

The Witchcraft Compiler Collection

<http://github.com/endrazine/wcc/>

Contents

- 1 Data Structure Index** **1**
- 1.1 Data Structures 1

- 2 File Index** **3**
- 2.1 File List 3

- 3 Data Structure Documentation** **5**
- 3.1 breakpoint_t Struct Reference 5
- 3.1.1 Detailed Description 5
- 3.1.2 Field Documentation 5
- 3.1.2.1 backup 5
- 3.1.2.2 ptr 5
- 3.1.2.3 weight 5
- 3.2 ctx_t Struct Reference 5
- 3.2.1 Detailed Description 6
- 3.2.2 Field Documentation 6
- 3.2.2.1 abfd 6
- 3.2.2.2 archsz 6
- 3.2.2.3 base_address 7
- 3.2.2.4 binname 7
- 3.2.2.5 corefile 7
- 3.2.2.6 fdout 7
- 3.2.2.7 has_relativerelocations 7
- 3.2.2.8 mphdrs 7
- 3.2.2.9 mphnum 7
- 3.2.2.10 mshdrs 7
- 3.2.2.11 mshnum 7
- 3.2.2.12 opt_arch 7
- 3.2.2.13 opt_asmdebug 7
- 3.2.2.14 opt_binname 7
- 3.2.2.15 opt_core 8
- 3.2.2.16 opt_debug 8

3.2.2.17	opt_entrypoint	8
3.2.2.18	opt_exec	8
3.2.2.19	opt_flags	8
3.2.2.20	opt_interp	8
3.2.2.21	opt_original	8
3.2.2.22	opt_poison	8
3.2.2.23	opt_reloc	8
3.2.2.24	opt_shared	8
3.2.2.25	opt_sstrip	8
3.2.2.26	opt_static	8
3.2.2.27	opt_strip	9
3.2.2.28	opt_verbose	9
3.2.2.29	phnum	9
3.2.2.30	shnum	9
3.2.2.31	start_phdrs	9
3.2.2.32	start_shdrs	9
3.2.2.33	strndx	9
3.2.2.34	strndx_index	9
3.2.2.35	strndx_len	9
3.3	elfdata_t Struct Reference	9
3.3.1	Detailed Description	10
3.3.2	Field Documentation	10
3.3.2.1	base	10
3.3.2.2	dyn_index	10
3.3.2.3	dyns	10
3.3.2.4	ehdr	10
3.3.2.5	et_dyn	10
3.3.2.6	limit	10
3.3.2.7	link_map	10
3.3.2.8	p_pltgot	10
3.3.2.9	phdrs	10
3.3.2.10	r_debug	11
3.4	eps_t Struct Reference	11
3.4.1	Detailed Description	11
3.4.2	Field Documentation	11
3.4.2.1	addr	11
3.4.2.2	name	11
3.4.2.3	next	11
3.4.2.4	prev	11
3.5	gimport_t Struct Reference	11

3.5.1	Detailed Description	12
3.5.2	Field Documentation	12
3.5.2.1	r	12
3.5.2.2	rtype	12
3.5.2.3	sec	12
3.5.2.4	sindex	12
3.5.2.5	sname	12
3.6	help_t Struct Reference	12
3.6.1	Detailed Description	12
3.6.2	Field Documentation	13
3.6.2.1	descr	13
3.6.2.2	name	13
3.6.2.3	proto	13
3.6.2.4	protoprefix	13
3.6.2.5	retval	13
3.7	learn_key_t Struct Reference	13
3.7.1	Detailed Description	13
3.7.2	Field Documentation	13
3.7.2.1	targ	13
3.7.2.2	tfunction	13
3.7.2.3	tlib	14
3.7.2.4	ttype	14
3.7.2.5	tvalue	14
3.8	learn_t Struct Reference	14
3.8.1	Detailed Description	14
3.8.2	Field Documentation	14
3.8.2.1	hh	14
3.8.2.2	key	14
3.8.2.3	toffset	14
3.9	linenoiseCompletions Struct Reference	14
3.9.1	Detailed Description	15
3.9.2	Field Documentation	15
3.9.2.1	cvec	15
3.9.2.2	len	15
3.10	lua_Debug Struct Reference	15
3.10.1	Detailed Description	15
3.10.2	Field Documentation	16
3.10.2.1	currentline	16
3.10.2.2	event	16
3.10.2.3	i_ci	16

