 get_file_size()

```
size_t FileInfo::get_file_size ( )
```

Returns

the size of this file in bytes

5.19.2.9 get_finode()

```
Finode FileInfo::get_finode ( )
```

Returns

the Finode associated with this file

5.19.2.10 get_tags()

```
unordered_set< string > FileInfo::get_tags ( )
```

Returns

The tags associated with this file

5.19.2.11 get_vec_tags()

```
vector< string > FileInfo::get_vec_tags ( )
```

5.19.2.12 insert()

```
void FileInfo::insert (
    string name,
    TreeObject * obj ) [virtual]
```

Associate a [TreeObject](#) with this object

Parameters

<i>name</i>	name of the object, mangled if inserting a FileInfo
<i>obj</i>	the object to be inserted

Exceptions

tag_error	
---------------------------	--

See also

[FileInfo::insert\(\)](#)

Reimplemented from [TreeObject](#).

5.19.2.13 insert_addition()

```
void FileInfo::insert_addition (
    TreeObject * add ) [virtual]
```

Do not call on [FileInfo](#)

Reimplemented from [TreeObject](#).

5.19.2.14 insert_deletion()

```
void FileInfo::insert_deletion (
    TreeObject * del ) [virtual]
```

Do not call on [FileInfo](#)

Reimplemented from [TreeObject](#).

5.19.2.15 mangle() [1/3]

```
string FileInfo::mangle ( )
```

mangles the filename with its tags

The name is mangled as follows: Each tag is placed in alphabetical order and appended to the filename using '_' as the separator.

Returns

the mangled name of this file.

See also

[mangle\(vector<string>&\)](#) [mangle\(unordered_set<string>& tags\)](#)

5.19.2.16 mangle() [2/3]

```
string FileInfo::mangle (
    vector< string > & tags )
```

mangles the filename with the specified tags

The name is mangled as follows: Each tag is placed in alphabetical order and appended to the filename using '_' as the separator.

Returns

the mangled name of this file.

Parameters

<i>tags</i>	the tags you wish to mangle the filename with
-------------	---

See also

[mangle\(\)](#) [mangle\(unordered_set<string>& tags\)](#)

5.19.2.17 [mangle\(\)](#) [3/3]

```
string FileInfo::mangle (
    unordered_set< string > & tags )
```

mangles the filename with the specified tags

) The name is mangled as follows: Each tag is placed in alphabetical order and appended to the filename using '_' as the seperator.

Returns

the mangled name of this file.

Parameters

<i>tags</i>	the tags you wish to mangle the filename with
-------------	---

See also

[mangle\(\)](#) [mangle\(unordered_set<string>& tags\)](#)

5.19.2.18 [read_in\(\)](#)

```
void FileInfo::read_in (
    unordered_multimap< string, FileInfo *> * allFiles,
    RootTree * rootTree ) [virtual]
```

Will read in all object data from disk

Parameters

<i>allFiles</i>	a pointer to the map of all files
<i>rootTree</i>	a pointer to the root tree

Implements [TreeObject](#).

5.19.2.19 `serialize()`

```
string * FileInfo::serialize (
    FileInfo * file ) [static]
```

Will serialize a [FileInfo](#) object such that it can be read in as a [File](#) object

Parameters

<i>file</i>	the FileInfo object to be serialized
-------------	--

Returns

The serialized object in string form

See also

[File::read_buff\(\)](#)

5.19.2.20 `set_access()`

```
void FileInfo::set_access ( )
```

marks the file as accessed at the current UNIX time

5.19.2.21 `set_edit()`

```
void FileInfo::set_edit ( )
```

marks the file as edited at the current UNIX time

5.19.2.22 `set_permissions()`

```
void FileInfo::set_permissions (
    char * perms )
```

sets the permissions for this file

The permissions format is as follows. a 1 for true 0 false Byte 0, 1, 2 : reserved, for now Byte 3 - 5 : read write and execute permissions for the user Byte 6 - 8 : read write and execute permissions for the group Byte 9 - 11 : read write and execute permissions for the world Currently there is no differentiation between user group and world

Parameters

<i>perms</i>	the permisssons in the correct format
--------------	---------------------------------------

5.19.2.23 update_file_size()

```
void FileInfo::update_file_size (
    size_t bytes )
```

Sets the file size to the specified bytes. Only the filesystem should call.

Parameters

<i>bytes</i>	the new file size
--------------	-------------------

5.19.2.24 write_out()

```
void FileInfo::write_out ( ) [virtual]
```

Intended to write out the object to disk

Implements [TreeObject](#).

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

- Filesystem/DaemonDependancies/Trees/[Trees.h](#)
- Filesystem/DaemonDependancies/Trees/[Trees.cpp](#)

5.20 FileOpen Class Reference

```
#include <FileSystem.h>
```

Public Member Functions

- [FileOpen](#) ([FileInfo](#) *file, char mode, [PartitionManager](#) *pm)
- [FileInfo](#) * [get_file](#) ()
- [size_t](#) [get_seek](#) ()
- char [get_mode](#) ()
- void [increment_seek](#) ([size_t](#) bytes, bool write=false)
- void [decrement_seek](#) ([size_t](#) bytes)
- [Index](#) [byte_to_index](#) (short offset)
- [Index](#) [increment_index](#) ()
- void [set_EOF](#) ()
- void [reset_seek](#) ()
- bool [get_EOF](#) ()
- void [go_past_last_byte](#) ()
- void [refresh](#) ()

5.20.1 Constructor & Destructor Documentation

5.20.1.1 FileOpen()

```
FileOpen::FileOpen (
    FileInfo * file,
    char mode,
    PartitionManager * pm )
```

5.20.2 Member Function Documentation

5.20.2.1 byte_to_index()

```
Index FileOpen::byte_to_index (
    short offset )
```

5.20.2.2 decrement_seek()

```
void FileOpen::decrement_seek (
    size_t bytes )
```

5.20.2.3 get_EOF()

```
bool FileOpen::get_EOF ( )
```

5.20.2.4 get_file()

```
FileInfo * FileOpen::get_file ( )
```

5.20.2.5 get_mode()

```
char FileOpen::get_mode ( )
```

5.20.2.6 get_seek()

```
size_t FileOpen::get_seek ( )
```

5.20.2.7 go_past_last_byte()

```
void FileOpen::go_past_last_byte ( )
```

5.20.2.8 increment_index()

```
Index FileOpen::increment_index ( )
```

5.20.2.9 increment_seek()

```
void FileOpen::increment_seek (
    size_t bytes,
    bool write = false )
```

5.20.2.10 refresh()

```
void FileOpen::refresh ( )
```

5.20.2.11 reset_seek()

```
void FileOpen::reset_seek ( )
```

5.20.2.12 set_EOF()

```
void FileOpen::set_EOF ( )
```

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

- Filesystem/DaemonDependancies/FileSystem/[FileSystem.h](#)
- Filesystem/DaemonDependancies/FileSystem/[FileSystem.cpp](#)

5.21 FileSystem Class Reference

```
#include <FileSystem.h>
```

Public Member Functions

- [FileSystem](#) ([DiskManager](#) *dm, string partitionName)
- [~FileSystem](#) ()
- void [write_changes](#) ()
- [FileInfo](#) * [path_to_file](#) (vector< string > &path)
- int [get_file_name_size](#) ()
- size_t [num_of_files](#) ()
- size_t [num_of_tags](#) ()

Tag Operations

- vector< [FileInfo](#) * > * [tag_search](#) (unordered_set< string > &tags)
- void [create_tag](#) (string tagName)
- void [delete_tag](#) (string tagName)
- void [merge_tags](#) (string tag1, string tag2)
- void [tag_file](#) ([FileInfo](#) *file, unordered_set< string > tagsToAdd)
- void [tag_file](#) (vector< string > &filePath, unordered_set< string > tags)
- void [untag_file](#) ([FileInfo](#) *file, unordered_set< string > tagsToRemove, bool deleting=false)
- void [untag_file](#) (vector< string > &filePath, unordered_set< string > tags)
- void [rename_tag](#) (string originalTagName, string newTagName)

File Operations

- vector< [FileInfo](#) * > * [file_search](#) (string name)
- [FileInfo](#) * [create_file](#) (string filename, unordered_set< string > &tags)
- int [open_file](#) (vector< string > &filePath, char mode)
- void [close_file](#) (unsigned int fileDesc)
- size_t [read_file](#) (unsigned int fileDesc, char *data, size_t len)
- size_t [write_file](#) (unsigned int fileDesc, const char *data, size_t len)
- size_t [append_file](#) (unsigned int fileDesc, const char *data, size_t len)
- void [seek_file_absolute](#) (unsigned int fileDesc, size_t offset)
- void [seek_file_relative](#) (unsigned int fileDesc, long int offset)
- void [rename_file](#) (vector< string > &originalFilePath, string newFileName)
- [Attributes](#) * [get_attributes](#) (vector< string > &filePath)
- void [set_permissions](#) (vector< string > &filePath, char *perms)
- void [delete_file](#) ([FileInfo](#) *file)
- void [delete_file](#) (vector< string > &filePath)

Debug Functions

- void [print_root](#) ()
- void [print_tags](#) ()
- void [print_files](#) ()

5.21.1 Constructor & Destructor Documentation

5.21.1.1 FileSystem()

```
FileSystem::FileSystem (
    DiskManager * dm,
    string partitionName )
```

Parameters

<i>dm</i>	the Disk manager for the disk that this Filesystem will be accessing
<i>partitionName</i>	the name of the partition that this FileSystem will be associated with

5.21.1.2 ~FileSystem()

```
FileSystem::~FileSystem ( )
```

5.21.2 Member Function Documentation

5.21.2.1 append_file()

```
size_t FileSystem::append_file (
    unsigned int fileDesc,
    const char * data,
    size_t len )
```

Will Append a number of bytes to an open file. The file must have been opened with write permissions.

Parameters

<i>fileDesc</i>	the file descriptor returned from open_file
<i>data</i>	a buffer to be read from to write to the file. must be at least of len size
<i>len</i>	the number of bytes to write from data.

5.21.2.2 close_file()

```
void FileSystem::close_file (
    unsigned int fileDesc )
```

Will close a file. the [File](#) must have been opened.

Parameters

<i>fileDesc</i>	the file descriptor returned from open_file
-----------------	---

5.21.2.3 create_file()

```
FileInfo * FileSystem::create_file (
    string filename,
    unordered_set< string > & tags )
```

Will create a new file with the specified name and tags. The new file must not already exist.

Parameters

<i>filename</i>	the name of the new file
<i>tags</i>	the tag set to tag the file with. If empty, will be tagged with default.

Returns

a [FileInfo](#) to the created file, in case the calling code needs it

5.21.2.4 create_tag()

```
void FileSystem::create_tag (
    string tagName )
```

Will create a new tag if that tag name does not already exist

Parameters

<i>tagName</i>	the name of the Tag to create
----------------	-------------------------------

5.21.2.5 delete_file() [1/2]

```
void FileSystem::delete_file (
    FileInfo * file )
```

Delete a particular file. The file must exist.

Parameters

<i>file</i>	the FileInfo object to be deleted.
-------------	--

See also

[delete_file\(vector<string>&\)](#)

5.21.2.6 delete_file() [2/2]

```
void FileSystem::delete_file (
    vector< string > & filePath )
```

Delete a particular file. The file must exist.

Parameters

<i>filePath</i>	the full path to the file to you wish to delete
-----------------	---

See also

[delete_file\(FileInfo*\)](#)

5.21.2.7 delete_tag()

```
void FileSystem::delete_tag (
    string tagName )
```

Will delete the specified tag only if it has no files associated with it(it is empty) and it does in fact exist.

Parameters

<i>tagName</i>	the name of the tag to be deleted
----------------	-----------------------------------

5.21.2.8 file_search()

```
vector< FileInfo * > * FileSystem::file_search (
    string name )
```

Will search for a specified file name. Searches the entire [FileSystem](#)

Parameters

<i>name</i>	the name of the file to search for.
-------------	-------------------------------------

Returns

a pointer to a vector of [FileInfo](#) objects that have the specified name. This should be freed by the calling code

5.21.2.9 `get_attributes()`

```
Attributes * FileSystem::get_attributes (
    vector< string > & filePath )
```

Will search for a file and return its [Attributes](#)

Parameters

<i>filePath</i>	the full path to the file to you wish to get the Attributes of
-----------------	--

Returns

the [Attributes](#) object associated with a particular file.

5.21.2.10 `get_file_name_size()`

```
int FileSystem::get_file_name_size ( )
```

Returns

the Maximum file name size for the partition associated with this [FileSystem](#) object

5.21.2.11 `merge_tags()`

```
void FileSystem::merge_tags (
    string tag1,
    string tag2 )
```

TODO: description and Function

Parameters

<i>tag1</i>	
<i>tag</i>	2

5.21.2.12 `num_of_files()`

```
size_t FileSystem::num_of_files ( )
```


5.21.2.13 num_of_tags()

```
size_t FileSystem::num_of_tags ( )
```

5.21.2.14 open_file()

```
int FileSystem::open_file (
    vector< string > & filePath,
    char mode )
```

Will open a file. The file must exist. There is a cap on the Maximum number of open files. You can open the same file as many times as you want.

Parameters

<i>filePath</i>	the full path including the file name as the last entry
<i>mode</i>	the mode to open the file with. r, w, or x. x is read and write ability.

Returns

a file descriptor that can later be used to reference the opened file

5.21.2.15 path_to_file()

```
FileInfo * FileSystem::path_to_file (
    vector< string > & path )
```

Will find a [FileInfo](#) object if it exists, given the full path

Parameters

<i>path</i>	The full path to the file. The filename must be the last entry in the vector. an file name with no path is considered to be in the default path
-------------	---

Returns

the found [FileInfo](#) object

5.21.2.16 print_files()

```
void FileSystem::print_files ( )
```

Print out all files and their blocknumbers

5.21.2.17 print_root()

```
void FileSystem::print_root ( )
```

Print out the root Tree

5.21.2.18 print_tags()

```
void FileSystem::print_tags ( )
```

Print out all the tag trees and their contents

5.21.2.19 read_file()

```
size_t FileSystem::read_file (
    unsigned int fileDesc,
    char * data,
    size_t len )
```

Will read a number of bytes from an open file. The file must have been opened with read permissions. If you read past the end of the file, EOF will be tripped and you cannot continue reading. will return all the data up to that point

Parameters

<i>fileDesc</i>	the file descriptor returned from open_file
<i>data</i>	a buffer to store the read data must be at least len size
<i>len</i>	the number of bytes to read.

5.21.2.20 rename_file()

```
void FileSystem::rename_file (
    vector< string > & originalFilePath,
    string newFileName )
```

Will rename a file. The new file must not already exist in the emulated directory

Parameters

<i>originalFilePath</i>	the full path to the file to be renamed
<i>newFileName</i>	the name that the file will be renamed to.

5.21.2.21 rename_tag()

```
void FileSystem::rename_tag (
```

```
string originalTagName,  
string newTagName )
```

Will rename the tag. The tag must exist. The new tag name must already exist. This is a slow operation.

Parameters

<i>originalTagName</i>	the name of the tag to be renamed
<i>newTagName</i>	the new tag name for that tag

5.21.2.22 seek_file_absolute()

```
void FileSystem::seek_file_absolute (   
    unsigned int fileDesc,  
    size_t offset )
```

Seek to an absolute position in the file. Will trip EOF if the offset is larger than the file size. The position in the file is indexed at 1.

Parameters

<i>fileDesc</i>	the file descriptor returned from open_file
<i>offset</i>	the absolute position in the file to seek to.

5.21.2.23 seek_file_relative()

```
void FileSystem::seek_file_relative (   
    unsigned int fileDesc,  
    long int offset )
```

Seek to a relative position in the file. Will trip EOF if you try to seek too far in a direction. The position in the file is indexed at 1.

Parameters

<i>fileDesc</i>	the file descriptor returned from open_file
<i>offset</i>	the relative position in the file to seek to. may be a negative number.

5.21.2.24 set_permissions()

```
void FileSystem::set_permissions (   
    vector< string > & filePath,  
    char * perms )
```

Set the permissions for a file. The format is defined in [FileInfo](#).

Parameters

<i>filePath</i>	the full path to the file to you wish to get the Attributes of
<i>perms</i>	the permissions following the correct format to set to this file

See also

[FileInfo::set_permissions\(\)](#)

5.21.2.25 tag_file() [1/2]

```
void FileSystem::tag_file (
    FileInfo * file,
    unordered_set< string > tagsToAdd )
```

Will tag a file with the specified tags. If some or all of the tags do not exist, a warning is printed and the operation continues. The file must exist. The file that would be created by adding tags must not already exist.

Parameters

<i>file</i>	the FileInfo* that will be tagged with the specified tags
<i>tagsToAdd</i>	the tags that will be added to the file's tag set

See also

[tag_file\(vector<string>&, unordered_set<string>\)](#)

5.21.2.26 tag_file() [2/2]

```
void FileSystem::tag_file (
    vector< string > & filePath,
    unordered_set< string > tags )
```

An alternate way to tag a file using a file path instead. Will tag a file with the specified tags. If some or all of the tags do not exist, a warning is printed and the operation continues. The file must exist. The file that would be created by adding tags must not already exist.

Parameters

<i>filePath</i>	the FileInfo* that will be tagged with the specified tags
<i>tagsToAdd</i>	the tags that will be added to the file's tag set

See also

[tag_file\(FileInfo*, unordered_set<string>\)](#)

5.21.2.27 tag_search()

```
vector< FileInfo * > * FileSystem::tag_search (
    unordered_set< string > & tags )
```

Search for files by tags. The tag search is an "and" operation, meaning the files returned will have at least all the specified tags.

Parameters

<i>tags</i>	the tags that the files will be tagged with in the return vector
-------------	--

Returns

a pointer to a vector of the [FileInfo](#) objects which then can be serialized. The returned vector should be freed by the calling code

5.21.2.28 untag_file() [1/2]

```
void FileSystem::untag_file (
    FileInfo * file,
    unordered_set< string > tagsToRemove,
    bool deleting = false )
```

Will remove tags associated with the specified file. The tags must exist. The file must exist. The file that would be created by removing tags must not already exist.

Parameters

<i>file</i>	the FileInfo* that will be untagged with the specified tags
<i>tagsToRemove</i>	the tags that will be removed from the file's tag set
<i>deleting</i>	this is a tag only used by the FileSystem itself for deleting a file

See also

[tag_file\(FileInfo*, unordered_set<string>\)](#)

5.21.2.29 `untag_file()` [2/2]

```
void FileSystem::untag_file (
    vector< string > & filePath,
    unordered_set< string > tags )
```

Will remove tags associated with the specified file. The tags must exist. The file must exist. The file that would be created by removing tags must not already exist.

Parameters

<i>file</i>	the FileInfo* that will be untagged with the specified tags
<i>tagsToRemove</i>	the tags that will be removed from the file's tag set
<i>deleting</i>	this is a tag only used by the FileSystem itself for deleting a file

See also

[tag_file\(FileInfo*, unordered_set<string>\)](#)

5.21.2.30 `write_changes()`

```
void FileSystem::write_changes ( )
```

Since the [FileSystem](#) is journaling. The changes to tag trees and the Root tree are only written out when this is called. [File](#) Operations are not journaled.

5.21.2.31 `write_file()`

```
size_t FileSystem::write_file (
    unsigned int fileDesc,
    const char * data,
    size_t len )
```

Will write a number of bytes to an open file. The file must have been opened with write permissions. You can write past the EOF with no problems.

Parameters

<i>fileDesc</i>	the file descriptor returned from <code>open_file</code>
<i>data</i>	a buffer to be read from to write to the file. must be at least of len size
<i>len</i>	the number of bytes to write from data.