3.10.2.4	istailcall	16
3.10.2.5	isvararg	16
3.10.2.6	lastlinedefined	16
3.10.2.7	linedefined	16
3.10.2.8	name	16
3.10.2.9	namewhat	16
3.10.2.10	nparams	16
3.10.2.11	nups	16
3.10.2.12	short_src	16
3.10.2.13	source	17
3.10.2.14	what	17
3.11	luaL_Buffer Struct Reference	17
3.11.1	Detailed Description	17
3.11.2	Field Documentation	17
3.11.2.1	b	17
3.11.2.2	initb	17
3.11.2.3	L	17
3.11.2.4	n	17
3.11.2.5	size	18
3.12	luaL_Reg Struct Reference	18
3.12.1	Detailed Description	18
3.12.2	Field Documentation	18
3.12.2.1	func	18
3.12.2.2	name	18
3.13	luaL_Stream Struct Reference	18
3.13.1	Detailed Description	18
3.13.2	Field Documentation	19
3.13.2.1	closef	19
3.13.2.2	f	19
3.14	msec_t Struct Reference	19
3.14.1	Detailed Description	19
3.14.2	Field Documentation	19
3.14.2.1	data	19
3.14.2.2	flags	19
3.14.2.3	len	19
3.14.2.4	name	19
3.14.2.5	next	20
3.14.2.6	outoffset	20
3.14.2.7	prev	20
3.14.2.8	s_bfd	20

3.14.2.9	s_elf	20
3.15	mseg_t Struct Reference	20
3.15.1	Detailed Description	20
3.15.2	Field Documentation	20
3.15.2.1	next	20
3.15.2.2	p_align	21
3.15.2.3	p_filesz	21
3.15.2.4	p_flags	21
3.15.2.5	p_memsz	21
3.15.2.6	p_offset	21
3.15.2.7	p_paddr	21
3.15.2.8	p_type	21
3.15.2.9	p_vaddr	21
3.15.2.10	prev	21
3.16	preload_t Struct Reference	21
3.16.1	Detailed Description	22
3.16.2	Field Documentation	22
3.16.2.1	name	22
3.16.2.2	next	22
3.16.2.3	prev	22
3.17	range_t Struct Reference	22
3.17.1	Detailed Description	22
3.17.2	Field Documentation	22
3.17.2.1	max	22
3.17.2.2	min	22
3.18	script_t Struct Reference	23
3.18.1	Detailed Description	23
3.18.2	Field Documentation	23
3.18.2.1	name	23
3.18.2.2	next	23
3.18.2.3	prev	23
3.19	section Struct Reference	23
3.19.1	Detailed Description	24
3.19.2	Field Documentation	24
3.19.2.1	end	24
3.19.2.2	hperms	24
3.19.2.3	init	24
3.19.2.4	name	24
3.19.2.5	next	24
3.19.2.6	num	24

3.19.2.7	perms	24
3.19.2.8	proba	24
3.19.2.9	probableval	24
3.19.2.10	size	24
3.20	sections_t Struct Reference	25
3.20.1	Detailed Description	25
3.20.2	Field Documentation	25
3.20.2.1	addr	25
3.20.2.2	flags	25
3.20.2.3	libname	25
3.20.2.4	name	25
3.20.2.5	next	25
3.20.2.6	perms	25
3.20.2.7	prev	25
3.20.2.8	size	26
3.21	segments_t Struct Reference	26
3.21.1	Detailed Description	26
3.21.2	Field Documentation	26
3.21.2.1	addr	26
3.21.2.2	flags	26
3.21.2.3	libname	26
3.21.2.4	next	26
3.21.2.5	perms	26
3.21.2.6	prev	27
3.21.2.7	size	27
3.21.2.8	type	27
3.22	signame_t Struct Reference	27
3.22.1	Detailed Description	27
3.22.2	Field Documentation	27
3.22.2.1	name	27
3.22.2.2	signal	27
3.23	symaddr Struct Reference	27
3.23.1	Detailed Description	28
3.23.2	Field Documentation	28
3.23.2.1	addr	28
3.23.2.2	name	28
3.23.2.3	next	28
3.24	symbols_t Struct Reference	28
3.24.1	Detailed Description	28
3.24.2	Field Documentation	28

3.24.2.1	addr	28
3.24.2.2	hbind	29
3.24.2.3	htype	29
3.24.2.4	libname	29
3.24.2.5	next	29
3.24.2.6	prev	29
3.24.2.7	size	29
3.24.2.8	symbol	29
3.24.2.9	value	29
3.25	tuple_t Struct Reference	29
3.25.1	Detailed Description	29
3.25.2	Field Documentation	30
3.25.2.1	addr	30
3.25.2.2	name	30
3.26	wsh_t Struct Reference	30
3.26.1	Detailed Description	31
3.26.2	Field Documentation	31
3.26.2.1	bp_array	31
3.26.2.2	bp_num	31
3.26.2.3	bp_points	31
3.26.2.4	btcaller	31
3.26.2.5	eps	31
3.26.2.6	errcontext	31
3.26.2.7	faultaddr	32
3.26.2.8	firstcontext	32
3.26.2.9	firsterrno	32
3.26.2.10	firstsicode	32
3.26.2.11	firstsignal	32
3.26.2.12	globalsignals	32
3.26.2.13	interrupted	32
3.26.2.14	is_stdinscript	32
3.26.2.15	L	32
3.26.2.16	learnfile	32
3.26.2.17	learnlog	32
3.26.2.18	longjmp_ptr	32
3.26.2.19	longjmp_ptr_high	33
3.26.2.20	longjmp_ptr_high_cnt	33
3.26.2.21	mainhandle	33
3.26.2.22	opt_argc	33
3.26.2.23	opt_argv	33

3.26.2.24	opt_hollywood	33
3.26.2.25	opt_rescan	33
3.26.2.26	opt_verbose	33
3.26.2.27	opt_verbosetrace	33
3.26.2.28	phdrs	33
3.26.2.29	pltgot	33
3.26.2.30	pltz	33
3.26.2.31	preload	34
3.26.2.32	reason	34
3.26.2.33	script_argnum	34
3.26.2.34	script_args	34
3.26.2.35	scriptfile	34
3.26.2.36	scriptname	34
3.26.2.37	scripts	34
3.26.2.38	shdrs	34
3.26.2.39	sigbus_count	34
3.26.2.40	sigbus_hash	34
3.26.2.41	singlebranch_count	34
3.26.2.42	singlebranch_hash	34
3.26.2.43	singlestep_count	35
3.26.2.44	singlestep_hash	35
3.26.2.45	symbols	35
3.26.2.46	totsignals	35
3.26.2.47	trace_rtrace	35
3.26.2.48	trace_singlebranch	35
3.26.2.49	trace_singlestep	35
3.26.2.50	trace_strace	35
3.26.2.51	trace_unaligned	35
4	File Documentation	37
4.1	wcc/wcc.c File Reference	37
4.1.1	Macro Definition Documentation	40
4.1.1.1	__USE_GNU	40
4.1.1.2	_GNU_SOURCE	40
4.1.1.3	CS_MODE	41
4.1.1.4	DEFAULT_STRNDX_SIZE	41
4.1.1.5	Elf_Addr	41
4.1.1.6	Elf_Ehdr	41
4.1.1.7	Elf_Off	41
4.1.1.8	Elf_Phdr	41