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

- [Filesystem/DaemonDependancies/FileSystem/FileSystem.h](#)
- [Filesystem/DaemonDependancies/FileSystem/FileSystem.cpp](#)

5.22 finode Struct Reference

```
#include <types.h>
```

Public Attributes

- [BlkNumType attributes](#)
- [BlkNumType directBlocks](#) [12]
- [BlkNumType level1Indirect](#)
- [BlkNumType level2Indirect](#)
- [BlkNumType level3Indirect](#)

5.22.1 Member Data Documentation

5.22.1.1 attributes

```
BlkNumType finode::attributes
```

5.22.1.2 directBlocks

```
BlkNumType finode::directBlocks[12]
```

5.22.1.3 level1Indirect

```
BlkNumType finode::level1Indirect
```

5.22.1.4 level2Indirect

```
BlkNumType finode::level2Indirect
```

5.22.1.5 level3Indirect

```
BlkNumType finode::level3Indirect
```

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

- Filesystem/DaemonDependencies/Types/[types.h](#)

5.23 index Struct Reference

```
#include <types.h>
```

Public Attributes

- [BlkNumType](#) blknum
- [size_t](#) offset

5.23.1 Member Data Documentation

5.23.1.1 blknum

```
BlkNumType index::blknum
```

5.23.1.2 offset

```
size\_t index::offset
```

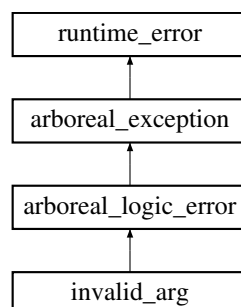
The documentation for this struct was generated from the following file:

- Filesystem/DaemonDependancies/Types/[types.h](#)

5.24 invalid_arg Class Reference

```
#include <Arboreal_Exceptions.h>
```

Inheritance diagram for `invalid_arg`:



Public Member Functions

- `invalid_arg` (const char **what*, const char **where*, const int *ecode*=99)
- `invalid_arg` (const char **what*, const string &*where*, const int *ecode*=99)
- `invalid_arg` (const string &*what*, const string &*where*, const int *ecode*=99)
- `invalid_arg` (const string &*what*, const char **where*, const int *ecode*=99)
- `~invalid_arg` () throw ()

Additional Inherited Members

5.24.1 Constructor & Destructor Documentation

5.24.1.1 `invalid_arg()` [1/4]

```
invalid_arg::invalid_arg (  
    const char * what,  
    const char * where,  
    const int ecode = 99 )
```

5.24.1.2 `invalid_arg()` [2/4]

```
invalid_arg::invalid_arg (  
    const char * what,  
    const string & where,  
    const int ecode = 99 )
```

5.24.1.3 `invalid_arg()` [3/4]

```
invalid_arg::invalid_arg (  
    const string & what,  
    const string & where,  
    const int ecode = 99 )
```

5.24.1.4 `invalid_arg()` [4/4]

```
invalid_arg::invalid_arg (  
    const string & what,  
    const char * where,  
    const int ecode = 99 )
```

5.24.1.5 ~invalid_arg()

```
invalid_arg::~invalid_arg ( ) throw ( )
```

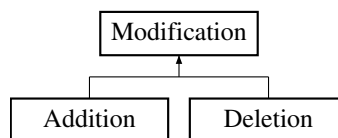
The documentation for this class was generated from the following files:

- SharedHeaders/[Arboreal_Exceptions.h](#)
- SharedCPPFiles/[Arboreal_Exceptions.cpp](#)

5.25 Modification Class Reference

```
#include <Trees.h>
```

Inheritance diagram for Modification:



Public Member Functions

- virtual [~Modification](#) ()
- virtual void [write_out](#) ([PartitionManager](#) *pm)=0

Protected Member Functions

- [Modification](#) ([TreeObject](#) *obj, [TreeObject](#) *parent)

Protected Attributes

- [TreeObject](#) * [_mod](#)
- [TreeObject](#) * [_parent](#)

5.25.1 Constructor & Destructor Documentation

5.25.1.1 Modification()

```
Modification::Modification (
    TreeObject * obj,
    TreeObject * parent ) [protected]
```

5.25.1.2 ~Modification()

```
Modification::~Modification ( ) [virtual]
```

5.25.2 Member Function Documentation

5.25.2.1 write_out()

```
virtual void Modification::write_out (
    PartitionManager * pm ) [pure virtual]
```

Implemented in [Deletion](#), and [Addition](#).

5.25.3 Member Data Documentation

5.25.3.1 _mod

```
TreeObject* Modification::_mod [protected]
```

5.25.3.2 _parent

```
TreeObject* Modification::_parent [protected]
```

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

- Filesystem/DaemonDependancies/Trees/[Trees.h](#)
- Filesystem/DaemonDependancies/Trees/[Trees.cpp](#)

5.26 ParseError Class Reference

```
#include <Parser.h>
```

Public Member Functions

- [ParseError](#) (const char *[where](#), const char *[what](#))
- std::string [where](#) ()
- std::string [what](#) ()

5.26.1 Constructor & Destructor Documentation

5.26.1.1 ParseError()

```
ParseError::ParseError (
    const char * where,
    const char * what ) [inline]
```

Parameters

<i>where</i>	Where the parse error took place
<i>what</i>	What the parse error consisted of

5.26.2 Member Function Documentation

5.26.2.1 what()

```
std::string ParseError::what ( ) [inline]
```

Returns

A std::string detailing what the parse error consisted of

5.26.2.2 where()

```
std::string ParseError::where ( ) [inline]
```

Returns

A std::string detailing where the parse error occurred

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

- SharedHeaders/[Parser.h](#)

5.27 Parser Class Reference

```
#include <Parser.h>
```

Public Member Functions

- [Parser](#) (char *buffer, char *cwd, int max_name_size)
- [Parser](#) (std::string string, std::string cwd, int max_name_size)
- [Parser](#) (const char *string_lit, const char *cwd, int max_name_size)
- [Parser](#) ()
- [~Parser](#) ()
- void [reset](#) (std::string string, std::string cwd="")
Changes the member _string of the parser class to whatever is passed.
- void [reset](#) (char *buffer, char *cwd=NULL)
Changes the member _string of the parser class to whatever is passed.
- void [reset](#) (const char *string_lit, const char *cwd="")
Changes the member _string of the parser class to whatever is passed.
- void [set_max_name_size](#) (int size)
Sets the maximum allowed file and tagname size that the [Parser](#) will use.
- void [set_cwd](#) (std::string cwd)
Sets the Current Working Directory that the [Parser](#) will use.
- std::vector< std::string > [parse](#) (int type)
Parse a string based on a certain rule.
- std::vector< std::string > [get_cwd_tags](#) ()
Returns a vector representation of the current working directory.

Static Public Member Functions

- static std::vector< std::string > [split_on_delim](#) (std::string string, char delim)
Splits a string at each instance of a particular char (the delimiter)

5.27.1 Constructor & Destructor Documentation

5.27.1.1 [Parser](#)() [1/4]

```
Parser::Parser (
    char * buffer,
    char * cwd,
    int max_name_size )
```

Parameters

<i>buffer</i>	A C-Style String representation of the string to be parsed
<i>cwd</i>	A C-Style String representation of the current working directory; (This value is typically provided by the Liaison process). The directory string is used to parse commands which act within directories only thus providing commands such as 'tag' a "path" to the file(s) which will be tagged without the user having to explicitly enter those file's entire paths themselves.
<i>max_name_size</i>	The maximum length that a file or tagname is allowed to have; (This value is typically provided by the Liaison process)

5.27.1.2 Parser() [2/4]

```
Parser::Parser (
    std::string string,
    std::string cwd,
    int max_name_size )
```

Parameters

<i>buffer</i>	A std::string representation of the string to be parsed
<i>cwd</i>	A std::string representation of the current working directory; (This value is typically provided by the Liaison process). The directory string is used to parse commands which act within directories only thus, providing commands such as 'tag' a "path" to the file(s) which will be tagged without the user having to explicitly enter those file's entire paths themselves.
<i>max_name_size</i>	The maximum length that a file or tagname is allowed to have; (This value is typically provided by the Liaison process)

5.27.1.3 Parser() [3/4]

```
Parser::Parser (
    const char * string_lit,
    const char * cwd,
    int max_name_size )
```

Parameters

<i>buffer</i>	A String Literal representation of the string to be parsed
<i>cwd</i>	A String Literal representation of the current working directory; (This value is typically provided by the Liaison process). The directory string is used to parse commands which act within directories only thus, providing commands such as 'tag' a "path" to the file(s) which will be tagged without the user having to explicitly enter those file's entire paths themselves.
<i>max_name_size</i>	The maximum length that a file or tagname is allowed to have; (This value is typically provided by the Liaison process)

5.27.1.4 Parser() [4/4]

```
Parser::Parser ( )
```

Default Constructor to be used in case initialization of values needs to be done elsewhere

5.27.1.5 ~Parser()

```
Parser::~~Parser ( )
```

Default Destructor; Does nothing

5.27.2 Member Function Documentation

5.27.2.1 `get_cwd_tags()`

```
std::vector< std::string > Parser::get_cwd_tags ( )
```

Returns a vector representation of the current working directory.

That is, it will decompose `'/string1/string2'` into a vector containing `[string1, string2]`. This is useful when the calling code requires the current working directory as a vector of strings rather than as a standard string representation.

Returns

A `std::vector` of `std::string` comprised of the non-`'/'` parts of the [Parser](#) member value `_cwd`

5.27.2.2 `parse()`

```
std::vector< std::string > Parser::parse (
    int type )
```

Parse a string based on a certain rule.

The rule generally corresponds to how a [CLI](#) command should be decomposed.

For example the [CLI](#) command for finding files takes a list of files, however the filesystem itself does not support batch commands, therefore, the [Parser](#) will decompose the command into its constituent parts (i.e. a single file).

This particular behavior is achieved by passing `'8'` as the "type" of decomposition that needs to take place (Note that this corresponds to the command's ID).

However the [Parser](#) can be extended to support any rule whatsoever, so long as it is added to the [Parser's](#) `parse()` function switch statement.

Parameters

<i>type</i>	The integer identification of the parse rule that will be executed
-------------	--

Returns

A `std::vector` of `std::string` comprised of the result after the chosen parse rule is executed.

5.27.2.3 `reset()` [1/3]

```
void Parser::reset (
    std::string string,
    std::string cwd = "" )
```

Changes the member `_string` of the parser class to whatever is passed.

The [Parser](#) class conducts all operations on its member `_string` rather than requiring that a string value be passed to its [parse\(\)](#) method. This was done in order to make use of the class as streamlined as possible.

Parameters

<i>string</i>	A <code>std::string</code> representation of the string to be parsed
<i>cwd</i>	A <code>std::string</code> representation of the current working directory; Note that this argument is optional and allows the user to both reset the string the Parser will work with as well as the directory string the Parser will use. The directory string is used to parse commands which act within directories only thus providing commands such as 'tag' a "path" to the file(s) which will be tagged without the user having to explicitly enter those file's entire paths themselves.

5.27.2.4 [reset\(\)](#) [2/3]

```
void Parser::reset (
    char * buffer,
    char * cwd = NULL )
```

Changes the member `_string` of the parser class to whatever is passed.

The [Parser](#) class conducts all operations on its member `_string` rather than requiring that a string value be passed to its [parse\(\)](#) method. This was done in order to make use of the class as streamlined as possible.

Parameters

<i>string</i>	A C-Style String representation of the string to be parsed
<i>cwd</i>	A C-Style String representation of the current working directory; Note that this argument is optional and allows the user to both reset the string the Parser will work with as well as the directory string the Parser will use. The directory string is used to parse commands which act within directories only thus providing commands such as 'tag' a "path" to the file(s) which will be tagged without the user having to explicitly enter those file's entire paths themselves.

Returns

Void

5.27.2.5 [reset\(\)](#) [3/3]

```
void Parser::reset (
    const char * string_lit,
    const char * cwd = "" )
```

Changes the member `_string` of the parser class to whatever is passed.

The [Parser](#) class conducts all operations on its member `_string` rather than requiring that a string value be passed to its [parse\(\)](#) method. This was done in order to make use of the class as streamlined as possible.

Parameters

<i>string</i>	A String Literal representation of the string to be parsed
<i>cwd</i>	A String Literal representation of the current working directory; Note that this argument is optional and allows the user to both reset the string the Parser will work with as well as the directory string the Parser will use. The directory string is used to parse commands which act within directories only thus providing commands such as 'tag' a "path" to the file(s) which will be tagged without the user having to explicitly enter those file's entire paths themselves.

Returns

Void

5.27.2.6 `set_cwd()`

```
void Parser::set_cwd (
    std::string cwd )
```

Sets the Current Working Directory that the [Parser](#) will use.

The directory string is used to parse commands which act within directories only thus providing commands such as 'tag' a "path" to the file(s) which will be tagged without the user having to explicitly enter those file's entire paths themselves. This function does not have counterparts which take C-Style Strings or String Literals. This is because, in all situations, if the current working directory must be set using this method, it is highly likely that the calling code has a `std::string` representation of the current working directory rather than a representation in one of the other formats. If such functionality (C-Style Strings and others) is desired, extensibility is easy enough. Regardless the [Parser](#)'s `_cwd` member will always be a `std::string`.

Parameters

<i>cwd</i>	A <code>std::string</code> representation of the current working directory
------------	--

Returns

Void

5.27.2.7 `set_max_name_size()`

```
void Parser::set_max_name_size (
    int size )
```

Sets the maximum allowed file and tagname size that the [Parser](#) will use.

If this size is exceeded an error is thrown and the [Parser](#) will stop its current activities. This value is dictated by the [CLI](#) and is generally provided to the [Parser](#) by the Liaison Process.

Parameters

<i>size</i>	The maximum file/tag name length
-------------	----------------------------------

5.27.2.8 split_on_delim()

```
std::vector< std::string > Parser::split_on_delim (
    std::string string,
    char delim ) [static]
```

Splits a string at each instance of a particular char (the delimiter)

The delimiters are NOT included anywhere in the resulting vector. This function is static and is mainly used outside the [Parser](#) in order to split values that the parser returned. This can happen because the complexity of certain commands does not allow the parser to fully decompose the string and instead it can only reorganize the command into a form which can be easily split later. It is important to note that this function does not differentiate between the number of delimiter characters the string contains. That is, it will read the whole string and split it at any point where the delimiter is seen whether it is seen in 1 or 100 places.

Parameters

<i>string</i>	A std::string representation of whatever string needs to be split
<i>delim</i>	A char value representing where the string should be split

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

- SharedHeaders/[Parser.h](#)
- SharedCPPFiles/[Parser.cpp](#)

5.28 PartitionManager Class Reference

```
#include <PartitionManager.h>
```

Public Member Functions

- [PartitionManager](#) ([DiskManager](#) **dm*, string *partitionName*)
- [~PartitionManager](#) ()

Accessor Functions

- void [readDiskBlock](#) ([BlkNumType](#) *blknum*, char **blkdata*)
- [size_t](#) [getBlockSize](#) ()
- string [getPartitionName](#) ()
- int [get_file_name_size](#) ()

Modifier Functions

- void [writeDiskBlock](#) ([BlkNumType](#) *blknum*, char **blkdata*)
- [BlkNumType](#) [getFreeDiskBlock](#) ()
- void [returnDiskBlock](#) ([BlkNumType](#) *blknum*)

5.28.1 Constructor & Destructor Documentation

5.28.1.1 PartitionManager()

```
PartitionManager::PartitionManager (
    DiskManager * dm,
    string partitionName )
```

Parameters

<i>dm</i>	the DiskManager associated with this object
<i>partitionName</i>	the name of the partition that this will be managing

5.28.1.2 ~PartitionManager()

```
PartitionManager::~~PartitionManager ( )
```

5.28.2 Member Function Documentation

5.28.2.1 get_file_name_size()

```
int PartitionManager::get_file_name_size ( )
```

Returns

The maximum file name size for this partition in bytes

5.28.2.2 getBlockSize()

```
size_t PartitionManager::getBlockSize ( )
```

Returns

the blocksize of the [Disk](#)

5.28.2.3 `getFreeDiskBlock()`

```
BlkNumType PartitionManager::getFreeDiskBlock ( )
```

Allocates a block on disk if there is a free one. The [Disk](#) free list is updated accordingly

Returns

the block number of the newly allocated block

5.28.2.4 `getPartitionName()`

```
string PartitionManager::getPartitionName ( )
```

Returns

The name of the partition this [PartitionManager](#) is associated with

5.28.2.5 `readDiskBlock()`

```
void PartitionManager::readDiskBlock (
    BlkNumType blknum,
    char * blkdata )
```

Reads a block from the [Disk](#).

Parameters

<i>blknum</i>	the blocknumber to be read
<i>blkdata</i>	the buffer to put the read data. must be large enough to contain an entire block of data

5.28.2.6 `returnDiskBlock()`

```
void PartitionManager::returnDiskBlock (
    BlkNumType blknum )
```

returns a block to the [Disk](#) free list and zeros it out before writing.

Parameters

<i>blknum</i>	the blocknumber of the block to be freed
---------------	--

5.28.2.7 writeDiskBlock()

```
void PartitionManager::writeDiskBlock (
    BlkNumType blknum,
    char * blkdata )
```

Writes a block to the [Disk](#).

Parameters

<i>blknum</i>	the blocknumber to be written
<i>blkdata</i>	the buffer to write the data from. It Will write an entire block size of data.

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

- Filesystem/DaemonDependancies/PartitionManager/[PartitionManager.h](#)
- Filesystem/DaemonDependancies/PartitionManager/[PartitionManager.cpp](#)

5.29 rootSuperBlock Struct Reference

```
#include <types.h>
```

Public Attributes

- `size_t` [size](#)
- [Index](#) [lastEntry](#)
- [BlkNumType](#) [startBlock](#)

5.29.1 Member Data Documentation

5.29.1.1 lastEntry

```
Index rootSuperBlock::lastEntry
```

5.29.1.2 size

```
size_t rootSuperBlock::size
```

5.29.1.3 startBlock

```
BlkNumType rootSuperBlock::startBlock
```

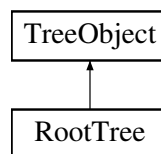
The documentation for this struct was generated from the following file:

- Filesystem/DaemonDependencies/Types/types.h

5.30 RootTree Class Reference

```
#include <Trees.h>
```

Inheritance diagram for RootTree:



Public Member Functions

- [RootTree](#) ([PartitionManager](#) *pm)
- [~RootTree](#) ()
- void [write_out](#) ()
- void [read_in](#) (unordered_multimap< string, [FileInfo](#) *> *allFiles, [RootTree](#) *rootTree)
- void [del](#) ()

Additional Inherited Members

5.30.1 Constructor & Destructor Documentation

5.30.1.1 RootTree()

```
RootTree::RootTree (
    PartitionManager * pm )
```

Parameters

<i>pm</i>	the PartitionManager to be associated with the RootTree
-----------	---

5.30.1.2 ~RootTree()

```
RootTree::~~RootTree ( )
```

5.30.2 Member Function Documentation

5.30.2.1 del()

```
void RootTree::del ( ) [virtual]
```

Will completely remove the [TreeObject](#)'s presence on disk

Implements [TreeObject](#).

5.30.2.2 read_in()

```
void RootTree::read_in (
    unordered_multimap< string, FileInfo *> * allFiles,
    RootTree * rootTree ) [virtual]
```

Will read in all object data from disk

Parameters

<i>allFiles</i>	a pointer to the map of all files
<i>rootTree</i>	a pointer to the root tree

Implements [TreeObject](#).

5.30.2.3 write_out()

```
void RootTree::write_out ( ) [virtual]
```

Intended to write out the object to disk

Implements [TreeObject](#).

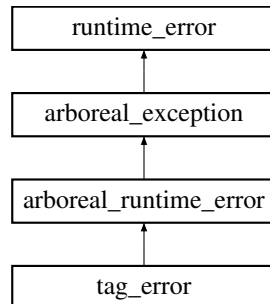
The documentation for this class was generated from the following files:

- Filesystem/DaemonDependancies/Trees/[Trees.h](#)
- Filesystem/DaemonDependancies/Trees/[Trees.cpp](#)

5.31 tag_error Class Reference

```
#include <Arboreal_Exceptions.h>
```

Inheritance diagram for tag_error:



Public Member Functions

- [tag_error](#) (const char *what, const char *where, const int [ecode](#)=99)
- [tag_error](#) (const char *what, const string &where, const int [ecode](#)=99)
- [tag_error](#) (const string &what, const string &where, const int [ecode](#)=99)
- [tag_error](#) (const string &what, const char *where, const int [ecode](#)=99)
- [~tag_error](#) () throw ()

Additional Inherited Members

5.31.1 Constructor & Destructor Documentation

5.31.1.1 tag_error() [1/4]

```
tag_error::tag_error (
    const char * what,
    const char * where,
    const int ecode = 99 )
```

5.31.1.2 tag_error() [2/4]

```
tag_error::tag_error (
    const char * what,
    const string & where,
    const int ecode = 99 )
```


5.31.1.3 tag_error() [3/4]

```
tag_error::tag_error (
    const string & what,
    const string & where,
    const int ecode = 99 )
```

5.31.1.4 tag_error() [4/4]

```
tag_error::tag_error (
    const string & what,
    const char * where,
    const int ecode = 99 )
```

5.31.1.5 ~tag_error()

```
tag_error::~tag_error ( ) throw ( )
```

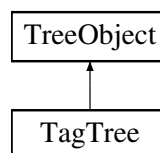
The documentation for this class was generated from the following files:

- SharedHeaders/[Arboreal_Exceptions.h](#)
- SharedCPPFiles/[Arboreal_Exceptions.cpp](#)

5.32 TagTree Class Reference

```
#include <Trees.h>
```

Inheritance diagram for TagTree:



Public Member Functions

- [TagTree](#) (string tagName, [BlkNumType](#) blknum, [PartitionManager](#) *pm)
- [~TagTree](#) ()
- void [write_out](#) ()
- void [read_in](#) (unordered_multimap< string, [FileInfo](#) *> *allFiles, [RootTree](#) *rootTree)
- void [del](#) ()

Additional Inherited Members

5.32.1 Constructor & Destructor Documentation

5.32.1.1 TagTree()

```
TagTree::TagTree (
    string tagName,
    BlkNumType blknum,
    PartitionManager * pm )
```

Parameters

<i>tagName</i>	the name of this tag
<i>blknum</i>	the blocknumber for the superblock of this tagTree

5.32.1.2 ~TagTree()

```
TagTree::~~TagTree ( )
```

5.32.2 Member Function Documentation

5.32.2.1 del()

```
void TagTree::del ( ) [virtual]
```

Will completely remove the [TreeObject](#)'s presence on disk

Implements [TreeObject](#).

5.32.2.2 read_in()

```
void TagTree::read_in (
    unordered_multimap< string, FileInfo *> * allFiles,
    RootTree * rootTree ) [virtual]
```

Will read in all object data from disk

Parameters

<i>allFiles</i>	a pointer to the map of all files
<i>rootTree</i>	a pointer to the root tree

Implements [TreeObject](#).

5.32.2.3 write_out()

```
void TagTree::write_out ( ) [virtual]
```

Intended to write out the object to disk

Implements [TreeObject](#).