4.1.1.9	ELF_R_INFO	41
4.1.1.10	ELF_R_SYM	41
4.1.1.11	ELF_R_TYPE	41
4.1.1.12	Elf_Rel	41
4.1.1.13	Elf_Rela	41
4.1.1.14	Elf_Section	41
4.1.1.15	Elf_Shdr	42
4.1.1.16	ELF_ST_BIND	42
4.1.1.17	ELF_ST_TYPE	42
4.1.1.18	Elf_Sword	42
4.1.1.19	Elf_Sym	42
4.1.1.20	Elf_Word	42
4.1.1.21	Elf_Xword	42
4.1.1.22	ELFCLASS	42
4.1.1.23	ELFMACHINE	42
4.1.1.24	elis	42
4.1.1.25	EXTRA_CREATED_SECTIONS	42
4.1.1.26	FLAG_BSS	42
4.1.1.27	FLAG_NOBIT	43
4.1.1.28	FLAG_NOWRITE	43
4.1.1.29	FLAG_TEXT	43
4.1.1.30	ifis	43
4.1.1.31	MAXPADLEN	43
4.1.1.32	nullstr	43
4.1.1.33	RELOC_MODE	43
4.1.1.34	RELOC_X86_32	43
4.1.1.35	RELOC_X86_64	43
4.1.2	Typedef Documentation	43
4.1.2.1	ctx_t	43
4.1.2.2	gimport_t	43
4.1.2.3	msec_t	43
4.1.2.4	mseg_t	43
4.1.3	Function Documentation	44
4.1.3.1	add_extra_symbols	44
4.1.3.2	add_symaddr	44
4.1.3.3	adjust_baseaddress	44
4.1.3.4	alignfromname	44
4.1.3.5	alloc_phdr	44
4.1.3.6	analyze_text	44
4.1.3.7	append_reloc	44

4.1.3.8	append_strtab	44
4.1.3.9	append_sym	44
4.1.3.10	check_global_import	45
4.1.3.11	copy_body	45
4.1.3.12	craft_section	45
4.1.3.13	create_phdrs	45
4.1.3.14	ctx_getopt	45
4.1.3.15	ctx_init	45
4.1.3.16	desired_arch	45
4.1.3.17	entszfromname	45
4.1.3.18	fixup_strtab_and_symtab	45
4.1.3.19	fixup_symtab_section_index	45
4.1.3.20	fixup_text	46
4.1.3.21	flags_from_name	46
4.1.3.22	hexdump	46
4.1.3.23	info_from_name	46
4.1.3.24	internal_function_store	46
4.1.3.25	libify	46
4.1.3.26	link_from_name	47
4.1.3.27	load_binary	47
4.1.3.28	main	47
4.1.3.29	max	47
4.1.3.30	merge_phdrs	47
4.1.3.31	mk_section	47
4.1.3.32	open_best	47
4.1.3.33	open_target	48
4.1.3.34	patch_symbol_index	48
4.1.3.35	pflag_from_section	48
4.1.3.36	phdr_cmp	48
4.1.3.37	phdr_cmp_premerge	48
4.1.3.38	print_bfd_sections	48
4.1.3.39	print_maps	48
4.1.3.40	print_msec	48
4.1.3.41	print_version	48
4.1.3.42	protect_perms	48
4.1.3.43	ptype_from_section	49
4.1.3.44	rd_sections	49
4.1.3.45	rd_symbols	49
4.1.3.46	rd_symtab	49
4.1.3.47	reloc_htype	49

4.1.3.48	reloc_htype_x86_32	49
4.1.3.49	reloc_htype_x86_64	49
4.1.3.50	rm_section	49
4.1.3.51	save_dynstr	49
4.1.3.52	save_dynsym	49
4.1.3.53	save_global_import	50
4.1.3.54	save_reloc	50
4.1.3.55	sec_name_from_index_after_strip	50
4.1.3.56	secindex_from_name	50
4.1.3.57	secindex_from_name_after_strip	50
4.1.3.58	section_from_addr	50
4.1.3.59	section_from_index	50
4.1.3.60	section_from_name	50
4.1.3.61	sort_phdrs	50
4.1.3.62	sort_phdrs_premerge	50
4.1.3.63	strip_binary_reloc	51
4.1.3.64	typefromname	51
4.1.3.65	usage	51
4.1.4	Variable Documentation	51
4.1.4.1	allowed_sections	51
4.1.4.2	blnames	51
4.1.4.3	datavma	51
4.1.4.4	deltastrtab	51
4.1.4.5	gimports	51
4.1.4.6	gimportslen	51
4.1.4.7	globalreloc	52
4.1.4.8	globalrelocen	52
4.1.4.9	globalrelocoffset	52
4.1.4.10	globalstrtab	52
4.1.4.11	globalstrtablen	52
4.1.4.12	globalstrtableoffset	52
4.1.4.13	globalsymindex	52
4.1.4.14	globalsymtab	52
4.1.4.15	globalsymtablen	52
4.1.4.16	globalsymtableoffset	52
4.1.4.17	maxdata	52
4.1.4.18	maxnewsec	52
4.1.4.19	maxoldsec	53
4.1.4.20	maxtext	53
4.1.4.21	mindata	53

4.1.4.22	mintext	53
4.1.4.23	orig_sz	53
4.1.4.24	orig_text	53
4.1.4.25	symaddrs	53
4.1.4.26	textvma	53
4.2	wld/wld.c File Reference	53
4.2.1	Macro Definition Documentation	54
4.2.1.1	DEFAULT_NAME	54
4.2.2	Function Documentation	54
4.2.2.1	main	54
4.2.2.2	mk_lib	54
4.2.2.3	print_version	54
4.3	wsh/helper.c File Reference	54
4.3.1	Macro Definition Documentation	55
4.3.1.1	_FILE_OFFSET_BITS	55
4.3.1.2	_XOPEN_SOURCE	55
4.3.1.3	HAS_ZFIRST	55
4.3.2	Function Documentation	55
4.3.2.1	is_mapped	55
4.3.2.2	read_maps	55
4.3.3	Variable Documentation	55
4.3.3.1	lastsignal	55
4.3.3.2	nsections	55
4.3.3.3	zfirst	56
4.4	wsh/include/colors.h File Reference	56
4.4.1	Macro Definition Documentation	56
4.4.1.1	BLACK	56
4.4.1.2	BLUE	56
4.4.1.3	BROWN	56
4.4.1.4	CLEAR	56
4.4.1.5	CYAN	56
4.4.1.6	DARKGRAY	56
4.4.1.7	GRAY	56
4.4.1.8	GREEN	57
4.4.1.9	MAGENTA	57
4.4.1.10	NORMAL	57
4.4.1.11	RED	57
4.4.1.12	YELLOW	57
4.5	wsh/include/lauxlib.h File Reference	57
4.5.1	Macro Definition Documentation	59

4.5.1.1	LUA_ERRFILE	59
4.5.1.2	LUA_FILEHANDLE	59
4.5.1.3	LUA_NOREF	59
4.5.1.4	LUA_REFNIL	59
4.5.1.5	lua_writeline	59
4.5.1.6	lua_writestring	59
4.5.1.7	lua_writestringerror	59
4.5.1.8	luaL_addchar	59
4.5.1.9	luaL_addsize	59
4.5.1.10	luaL_argcheck	60
4.5.1.11	luaL_checkstring	60
4.5.1.12	luaL_checkversion	60
4.5.1.13	luaL_dofile	60
4.5.1.14	luaL_dostring	60
4.5.1.15	luaL_getmetatable	60
4.5.1.16	luaL_loadbuffer	60
4.5.1.17	luaL_loadfile	60
4.5.1.18	luaL_newlib	60
4.5.1.19	luaL_newlibtable	60
4.5.1.20	LUAL_NUMSIZES	60
4.5.1.21	luaL_opt	60
4.5.1.22	luaL_optstring	61
4.5.1.23	luaL_prepbuffer	61
4.5.1.24	luaL_typename	61
4.5.2	Typedef Documentation	61
4.5.2.1	luaL_Buffer	61
4.5.2.2	luaL_Reg	61
4.5.2.3	luaL_Stream	61
4.5.3	Function Documentation	61
4.5.3.1	luaL_addstring	61
4.5.3.2	luaL_addstring	61
4.5.3.3	luaL_addvalue	61
4.5.3.4	luaL_argerror	61
4.5.3.5	luaL_buffinit	61
4.5.3.6	luaL_buffinitsize	61
4.5.3.7	luaL_callmeta	61
4.5.3.8	luaL_checkany	61
4.5.3.9	luaL_checkinteger	61
4.5.3.10	luaL_checklstring	61
4.5.3.11	luaL_checknumber	61