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

- Filesystem/DaemonDependancies/Trees/[Trees.h](#)
- Filesystem/DaemonDependancies/Trees/[Trees.cpp](#)

5.33 tagTreeSuperBlock Struct Reference

```
#include <types.h>
```

Public Attributes

- `size_t` [size](#)
- `Index` [lastEntry](#)
- `BlkNumType` [startBlock](#)

5.33.1 Member Data Documentation

5.33.1.1 lastEntry

```
Index tagTreeSuperBlock::lastEntry
```

5.33.1.2 size

```
size_t tagTreeSuperBlock::size
```

5.33.1.3 startBlock

```
BlkNumType tagTreeSuperBlock::startBlock
```

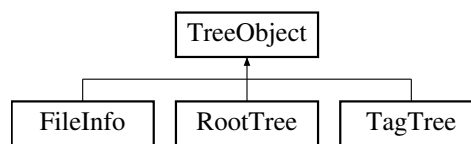
The documentation for this struct was generated from the following file:

- Filesystem/DaemonDependencies/Types/types.h

5.34 TreeObject Class Reference

```
#include <Trees.h>
```

Inheritance diagram for TreeObject:



Public Member Functions

- virtual `~TreeObject()`
- `TreeObject` (string name, `BlkNumType` blknum, `PartitionManager` *pm)

Accessor Functions

- string `get_name()` const
- `BlkNumType` `get_block_number()` const
- `Index` `get_index` (`TreeObject` *obj) const
- `Index` `get_last_entry()` const
- `BlkNumType` `get_start_block()` const
- `size_t` `size()` const
- `unordered_map`< string, `TreeObject` * >::iterator `begin()`
- `unordered_map`< string, `TreeObject` * >::iterator `end()`
- `TreeObject` * `find` (string name) const
- `queue`< `Index` > * `get_free_spots()`

Modifier Functions

- void `set_name` (string name)
- void `add_index` (`TreeObject` *obj, `Index` index)
- void `set_last_entry` (`Index` index)
- virtual void `insert` (string name, `TreeObject` *obj)
- virtual void `erase` (string name)
- virtual void `insert_addition` (`TreeObject` *add)
- virtual void `insert_deletion` (`TreeObject` *del)

Disk Functions

- virtual void `write_out` ()=0
- virtual void `read_in` (`unordered_multimap`< string, `FileInfo` * > *allFiles, `RootTree` *rootTree)=0
- virtual void `del` ()=0
- void `increment_allocate` (`Index` *index)
- void `increment_follow` (`Index` *index)

Protected Member Functions

- virtual void [delete_cont_blocks](#) ([BlkNumType](#) blknum)

Protected Attributes

- queue< [Modification](#) * > [_modifications](#)
A collection of associated Modifications.
- unordered_map< string, [TreeObject](#) * > [_myTree](#)
A collection of contained TreeObjects.
- string [_name](#)
name or value
- [BlkNumType](#) [_blockNumber](#)
Blocknumber of the superblock on disk.
- unordered_map< [TreeObject](#) *, [Index](#) > [_indeces](#)
location(s) of the superblock entry(ies) on disk
- [Index](#) [_lastEntry](#)
Index of the last entry of data on disk.
- [BlkNumType](#) [_startBlock](#)
blocknumber of the start of this data on disk
- [PartitionManager](#) * [_myPartitionManager](#)
Associated [PartitionManager](#).
- queue< [Index](#) > [_freeSpots](#)

5.34.1 Constructor & Destructor Documentation

5.34.1.1 ~TreeObject()

```
TreeObject::~TreeObject ( ) [virtual]
```

5.34.1.2 TreeObject()

```
TreeObject::TreeObject (
    string name,
    BlkNumType blknum,
    PartitionManager * pm )
```

Parameters

<i>name</i>	name of this object
<i>blknum</i>	blocknumber of the superblock
<i>pm</i>	PartitionManager object to be associated with this object

5.34.2 Member Function Documentation

5.34.2.1 add_index()

```
void TreeObject::add_index (
    TreeObject * obj,
    Index index )
```

Add an index to `_indeces` for the specified [TreeObject](#). If the index already existed. nothing happens

Parameters

<i>obj</i>	the object that the Index references to
<i>index</i>	the Index of obj

5.34.2.2 begin()

```
unordered_map< string, TreeObject * >::iterator TreeObject::begin ( )
```

Returns

An iterator to the beginning of the TreeObjects associated with this object

5.34.2.3 del()

```
virtual void TreeObject::del ( ) [pure virtual]
```

Will completely remove the [TreeObject](#)'s presence on disk

Implemented in [RootTree](#), [TagTree](#), and [FileInfo](#).

5.34.2.4 delete_cont_blocks()

```
void TreeObject::delete_cont_blocks (
    BlkNumType blknum ) [protected], [virtual]
```

Will follow the chain of continuation blocks and free all of them

Parameters

<i>blknum</i>	will free the blknum and use it to follow the chain of continuation blocks
---------------	--

Reimplemented in [FileInfo](#).

5.34.2.5 end()

```
unordered_map< string, TreeObject * >::iterator TreeObject::end ( )
```

Returns

An iterator to the end of the TreeObjects associated with this object

5.34.2.6 erase()

```
void TreeObject::erase (
    string name ) [virtual]
```

Disassociate the given name from this object

Parameters

<i>name</i>	the name of the object to be erased.
-------------	--------------------------------------

Exceptions

arboreal_logic_error	
--------------------------------------	--

Reimplemented in [FileInfo](#).

5.34.2.7 find()

```
TreeObject * TreeObject::find (
    string name ) const
```

Search `_myTree` for the specified name

Parameters

<i>name</i>	the name of the desired object
-------------	--------------------------------

Returns

a pointer to the object if found, 0 otherwise

5.34.2.8 get_block_number()

```
BlkNumType TreeObject::get_block_number ( ) const
```

Returns

The blocknumber of the superblock

5.34.2.9 get_free_spots()

```
queue< Index > * TreeObject::get_free_spots ( )
```

Returns

a pointer to the queue of empty spaces where new entries can be added

5.34.2.10 get_index()

```
Index TreeObject::get_index (
    TreeObject * obj ) const
```

Searches for obj and returns the Index of obj on disk, if found

Parameters

<i>obj</i>	object whose position is desired
------------	----------------------------------

Returns

The Index of obj on disk,

Exceptions

<i>arboreal_logic_error</i>	
---	--

5.34.2.11 get_last_entry()

```
Index TreeObject::get_last_entry ( ) const
```

Find the Index of the last entry for this object on disk

Returns

Index of the last entry on disk

5.34.2.12 get_name()

```
string TreeObject::get_name ( ) const
```

Returns

The name

5.34.2.13 get_start_block()

```
BlkNumType TreeObject::get_start_block ( ) const
```

Returns

The start block of data for this object

5.34.2.14 increment_allocate()

```
void TreeObject::increment_allocate (
    Index * index )
```

Will increment the Index passed and allocate blocks if necessary to do so

Parameters

<i>index</i>	the Index to be incremented
--------------	-----------------------------

5.34.2.15 increment_follow()

```
void TreeObject::increment_follow (
    Index * index )
```

Will increment the Index passed but only follow the chain of already allocated blocks

Parameters

<i>index</i>	the Index to be incremented
--------------	-----------------------------

5.34.2.16 insert()

```
void TreeObject::insert (
    string name,
    TreeObject * obj ) [virtual]
```

Associate a [TreeObject](#) with this object

Parameters

<i>name</i>	name of the object, mangled if inserting a FileInfo
<i>obj</i>	the object to be inserted

Exceptions

tag_error	
---------------------------	--

See also

[FileInfo::insert\(\)](#)

Reimplemented in [FileInfo](#).

5.34.2.17 insert_addition()

```
void TreeObject::insert_addition (
    TreeObject * add ) [virtual]
```

Add an [Addition](#) to the list of Modifications so that it can be written out later. Note: Do not call this on a [FileInfo](#).

Parameters

<i>add</i>	the object that was previously inserted to this object which will be added to the list of Modifications
------------	---

See also

FileSystem::write_out() [TreeObject::insert\(\)](#)

Reimplemented in [FileInfo](#).

5.34.2.18 insert_deletion()

```
void TreeObject::insert_deletion (
    TreeObject * del ) [virtual]
```

Add a [Deletion](#) to the list of Modifications so that it can be written out later. Note: Do not call this on a [FileInfo](#).

Parameters

<i>del</i>	the object that was previously erased from this object which will be added to the list of Modifications
------------	---

See also

FileSystem::write_out() [TreeObject::erase\(\)](#)

Reimplemented in [FileInfo](#).

5.34.2.19 read_in()

```
virtual void TreeObject::read_in (
    unordered_multimap< string, FileInfo *> * allFiles,
    RootTree * rootTree ) [pure virtual]
```

Will read in all object data from disk

Parameters

<i>allFiles</i>	a pointer to the map of all files
<i>rootTree</i>	a pointer to the root tree

Implemented in [RootTree](#), [TagTree](#), and [FileInfo](#).

5.34.2.20 set_last_entry()

```
void TreeObject::set_last_entry (
    Index index )
```

Set the last Index for the last entry belonging to this object on disk

Parameters

<i>index</i>	The last Index
--------------	----------------

5.34.2.21 set_name()

```
void TreeObject::set_name (
    string name )
```

Set the name

Parameters

<i>name</i>	The new name
-------------	--------------

5.34.2.22 size()

```
size_t TreeObject::size ( ) const
```

Returns

The size of `_myTree`

5.34.2.23 write_out()

```
virtual void TreeObject::write_out ( ) [pure virtual]
```

Intended to write out the object to disk

Implemented in [RootTree](#), [TagTree](#), and [FileInfo](#).

5.34.3 Member Data Documentation**5.34.3.1 _blockNumber**

```
BlkNumType TreeObject::_blockNumber [protected]
```

Blocknumber of the superblock on disk.

5.34.3.2 `_freeSpots`

`queue<Index> TreeObject::_freeSpots` [protected]

5.34.3.3 `_indeces`

`unordered_map<TreeObject*, Index> TreeObject::_indeces` [protected]

location(s) of the superblock entry(ies) on disk

5.34.3.4 `_lastEntry`

`Index TreeObject::_lastEntry` [protected]

Index of the last entry of data on disk.

5.34.3.5 `_modifications`

`queue<Modification*> TreeObject::_modifications` [protected]

A collection of associated Modifications.

5.34.3.6 `_myPartitionManager`

`PartitionManager* TreeObject::_myPartitionManager` [protected]

Associated [PartitionManager](#).

5.34.3.7 `_myTree`

`unordered_map<string, TreeObject*> TreeObject::_myTree` [protected]

A collection of contained TreeObjects.

5.34.3.8 `_name`

`string TreeObject::_name` [protected]

name or value

5.34.3.9 `_startBlock`

`BlkNumType TreeObject::_startBlock` [protected]

blocknumber of the start of this data on disk

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

- Filesystem/DaemonDependancies/Trees/[Trees.h](#)
- Filesystem/DaemonDependancies/Trees/[Trees.cpp](#)

Chapter 6

File Documentation

6.1 CommandLineInterface/CLDependancies/cli_helper.hpp File Reference

Macros

- `#define INCLUSIVE 0`
- `#define EXCLUSIVE 1`
- `#define NEW_AND_TAG 2`
- `#define NEW_AND_TAG_EXC 3`
- `#define MERGE_1 4`
- `#define MERGE_2 5`
- `#define TAG_1 6`
- `#define TAG_2 7`
- `#define TAG_3 8`
- `#define OPEN 9`

Functions

- void `clean` (int signal)
- void `bad_clean` (int signal)
- void `delete_shm` (int shm_id, char *shm)
- char * `create_shm_seg` (key_t key, int &id)
- int `get_cmnd_id` (char *cmnd)
- int `set_up_socket` (std::string client_sockpath, struct sockaddr_un &client_sockaddr)
- void `connect_to_server` (int client_sock, std::string client_sockpath, std::string server_sockpath, struct sockaddr_un &server_sockaddr, socklen_t len)
- void `send_to_server` (int client_sock, std::string client_sockpath, const char *cmnd, int size, int flag)
- char * `receive_from_server` (int client_sock, std::string client_sockpath, int size, int flag)

6.1.1 Macro Definition Documentation

6.1.1.1 EXCLUSIVE

```
#define EXCLUSIVE 1
```

6.1.1.2 INCLUSIVE

```
#define INCLUSIVE 0
```

6.1.1.3 MERGE_1

```
#define MERGE_1 4
```

6.1.1.4 MERGE_2

```
#define MERGE_2 5
```

6.1.1.5 NEW_AND_TAG

```
#define NEW_AND_TAG 2
```

6.1.1.6 NEW_AND_TAG_EXC

```
#define NEW_AND_TAG_EXC 3
```

6.1.1.7 OPEN

```
#define OPEN 9
```

6.1.1.8 TAG_1

```
#define TAG_1 6
```


6.1.1.9 TAG_2

```
#define TAG_2 7
```

6.1.1.10 TAG_3

```
#define TAG_3 8
```

6.1.2 Function Documentation

6.1.2.1 bad_clean()

```
void bad_clean (
    int signal )
```

Remove Socket Files in case of interrupt signals Called when signals indicating illegal operations (such as SIGSEGV) are thrown

Parameters

<i>signal</i>	Value returned by signal() function call
---------------	--

6.1.2.2 clean()

```
void clean (
    int signal )
```

Remove Socket Files in case of interrupt signals Called when signals originating from the user (such as SIGINT (control-C)) are thrown

Parameters

<i>signal</i>	Value returned by signal() function call
---------------	--

6.1.2.3 connect_to_server()

```
void connect_to_server (
    int client_sock,
```

```

std::string client_sockpath,
std::string server_sockpath,
struct sockaddr_un & server_sockaddr,
socklen_t len )

```

Attempt to initiate a connection to the Liaison process

Parameters

<i>client_sock</i>	Client socket identification number
<i>client_sockpath</i>	Client socket pathname
<i>server_sockpath</i>	Server socket pathname
<i>server_sockaddr</i>	A reference to a standard structure whose components I will not describe here and can be viewed in a Unix manual. Suffice it to say, it stores the socket type and the socket path. (Note that the "type" of the struct is <code>sockaddr_un</code> signifying that this is a unix domain socket)
<i>len</i>	Size of <code>server_sockaddr</code> in bytes (from <code>sizeof()</code>)

6.1.2.4 create_shm_seg()

```

char* create_shm_seg (
    key_t key,
    int & id )

```

Create and attach a Shared Memory Segment

Parameters

<i>key</i>	The Key required to access the Shared Memory Segment
<i>id</i>	Address of an integer variable that will store the created segments identification number

6.1.2.5 delete_shm()

```

void delete_shm (
    int shm_id,
    char * shm )

```

Delete a Shared Memory Fragment Shared Memory Fragments can only be deleted if they are not attached to anything Calling this function without having previously unattached a process from a segment will result in failure

shm_id: The Shared Memory Fragment's identifier shm: The pointer to the Shared Memory

6.1.2.6 get_cmnd_id()

```

int get_cmnd_id (
    char * cmnd )

```

Extracts the Command ID from a buffer created using [CLI::build\(\)](#) And returns it as an integer.

Parameters

<i>cmdnd</i>	A C-Style string created using CLI::build()
--------------	---

6.1.2.7 `receive_from_server()`

```
char* receive_from_server (
    int client_sock,
    std::string client_sockpath,
    int size,
    int flag )
```

Receive data from [File](#) System, returns a C-String containing the Data

Parameters

<i>client_sock</i>	Client socket identification number
<i>client_sockpath</i>	Client socket pathname
<i>size</i>	Size of command to be recieved
<i>flag</i>	Flag for 'recv()' call (see 'man recv')

6.1.2.8 `send_to_server()`

```
void send_to_server (
    int client_sock,
    std::string client_sockpath,
    const char * cmdnd,
    int size,
    int flag )
```

Send a command to the Liaison process

Parameters

<i>client_sock</i>	Client socket identification number
<i>client_sockpath</i>	Client socket pathname
<i>cmdnd</i>	Command to be sent
<i>size</i>	Size of 'cmdnd'
<i>flag</i>	Flag for 'send()' call (see 'man send')

6.1.2.9 `set_up_socket()`

```
int set_up_socket (
```

```
std::string client_sockpath,
struct sockaddr_un & client_sockaddr )
```

Create and set-up a socket used for communication with Liaison process Returns the client socket's identification number

Parameters

<i>client_sockpath</i>	Client Socket's pathname
<i>client_sockaddr</i>	A reference to a standard structure whose components I will not describe here and can be viewed in a Unix manual. Suffice it to say, it stores the socket type and the socket path. (Note that the "type" of the struct is <code>sockaddr_un</code> signifying that this is a unix domain socket)

6.2 CommandLineInterface/CLHeaders/Cli.h File Reference

```
#include <string>
#include <iostream>
#include <vector>
#include <errno.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>
#include "../SharedHeaders/Arboreal_Exceptions.h"
#include "../SharedHeaders/Print.h"
#include "../SharedHeaders/DebugMessages.hpp"
```

Classes

- class [CLI](#)

Variables

- static const int [MaxBufferSize](#) = 4096
- static const int [SharedMemorySize](#) = 1
- static const int [Permissions](#) = 0666
- static const int [Flag](#) = 0
- [DebugMessages](#) [Debug](#)

6.2.1 Variable Documentation

6.2.1.1 Debug

[DebugMessages](#) Debug

Socket Send/Recv Flag

6.2.1.2 Flag

```
const int Flag = 0 [static]
```

Socket Permissions

6.2.1.3 MaxBufferSize

```
const int MaxBufferSize = 4096 [static]
```

Strings

cout

Vectors

errno Definitions

Unix Std. Stuff

Socket Handling

Unix Domain Socket Stuff

Inter Process Communication Stds.

Shared Memory Handling

Signal Handling

System Types Definitions

Wait Calls

6.2.1.4 Permissions

```
const int Permissions = 0666 [static]
```

Size of Shared Memory Segment

6.2.1.5 SharedMemorySize

```
const int SharedMemorySize = 1 [static]
```

Maximum size a command can be

6.3 CommandLineInterface/Cli.cpp File Reference

```
#include "CLHeaders/Cli.h"  
#include "CLDependancies/cli_helper.hpp"
```

Functions

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

6.3.1 Function Documentation

6.3.1.1 main()

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

The Command Line has several different modes of execution

Mode 1: The most standard mode requires that a partition name be passed as an argument. The partition name must exist on the filesystem if it does not, the commandline will quit. This mode expects the user to manually type commands into the command line interface. This mode's run command looks like: './commandline PartitionName'

Mode 2: The second mode adds debugging information to Mode 1 The flag that must be passed to enable this mode is '-d' This mode's run command looks like: './commandline PartitionName -d'

Mode 3: The third mode operates similar to Mode 1 except that rather than expecting users to manually type commands in, it expects a file containing all of the commands that will be executed, to be piped to it. This mode still requires that a legal partition be passed. The flag that must be passed to enable this mode is '-d' This mode's run command looks like: './commandline PartitionName -s'

Mode 4: The fourth and final mode adds debugging support to Mode 3. The flag that must be passed to enable this mode is '-d' This mode's run command looks like: './commandline PartitionName -s -d'

Parameters

<i>argc</i>	The argument count (Not passed by user)
<i>argv</i>	The argument values (Passed by user)

Returns

An integer always equal to 0

6.4 diskInfo.d File Reference

6.5 exthd.d File Reference

6.6 Filesystem/daemon.cpp File Reference

```
#include <thread>
#include <errno.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <sys/ipc.h>
#include <sys/shm.h>
```

```
#include <netinet/in.h>
#include <netdb.h>
#include <sys/ioctl.h>
#include <signal.h>
#include <chrono>
#include <ctime>
#include "DaemonDependancies/FileSystem/FileSystem.h"
#include "DaemonDependancies/File/File.h"
#include "DaemonHeaders/daemon.h"
#include "../SharedHeaders/Print.h"
```

Macros

- #define [STARTTUPDATA](#) "Data/startup_time.txt"

Functions

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

6.6.1 Macro Definition Documentation

6.6.1.1 STARTTUPDATA

```
#define STARTTUPDATA "Data/startup_time.txt"
```

6.6.2 Function Documentation

6.6.2.1 main()

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

6.7 Filesystem/DaemonDependancies/Disk/Disk.cpp File Reference

```
#include "Disk.h"
```


6.8 Filesystem/DaemonDependencies/Disk/Disk.h File Reference

```
#include "../Types/types.h"
```

Classes

- class [Disk](#)

6.9 Filesystem/DaemonDependencies/DiskManager/DiskManager.cpp File Reference

```
#include "DiskManager.h"
```

Functions

- bool [operator==](#) (const [DiskPartition](#) *lhs, const [DiskPartition](#) &rhs)

6.9.1 Function Documentation

6.9.1.1 [operator==\(\)](#)

```
bool operator== (
    const DiskPartition * lhs,
    const DiskPartition & rhs )
```

6.10 Filesystem/DaemonDependencies/DiskManager/DiskManager.h File Reference

```
#include "../Types/types.h"
#include "../Disk/Disk.h"
```

Classes

- struct [DiskPartition](#)
- class [DiskManager](#)

Functions

- bool [operator==](#) (const [DiskPartition](#) *lhs, const [DiskPartition](#) &rhs)

6.10.1 Function Documentation

6.10.1.1 operator==()

```
bool operator== (
    const DiskPartition * lhs,
    const DiskPartition & rhs )
```

6.11 Filesystem/DaemonDependencies/File/File.cpp File Reference

```
#include "File.h"
```

6.12 Filesystem/DaemonDependencies/File/File.h File Reference

```
#include "../Types/types.h"
```

Classes

- class [File](#)