4.5.3.12	luaL_checkoption	61
4.5.3.13	luaL_checkstack	61
4.5.3.14	luaL_checktype	61
4.5.3.15	luaL_checkudata	61
4.5.3.16	luaL_checkversion_	61
4.5.3.17	luaL_error	62
4.5.3.18	luaL_execresult	62
4.5.3.19	luaL_fileresult	62
4.5.3.20	luaL_getmetafield	62
4.5.3.21	luaL_getsubtable	62
4.5.3.22	luaL_gsub	62
4.5.3.23	luaL_len	62
4.5.3.24	luaL_loadbufferx	62
4.5.3.25	luaL_loadfilex	62
4.5.3.26	luaL_loadstring	62
4.5.3.27	luaL_newmetatable	62
4.5.3.28	luaL_newstate	62
4.5.3.29	luaL_optinteger	62
4.5.3.30	luaL_optlstring	62
4.5.3.31	luaL_optnumber	62
4.5.3.32	luaL_prebuffsize	62
4.5.3.33	luaL_pushresult	62
4.5.3.34	luaL_pushresultsizes	62
4.5.3.35	luaL_ref	62
4.5.3.36	luaL_requiref	62
4.5.3.37	luaL_setfuncs	62
4.5.3.38	luaL_setmetatable	62
4.5.3.39	luaL_testudata	62
4.5.3.40	luaL_tolstring	62
4.5.3.41	luaL_traceback	62
4.5.3.42	luaL_unref	62
4.5.3.43	luaL_where	62
4.6	wsh/include/libwitch/helper.h File Reference	63
4.6.1	Function Documentation	63
4.6.1.1	is_mapped	63
4.6.1.2	read_maps	63
4.6.2	Variable Documentation	63
4.6.2.1	nsections	63
4.6.2.2	zfirst	63
4.7	wsh/include/libwitch/mylaux.h File Reference	63

4.7.1	Macro Definition Documentation	64
4.7.1.1	luaL_argcheck	64
4.7.1.2	luaL_checkstring	64
4.7.1.3	luaL_dofile	64
4.7.1.4	luaL_dostring	64
4.7.1.5	luaL_getmetatable	64
4.7.1.6	luaL_loadbuffer	64
4.7.1.7	luaL_newlib	64
4.7.1.8	luaL_newlibtable	64
4.7.1.9	luaL_opt	64
4.7.1.10	luaL_optstring	64
4.7.1.11	luaL_typename	64
4.8	wsh/include/libwitch/sigs.h File Reference	65
4.8.1	Typedef Documentation	65
4.8.1.1	signame_t	65
4.8.2	Variable Documentation	65
4.8.2.1	signames	65
4.9	wsh/include/libwitch/wsh.h File Reference	66
4.9.1	Macro Definition Documentation	70
4.9.1.1	_GNU_SOURCE	70
4.9.1.2	BIND_FLAGS	70
4.9.1.3	DEFAULT_LEARN_FILE	70
4.9.1.4	default_poison	70
4.9.1.5	DEFAULT_SCRIPT	70
4.9.1.6	DEFAULT_SCRIPT_INDEX	70
4.9.1.7	DMGL_ANSI	70
4.9.1.8	DMGL_ARM	70
4.9.1.9	DMGL_PARAMS	70
4.9.1.10	ELF32_ST_BIND	70
4.9.1.11	ELF32_ST_INFO	70
4.9.1.12	ELF32_ST_TYPE	70
4.9.1.13	ELF64_ST_BIND	71
4.9.1.14	ELF64_ST_INFO	71
4.9.1.15	ELF64_ST_TYPE	71
4.9.1.16	Elf_Dyn	71
4.9.1.17	Elf_Ehdr	71
4.9.1.18	Elf_Phdr	71
4.9.1.19	Elf_Shdr	71
4.9.1.20	Elf_Sym	71
4.9.1.21	FAULT_EXEC	71

4.9.1.22	FAULT_READ	71
4.9.1.23	FAULT_WRITE	71
4.9.1.24	HPERMSMAX	71
4.9.1.25	LINES_MAX	72
4.9.1.26	luaL_reg	72
4.9.1.27	MAX_SIGNALS	72
4.9.1.28	MIN_BIN_SIZE	72
4.9.1.29	MY_CPU	72
4.9.1.30	PROC_ASLR_PATH	72
4.9.1.31	read_arg	72
4.9.1.32	read_arg1	72
4.9.1.33	read_arg2	73
4.9.1.34	read_arg3	73
4.9.1.35	read_arg4	74
4.9.1.36	SHELL_HISTORY_NAME	74
4.9.1.37	SKIP_BOTTOM	74
4.9.1.38	SKIP_INIT	74
4.9.1.39	STB_GLOBAL	74
4.9.1.40	STB_GNU_SECONDARY	74
4.9.1.41	STB_GNU_UNIQUE	74
4.9.1.42	STB_LOCAL	74
4.9.1.43	STB_WEAK	75
4.9.1.44	STT_COMMON	75
4.9.1.45	STT_FILE	75
4.9.1.46	STT_FUNC	75
4.9.1.47	STT_NOTYPE	75
4.9.1.48	STT_OBJECT	75
4.9.1.49	STT_SECTION	75
4.9.1.50	STT_TLS	75
4.9.1.51	USE_LUA	75
4.9.2	Typedef Documentation	75
4.9.2.1	breakpoint_t	75
4.9.2.2	eps_t	75
4.9.2.3	preload_t	75
4.9.2.4	range_t	76
4.9.2.5	script_t	76
4.9.2.6	sections_t	76
4.9.2.7	segments_t	76
4.9.2.8	symbols_t	76
4.9.2.9	tuple_t	76

4.9.2.10	wsh_t	76
4.9.3	Function Documentation	76
4.9.3.1	add_symbol	76
4.9.3.2	alloccharbuf	76
4.9.3.3	bfmap	76
4.9.3.4	breakpoint	76
4.9.3.5	bsspolute	77
4.9.3.6	cplus_demangle	77
4.9.3.7	disable_aslr	77
4.9.3.8	disable_core	77
4.9.3.9	do_loadlib	77
4.9.3.10	empty_phdrs	77
4.9.3.11	empty_shdrs	77
4.9.3.12	enable_aslr	77
4.9.3.13	enable_core	78
4.9.3.14	entrypoints	78
4.9.3.15	execlib	78
4.9.3.16	gencore	78
4.9.3.17	getcharbuf	78
4.9.3.18	getsize	78
4.9.3.19	grep	78
4.9.3.20	grepptr	78
4.9.3.21	headers	78
4.9.3.22	help	78
4.9.3.23	hexdump	79
4.9.3.24	hollywood	79
4.9.3.25	info	79
4.9.3.26	libcall	79
4.9.3.27	loadbin	80
4.9.3.28	ltrace	80
4.9.3.29	man	80
4.9.3.30	map	80
4.9.3.31	newarray	80
4.9.3.32	phdrs	80
4.9.3.33	print_functions	80
4.9.3.34	print_libs	80
4.9.3.35	print_objects	80
4.9.3.36	print_phdrs	81
4.9.3.37	print_shdrs	81
4.9.3.38	print_symbols	81

4.9.3.39	print_version	81
4.9.3.40	priv_memcpy	81
4.9.3.41	priv_strcat	81
4.9.3.42	priv_strcpy	81
4.9.3.43	procmmap_lua	81
4.9.3.44	prototypes	81
4.9.3.45	ralloc	82
4.9.3.46	rawmemaddr	82
4.9.3.47	rawmemread	82
4.9.3.48	rawmemstr	82
4.9.3.49	rawmemstrlen	82
4.9.3.50	rawmemusage	82
4.9.3.51	rawmemwrite	82
4.9.3.52	rdnum	82
4.9.3.53	rdstr	83
4.9.3.54	reload_elfs	83
4.9.3.55	rescan	83
4.9.3.56	rtrace	83
4.9.3.57	script	83
4.9.3.58	segment_add	83
4.9.3.59	set_align_flag	83
4.9.3.60	set_branch_flag	83
4.9.3.61	set_trace_flag	83
4.9.3.62	setarray	83
4.9.3.63	setcharbuf	83
4.9.3.64	shdrs	84
4.9.3.65	sicode_strerror	84
4.9.3.66	signaltoname	84
4.9.3.67	singlebranch	84
4.9.3.68	singlestep	84
4.9.3.69	systrace	84
4.9.3.70	traceunaligned	84
4.9.3.71	unrtrace	84
4.9.3.72	unset_align_flag	84
4.9.3.73	unset_branch_flag	84
4.9.3.74	unset_trace_flag	84
4.9.3.75	unsinglebranch	84
4.9.3.76	unsinglestep	85
4.9.3.77	unsystrace	85
4.9.3.78	untraceunaligned	85