6.13 Filesystem/DaemonDependencies/FileSystem/FileSystem.cpp File Reference

```
#include "FileSystem.h"
```

Variables

- bool [EncryptionFlag](#) = false

6.13.1 Variable Documentation

6.13.1.1 EncryptionFlag

```
bool EncryptionFlag = false
```

6.14 Filesystem/DaemonDeps/FileSystem/FileSystem.h File Reference

```
#include "../Types/types.h"
#include "../Disk/Disk.h"
#include "../DiskManager/DiskManager.h"
#include "../PartitionManager/PartitionManager.h"
#include "../Trees/Trees.h"
```

Classes

- class [FileOpen](#)
- class [FileSystem](#)

6.15 Filesystem/DaemonDeps/PartitionManager/PartitionManager.cpp File Reference

```
#include "PartitionManager.h"
```

Variables

- bool [DEBUG](#) = false

6.15.1 Variable Documentation

6.15.1.1 [DEBUG](#)

```
bool DEBUG = false
```

6.16 Filesystem/DaemonDeps/PartitionManager/PartitionManager.h File Reference

```
#include "../Types/types.h"
#include "../DiskManager/DiskManager.h"
```

Classes

- class [PartitionManager](#)

6.17 Filesystem/DaemonDependancies/Trees/Trees.cpp File Reference

```
#include "Trees.h"
```

Functions

- bool `operator==` ([Index](#) &lhs, [Index](#) &rhs)
- bool `operator!=` ([Index](#) &lhs, [Index](#) &rhs)

6.17.1 Function Documentation

6.17.1.1 `operator!=()`

```
bool operator!= (  
    Index & lhs,  
    Index & rhs )
```

6.17.1.2 `operator==()`

```
bool operator== (  
    Index & lhs,  
    Index & rhs )
```

6.18 Filesystem/DaemonDependancies/Trees/Trees.h File Reference

```
#include "../Types/types.h"  
#include "../PartitionManager/PartitionManager.h"
```

Classes

- class [Attributes](#)
- class [Modification](#)
- class [Addition](#)
- class [Deletion](#)
- class [TreeObject](#)
- class [FileInfo](#)
- class [TagTree](#)
- class [RootTree](#)

Macros

- `#define` [DEFAULTOWNER](#) 1
- `#define` [DEFAULTPERMISSIONS](#) 0

6.18.1 Macro Definition Documentation

6.18.1.1 DEFAULTOWNER

```
#define DEFAULTOWNER 1
```

6.18.1.2 DEFAULTPERMISSIONS

```
#define DEFAULTPERMISSIONS 0
```

6.19 Filesystem/DaemonDependencies/Types/types.h File Reference

```
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <string>
#include <string.h>
#include <cstring>
#include <queue>
#include <vector>
#include <unordered_set>
#include <map>
#include <unordered_map>
#include <algorithm>
#include <utility>
#include <cstdlib>
#include <time.h>
#include "../SharedHeaders/Arboreal_Exceptions.h"
```

Classes

- struct [index](#)
- struct [rootSuperBlock](#)
- struct [tagTreeSuperBlock](#)
- struct [file_attributes](#)
- struct [finode](#)

Macros

- `#define` [MAXopen_fileS](#) 1000

Typedefs

- `typedef unsigned long` [BlkNumType](#)
- `typedef struct` [index](#) [Index](#)
- `typedef struct` [rootSuperBlock](#) [RootSuperBlock](#)
- `typedef struct` [tagTreeSuperBlock](#) [TagTreeSuperBlock](#)
- `typedef struct` [finode](#) [Finode](#)
- `typedef struct` [file_attributes](#) [FileAttributes](#)

Variables

- `bool` [DEBUG](#)

6.19.1 Macro Definition Documentation

6.19.1.1 MAXopen_fileS

```
#define MAXopen_fileS 1000
```

6.19.2 Typedef Documentation

6.19.2.1 BlkNumType

```
typedef unsigned long BlkNumType
```

6.19.2.2 FileAttributes

```
typedef struct file\_attributes FileAttributes
```

6.19.2.3 Finode

```
typedef struct finode Finode
```

6.19.2.4 Index

```
typedef struct index Index
```

6.19.2.5 RootSuperBlock

```
typedef struct rootSuperBlock RootSuperBlock
```

6.19.2.6 TagTreeSuperBlock

```
typedef struct tagTreeSuperBlock TagTreeSuperBlock
```

6.19.3 Variable Documentation

6.19.3.1 DEBUG

```
bool DEBUG
```

6.20 Filesystem/DaemonHeaders/daemon.h File Reference

```
#include "../SharedHeaders/Parser.h"  
#include "../SharedHeaders/DebugMessages.hpp"
```

Macros

- #define CREATEFILEDATA "Data/create_file_time.txt"
- #define CREATETAGDATA "Data/create_tag_time.txt"
- #define TAGSEARCHDATA "Data/tag_search_time.txt"
- #define FILESEARCHDATA "Data/file_search_time.txt"
- #define TAGFILEDATA "Data/tag_file_time.txt"
- #define RENAMETAGDATA "Data/rename_tag_time.txt"

Functions

- void [sig_caught](#) (int sig)
- void [save_to_disk](#) (void)
- std::string [command_to_string](#) (char *cmnd, int size)
- int [create_sock](#) (int timeout)
- void [set_socket_opt](#) (int daemon_sock, int sock_opt, int timeout)
- void [set_nonblocking](#) (int daemon_sock, int is_on)
- void [bind_socket](#) (int daemon_sock, struct sockaddr_in daemon_sockaddr, int timeout)
- void [listen_on_socket](#) (int daemon_sock, int backlog, int timeout)
- int [get_cmnd_id](#) (char *cmnd)
- std::string [get_partition](#) (char *cmnd)
- bool [is_number](#) (const char *str)
- std::string [pad_string](#) (std::string string, int size, char value)
- std::unordered_set< std::string > [get_set](#) (char *command, char delim)
- std::unordered_set< std::string > [get_set](#) (std::vector< std::string > vec)
- std::string [get_file_info](#) (File *file)
- std::string [get_file_info](#) (FileInfo *file)
- std::string [get_short_file_info](#) (FileInfo *file, int num_tags)
- std::vector< std::string > [serialize_fileinfo](#) (std::vector< FileInfo * > *fileinfo)
- void [execute](#) (int id, char *command, int fd, std::vector< std::string > &data)

Variables

- static const int [BACKLOG](#) = 10
- static const int [FLAG](#) = 0
- static const int [TIMEOUT](#) = 10
- static const int [TRUE](#) = 1
- static const int [FALSE](#) = 0
- static const int [PORT](#) = 70777
- static const int [MAX_COMMAND_SIZE](#) = 4096
- static const int [WRITE_CHANGES_WAIT](#) = 1
- static const bool [WILL_TIME](#) = false
- [DebugMessages](#) Debug
- fd_set [master_set](#)
- int [my_fid](#) = 999
- int [max_fid](#) = 0
- int [current_command_id](#) = 0
- bool [verbose](#) = false
- std::vector< std::string > [data](#)
- std::map< int, FileSystem * > [fd_fs_map](#)
- std::map< std::string, FileSystem * > [part_fs_map](#)
- std::map< std::string, unsigned int > [path_filedesc_map](#)
- Disk * [d](#) = 0
- DiskManager * [dm](#) = 0

6.20.1 Macro Definition Documentation

6.20.1.1 CREATEFILEDATA

```
#define CREATEFILEDATA "Data/create_file_time.txt"
```

Data file locations for timing tests

6.20.1.2 CREATETAGDATA

```
#define CREATETAGDATA "Data/create_tag_time.txt"
```

6.20.1.3 FILESEARCHDATA

```
#define FILESEARCHDATA "Data/file_search_time.txt"
```

6.20.1.4 RENAMETAGDATA

```
#define RENAMETAGDATA "Data/rename_tag_time.txt"
```

6.20.1.5 TAGFILEDATA

```
#define TAGFILEDATA "Data/tag_file_time.txt"
```

6.20.1.6 TAGSEARCHDATA

```
#define TAGSEARCHDATA "Data/tag_search_time.txt"
```

6.20.2 Function Documentation

6.20.2.1 bind_socket()

```
void bind_socket (
    int daemon_sock,
    struct sockaddr_in daemon_sockaddr,
    int timeout )
```

Bind socket to Port number

Parameters

<i>daemon_sock</i>	Daemon socket ID
<i>daemon_sockaddr</i>	Daemon socket address
<i>timeout</i>	Retry time length

Returns

VOID

6.20.2.2 command_to_string()

```
std::string command_to_string (
    char * cmd,
    int size )
```

Convert a command line interface command buffer into a string Used only for debugging puposes

Parameters

<i>cmd</i>	The command to be converted
<i>size</i>	The size of the command buffer

Returns

A std::string of the data within the buffer minus the first X bytes where X is the size of an integer

6.20.2.3 create_sock()

```
int create_sock (
    int timeout )
```

Create the daemon socket If socket creation fails, keep trying until you hit TIMEOUT

Parameters

<i>timeout</i>	Length of time in seconds which the function should attempt to create socket in the case of failure
----------------	---

Returns

An integer, socket ID

6.20.2.4 execute()

```
void execute (
    int id,
    char * command,
    int fd,
    std::vector< std::string > & data )
```

Execute the proper [File](#) System action based on a command id and command data Apply those actions to the correct FS object by using the fd_fs_map and the file descriptor passed

Parameters

<i>id</i>	The command to be executed's ID
<i>command</i>	The command to be executed's data
<i>fd</i>	The file descriptor that requested this command, the resulting data will be passed back to it and the changes will occur on the FS object that it is tied to

Returns

A std::vector of std::string's comprising the data returned by the command execution, this could be anything from an error message, to a success message, to a bunch of file information

6.20.2.5 get_cmnd_id()

```
int get_cmnd_id (
    char * cmnd )
```

Convert the first X characters in a 'Command Buffer' to an integer value X is the size of an integer

Parameters

<i>cmnd</i>	: The command buffer
-------------	----------------------

6.20.2.6 get_file_info() [1/2]

```
std::string get_file_info (
    File * file )
```

Returns a string containing some of a File's attributes

Parameters

<i>file</i>	A pointer to a File object containing the file's attributes
-------------	---

Returns

A std::string containing some of the file's attributes

6.20.2.7 get_file_info() [2/2]

```
std::string get_file_info (
    FileInfo * file )
```

Overloaded version of [get_file_info\(\)](#) which takes as a parameter a pointer to a [FileInfo](#) object rather than a [File](#) object

Parameters

<i>file</i>	A pointer to a FileInfo object containing the file's attributes
-------------	---

Returns

A std::string containing some of the file's attributes

6.20.2.8 get_partition()

```
std::string get_partition (
    char * cmd )
```

Get the partition a Command Line would like to connect to as a std::string rather than char*

Parameters

<i>cmd</i>	Command Line command buffer SPECIFICALLY, the one sent by start() in order to initiate the handshake process
------------	--

Returns

The partition name as a std::string

6.20.2.9 get_set() [1/2]

```
std::unordered_set<std::string> get_set (
    char * command,
    char delim )
```

Return a set representation of the data within a buffer sent by the Liaison process This is most commonly used in order to break down a string such as a path into its constituent parts using a character delimiter. For example, sending /tag1/tag2/tag3 to this function will return an unordered set containing [tag1,tag2,tag3]

Parameters

<i>command</i>	The command that needs to be split into parts
<i>delim</i>	The charachter that will be used as the delimiter marking where the function needs to split the command

Returns

An unordered set of the command contents minus the delimiting charachters

6.20.2.10 `get_set()` [2/2]

```
std::unordered_set<std::string> get_set (
    std::vector< std::string > vec )
```

Overloaded version of [get_set\(\)](#) which takes as its parameter a vector This function does not require a delimiter instead it just pushes the items from the vector into an unordered_set

Parameters

<i>vec</i>	The vector that needs to be converted into an unordered_set
------------	---

Returns

A std::unordered_set containing the vector's contents

6.20.2.11 `get_short_file_info()`

```
std::string get_short_file_info (
    FileInfo * file,
    int num_tags )
```

Get A shortened version of the file information The shortened file info conatains the file name, the first X tags were X = num_tags and the creation timestamp The number os tags is less than the value for num_tags, the actual number of tags will be used instead

Parameters

<i>file</i>	The file who's info we want
<i>num_tags</i>	Number of tags to display

Returns

A std::string containing the file information

6.20.2.12 `is_number()`

```
bool is_number (
    const char * str )
```

Returns true if a buffer sent by the Liaison process is a number or not Used to check when the Liaison has issued a new command rather than just more data for the previous command The buffer must first be converted into a string This function will only work with strings sent by the Liaison AFTER having completed a handshake, that is it is only valid for string constructed using the [Parser](#) and should NOT contain byte representations of numbers

Parameters

<i>str</i>	A string litteral
------------	-------------------

Returns

TRUE if the string is a number | FALSE otherwise

6.20.2.13 `listen_on_socket()`

```
void listen_on_socket (
    int daemon_sock,
    int backlog,
    int timeout )
```

Mark socket as open for receiving connections

Parameters

<i>daemon_sock</i>	Daemon socket ID
<i>backlog</i>	Number of connections that listen can queue up
<i>timeout</i>	Retry time length

Returns

VOID

6.20.2.14 `pad_string()`

```
std::string pad_string (
    std::string string,
    int size,
    char value )
```

Pad a std::string with a certain character to a certain length Pads from the back only

Parameters

<i>string</i>	String to be padded
<i>size</i>	Number of characters to append
<i>value</i>	Which character to pad the string with

Returns

The padded string

6.20.2.15 `save_to_disk()`

```
void save_to_disk (
    void )
```

Quit the Daemon; Delete data properly and signal other processes that need to be aware of the quit

This function is run by a thread that is detached from the main process

Returns

VOID

Periodically write all changes to disk Interval in between writes can be adjusted by changing the value of `WRITE↵_CHANGES_WAIT`

This function is run by a thread that is detached from the main process

Returns

VOID

6.20.2.16 `serialize_fileinfo()`

```
std::vector<std::string> serialize_fileinfo (
    std::vector< FileInfo *> * fileinfo )
```

Uses `get_file_info()` to return a vector of file info strings The `File` System functions which return file attributes, can return as many file attributes as there are files, typically this means that a vector of `FileInfo` pointers is returned, this function converts all of those `FileInfo` pointers into strings containing the respective file information

Parameters

<i>fileinfo</i>	A std::vector of <code>FileInfo</code> pointers
-----------------	---

Returns

A `std::vector` of `std::string`'s returned from [get_file_info\(\)](#)

6.20.2.17 set_nonblocking()

```
void set_nonblocking (
    int daemon_sock,
    int is_on )
```

Set socket to nonblocking mode in order to have continuous data streams this will also set any connecting sockets to nonblocking

Parameters

<i>daemon_sock</i>	Daemon socket ID
<i>is_on</i>	Whether nonblocking mode should be turned on or off (1 == ON 0 == OFF)

Returns

VOID

6.20.2.18 set_socket_opt()

```
void set_socket_opt (
    int daemon_sock,
    int sock_opt,
    int timeout )
```

Set socket options, this mainly allows the same socket address to be reused by the program when it starts up again. Normally socket addresses are one time use, this causes issues if you would like to quit the FS and then begin it again, so we must force a reuse

Parameters

<i>daemon_sock</i>	Daemon socket ID
<i>sock_opt</i>	Used by setsockopt() see man pages
<i>timeout</i>	Time in seconds the function should retry for if set options fails

Returns

VOID

6.20.2.19 sig_caught()

```
void sig_caught (
    int sig )
```

Catch either a user generated or system generated termination signal

Parameters

<i>sig</i>	The generated signals ID, passed to the function by the call to signal(), DO NOT supply this yourself, it is supplied automatically by the system.
------------	--

Returns

VOID

6.20.3 Variable Documentation

6.20.3.1 BACKLOG

```
const int BACKLOG = 10 [static]
```

Number of Connection Requests that the Server Can Queue

6.20.3.2 current_command_id

```
int current_command_id = 0
```

The Command Being Operated On's ID Some commands come in as lists and must be executed one part of the list at a time, in these cases it is paramount that the same command be executed. This value will not change until the daemon receives new data that begins with a number (the command ID)

6.20.3.3 d

```
Disk* d = 0
```

Disk Object

6.20.3.4 data

```
std::vector<std::string> data
```

The data the daemon has received

6.20.3.5 Debug

`DebugMessages` Debug

Handles Debugging

6.20.3.6 dm

`DiskManager*` dm = 0

Disk Manager

6.20.3.7 FALSE

```
const int FALSE = 0 [static]
```

Integer Boolean False

6.20.3.8 fd_fs_map

```
std::map<int, FileSystem*> fd_fs_map
```

Maps a file descriptor (socket) to a Partition

6.20.3.9 FLAG

```
const int FLAG = 0 [static]
```

Flag for recv()

6.20.3.10 master_set

```
fd_set master_set
```

Used for call to select() holds file descriptors

6.20.3.11 MAX_COMMAND_SIZE

```
const int MAX_COMMAND_SIZE = 4096 [static]
```

Maximum Buffer Size

6.20.3.12 max_fid

```
int max_fid = 0
```

Used by call to select() max_fid == 0 is FS socket

6.20.3.13 my_fid

```
int my_fid = 999
```

File system socket ID

6.20.3.14 part_fs_map

```
std::map<std::string, FileSystem*> part_fs_map
```

Maps a partition name to and FS object

6.20.3.15 path_filedesc_map

```
std::map<std::string, unsigned int> path_filedesc_map
```

Maps a pathname to a file descriptor (socket)

6.20.3.16 PORT

```
const int PORT = 70777 [static]
```

File System Port Number

6.20.3.17 TIMEOUT

```
const int TIMEOUT = 10 [static]
```

How Long Retries Should Take

6.20.3.18 TRUE

```
const int TRUE = 1 [static]
```

Integer Boolean True

6.20.3.19 verbose

```
bool verbose = false
```

Thread Synchronicity (No longer Used)

Thread Synchronicity (No longer Used)
More wordy return data for calls like 'find'

6.20.3.20 WILL_TIME

```
const bool WILL_TIME = false [static]
```

Whether or Not Timing Test Should Be Performed

6.20.3.21 WRITE_CHANGES_WAIT

```
const int WRITE_CHANGES_WAIT = 1 [static]
```

How Long To Wait Before Writing Changes

6.21 Filesystem/driver.cpp File Reference

```
#include "DaemonDependencies/FileSystem/FileSystem.h"
```

Functions

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

Variables

- bool [DEBUG](#) = false

6.21.1 Function Documentation

6.21.1.1 main()

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

6.21.2 Variable Documentation

6.21.2.1 DEBUG

```
bool DEBUG = false
```

6.22 Filesystem/timing.cpp File Reference

```
#include <chrono>
#include <ctime>
#include <fstream>
#include <string>
#include <stdlib.h>
#include <vector>
#include "DaemonDependancies/FileSystem/FileSystem.h"
```

Macros

- #define [CREATEFILEDATA](#) "Data/create_file_time.txt"
- #define [CREATETAGDATA](#) "Data/create_tag_time.txt"
- #define [TAGSEARCHDATA](#) "Data/tag_search_time.txt"
- #define [FILESEARCHDATA](#) "Data/file_search_time.txt"
- #define [TAGFILEDATA](#) "Data/tag_file_time.txt"
- #define [RENAMETAGDATA](#) "Data/rename_tag_time.txt"
- #define [STARTTUPDATA](#) "Data/startup_time.txt"

Functions

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

6.22.1 Macro Definition Documentation

6.22.1.1 CREATEFILEDATA

```
#define CREATEFILEDATA "Data/create_file_time.txt"
```

6.22.1.2 CREATETAGDATA

```
#define CREATETAGDATA "Data/create_tag_time.txt"
```

6.22.1.3 FILESEARCHDATA

```
#define FILESEARCHDATA "Data/file_search_time.txt"
```

6.22.1.4 RENAMETAGDATA

```
#define RENAMETAGDATA "Data/rename_tag_time.txt"
```

6.22.1.5 STARTTUPDATA

```
#define STARTTUPDATA "Data/startup_time.txt"
```

6.22.1.6 TAGFILEDATA

```
#define TAGFILEDATA "Data/tag_file_time.txt"
```

6.22.1.7 TAGSEARCHDATA

```
#define TAGSEARCHDATA "Data/tag_search_time.txt"
```

6.22.2 Function Documentation

6.22.2.1 main()

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

6.23 FSFormat/format.cpp File Reference

```
#include "../Filesystem/DaemonDependancies/Types/types.h"
```

Functions

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

6.23.1 Function Documentation

6.23.1.1 main()

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

6.24 LiaisonProcess/liaison.cpp File Reference

```
#include <stdlib.h>
#include <string>
#include <iostream>
#include <vector>
#include <errno.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <chrono>
#include <ctime>
#include <netinet/in.h>
#include <netdb.h>
#include <signal.h>
#include "../SharedHeaders/Parser.h"
#include "../SharedHeaders/DebugMessages.hpp"
#include "../SharedHeaders/Arboreal_Exceptions.h"
#include "../SharedHeaders/Print.h"
#include "LiaisonDependancies/liason_helper.hpp"
```

Functions

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

Variables

- static const int [Permissions](#) = 0666
- static const int [MaxBufferSize](#) = 4096
- static const int [SharedMemorySize](#) = 1
- static const int [Backlog](#) = 10
- static const int [Flag](#) = 0
- static const int [DaemonPort](#) = 70777
- static const int [Timeout](#) = 10
- static const bool [VERBOSE](#) = false
- [DebugMessages](#) Debug
- [Parser](#) * [Parser](#) = 0

6.24.1 Function Documentation

6.24.1.1 main()

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

6.24.2 Variable Documentation

6.24.2.1 Backlog

```
const int Backlog = 10 [static]
```

6.24.2.2 DaemonPort

```
const int DaemonPort = 70777 [static]
```

6.24.2.3 Debug

[DebugMessages](#) Debug

6.24.2.4 Flag

```
const int Flag = 0 [static]
```

6.24.2.5 MaxBufferSize

```
const int MaxBufferSize = 4096 [static]
```

6.24.2.6 Parser

```
Parser\* Parser = 0
```


6.24.2.7 Permissions

```
const int Permissions = 0666 [static]
```

6.24.2.8 SharedMemorySize

```
const int SharedMemorySize = 1 [static]
```

6.24.2.9 Timeout

```
const int Timeout = 10 [static]
```

6.24.2.10 VERBOSE

```
const bool VERBOSE = false [static]
```

6.25 LiaisonProcess/LiaisonDependencies/liason_helper.hpp File Reference

Macros

- `#define NEW_PLUS "n+"`

Functions

- void `clean` (int signal)
- void `bad_clean` (int signal)
- void `seg_fault` (int signal)
- int `get_cmnd_id` (const char *cmnd)
- std::string `get_command_string` (const char *cmnd, const int size)
- std::string `pad_string` (const std::string string, const int size, const char value)
- char * `get_shm_seg` (const key_t key, int &id)
- void `unat_shm` (const int shm_id, const char *shm)
- int `set_up_socket` (std::string server_sockpath, struct sockaddr_un &server_sockaddr)
- void `listen_for_client` (const int server_sock, const std::string server_sockpath)
- int `accept_client` (int server_sock, struct sockaddr_un &client_sockaddr, socklen_t length, std::string server_↵_sockpath)
- void `get_peername` (const int client_sock, const struct sockaddr_un &client_sockaddr, const int server_sock, const std::string server_sockpath)
- char * `recv_msg` (const int client_sock, const int size, const int flag, const int server_sock, const std::string server_sockpath, const std::string client_sockpath)
- void `send_response` (const int client_sock, const char *data, const int size, const int flag, const int server_↵_sock, const std::string server_sockpath, const std::string client_sockpath)
- void `shutdown` (const int liaison_fid, const int client_sock, const std::string client_sockpath, const int liaison_↵_sock, const std::string liaison_sockpath)
- int `create_daemon_sock` (const int client_sock, const std::string client_sockpath, const int liaison_sock, const std::string liaison_sockpath)
- void `connect_to_daemon` (int liaison_fid, struct sockaddr_in daemon_addr, const int client_sock, const std_↵::string client_sockpath, const int liaison_sock, const std::string liaison_sockpath)

6.25.1 Macro Definition Documentation

6.25.1.1 NEW_PLUS

```
#define NEW_PLUS "n+"
```

6.25.2 Function Documentation

6.25.2.1 accept_client()

```
int accept_client (
    int server_sock,
    struct sockaddr_un & client_sockaddr,
    socklen_t length,
    std::string server_sockpath )
```

Accept a connection request, returns the client socket's identifier

Parameters

<i>server_sock</i>	This server socket's identifier
<i>client_sockaddr</i>	A reference to a standard structure whose components I will not describe here and can be viewed in a Unix manual. Suffice it to say, it stores the socket type and the socket path. (Note that the "type" of the struct is <code>sockaddr_un</code> signifying that this is a unix domain socket). This will store the connecting client's information
<i>length</i>	The size of the <code>server_sockaddr</code> (This must be the size of the whole structure not just a single part and is most easily retrieved via a call to <code>sizeof()</code>)
<i>server_sockpath</i>	This server socket's pathname

Returns

An integer, the client socket's ID

Exceptions

<i>arboreal_liaison_error</i>	
---	--

6.25.2.2 bad_clean()

```
void bad_clean (
    int signal )
```

Remove socket files. Called when a system generated interrupt is caught

Parameters

<i>signal</i>	The sytem signal that was received
---------------	------------------------------------

Returns

VOID

6.25.2.3 clean()

```
void clean (
    int signal )
```

Remove socket files. Called when a user generated interrupt is caught

Parameters

<i>signal</i>	The sytem signal that was received
---------------	------------------------------------

Returns

VOID

6.25.2.4 connect_to_daemon()

```
void connect_to_daemon (
    int liaison_fid,
    struct sockaddr_in daemon_addr,
    const int client_sock,
    const std::string client_sockpath,
    const int liaison_sock,
    const std::string liaison_sockpath )
```

Connect to the [File](#) System, everything after *liaison_fid* and daemon address is used in case of failure in order to ensure proper cleanup

Parameters

<i>liaison_fid</i>	The ID of the [Liaison -> File System] Socket
<i>daemon_addr</i>	A sockaddr_in structure to store the filesystem daemon info
<i>client_sock</i>	ID of the [Command Line -> Liaison] Socket
<i>client_sockpath</i>	The [Command Line -> Liaison] Socket's path
<i>liaison_sock</i>	ID of the [Liaison -> Command Line] Socket
<i>liaison_sockpath</i>	The [Liaison -> Command Line] Socket's path

Returns

VOID

6.25.2.5 create_daemon_sock()

```
int create_daemon_sock (
    const int client_sock,
    const std::string client_sockpath,
    const int liaison_sock,
    const std::string liaison_sockpath )
```

Create a new socket for the [Liaison -> [File System](#)] connection. All parameters passed are purely in case of failure so that proper cleanup can be done

Parameters

<i>client_sock</i>	ID of the [Command Line -> Liaison] Socket
<i>client_sockpath</i>	The [Command Line -> Liaison] Socket's path
<i>liaison_sock</i>	ID of the [Liaison -> Command Line] Socket
<i>liaison_sockpath</i>	The [Liaison -> Command Line] Socket's path

Returns

The created socket's ID

6.25.2.6 get_cmnd_id()

```
int get_cmnd_id (
    const char * cmnd )
```

Convert the first X charachters of a command buffer into an integer. X is the size of an integer

Parameters

<i>cmnd</i>	A charachter array created by the Command Line Interface
-------------	--

Returns

An integer representing the ID of cmnd

6.25.2.7 get_command_string()

```
std::string get_command_string (
    const char * cmd,
    const int size )
```

Returns a string representation of a character array created with the Command Line

Parameters

<i>cmd</i>	A character array created via the command line process
<i>size</i>	The size of the character array (usually equal to MaxBufferSize)

Returns

A std::string representation of the command minus the first X bytes (Where X is the size of an integer) that store the command ID

6.25.2.8 get_peername()

```
void get_peername (
    const int client_sock,
    const struct sockaddr_un & client_sockaddr,
    const int server_sock,
    const std::string server_sockpath )
```

Retrieve a accepted client's information for use in send/receive functionality

Parameters

<i>client_sock</i>	The client socket's identifier
<i>client_sockaddr</i>	A reference to a standard structure whose components I will not describe here and can be viewed in a Unix manual. Suffice it to say, it stores the socket type and the socket path. (Note that the "type" of the struct is sockaddr_un signifying that this is a unix domain socket). This will store the connecting client's information
<i>server_sock</i>	This server socket's identifier
<i>server_sockpath</i>	This server socket's pathname

Returns

VOID

Exceptions

arboreal_liaison_error	
--	--

6.25.2.9 `get_shm_seg()`

```
char* get_shm_seg (
    const key_t key,
    int & id )
```

Request and attach to, a shared memory segment with a specific key. The shared memory segment will be used to synchronize the command line interface and this liason process. (Note that the only difference between the `shmget()` of the Command Line Process and the Liaison Process is the lack of `IPC_CREAT` as one of the flags passed. `IPC_CREAT` will create a new fragment leaving it off only)

Parameters

<i>key</i>	The unique key required to access the specific shared memory segment This is passed as a parameter from the CLI to the Liason process via main() arguments
<i>id</i>	A reference to an integer in which to store the shared memory id that <code>shmget()</code> returns

Returns

A pointer to the shared memory segment

Exceptions

arboreal_liaison_error	
--	--

6.25.2.10 `listen_for_client()`

```
void listen_for_client (
    const int server_sock,
    const std::string server_sockpath )
```

Mark the server socket as open for buisness (i.e. capable of accepting connections) The Server can queue up X number of connection requests were X = Backlog

Parameters

<i>server_sock</i>	This server socket's identifier
<i>server_sockpath</i>	This server socket's pathname

Returns

VOID

Exceptions

arboreal_liaison_error	
--	--

6.25.2.11 pad_string()

```
std::string pad_string (
    const std::string string,
    const int size,
    const char value )
```

Pad the end of a std::string with X charachters where X is a chosen value and the charachter is also chosen.

Parameters

<i>string</i>	String to be padded
<i>size</i>	Number of charachters to pad the string with
<i>value</i>	What charachter to pad the string with

Returns

The padded string

6.25.2.12 recv_msg()

```
char* recv_msg (
    const int client_sock,
    const int size,
    const int flag,
    const int server_sock,
    const std::string server_sockpath,
    const std::string client_sockpath )
```

Receive a message from an accepted socket. (Note that the client/server pathnames and the server socket id are only used when an exception is thrown, in order to correctly close the socket)

Parameters

<i>client_sock</i>	The client socket's identifier
<i>size</i>	The size of the message to be received
<i>flag</i>	Any flags for the recv() function (see 'man recv')
<i>server_sock</i>	This server socket's identifier
<i>server_sockpath</i>	This server socket's pathname
<i>client_sockpath</i>	The client socket's pathname

Returns

A charachter array comprising the message received

Exceptions

<i>arboreal_liaison_error</i>	
---	--

6.25.2.13 `seg_fault()`

```
void seg_fault (
    int signal )
```

Remove socket files. Called whenever a SISEGIV is thrown

Parameters

<i>signal</i>	The sytem signal that was received
---------------	------------------------------------

Returns

VOID

6.25.2.14 `send_response()`

```
void send_response (
    const int client_sock,
    const char * data,
    const int size,
    const int flag,
    const int server_sock,
    const std::string server_sockpath,
    const std::string client_sockpath )
```

Send a response to an accepted socket (Note that the client/server pathnames and the server socket id are only used when an exception is thrown in order to correctly close the socket)

Parameters

<i>client_sock</i>	The client socket's identifier
<i>size</i>	The size of the message to be received
<i>flag</i>	Any flags for the <code>recv()</code> function (see 'man recv')
<i>server_sock</i>	This server socket's identifier
<i>server_sockpath</i>	This server socket's pathname
<i>client_sockpath</i>	The client socket's pathname

Returns

VOID

Exceptions

arboreal_liaison_error	
--	--

6.25.2.15 set_up_socket()

```
int set_up_socket (
    std::string server_sockpath,
    struct sockaddr_un & server_sockaddr )
```

Set up a server socket to receive incoming connections

@ param server_sockpath: The pathname fo the server's socket (In this case the pathame will not be static, as each CLI process will fork its own Liaison process, therefore the server pathname is passed as an argument to the Liaison process' [main\(\)](#) function)

Parameters

<i>server_sockaddr</i>	A reference to a standard structure whose components I will not describe here and can be viewed in a Unix manual. Suffice it to say, it stores the socket type and the socket path. (Note that the "type" of the struct is <code>sockaddr_un</code> signifying that this is a unix domain socket)
------------------------	---

Returns

An integer, the socket ID

Exceptions

arboreal_liaison_error	
--	--

6.25.2.16 shutdown()

```
void shutdown (
    const int liaison_fid,
    const int client_sock,
    const std::string client_sockpath,
    const int liaison_sock,
    const std::string liaison_sockpath )
```

Perform proper cleanup when quit command or interrupt signal is received. This mainly involves closing all open connections and properly deleting any socket files on the system and finally, exiting the process via a call to `exit()`.

Parameters

<i>liaison_fid</i>	ID of the [Liaison -> File System] Socket
<i>client_sock</i>	ID of the [Command Line -> Liaison] Socket
<i>client_sockpath</i>	The [Command Line -> Liaison] Socket's path
<i>liaison_sock</i>	ID of the [Liaison -> Command Line] Socket
<i>liaison_sockpath</i>	The [Liaison -> Command Line] Socket's path

Exceptions

arboreal_liaison_error	
--	--

6.25.2.17 unat_shm()

```
void unat_shm (
    const int shm_id,
    const char * shm )
```

Un-attach a shared memory segment from this process. (Process will not be able to access the shared memory segment until it is reattached)

Parameters

<i>shm_↔_id</i>	The id of the shared memory segment that will be detached
<i>shm</i>	The actual pointer to the shared memory segment

Returns

VOID

Exceptions

arboreal_liaison_error	
--	--

6.26 README.md File Reference

6.27 SharedCPPFiles/Arboreal_Exceptions.cpp File Reference

```
#include "../SharedHeaders/Arboreal_Exceptions.h"
```

6.28 SharedCPPFiles/Parser.cpp File Reference

```
#include "../SharedHeaders/Parser.h"
```

6.29 SharedHeaders/Arboreal_Exceptions.h File Reference

```
#include <string>
#include <stdexcept>
#include "ErrorCodes.h"
```

Classes

- class [arboreal_exception](#)
- class [arboreal_runtime_error](#)
- class [arboreal_cli_error](#)
- class [arboreal_liaison_error](#)
- class [arboreal_daemon_error](#)
- class [disk_error](#)
- class [tag_error](#)
- class [file_error](#)
- class [arboreal_logic_error](#)
- class [invalid_arg](#)

6.30 SharedHeaders/CommandCodes.h File Reference

Variables

- static const int [FIND_TS](#) = 400
- static const int [FIND_FS](#) = 401
- static const int [NEW_FP](#) = 300
- static const int [NEW_TS](#) = 301
- static const int [NEW_FS](#) = 302
- static const int [DEL_FP](#) = 500
- static const int [DEL_TS](#) = 501
- static const int [DEL_FS](#) = 502
- static const int [OPEN_FP](#) = 200
- static const int [OPEN_F](#) = 201
- static const int [CLOSE_FP](#) = 600
- static const int [CLOSE_F](#) = 601
- static const int [RNAME_FP](#) = 100
- static const int [RNAME_TS](#) = 101
- static const int [RNAME_FS](#) = 102
- static const int [ATTR_FP](#) = 700
- static const int [ATTR_FS](#) = 701
- static const int [MERG_1_1](#) = 801
- static const int [MERG_M_1](#) = 802
- static const int [TAG_FP](#) = 900

- static const int [TAG_FS](#) = 901
- static const int [UTAG_FP](#) = 1000
- static const int [UTAG_FS](#) = 1001
- static const int [CD_ABS](#) = 2222
- static const int [CD_RLP](#) = 1112
- static const int [READ_XP](#) = 3000
- static const int [READ_FP](#) = 3300
- static const int [READ_XCWD](#) = 3001
- static const int [READ_FCWD](#) = 3002
- static const int [WRITE_FP](#) = 4000
- static const int [APPND_FP](#) = 4400
- static const int [WRITE_XFPF](#) = 4440
- static const int [APPND_XFPF](#) = 4444
- static const int [WRITE_FCWD](#) = 4001
- static const int [APPND_FCWD](#) = 4002
- static const int [WRITE_XFCWDF](#) = 4003
- static const int [APPND_XFCWDF](#) = 4004
- static const int [CPY_FP](#) = 6000
- static const int [CPY_FCWD](#) = 6001
- static const int [QUIT](#) = 999
- static const int [FTL_ERR](#) = 9999
- static const int [HANDSHK](#) = 0
- static const int [UHELP](#) = 10001
- static const int [UQUIT](#) = 10002
- static const int [UFIND](#) = 10003
- static const int [UNEW](#) = 10004
- static const int [UDEL](#) = 10005
- static const int [UOPEN](#) = 10006
- static const int [UCLOSE](#) = 10007
- static const int [URNAME](#) = 10008
- static const int [UATTR](#) = 10009
- static const int [UMERG](#) = 10010
- static const int [UTAG](#) = 10011
- static const int [UUTAG](#) = 10012
- static const int [UCD](#) = 10013
- static const int [UREAD](#) = 10014
- static const int [UWRITE](#) = 10015
- static const int [UCOPY](#) = 10016

6.30.1 Variable Documentation

6.30.1.1 APPND_FCWD

```
const int APPND_FCWD = 4002 [static]
```

Append To [File](#) (In Current Working Directory)

6.30.1.2 APPND_FP

```
const int APPND_FP = 4400 [static]
```

Append To [File](#) (Must Supply [File](#) Path)

6.30.1.3 APPND_XFCWDF

```
const int APPND_XFCWDF = 4004 [static]
```

Append X Bytes From [File](#) To [File](#) (In Current Directory)

6.30.1.4 APPND_XFPF

```
const int APPND_XFPF = 4444 [static]
```

Append X Bytes From [File](#) To [File](#) (Must Supply [File](#) Paths)

6.30.1.5 ATTR_FP

```
const int ATTR_FP = 700 [static]
```

Get [File Attributes](#) (Must Supply [File](#) Path)

6.30.1.6 ATTR_FS

```
const int ATTR_FS = 701 [static]
```

Get [File Attributes](#) (In Current Working Directory)

6.30.1.7 CD_ABS

```
const int CD_ABS = 2222 [static]
```

Change Directory (Absolute Path)

6.30.1.8 CD_RLP

```
const int CD_RLP = 1112 [static]
```

Change Directory (Relative Path)

6.30.1.9 CLOSE_F

```
const int CLOSE_F = 601 [static]
```

Close A [File](#) (In Current Working Directory)

6.30.1.10 CLOSE_FP

```
const int CLOSE_FP = 600 [static]
```

Close A [File](#) (Must Supply [File](#) Path)

6.30.1.11 CPY_FCWD

```
const int CPY_FCWD = 6001 [static]
```

Copy Contents Of One [File](#) To Another (Overwrites [File](#); In Current Working Directory)

6.30.1.12 CPY_FP

```
const int CPY_FP = 6000 [static]
```

Copy Contents Of One [File](#) To Another (Overwrites [File](#); Must Supply [File](#) Paths)

6.30.1.13 DEL_FP

```
const int DEL_FP = 500 [static]
```

Delete A [File](#) (Must Supply [File](#) Path)

6.30.1.14 DEL_FS

```
const int DEL_FS = 502 [static]
```

Delete A [File\(s\)](#) (In Current Working Directory)

6.30.1.15 DEL_TS

```
const int DEL_TS = 501 [static]
```

Delete A Tag(s) (Must Be Empty)

6.30.1.16 FIND_FS

```
const int FIND_FS = 401 [static]
```

Find Files By Name

6.30.1.17 FIND_TS

```
const int FIND_TS = 400 [static]
```

Find Files By Tag

6.30.1.18 FTL_ERR

```
const int FTL_ERR = 9999 [static]
```

Fatal Error

6.30.1.19 HANDSHK

```
const int HANDSHK = 0 [static]
```

Handshake

6.30.1.20 MERG_1_1

```
const int MERG_1_1 = 801 [static]
```

Merge One Tag Into Another

6.30.1.21 MERG_M_1

```
const int MERG_M_1 = 802 [static]
```

Merge Many Tags Into One

6.30.1.22 NEW_FP

```
const int NEW_FP = 300 [static]
```

Create A New [File](#) From Anywhere (Must Supply [File](#) Path)

6.30.1.23 NEW_FS

```
const int NEW_FS = 302 [static]
```

Create 1 Or More New Files Within The Current Working Directory

6.30.1.24 NEW_TS

```
const int NEW_TS = 301 [static]
```

Create 1 Or More New Tags

6.30.1.25 OPEN_F

```
const int OPEN_F = 201 [static]
```

Open A [File](#) (In Current Working Directory)

6.30.1.26 OPEN_FP

```
const int OPEN_FP = 200 [static]
```

Open A [File](#) For Operations (Must Supply [File](#) Path)

6.30.1.27 QUIT

```
const int QUIT = 999 [static]
```

Quit Interface

6.30.1.28 READ_FCWD

```
const int READ_FCWD = 3002 [static]
```

Read Whole [File](#) (In Current Working Directory)

6.30.1.29 READ_FP

```
const int READ_FP = 3300 [static]
```

Read Whole [File](#) (Must Supply Path)

6.30.1.30 READ_XCWD

```
const int READ_XCWD = 3001 [static]
```

Read X Bytes From [File](#) (In Current Working Directory)

6.30.1.31 READ_XP

```
const int READ_XP = 3000 [static]
```

Read X Bytes From [File](#) (Must Supply Path)

6.30.1.32 RNAME_FP

```
const int RNAME_FP = 100 [static]
```

Rename [File\(s\)](#) (Must Supply [File](#) Path)

6.30.1.33 RNAME_FS

```
const int RNAME_FS = 102 [static]
```

Rename [File\(s\)](#) (In Current Working Directory)

6.30.1.34 RNAME_TS

```
const int RNAME_TS = 101 [static]
```

Rename Tag(s)

6.30.1.35 TAG_FP

```
const int TAG_FP = 900 [static]
```

Tag [File](#) (Must Supply [File](#) Path)

6.30.1.36 TAG_FS

```
const int TAG_FS = 901 [static]
```

Tag [File\(s\)](#) (In Current Working Directory)

6.30.1.37 UATTR

```
const int UATTR = 10009 [static]
```

Usage [Attributes](#)

6.30.1.38 UCD

```
const int UCD = 10013 [static]
```

Usage Change Directory

6.30.1.39 UCLOSE

```
const int UCLOSE = 10007 [static]
```

Usage Close

6.30.1.40 UCOPY

```
const int UCOPY = 10016 [static]
```

Usage Copy

6.30.1.41 UDEL

```
const int UDEL = 10005 [static]
```

Usage Delete

6.30.1.42 UFIND

```
const int UFIND = 10003 [static]
```

Usage Find

6.30.1.43 UHELP

```
const int UHELP = 10001 [static]
```

Usage Help

6.30.1.44 UMERG

```
const int UMERG = 10010 [static]
```

Usage Merge

6.30.1.45 UNEW

```
const int UNEW = 10004 [static]
```

Usage New

6.30.1.46 UOPEN

```
const int UOPEN = 10006 [static]
```

Usage Open

6.30.1.47 UQUIT

```
const int UQUIT = 10002 [static]
```

Usage Quit

6.30.1.48 UREAD

```
const int UREAD = 10014 [static]
```

Usage Read

6.30.1.49 URNAME

```
const int URNAME = 10008 [static]
```

Usage Rename

6.30.1.50 UTAG

```
const int UTAG = 10011 [static]
```

Usage Tag

6.30.1.51 UTAG_FP

```
const int UTAG_FP = 1000 [static]
```

Untag [File](#) (Must Supply [File](#) Path)

6.30.1.52 UTAG_FS

```
const int UTAG_FS = 1001 [static]
```

Untag [File\(s\)](#) (In Current Working Directory)

6.30.1.53 UUTAG

```
const int UUTAG = 10012 [static]
```

Usage Untag

6.30.1.54 UWRITE

```
const int UWRITE = 10015 [static]
```

Usage Write

6.30.1.55 WRITE_FCWD

```
const int WRITE_FCWD = 4001 [static]
```

Write To [File](#) (In Current Working Directory)

6.30.1.56 WRITE_FP

```
const int WRITE_FP = 4000 [static]
```

Write To [File](#) (Must Supply [File](#) Path)

6.30.1.57 WRITE_XFCWDF

```
const int WRITE_XFCWDF = 4003 [static]
```

Write X Bytes From [File](#) To [File](#) (In Current Working Directory)

6.30.1.58 WRITE_XFPF

```
const int WRITE_XFPF = 4440 [static]
```

Write X Bytes From [File](#) To [File](#) (Must Supply [File](#) Paths)

6.31 SharedHeaders/CommandValidation.h File Reference

```
#include <regex>
#include "CommandCodes.h"
```

Functions

- std::regex [change_dir](#) ("cd /[0-9a-zA-Z_]*+")
- std::regex [change_dir_rl](#) ("cd \\/[0-9a-zA-Z_]*+")
- std::regex [usage_help](#) ("--help")
- std::regex [usage_quit](#) ("--quit")
- std::regex [usage_find](#) ("--find")
- std::regex [usage_new](#) ("--new")
- std::regex [usage_delete](#) ("--delete")
- std::regex [usage_open](#) ("--open")
- std::regex [usage_close](#) ("--close")
- std::regex [usage_rename](#) ("--rename")
- std::regex [usage_attr](#) ("--attr")
- std::regex [usage_merge](#) ("--merge")
- std::regex [usage_tag](#) ("--tag")
- std::regex [usage_untag](#) ("--untag")
- std::regex [usage_cd](#) ("--cd")
- std::regex [usage_read](#) ("--read")
- std::regex [usage_write](#) ("--write")
- std::regex [usage_copy](#) ("--copy")
- std::regex [help_1](#) ("-h --help")
- std::regex [help_2](#) ("-h --quit")
- std::regex [help_3](#) ("-h --find")
- std::regex [help_4](#) ("-h --new")
- std::regex [help_5](#) ("-h --delete")
- std::regex [help_6](#) ("-h --open")
- std::regex [help_7](#) ("-h --close")
- std::regex [help_8](#) ("-h --rename")
- std::regex [help_9](#) ("-h --attr")
- std::regex [help_10](#) ("-h --merge")
- std::regex [help_11](#) ("-h --tag")
- std::regex [help_12](#) ("-h --untag")
- std::regex [help_13](#) ("-h --cd")
- std::regex [help_14](#) ("-h --read")
- std::regex [help_15](#) ("-h --write")
- std::regex [help_16](#) ("-h --copy")
- std::regex [find_tags](#) ("find -t [\\,0-9a-zA-Z_\\]*")
- std::regex [find_files](#) ("find -f \\([0-9a-zA-Z_]+\\)(\\[a-zA-Z_]+)?(\\([0-9a-zA-Z_]+\\)(\\[0-9a-zA-Z_]+)?)*\\")
- std::regex [new_tags](#) ("new -t \\([0-9a-zA-Z_]+\\)(\\[0-9a-zA-Z_]+)*\\")

- `std::regex new_files` ("new -f \([0-9a-zA-Z_]+\)(\[a-zA-Z]+\)?,\([0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)?*\)")
- `std::regex new_file` ("new \([0-9a-zA-Z_]+\)+[0-9a-zA-Z_]+\)(\[a-zA-Z]+\)?")
- `std::regex del_tags` ("delete -t \([0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)*\)")
- `std::regex del_files` ("delete -f \([0-9a-zA-Z_]+\)(\[a-zA-Z]+\)?,\([0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)?*\)")
- `std::regex del_file` ("delete \([0-9a-zA-Z_]+\)*\[0-9a-zA-Z]+\)(\[a-zA-Z]+\)?")
- `std::regex open_files` ("open (-r|-w|-x) \([0-9a-zA-Z_]+\)+[0-9a-zA-Z_]+\)(\[a-zA-Z]+\)?")
- `std::regex open_file_cd` ("open (-r|-w|-x) [0-9a-zA-Z_]+\)(\[a-zA-Z]+\)?")
- `std::regex close_files` ("close \([0-9a-zA-Z_]+\)+[0-9a-zA-Z_]+\)(\[a-zA-Z]+\)?")
- `std::regex close_file_cd` ("close [0-9a-zA-Z_]+\)(\[a-zA-Z]+\)?")
- `std::regex rename_tags` ("rename -t \([0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)*\ => \([0-9a-zA-Z_]+\)(\[a-zA-Z_↵\]+\)*\)")
- `std::regex rename_files` ("rename \([0-9a-zA-Z_]+\)*\[0-9a-zA-Z]+\)(\[a-zA-Z]+\)? => [0-9a-zA-Z_]+\)(\[a-zA-Z_↵\]+\)?")
- `std::regex rename_file_cd` ("rename [0-9a-zA-Z_]+\)(\[a-zA-Z]+\)? => [0-9a-zA-Z_]+\)(\[a-zA-Z]+\)?")
- `std::regex get_attrs` ("attr \([0-9a-zA-Z_]+\)*\[0-9a-zA-Z_]+\)(\[a-zA-Z]+\)?")
- `std::regex get_attr_cd` ("attr [0-9a-zA-Z_]+\)(\[a-zA-Z]+\)?")
- `std::regex merge_1_1` ("merge [0-9a-zA-Z_]+ -> [0-9a-zA-Z_]+")
- `std::regex merge_m_1` ("merge \([0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)*\ -> [0-9a-zA-Z_]+\)[a-zA-Z_0-9]*")
- `std::regex add_tags` ("tag \([0-9a-zA-Z_]+\)*\[0-9a-zA-Z_]+\)(\[a-zA-Z]+\)? \> \([0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)*\)")
- `std::regex tag_files` ("tag \([0-9a-zA-Z_]+\)(\[a-zA-Z]+\)?,\([0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)?*\) \> \([0-9a-zA-Z_↵\]+\)(\[a-zA-Z_]+\)*\)")
- `std::regex untag_file` ("untag \([0-9a-zA-Z_]+\)*\[0-9a-zA-Z_]+\)(\[a-zA-Z]+\)? \> \([0-9a-zA-Z_]+\)(\[a-zA-Z_↵\]+\)*\)")
- `std::regex untag_files` ("untag \([0-9a-zA-Z_]+\)(\[a-zA-Z]+\)?,\([0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)?*\) \> \([0-9a-↵\]+)\([0-9a-zA-Z_]+\)*\)")
- `std::regex read_x_path` ("read \([0-9a-zA-Z_]+\)+[0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)? -b [0-9]+")
- `std::regex read_x_cwd` ("read [0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)? -b [0-9]+")
- `std::regex read_path` ("read \([0-9a-zA-Z_]+\)+[0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)?")
- `std::regex read_cwd` ("read [0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)?")
- `std::regex write_x_path` ("write \([0-9a-zA-Z_]+\)+[0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)? -b [0-9]+")
- `std::regex write_x_cwd` ("write [0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)? -b [0-9]+")
- `std::regex write_path` ("")
- `std::regex write_cwd` ("")
- `std::regex append_path` ("")
- `std::regex append_x_path` ("")
- `std::regex append_cwd` ("")
- `std::regex append_x_cwd` ("")
- `std::regex copy_path` ("")
- `std::regex copy_cwd` ("")
- `int check_command` (std::string command)
- `int check_usage` (std::string input)
- `int check_help` (std::string input)

6.31.1 Function Documentation

6.31.1.1 add_tags()

```
std::regex add_tags (
    "tag \([0-9a-zA-Z_]+\)*\[0-9a-zA-Z_]+\)(\[a-zA-Z]+\)? \> \([0-9a-zA-Z_]+\)(\[a-zA-Z_]+\)*\)" )
```

Regex For File System "tag" Commands

6.31.1.2 append_cwd()

```
std::regex append_cwd (
    "" )
```

6.31.1.3 append_path()

```
std::regex append_path (
    "" )
```

6.31.1.4 append_x_cwd()

```
std::regex append_x_cwd (
    "" )
```

6.31.1.5 append_x_path()

```
std::regex append_x_path (
    "" )
```

6.31.1.6 change_dir()

```
std::regex change_dir (
    "cd ([0-9a-zA-Z_]*)" )
```

General Regular Expression

- Relative Directory Change
- Absolute Directory Change
- Identifying Correct Help Command Syntax

6.31.1.7 change_dir_rl()

```
std::regex change_dir_rl (
    "cd \\[0-9a-zA-Z_\\+\\+)" )
```

6.31.1.8 check_command()

```
int check_command (
    std::string command )
```

6.31.1.9 check_help()

```
int check_help (
    std::string input )
```

6.31.1.10 check_usage()

```
int check_usage (
    std::string input )
```

6.31.1.11 close_file_cd()

```
std::regex close_file_cd (
    "close +(\-zA-Z)+?" [0-9a-zA-Z] )
```

6.31.1.12 close_files()

```
std::regex close_files (
    "close ([0-9a-zA-Z_]+)((\a-zA-Z_)+?" [0-9a-zA-Z_] )
```

Regex For [File](#) System "close" Commands

6.31.1.13 copy_cwd()

```
std::regex copy_cwd (
    "" )
```

6.31.1.14 copy_path()

```
std::regex copy_path (
    "" )
```

Regex For [File](#) System "copy" Commands Not Yet Available

6.31.1.15 del_file()

```
std::regex del_file (
    "delete (/ [0-9a-zA-Z_]*)+ (\-zA-Z)+) ?" [0-9a-zA-Z] )
```

6.31.1.16 del_files()

```
std::regex del_files (
    "delete -f \0-9a-zA-Z_]+) (\-zA-Z)+) ? (, ([0-9a-zA-Z_]+) (\-9a-zA-Z_]+) ?)*\ " )
```

6.31.1.17 del_tags()

```
std::regex del_tags (
    "delete -t \0-9a-zA-Z_]+) (, [0-9a-zA-Z_]+)*\ " )
```

Regex For [File](#) System "delete" Commands

6.31.1.18 find_files()

```
std::regex find_files (
    "find -f \0-9a-zA-Z_]+) (\-zA-Z)+) ? (, ([0-9a-zA-Z_]+) (\-9a-zA-Z_]+) ?)*\ " )
```

6.31.1.19 find_tags()

```
std::regex find_tags (
    "find -t *" [\\, 0-9a-zA-Z_\\] )
```

Regex For [File](#) System "find" Commands

6.31.1.20 get_attr_cd()

```
std::regex get_attr_cd (
    "attr +(\-zA-Z)+) ?" [0-9a-zA-Z] )
```

6.31.1.21 get_attrs()

```
std::regex get_attrs (
    "attr (/ [0-9a-zA-Z_]*)+ (\-zA-Z)+) ?" [0-9a-zA-Z] )
```

Regex For [File](#) System "attr" Commands

6.31.1.22 help_1()

```
std::regex help_1 (
    "-h --help" )
```

6.31.1.23 help_10()

```
std::regex help_10 (
    "-h --merge" )
```

6.31.1.24 help_11()

```
std::regex help_11 (
    "-h --tag" )
```

6.31.1.25 help_12()

```
std::regex help_12 (
    "-h --untag" )
```

6.31.1.26 help_13()

```
std::regex help_13 (
    "-h --cd" )
```

6.31.1.27 help_14()

```
std::regex help_14 (
    "-h --read" )
```

6.31.1.28 help_15()

```
std::regex help_15 (
    "-h --write" )
```

6.31.1.29 help_16()

```
std::regex help_16 (
    "-h --copy" )
```

6.31.1.30 help_2()

```
std::regex help_2 (
    "-h --quit" )
```

6.31.1.31 help_3()

```
std::regex help_3 (
    "-h --find" )
```

6.31.1.32 help_4()

```
std::regex help_4 (
    "-h --new" )
```

6.31.1.33 help_5()

```
std::regex help_5 (
    "-h --delete" )
```

6.31.1.34 help_6()

```
std::regex help_6 (
    "-h --open" )
```

6.31.1.35 help_7()

```
std::regex help_7 (
    "-h --close" )
```

6.31.1.36 help_8()

```
std::regex help_8 (
    "-h --rename" )
```

6.31.1.37 help_9()

```
std::regex help_9 (
    "-h --attr" )
```

6.31.1.38 merge_1_1()

```
std::regex merge_1_1 (
    "merge + -> +" [0-9a-zA-Z_] [0-9a-zA-Z_] )
```

Regex For [File](#) System "merge" Commands Not Yet Available

6.31.1.39 merge_m_1()

```
std::regex merge_m_1 (
    "merge \0-9a-zA-Z_+)(,[0-9a-zA-Z_+)*\> +" [0-9a-zA-Z_] [a-zA-Z_0-9] )
```

6.31.1.40 new_file()

```
std::regex new_file (
    "new (/ [0-9a-zA-Z_]+)((\a-zA-Z_)?)" [0-9a-zA-Z_] )
```

6.31.1.41 new_files()

```
std::regex new_files (
    "new -f \0-9a-zA-Z_+)(\a-zA-Z_+)?(,[0-9a-zA-Z_+)(\a-zA-Z_+)?)*\ " )
```

6.31.1.42 new_tags()

```
std::regex new_tags (
    "new -t \0-9a-zA-Z_+)(,[0-9a-zA-Z_+)*\ " )
```

Regex For [File](#) System "new" Commands

6.31.1.43 open_file_cd()

```
std::regex open_file_cd (
    "open (-r|-w|-x) +(\-zA-Z)?" [0-9a-zA-Z] )
```

6.31.1.44 open_files()

```
std::regex open_files (
    "open (-r|-w|-x) ([0-9a-zA-Z_]+)((\a-zA-Z_)+)?" [0-9a-zA-Z_] )
```

Regex For [File](#) System "open" Commands

6.31.1.45 read_cwd()

```
std::regex read_cwd (
    "read +((\a-zA-Z_)+)?" [0-9a-zA-Z_] )
```

6.31.1.46 read_path()

```
std::regex read_path (
    "read ([0-9a-zA-Z_]+)((\a-zA-Z_)+)?" [0-9a-zA-Z_] )
```

6.31.1.47 read_x_cwd()

```
std::regex read_x_cwd (
    "read +((\a-zA-Z_)+)? -b +" [0-9a-zA-Z_] [0-9] )
```

6.31.1.48 read_x_path()

```
std::regex read_x_path (
    "read ([0-9a-zA-Z_]+)((\a-zA-Z_)+)? -b +" [0-9a-zA-Z_] [0-9] )
```

Regex For [File](#) System "read" Commands Not Yet Available

6.31.1.49 rename_file_cd()

```
std::regex rename_file_cd (
    "rename +(\-zA-Z)?" [0-9a-zA-Z],
    [0-9a-zA-Z_] +(\a-zA-Z_)+)?" )
```

6.31.1.50 rename_files()

```
std::regex rename_files (
    "rename (/ [0-9a-zA-Z_]* )*/+ (\\-zA-Z)+)? [0-9a-zA-Z],
    [0-9a-zA-Z] + (\\[a-zA-Z]+)? " )
```

6.31.1.51 rename_tags()

```
std::regex rename_tags (
    "rename -t \\0-9a-zA-Z_+ ) (, [0-9a-zA-Z_+)*\\ ,
    \\ ([0-9a-zA-Z_+ ) (, [0-9a-zA-Z_+)*\\" )
```

Regex For [File](#) System "rename" Commands

6.31.1.52 tag_files()

```
std::regex tag_files (
    "tag \\0-9a-zA-Z_+ ) (\\-zA-Z)+)? (, ([0-9a-zA-Z_+ ) (\\-9a-zA-Z_+)?)*\\ ,
    \\ ([0-9a-zA-Z_+ ) (, [0-9a-zA-Z_+)*\\" )
```

6.31.1.53 untag_file()

```
std::regex untag_file (
    "untag (/ [0-9a-zA-Z_]* )*/+ (\\-zA-Z)+)? \\ [0-9a-zA-Z],
    \\ ([0-9a-zA-Z_+ ) (, [0-9a-zA-Z_+)*\\" )
```

Regex For [File](#) System "untag" Commands

6.31.1.54 untag_files()

```
std::regex untag_files (
    "untag \\0-9a-zA-Z_+ ) (\\-zA-Z)+)? (, ([0-9a-zA-Z_+ ) (\\-9a-zA-Z_+)?)*\\ ,
    \\ ([0-9a-zA-Z_+ ) (, [0-9a-zA-Z_+)*\\" )
```

6.31.1.55 usage_attr()

```
std::regex usage_attr (
    "--attr" )
```

6.31.1.56 usage_cd()

```
std::regex usage_cd (
    "--cd" )
```

6.31.1.57 usage_close()

```
std::regex usage_close (
    "--close" )
```

6.31.1.58 usage_copy()

```
std::regex usage_copy (
    "--copy" )
```

6.31.1.59 usage_delete()

```
std::regex usage_delete (
    "--delete" )
```

6.31.1.60 usage_find()

```
std::regex usage_find (
    "--find" )
```

6.31.1.61 usage_help()

```
std::regex usage_help (
    "--help" )
```

6.31.1.62 usage_merge()

```
std::regex usage_merge (
    "--merge" )
```

6.31.1.63 usage_new()

```
std::regex usage_new (
    "--new" )
```

6.31.1.64 usage_open()

```
std::regex usage_open (
    "--open" )
```

6.31.1.65 usage_quit()

```
std::regex usage_quit (
    "--quit" )
```

6.31.1.66 usage_read()

```
std::regex usage_read (
    "--read" )
```

6.31.1.67 usage_rename()

```
std::regex usage_rename (
    "--rename" )
```

6.31.1.68 usage_tag()

```
std::regex usage_tag (
    "--tag" )
```

6.31.1.69 usage_untag()

```
std::regex usage_untag (
    "--untag" )
```

6.31.1.70 usage_write()

```
std::regex usage_write (
    "--write" )
```

6.31.1.71 write_cwd()

```
std::regex write_cwd (
    "" )
```

6.31.1.72 write_path()

```
std::regex write_path (
    "" )
```

6.31.1.73 write_x_cwd()

```
std::regex write_x_cwd (
    "write +((\\a-zA-Z_)+)? -b " [0-9a-zA-Z_][0-9] )
```

6.31.1.74 write_x_path()

```
std::regex write_x_path (
    "write ([0-9a-zA-Z_]+)((\\a-zA-Z_)+ -b +" [0-9a-zA-Z_][0-9] )
```

Regex For [File](#) System "write" Commands Not Yet Available

6.32 SharedHeaders/DebugMessages.hpp File Reference

```
#include <map>
#include <string>
#include <iostream>
#include <fstream>
#include <mutex>
```

Classes

- class [DebugMessages](#)

Functions

- `std::unique_lock< std::mutex > lk (m)`

Variables

- `std::mutex m`

6.32.1 Function Documentation

6.32.1.1 lk()

```
std::unique_lock<std::mutex> lk (  
    m )
```

6.32.2 Variable Documentation

6.32.2.1 m

```
std::mutex m
```

6.33 SharedHeaders/ErrorsCodes.h File Reference

6.34 SharedHeaders/Parser.h File Reference

```
#include <string>  
#include <iostream>  
#include <vector>  
#include "ErrorsCodes.h"  
#include "CommandCodes.h"
```

Classes

- class [ParseError](#)
- class [Parser](#)

Typedefs

- typedef unsigned int [uint](#)

6.34.1 Typedef Documentation

6.34.1.1 uint

```
typedef unsigned int uint
```

6.35 SharedHeaders/Print.h File Reference

```
#include "CommandValidation.h"
```

Functions

- void [print_cmnd_lst](#) ()
- void [print_help](#) ()
- void [print_quit](#) ()
- void [print_find](#) ()
- void [print_new](#) ()
- void [print_del](#) ()
- void [print_open](#) ()
- void [print_close](#) ()
- void [print_rname](#) ()
- void [print_attr](#) ()
- void [print_merge](#) ()
- void [print_tag](#) ()
- void [print_utag](#) ()
- void [print_cd](#) ()
- void [print_read](#) ()
- void [print_write](#) ()
- void [print_copy](#) ()
- void [help](#) ()
- void [print_header](#) ()
- void [print_command](#) (char *cmnd, int size)
Print a command buffer.
- template<typename T >
void [print_vector](#) (const std::vector< T > &vec)

6.35.1 Function Documentation

6.35.1.1 help()

```
void help ( )
```

Run helper applet

6.35.1.2 print_attr()

```
void print_attr ( )
```

Print usage for 'attr' command

6.35.1.3 print_cd()

```
void print_cd ( )
```

Print usage for 'cd' command

6.35.1.4 print_close()

```
void print_close ( )
```

Print usage for 'close' command

6.35.1.5 print_cmnd_lst()

```
void print_cmnd_lst ( )
```

Print a table of all of the available command archetypes

6.35.1.6 print_command()

```
void print_command (
    char * cmnd,
    int size )
```

Print a command buffer.

Because the command ID is saved as littoral bytes (as opposed to the string representation) and `std::cout` does not print those well, the first X bytes of the C-String are skipped where X is the size of an integer

Parameters

<i>cmnd</i>	The command buffer as a C-String
<i>size</i>	The size of the command buffer (should always be whatever <code>MaxBufferSize</code> is although this is not strictly speaking mandatory)

6.35.1.7 print_copy()

```
void print_copy ( )
```

Print usage for 'copy' command

6.35.1.8 print_del()

```
void print_del ( )
```

Print usage for 'delete' command

6.35.1.9 print_find()

```
void print_find ( )
```

Print usage for 'find' command

6.35.1.10 print_header()

```
void print_header ( )
```

Print a welcome header

6.35.1.11 print_help()

```
void print_help ( )
```

Print usage for 'help' command

6.35.1.12 print_merge()

```
void print_merge ( )
```

Print usage for 'merge' command

6.35.1.13 print_new()

```
void print_new ( )
```

Print usage for 'new' command

6.35.1.14 print_open()

```
void print_open ( )
```

Print usage for 'open' command

6.35.1.15 print_quit()

```
void print_quit ( )
```

Print usage for 'quit' command

6.35.1.16 print_read()

```
void print_read ( )
```

Print usage for 'read' command

6.35.1.17 print_rname()

```
void print_rname ( )
```

Print usage for 'rename' command

6.35.1.18 print_tag()

```
void print_tag ( )
```

Print usage for 'tag' command

6.35.1.19 print_utag()

```
void print_utag ( )
```

Print usage for 'untag' command

6.35.1.20 print_vector()

```
template<typename T >  
void print_vector (   
    const std::vector< T > & vec )
```

Prints the contents of any vector as long as the contents of the vector can be piped to stdout

Parameters

<i>vec</i>	A reference to the vector to be printed
------------	---

6.35.1.21 print_write()

```
void print_write ( )
```

Print usage for 'write' command

Index

- _blockNumber
 - TreeObject, [96](#)
 - _ecode
 - arboreal_exception, [15](#)
 - arboreal_runtime_error, [20](#)
 - _freeSpots
 - TreeObject, [96](#)
 - _indeces
 - TreeObject, [97](#)
 - _lastEntry
 - TreeObject, [97](#)
 - _mod
 - Modification, [71](#)
 - _modifications
 - TreeObject, [97](#)
 - _myPartitionManager
 - TreeObject, [97](#)
 - _myTree
 - TreeObject, [97](#)
 - _name
 - TreeObject, [97](#)
 - _parent
 - Modification, [71](#)
 - _startBlock
 - TreeObject, [97](#)
 - _where
 - arboreal_exception, [15](#)
 - arboreal_runtime_error, [20](#)
- ~Addition
 - Addition, [9](#)
- ~CLI
 - CLI, [26](#)
- ~DebugMessages
 - DebugMessages, [29](#)
- ~Deletion
 - Deletion, [32](#)
- ~Disk
 - Disk, [34](#)
- ~DiskManager
 - DiskManager, [37](#)
- ~FileInfo
 - FileInfo, [46](#)
- ~FileSystem
 - FileSystem, [57](#)
- ~Modification
 - Modification, [70](#)
- ~Parser
 - Parser, [74](#)
- ~PartitionManager
 - PartitionManager, [79](#)
- ~RootTree
 - RootTree, [82](#)
- ~TagTree
 - TagTree, [86](#)
- ~TreeObject
 - TreeObject, [89](#)
- ~arboreal_cli_error
 - arboreal_cli_error, [11](#)
- ~arboreal_daemon_error
 - arboreal_daemon_error, [13](#)
- ~arboreal_exception
 - arboreal_exception, [14](#)
- ~arboreal_liaison_error
 - arboreal_liaison_error, [16](#)
- ~arboreal_logic_error
 - arboreal_logic_error, [18](#)
- ~arboreal_runtime_error
 - arboreal_runtime_error, [20](#)
- ~disk_error
 - disk_error, [36](#)
- ~file_error
 - file_error, [44](#)
- ~invalid_arg
 - invalid_arg, [69](#)
- ~tag_error
 - tag_error, [85](#)
- APPND_FCWD
 - CommandCodes.h, [144](#)
- APPND_FP
 - CommandCodes.h, [144](#)
- APPND_XFCWDF
 - CommandCodes.h, [145](#)
- APPND_XFPF
 - CommandCodes.h, [145](#)
- ATTR_FP
 - CommandCodes.h, [145](#)
- ATTR_FS
 - CommandCodes.h, [145](#)
- accept_client
 - liason_helper.hpp, [134](#)
- add_direct_block
 - FileInfo, [46](#)
- add_index
 - TreeObject, [90](#)
- add_indirect_block
 - FileInfo, [47](#)
- add_tags
 - CommandValidation.h, [153](#)

- Addition, 9
 - ~Addition, 9
 - Addition, 9
 - write_out, 10
- append_cwd
 - CommandValidation.h, 153
- append_file
 - FileSystem, 57
- append_path
 - CommandValidation.h, 154
- append_x_cwd
 - CommandValidation.h, 154
- append_x_path
 - CommandValidation.h, 154
- arboreal_cli_error, 10
 - ~arboreal_cli_error, 11
 - arboreal_cli_error, 10, 11
- arboreal_daemon_error, 12
 - ~arboreal_daemon_error, 13
 - arboreal_daemon_error, 12, 13
- arboreal_exception, 13
 - _ecode, 15
 - _where, 15
 - ~arboreal_exception, 14
 - arboreal_exception, 14
 - ecode, 15
 - where, 15
- arboreal_liaison_error, 15
 - ~arboreal_liaison_error, 16
 - arboreal_liaison_error, 16
- arboreal_logic_error, 17
 - ~arboreal_logic_error, 18
 - arboreal_logic_error, 17, 18
- arboreal_runtime_error, 18
 - _ecode, 20
 - _where, 20
 - ~arboreal_runtime_error, 20
 - arboreal_runtime_error, 19
- Attributes, 20
 - Attributes, 21
 - del, 21
 - get_access, 21
 - get_creation_time, 22
 - get_edit, 22
 - get_file_attributes, 22
 - get_owner, 22
 - get_permissions, 22
 - get_size, 23
 - read_in, 23
 - set_access, 23
 - set_creation_time, 23
 - set_edit, 23
 - set_owner, 23
 - set_permissions, 24
 - update_size, 24
 - write_out, 24
- attributes
 - finode, 67
- await_response
 - CLI, 26
- BACKLOG
 - daemon.h, 125
- Backlog
 - liaison.cpp, 132
- bad_clean
 - cli_helper.hpp, 101
 - liaison_helper.hpp, 134
- begin
 - TreeObject, 90
- bind_socket
 - daemon.h, 117
- BlkNumType
 - types.h, 114
- blknum
 - index, 68
- build
 - CLI, 26
- byte_to_index
 - FileOpen, 54
- CD_ABS
 - CommandCodes.h, 145
- CD_RLP
 - CommandCodes.h, 145
- CLOSE_FP
 - CommandCodes.h, 145
- CLOSE_F
 - CommandCodes.h, 145
- CLI, 24
 - ~CLI, 26
 - await_response, 26
 - build, 26
 - CLI, 25, 26
 - run, 27, 28
 - send_cmnd, 28
 - start, 28
- CPY_FCWD
 - CommandCodes.h, 146
- CPY_FP
 - CommandCodes.h, 146
- CREATEFILEDATA
 - daemon.h, 116
 - timing.cpp, 129
- CREATETAGDATA
 - daemon.h, 117
 - timing.cpp, 129
- change_dir
 - CommandValidation.h, 154
- change_dir_rl
 - CommandValidation.h, 154
- check_command
 - CommandValidation.h, 154
- check_help
 - CommandValidation.h, 155
- check_usage
 - CommandValidation.h, 155

- clean
 - cli_helper.hpp, 101
 - liason_helper.hpp, 135
- Cli.cpp
 - main, 107
- Cli.h
 - Debug, 104
 - Flag, 105
 - MaxBufferSize, 105
 - Permissions, 106
 - SharedMemorySize, 106
- cli_helper.hpp
 - bad_clean, 101
 - clean, 101
 - connect_to_server, 101
 - create_shm_seg, 102
 - delete_shm, 102
 - EXCLUSIVE, 99
 - get_cmnd_id, 102
 - INCLUSIVE, 100
 - MERGE_1, 100
 - MERGE_2, 100
 - NEW_AND_TAG_EXC, 100
 - NEW_AND_TAG, 100
 - OPEN, 100
 - receive_from_server, 103
 - send_to_server, 103
 - set_up_socket, 103
 - TAG_1, 100
 - TAG_2, 100
 - TAG_3, 101
- close_file
 - FileSystem, 57
- close_file_cd
 - CommandValidation.h, 155
- close_files
 - CommandValidation.h, 155
- command_to_string
 - daemon.h, 118
- CommandCodes.h
 - APPND_FCWD, 144
 - APPND_FP, 144
 - APPND_XFCWDF, 145
 - APPND_XFPF, 145
 - ATTR_FP, 145
 - ATTR_FS, 145
 - CD_ABS, 145
 - CD_RLP, 145
 - CLOSE_FP, 145
 - CLOSE_F, 145
 - CPY_FCWD, 146
 - CPY_FP, 146
 - DEL_FP, 146
 - DEL_FS, 146
 - DEL_TS, 146
 - FIND_FS, 146
 - FIND_TS, 146
 - FTL_ERR, 146
 - HANDSHK, 147
 - MERG_1_1, 147
 - MERG_M_1, 147
 - NEW_FP, 147
 - NEW_FS, 147
 - NEW_TS, 147
 - OPEN_FP, 147
 - OPEN_F, 147
 - QUIT, 148
 - READ_FCWD, 148
 - READ_FP, 148
 - READ_XCWD, 148
 - READ_XP, 148
 - RNAME_FP, 148
 - RNAME_FS, 148
 - RNAME_TS, 148
 - TAG_FP, 149
 - TAG_FS, 149
 - UATTR, 149
 - UCLOSE, 149
 - UCOPY, 149
 - UCD, 149
 - UDEL, 149
 - UFIND, 149
 - UHELP, 150
 - UMERG, 150
 - UNEW, 150
 - UOPEN, 150
 - UQUIT, 150
 - UREAD, 150
 - URNAME, 150
 - UTAG_FP, 151
 - UTAG_FS, 151
 - UTAG, 150
 - UUTAG, 151
 - UWRITE, 151
 - WRITE_FCWD, 151
 - WRITE_FP, 151
 - WRITE_XFCWDF, 151
 - WRITE_XFPF, 151
- CommandLineInterface/CLDependancies/cli_helper.↵
 - hpp, 99
- CommandLineInterface/CLHeaders/Cli.h, 104
- CommandLineInterface/Cli.cpp, 106
- CommandValidation.h
 - add_tags, 153
 - append_cwd, 153
 - append_path, 154
 - append_x_cwd, 154
 - append_x_path, 154
 - change_dir, 154
 - change_dir_rl, 154
 - check_command, 154
 - check_help, 155
 - check_usage, 155
 - close_file_cd, 155
 - close_files, 155
 - copy_cwd, 155

- copy_path, 155
- del_file, 155
- del_files, 156
- del_tags, 156
- find_files, 156
- find_tags, 156
- get_attr_cd, 156
- get_attrs, 156
- help_1, 156
- help_10, 157
- help_11, 157
- help_12, 157
- help_13, 157
- help_14, 157
- help_15, 157
- help_16, 157
- help_2, 158
- help_3, 158
- help_4, 158
- help_5, 158
- help_6, 158
- help_7, 158
- help_8, 158
- help_9, 159
- merge_1_1, 159
- merge_m_1, 159
- new_file, 159
- new_files, 159
- new_tags, 159
- open_file_cd, 159
- open_files, 160
- read_cwd, 160
- read_path, 160
- read_x_cwd, 160
- read_x_path, 160
- rename_file_cd, 160
- rename_files, 160
- rename_tags, 161
- tag_files, 161
- untag_file, 161
- untag_files, 161
- usage_attr, 161
- usage_cd, 161
- usage_close, 162
- usage_copy, 162
- usage_delete, 162
- usage_find, 162
- usage_help, 162
- usage_merge, 162
- usage_new, 162
- usage_open, 163
- usage_quit, 163
- usage_read, 163
- usage_rename, 163
- usage_tag, 163
- usage_untag, 163
- usage_write, 163
- write_cwd, 164
- write_path, 164
- write_x_cwd, 164
- write_x_path, 164
- connect_to_daemon
 - liason_helper.hpp, 135
- connect_to_server
 - cli_helper.hpp, 101
- copy_cwd
 - CommandValidation.h, 155
- copy_path
 - CommandValidation.h, 155
- create_daemon_sock
 - liason_helper.hpp, 136
- create_file
 - FileSystem, 57
- create_shm_seg
 - cli_helper.hpp, 102
- create_sock
 - daemon.h, 118
- create_tag
 - FileSystem, 58
- creationTime
 - file_attributes, 42
- current_command_id
 - daemon.h, 125
- d
 - daemon.h, 125
- DEBUG
 - driver.cpp, 128
 - PartitionManager.cpp, 111
 - types.h, 115
- DEFAULTOWNER
 - Trees.h, 113
- DEFAULTPERMISSIONS
 - Trees.h, 113
- DEL_FP
 - CommandCodes.h, 146
- DEL_FS
 - CommandCodes.h, 146
- DEL_TS
 - CommandCodes.h, 146
- daemon.cpp
 - main, 108
 - STARTTUPDATA, 108
- daemon.h
 - BACKLOG, 125
 - bind_socket, 117
 - CREATEFILEDATA, 116
 - CREATETAGDATA, 117
 - command_to_string, 118
 - create_sock, 118
 - current_command_id, 125
 - d, 125
 - data, 125
 - Debug, 125
 - dm, 126
 - execute, 118
 - FALSE, 126

- FILESEARCHDATA, 117
- FLAG, 126
- fd_fs_map, 126
- get_cmnd_id, 119
- get_file_info, 119, 120
- get_partition, 120
- get_set, 120, 121
- get_short_file_info, 121
- is_number, 121
- listen_on_socket, 122
- MAX_COMMAND_SIZE, 126
- master_set, 126
- max_fid, 126
- my_fid, 126
- PORT, 127
- pad_string, 122
- part_fs_map, 127
- path_filedesc_map, 127
- RENAMETAGDATA, 117
- save_to_disk, 123
- serialize_fileinfo, 123
- set_nonblocking, 124
- set_socket_opt, 124
- sig_caught, 124
- TAGFILEDATA, 117
- TAGSEARCHDATA, 117
- TIMEOUT, 127
- TRUE, 127
- verbose, 127
- WILL_TIME, 127
- WRITE_CHANGES_WAIT, 128
- DaemonPort
 - liaison.cpp, 132
- data
 - daemon.h, 125
- Debug
 - Cli.h, 104
 - daemon.h, 125
 - liaison.cpp, 132
- debug
 - DebugMessages, 30
- DebugMessages, 29
 - ~DebugMessages, 29
 - debug, 30
 - DebugMessages, 29
 - display, 30
 - lock, 30
 - log, 31
 - OFF, 31
 - ON, 31
 - unlock, 31
- DebugMessages.hpp
 - lk, 165
 - m, 165
- decrement_seek
 - FileOpen, 54
- del
 - Attributes, 21
 - FileInfo, 47
 - RootTree, 83
 - TagTree, 86
 - TreeObject, 90
- del_file
 - CommandValidation.h, 155
- del_files
 - CommandValidation.h, 156
- del_tags
 - CommandValidation.h, 156
- delete_cont_blocks
 - FileInfo, 47
 - TreeObject, 90
- delete_file
 - FileSystem, 58
- delete_shm
 - cli_helper.hpp, 102
- delete_tag
 - FileSystem, 59
- Deletion, 32
 - ~Deletion, 32
 - Deletion, 32
 - write_out, 32
- directBlocks
 - finode, 67
- Disk, 33
 - ~Disk, 34
 - Disk, 33
 - getBlockCount, 34
 - getBlockSize, 34
 - readDiskBlock, 34
 - writeDiskBlock, 35
- disk_error, 35
 - ~disk_error, 36
 - disk_error, 36
- diskInfo.d, 107
- DiskManager, 37
 - ~DiskManager, 37
 - DiskManager, 37
 - findPart, 37
 - getBlockSize, 38
 - getPartitionSize, 38
 - readDiskBlock, 38
 - writeDiskBlock, 39
- DiskManager.cpp
 - operator==, 109
- DiskManager.h
 - operator==, 110
- DiskPartition, 39
 - fileNameSize, 39
 - partitionBlkStart, 40
 - partitionName, 40
 - partitionSize, 40
- display
 - DebugMessages, 30
- dm
 - daemon.h, 126
- driver.cpp

- DEBUG, 128
 - main, 128
- EXCLUSIVE
 - cli_helper.hpp, 99
- ecode
 - arboreal_exception, 15
- EncryptionFlag
 - FileSystem.cpp, 110
- end
 - TreeObject, 91
- erase
 - FileInfo, 48
 - TreeObject, 91
- execute
 - daemon.h, 118
- exthd.d, 107
- FALSE
 - daemon.h, 126
- FILESEARCHDATA
 - daemon.h, 117
 - timing.cpp, 129
- FIND_FS
 - CommandCodes.h, 146
- FIND_TS
 - CommandCodes.h, 146
- FLAG
 - daemon.h, 126
- FSFormat/format.cpp, 130
- FTL_ERR
 - CommandCodes.h, 146
- fd_fs_map
 - daemon.h, 126
- File, 40
 - File, 40
 - get_attributes, 41
 - get_name, 41
 - get_tags, 41
 - read_buff, 41
- file_attributes, 42
 - creationTime, 42
 - lastAccess, 42
 - lastEdit, 42
 - owner, 43
 - permissions, 43
 - size, 43
- file_error, 43
 - ~file_error, 44
 - file_error, 44
- file_search
 - FileSystem, 59
- FileAttributes
 - types.h, 114
- FileInfo, 45
 - ~FileInfo, 46
 - add_direct_block, 46
 - add_indirect_block, 47
 - del, 47
 - delete_cont_blocks, 47
 - erase, 48
 - FileInfo, 46
 - get_attributes, 48
 - get_file_attributes, 48
 - get_file_size, 48
 - get_finode, 48
 - get_tags, 49
 - get_vec_tags, 49
 - insert, 49
 - insert_addition, 50
 - insert_deletion, 50
 - mangle, 50, 51
 - read_in, 51
 - serialize, 52
 - set_access, 52
 - set_edit, 52
 - set_permissions, 52
 - update_file_size, 53
 - write_out, 53
- fileNameSize
 - DiskPartition, 39
- FileOpen, 53
 - byte_to_index, 54
 - decrement_seek, 54
 - FileOpen, 54
 - get_EOF, 54
 - get_file, 54
 - get_mode, 54
 - get_seek, 54
 - go_past_last_byte, 55
 - increment_index, 55
 - increment_seek, 55
 - refresh, 55
 - reset_seek, 55
 - set_EOF, 55
- FileSystem, 56
 - ~FileSystem, 57
 - append_file, 57
 - close_file, 57
 - create_file, 57
 - create_tag, 58
 - delete_file, 58
 - delete_tag, 59
 - file_search, 59
 - FileSystem, 56
 - get_attributes, 59
 - get_file_name_size, 60
 - merge_tags, 60
 - num_of_files, 60
 - num_of_tags, 60
 - open_file, 61
 - path_to_file, 61
 - print_files, 61
 - print_root, 61
 - print_tags, 62
 - read_file, 62
 - rename_file, 62

- rename_tag, 62
- seek_file_absolute, 63
- seek_file_relative, 63
- set_permissions, 63
- tag_file, 64
- tag_search, 65
- untag_file, 65
- write_changes, 66
- write_file, 66
- FileSystem.cpp
 - EncryptionFlag, 110
- Filesystem/DaemonDependencies/Disk/Disk.cpp, 108
- Filesystem/DaemonDependencies/Disk/Disk.h, 109
- Filesystem/DaemonDependencies/DiskManager/Disk↔
 - Manager.cpp, 109
- Filesystem/DaemonDependencies/DiskManager/Disk↔
 - Manager.h, 109
- Filesystem/DaemonDependencies/File/File.cpp, 110
- Filesystem/DaemonDependencies/File/File.h, 110
- Filesystem/DaemonDependencies/FileSystem/File↔
 - System.cpp, 110
- Filesystem/DaemonDependencies/FileSystem/File↔
 - System.h, 111
- Filesystem/DaemonDependencies/PartitionManager/↔
 - PartitionManager.cpp, 111
- Filesystem/DaemonDependencies/PartitionManager/↔
 - PartitionManager.h, 111
- Filesystem/DaemonDependencies/Trees/Trees.cpp, 112
- Filesystem/DaemonDependencies/Trees/Trees.h, 112
- Filesystem/DaemonDependencies/Types/types.h, 113
- Filesystem/DaemonHeaders/daemon.h, 115
- Filesystem/daemon.cpp, 107
- Filesystem/driver.cpp, 128
- Filesystem/timing.cpp, 129
- find
 - TreeObject, 91
- find_files
 - CommandValidation.h, 156
- find_tags
 - CommandValidation.h, 156
- findPart
 - DiskManager, 37
- Finode
 - types.h, 114
- finode, 67
 - attributes, 67
 - directBlocks, 67
 - level1Indirect, 67
 - level2Indirect, 67
 - level3Indirect, 67
- Flag
 - Cli.h, 105
 - liaison.cpp, 132
- format.cpp
 - main, 130
- get_EOF
 - FileOpen, 54
- get_access
 - Attributes, 21
- get_attr_cd
 - CommandValidation.h, 156
- get_attributes
 - File, 41
 - FileInfo, 48
 - FileSystem, 59
- get_attrs
 - CommandValidation.h, 156
- get_block_number
 - TreeObject, 92
- get_cmnd_id
 - cli_helper.hpp, 102
 - daemon.h, 119
 - liaison_helper.hpp, 136
- get_command_string
 - liaison_helper.hpp, 136
- get_creation_time
 - Attributes, 22
- get_cwd_tags
 - Parser, 75
- get_edit
 - Attributes, 22
- get_file
 - FileOpen, 54
- get_file_attributes
 - Attributes, 22
 - FileInfo, 48
- get_file_info
 - daemon.h, 119, 120
- get_file_name_size
 - FileSystem, 60
 - PartitionManager, 79
- get_file_size
 - FileInfo, 48
- get_finode
 - FileInfo, 48
- get_free_spots
 - TreeObject, 92
- get_index
 - TreeObject, 92
- get_last_entry
 - TreeObject, 92
- get_mode
 - FileOpen, 54
- get_name
 - File, 41
 - TreeObject, 93
- get_owner
 - Attributes, 22
- get_partition
 - daemon.h, 120
- get_peername
 - liaison_helper.hpp, 137
- get_permissions
 - Attributes, 22
- get_seek
 - FileOpen, 54

- get_set
 - daemon.h, [120](#), [121](#)
- get_shm_seg
 - liason_helper.hpp, [137](#)
- get_short_file_info
 - daemon.h, [121](#)
- get_size
 - Attributes, [23](#)
- get_start_block
 - TreeObject, [93](#)
- get_tags
 - File, [41](#)
 - FileInfo, [49](#)
- get_vec_tags
 - FileInfo, [49](#)
- getBlockCount
 - Disk, [34](#)
- getBlockSize
 - Disk, [34](#)
 - DiskManager, [38](#)
 - PartitionManager, [79](#)
- getFreeDiskBlock
 - PartitionManager, [79](#)
- getPartitionName
 - PartitionManager, [80](#)
- getPartitionSize
 - DiskManager, [38](#)
- go_past_last_byte
 - FileOpen, [55](#)
- HANDSHK
 - CommandCodes.h, [147](#)
- help
 - Print.h, [166](#)
- help_1
 - CommandValidation.h, [156](#)
- help_10
 - CommandValidation.h, [157](#)
- help_11
 - CommandValidation.h, [157](#)
- help_12
 - CommandValidation.h, [157](#)
- help_13
 - CommandValidation.h, [157](#)
- help_14
 - CommandValidation.h, [157](#)
- help_15
 - CommandValidation.h, [157](#)
- help_16
 - CommandValidation.h, [157](#)
- help_2
 - CommandValidation.h, [158](#)
- help_3
 - CommandValidation.h, [158](#)
- help_4
 - CommandValidation.h, [158](#)
- help_5
 - CommandValidation.h, [158](#)
- help_6
 - CommandValidation.h, [158](#)
- help_7
 - CommandValidation.h, [158](#)
- help_8
 - CommandValidation.h, [158](#)
- help_9
 - CommandValidation.h, [159](#)
- INCLUSIVE
 - cli_helper.hpp, [100](#)
- increment_allocate
 - TreeObject, [93](#)
- increment_follow
 - TreeObject, [93](#)
- increment_index
 - FileOpen, [55](#)
- increment_seek
 - FileOpen, [55](#)
- Index
 - types.h, [114](#)
- index, [68](#)
 - blknum, [68](#)
 - offset, [68](#)
- insert
 - FileInfo, [49](#)
 - TreeObject, [94](#)
- insert_addition
 - FileInfo, [50](#)
 - TreeObject, [94](#)
- insert_deletion
 - FileInfo, [50](#)
 - TreeObject, [95](#)
- invalid_arg, [68](#)
 - ~invalid_arg, [69](#)
 - invalid_arg, [69](#)
- is_number
 - daemon.h, [121](#)
- lastAccess
 - file_attributes, [42](#)
- lastEdit
 - file_attributes, [42](#)
- lastEntry
 - rootSuperBlock, [81](#)
 - tagTreeSuperBlock, [87](#)
- level1Indirect
 - finode, [67](#)
- level2Indirect
 - finode, [67](#)
- level3Indirect
 - finode, [67](#)
- liaison.cpp
 - Backlog, [132](#)
 - DaemonPort, [132](#)
 - Debug, [132](#)
 - Flag, [132](#)
 - main, [131](#)
 - MaxBufferSize, [132](#)
 - Parser, [132](#)

- Permissions, 132
- SharedMemorySize, 133
- Timeout, 133
- VERBOSE, 133
- LiaisonProcess/LiaisonDependencies/liason_helper.↵
 - hpp, 133
- LiaisonProcess/liaison.cpp, 131
- liason_helper.hpp
 - accept_client, 134
 - bad_clean, 134
 - clean, 135
 - connect_to_daemon, 135
 - create_daemon_sock, 136
 - get_cmnd_id, 136
 - get_command_string, 136
 - get_peername, 137
 - get_shm_seg, 137
 - listen_for_client, 138
 - NEW_PLUS, 134
 - pad_string, 139
 - recv_msg, 139
 - seg_fault, 140
 - send_response, 140
 - set_up_socket, 141
 - shutdown, 141
 - unat_shm, 142
- listen_for_client
 - liason_helper.hpp, 138
- listen_on_socket
 - daemon.h, 122
- lk
 - DebugMessages.hpp, 165
- lock
 - DebugMessages, 30
- log
 - DebugMessages, 31
- m
 - DebugMessages.hpp, 165
- MAX_COMMAND_SIZE
 - daemon.h, 126
- MAXopen_fileS
 - types.h, 114
- MERG_1_1
 - CommandCodes.h, 147
- MERG_M_1
 - CommandCodes.h, 147
- MERGE_1
 - cli_helper.hpp, 100
- MERGE_2
 - cli_helper.hpp, 100
- main
 - Cli.cpp, 107
 - daemon.cpp, 108
 - driver.cpp, 128
 - format.cpp, 130
 - liaison.cpp, 131
 - timing.cpp, 130
- mangle
 - FileInfo, 50, 51
- master_set
 - daemon.h, 126
- max_fid
 - daemon.h, 126
- MaxBufferSize
 - Cli.h, 105
 - liaison.cpp, 132
- merge_1_1
 - CommandValidation.h, 159
- merge_m_1
 - CommandValidation.h, 159
- merge_tags
 - FileSystem, 60
- Modification, 70
 - _mod, 71
 - _parent, 71
 - ~Modification, 70
 - Modification, 70
 - write_out, 71
- my_fid
 - daemon.h, 126
- NEW_AND_TAG_EXC
 - cli_helper.hpp, 100
- NEW_AND_TAG
 - cli_helper.hpp, 100
- NEW_FP
 - CommandCodes.h, 147
- NEW_FS
 - CommandCodes.h, 147
- NEW_PLUS
 - liason_helper.hpp, 134
- NEW_TS
 - CommandCodes.h, 147
- new_file
 - CommandValidation.h, 159
- new_files
 - CommandValidation.h, 159
- new_tags
 - CommandValidation.h, 159
- num_of_files
 - FileSystem, 60
- num_of_tags
 - FileSystem, 60
- OFF
 - DebugMessages, 31
- OPEN_FP
 - CommandCodes.h, 147
- OPEN_F
 - CommandCodes.h, 147
- OPEN
 - cli_helper.hpp, 100
- offset
 - index, 68
- ON
 - DebugMessages, 31
- open_file

- FileSystem, 61
- open_file_cd
 - CommandValidation.h, 159
- open_files
 - CommandValidation.h, 160
- operator!=
 - Trees.cpp, 112
- operator==
 - DiskManager.cpp, 109
 - DiskManager.h, 110
 - Trees.cpp, 112
- owner
 - file_attributes, 43
- PORT
 - daemon.h, 127
- pad_string
 - daemon.h, 122
 - liaison_helper.hpp, 139
- parse
 - Parser, 75
- ParseError, 71
 - ParseError, 71
 - what, 72
 - where, 72
- Parser, 72
 - ~Parser, 74
 - get_cwd_tags, 75
 - liaison.cpp, 132
 - parse, 75
 - Parser, 73, 74
 - reset, 75, 76
 - set_cwd, 77
 - set_max_name_size, 77
 - split_on_delim, 78
- Parser.h
 - uint, 166
- part_fs_map
 - daemon.h, 127
- partitionBlkStart
 - DiskPartition, 40
- PartitionManager, 78
 - ~PartitionManager, 79
 - get_file_name_size, 79
 - getBlockSize, 79
 - getFreeDiskBlock, 79
 - getPartitionName, 80
 - PartitionManager, 79
 - readDiskBlock, 80
 - returnDiskBlock, 80
 - writeDiskBlock, 81
- PartitionManager.cpp
 - DEBUG, 111
- partitionName
 - DiskPartition, 40
- partitionSize
 - DiskPartition, 40
- path_filedesc_map
 - daemon.h, 127
- path_to_file
 - FileSystem, 61
- Permissions
 - Cli.h, 106
 - liaison.cpp, 132
- permissions
 - file_attributes, 43
- Print.h
 - help, 166
 - print_attr, 166
 - print_cd, 167
 - print_close, 167
 - print_cmnd_lst, 167
 - print_command, 167
 - print_copy, 167
 - print_del, 167
 - print_find, 168
 - print_header, 168
 - print_help, 168
 - print_merge, 168
 - print_new, 168
 - print_open, 168
 - print_quit, 168
 - print_read, 168
 - print_rname, 169
 - print_tag, 169
 - print_utag, 169
 - print_vector, 169
 - print_write, 169
- print_attr
 - Print.h, 166
- print_cd
 - Print.h, 167
- print_close
 - Print.h, 167
- print_cmnd_lst
 - Print.h, 167
- print_command
 - Print.h, 167
- print_copy
 - Print.h, 167
- print_del
 - Print.h, 167
- print_files
 - FileSystem, 61
- print_find
 - Print.h, 168
- print_header
 - Print.h, 168
- print_help
 - Print.h, 168
- print_merge
 - Print.h, 168
- print_new
 - Print.h, 168
- print_open
 - Print.h, 168
- print_quit

- Print.h, 168
- print_read
 - Print.h, 168
- print_rname
 - Print.h, 169
- print_root
 - FileSystem, 61
- print_tag
 - Print.h, 169
- print_tags
 - FileSystem, 62
- print_utag
 - Print.h, 169
- print_vector
 - Print.h, 169
- print_write
 - Print.h, 169
- QUIT
 - CommandCodes.h, 148
- READ_FCWD
 - CommandCodes.h, 148
- READ_FP
 - CommandCodes.h, 148
- READ_XCWD
 - CommandCodes.h, 148
- READ_XP
 - CommandCodes.h, 148
- README.md, 142
- RENAMETAGDATA
 - daemon.h, 117
 - timing.cpp, 129
- RNAME_FP
 - CommandCodes.h, 148
- RNAME_FS
 - CommandCodes.h, 148
- RNAME_TS
 - CommandCodes.h, 148
- read_buff
 - File, 41
- read_cwd
 - CommandValidation.h, 160
- read_file
 - FileSystem, 62
- read_in
 - Attributes, 23
 - FileInfo, 51
 - RootTree, 83
 - TagTree, 86
 - TreeObject, 95
- read_path
 - CommandValidation.h, 160
- read_x_cwd
 - CommandValidation.h, 160
- read_x_path
 - CommandValidation.h, 160
- readDiskBlock
 - Disk, 34
- DiskManager, 38
- PartitionManager, 80
- receive_from_server
 - cli_helper.hpp, 103
- recv_msg
 - liason_helper.hpp, 139
- refresh
 - FileOpen, 55
- rename_file
 - FileSystem, 62
- rename_file_cd
 - CommandValidation.h, 160
- rename_files
 - CommandValidation.h, 160
- rename_tag
 - FileSystem, 62
- rename_tags
 - CommandValidation.h, 161
- reset
 - Parser, 75, 76
- reset_seek
 - FileOpen, 55
- returnDiskBlock
 - PartitionManager, 80
- RootSuperBlock
 - types.h, 115
- rootSuperBlock, 81
 - lastEntry, 81
 - size, 81
 - startBlock, 81
- RootTree, 82
 - ~RootTree, 82
 - del, 83
 - read_in, 83
 - RootTree, 82
 - write_out, 83
- run
 - CLI, 27, 28
- STARTTUPDATA
 - daemon.cpp, 108
 - timing.cpp, 130
- save_to_disk
 - daemon.h, 123
- seek_file_absolute
 - FileSystem, 63
- seek_file_relative
 - FileSystem, 63
- seg_fault
 - liason_helper.hpp, 140
- send_cmnd
 - CLI, 28
- send_response
 - liason_helper.hpp, 140
- send_to_server
 - cli_helper.hpp, 103
- serialize
 - FileInfo, 52
- serialize_fileinfo

- daemon.h, 123
- set_EOF
 - FileOpen, 55
- set_access
 - Attributes, 23
 - FileInfo, 52
- set_creation_time
 - Attributes, 23
- set_cwd
 - Parser, 77
- set_edit
 - Attributes, 23
 - FileInfo, 52
- set_last_entry
 - TreeObject, 95
- set_max_name_size
 - Parser, 77
- set_name
 - TreeObject, 96
- set_nonblocking
 - daemon.h, 124
- set_owner
 - Attributes, 23
- set_permissions
 - Attributes, 24
 - FileInfo, 52
 - FileSystem, 63
- set_socket_opt
 - daemon.h, 124
- set_up_socket
 - cli_helper.hpp, 103
 - liason_helper.hpp, 141
- SharedCPPFiles/Arboreal_Exceptions.cpp, 142
- SharedCPPFiles/Parser.cpp, 143
- SharedHeaders/Arboreal_Exceptions.h, 143
- SharedHeaders/CommandCodes.h, 143
- SharedHeaders/CommandValidation.h, 152
- SharedHeaders/DebugMessages.hpp, 164
- SharedHeaders/ErrorCodes.h, 165
- SharedHeaders/Parser.h, 165
- SharedHeaders/Print.h, 166
- SharedMemorySize
 - Cli.h, 106
 - liaison.cpp, 133
- shutdown
 - liason_helper.hpp, 141
- sig_caught
 - daemon.h, 124
- size
 - file_attributes, 43
 - rootSuperBlock, 81
 - tagTreeSuperBlock, 87
 - TreeObject, 96
- split_on_delim
 - Parser, 78
- start
 - CLI, 28
- startBlock
 - rootSuperBlock, 81
 - tagTreeSuperBlock, 87
- TAG_1
 - cli_helper.hpp, 100
- TAG_2
 - cli_helper.hpp, 100
- TAG_3
 - cli_helper.hpp, 101
- TAG_FP
 - CommandCodes.h, 149
- TAG_FS
 - CommandCodes.h, 149
- TAGFILEDATA
 - daemon.h, 117
 - timing.cpp, 130
- TAGSEARCHDATA
 - daemon.h, 117
 - timing.cpp, 130
- TIMEOUT
 - daemon.h, 127
- TRUE
 - daemon.h, 127
- tag_error, 84
 - ~tag_error, 85
 - tag_error, 84, 85
- tag_file
 - FileSystem, 64
- tag_files
 - CommandValidation.h, 161
- tag_search
 - FileSystem, 65
- TagTree, 85
 - ~TagTree, 86
 - del, 86
 - read_in, 86
 - TagTree, 86
 - write_out, 87
- TagTreeSuperBlock
 - types.h, 115
- tagTreeSuperBlock, 87
 - lastEntry, 87
 - size, 87
 - startBlock, 87
- Timeout
 - liaison.cpp, 133
- timing.cpp
 - CREATEFILEDATA, 129
 - CREATETAGDATA, 129
 - FILESEARCHDATA, 129
 - main, 130
 - RENAMETAGDATA, 129
 - STARTTUPDATA, 130
 - TAGFILEDATA, 130
 - TAGSEARCHDATA, 130
- TreeObject, 88
 - _blockNumber, 96
 - _freeSpots, 96
 - _indeces, 97

- [_lastEntry, 97](#)
- [_modifications, 97](#)
- [_myPartitionManager, 97](#)
- [_myTree, 97](#)
- [_name, 97](#)
- [_startBlock, 97](#)
- [~TreeObject, 89](#)
- [add_index, 90](#)
- [begin, 90](#)
- [del, 90](#)
- [delete_cont_blocks, 90](#)
- [end, 91](#)
- [erase, 91](#)
- [find, 91](#)
- [get_block_number, 92](#)
- [get_free_spots, 92](#)
- [get_index, 92](#)
- [get_last_entry, 92](#)
- [get_name, 93](#)
- [get_start_block, 93](#)
- [increment_allocate, 93](#)
- [increment_follow, 93](#)
- [insert, 94](#)
- [insert_addition, 94](#)
- [insert_deletion, 95](#)
- [read_in, 95](#)
- [set_last_entry, 95](#)
- [set_name, 96](#)
- [size, 96](#)
- [TreeObject, 89](#)
- [write_out, 96](#)
- Trees.cpp
 - [operator!=, 112](#)
 - [operator==, 112](#)
- Trees.h
 - [DEFAULTOWNER, 113](#)
 - [DEFAULTPERMISSIONS, 113](#)
- types.h
 - [BlkNumType, 114](#)
 - [DEBUG, 115](#)
 - [FileAttributes, 114](#)
 - [Finode, 114](#)
 - [Index, 114](#)
 - [MAXopen_fileS, 114](#)
 - [RootSuperBlock, 115](#)
 - [TagTreeSuperBlock, 115](#)
- UATTR
 - [CommandCodes.h, 149](#)
- UCLOSE
 - [CommandCodes.h, 149](#)
- UCOPY
 - [CommandCodes.h, 149](#)
- UCD
 - [CommandCodes.h, 149](#)
- UDEL
 - [CommandCodes.h, 149](#)
- UFIND
 - [CommandCodes.h, 149](#)
- UHELP
 - [CommandCodes.h, 150](#)
- UMERG
 - [CommandCodes.h, 150](#)
- UNEW
 - [CommandCodes.h, 150](#)
- UOPEN
 - [CommandCodes.h, 150](#)
- UQUIT
 - [CommandCodes.h, 150](#)
- UREAD
 - [CommandCodes.h, 150](#)
- URNAME
 - [CommandCodes.h, 150](#)
- UTAG_FP
 - [CommandCodes.h, 151](#)
- UTAG_FS
 - [CommandCodes.h, 151](#)
- UTAG
 - [CommandCodes.h, 150](#)
- UUTAG
 - [CommandCodes.h, 151](#)
- UWRITE
 - [CommandCodes.h, 151](#)
- uint
 - [Parser.h, 166](#)
- unat_shm
 - [liason_helper.hpp, 142](#)
- unlock
 - [DebugMessages, 31](#)
- untag_file
 - [CommandValidation.h, 161](#)
 - [FileSystem, 65](#)
- untag_files
 - [CommandValidation.h, 161](#)
- update_file_size
 - [FileInfo, 53](#)
- update_size
 - [Attributes, 24](#)
- usage_attr
 - [CommandValidation.h, 161](#)
- usage_cd
 - [CommandValidation.h, 161](#)
- usage_close
 - [CommandValidation.h, 162](#)
- usage_copy
 - [CommandValidation.h, 162](#)
- usage_delete
 - [CommandValidation.h, 162](#)
- usage_find
 - [CommandValidation.h, 162](#)
- usage_help
 - [CommandValidation.h, 162](#)
- usage_merge
 - [CommandValidation.h, 162](#)
- usage_new
 - [CommandValidation.h, 162](#)
- usage_open

- CommandValidation.h, [163](#)
- usage_quit
 - CommandValidation.h, [163](#)
- usage_read
 - CommandValidation.h, [163](#)
- usage_rename
 - CommandValidation.h, [163](#)
- usage_tag
 - CommandValidation.h, [163](#)
- usage_untag
 - CommandValidation.h, [163](#)
- usage_write
 - CommandValidation.h, [163](#)
- VERBOSE
 - liaison.cpp, [133](#)
- verbose
 - daemon.h, [127](#)
- WILL_TIME
 - daemon.h, [127](#)
- WRITE_CHANGES_WAIT
 - daemon.h, [128](#)
- WRITE_FCWD
 - CommandCodes.h, [151](#)
- WRITE_FP
 - CommandCodes.h, [151](#)
- WRITE_XFCWDF
 - CommandCodes.h, [151](#)
- WRITE_XFPF
 - CommandCodes.h, [151](#)
- what
 - ParseError, [72](#)
- where
 - arboreal_exception, [15](#)
 - ParseError, [72](#)
- write_changes
 - FileSystem, [66](#)
- write_cwd
 - CommandValidation.h, [164](#)
- write_file
 - FileSystem, [66](#)
- write_out
 - Addition, [10](#)
 - Attributes, [24](#)
 - Deletion, [32](#)
 - FileInfo, [53](#)
 - Modification, [71](#)
 - RootTree, [83](#)
 - TagTree, [87](#)
 - TreeObject, [96](#)
- write_path
 - CommandValidation.h, [164](#)
- write_x_cwd
 - CommandValidation.h, [164](#)
- write_x_path
 - CommandValidation.h, [164](#)
- writeDiskBlock
 - Disk, [35](#)
- DiskManager, [39](#)
- PartitionManager, [81](#)