4.9.3.79	<code>unverbosetrace</code>	85
4.9.3.80	<code>usage</code>	85
4.9.3.81	<code>verbose</code>	85
4.9.3.82	<code>verbosetrace</code>	85
4.9.3.83	<code>wsh_getopt</code>	85
4.9.3.84	<code>wsh_init</code>	85
4.9.3.85	<code>wsh_loadlibs</code>	85
4.9.3.86	<code>wsh_run</code>	85
4.9.3.87	<code>xalloc</code>	86
4.9.3.88	<code>xfree</code>	86
4.9.4	Variable Documentation	86
4.9.4.1	<code>__programe_full</code>	86
4.10	<code>wsh/include/libwitch/wsh_functions.h</code> File Reference	86
4.10.1	Variable Documentation	86
4.10.1.1	<code>default_options</code>	86
4.10.1.2	<code>exposed</code>	86
4.10.1.3	<code>global_xalloc</code>	86
4.10.1.4	<code>lua_blacklist</code>	86
4.10.1.5	<code>lua_default_functions</code>	87
4.10.1.6	<code>ranges</code>	87
4.11	<code>wsh/include/linenoise.h</code> File Reference	87
4.11.1	Typedef Documentation	88
4.11.1.1	<code>linenoiseCompletionCallback</code>	88
4.11.1.2	<code>linenoiseCompletions</code>	88
4.11.2	Function Documentation	88
4.11.2.1	<code>linenoise</code>	88
4.11.2.2	<code>linenoiseAddCompletion</code>	88
4.11.2.3	<code>linenoiseClearScreen</code>	88
4.11.2.4	<code>linenoiseHistoryAdd</code>	88
4.11.2.5	<code>linenoiseHistoryLoad</code>	88
4.11.2.6	<code>linenoiseHistorySave</code>	88
4.11.2.7	<code>linenoiseHistorySetMaxLen</code>	88
4.11.2.8	<code>linenoisePrintKeyCodes</code>	88
4.11.2.9	<code>linenoiseSetCompletionCallback</code>	88
4.11.2.10	<code>linenoiseSetMultiLine</code>	88
4.12	<code>wsh/include/longjmp.h</code> File Reference	88
4.12.1	Macro Definition Documentation	89
4.12.1.1	<code>CATCH</code>	89
4.12.1.2	<code>ENTRY</code>	89
4.12.1.3	<code>FINALLY</code>	89

4.12.1.4	THROW	89
4.12.1.5	TRY	89
4.13	wsh/include/lua.h File Reference	89
4.13.1	Macro Definition Documentation	93
4.13.1.1	LUA_AUTHORS	93
4.13.1.2	lua_call	93
4.13.1.3	LUA_COPYRIGHT	93
4.13.1.4	LUA_ERRERR	93
4.13.1.5	LUA_ERRGCOMM	94
4.13.1.6	LUA_ERRMEM	94
4.13.1.7	LUA_ERRRUN	94
4.13.1.8	LUA_ERRSYNTAX	94
4.13.1.9	LUA_GCCOLLECT	94
4.13.1.10	LUA_GCCOUNT	94
4.13.1.11	LUA_GCCOUNTB	94
4.13.1.12	LUA_GCISRUNNING	94
4.13.1.13	LUA_GCRESTART	94
4.13.1.14	LUA_GCSETPAUSE	94
4.13.1.15	LUA_GCSETSTEPMUL	94
4.13.1.16	LUA_GCSTEP	94
4.13.1.17	LUA_GCSTOP	95
4.13.1.18	lua_getextraspaces	95
4.13.1.19	LUA_HOOKCALL	95
4.13.1.20	LUA_HOOKCOUNT	95
4.13.1.21	LUA_HOOKLINE	95
4.13.1.22	LUA_HOOKRET	95
4.13.1.23	LUA_HOOKTAILCALL	95
4.13.1.24	lua_insert	95
4.13.1.25	lua_isboolean	95
4.13.1.26	lua_isfunction	95
4.13.1.27	lua_isthread	95
4.13.1.28	lua_isnil	95
4.13.1.29	lua_isnone	96
4.13.1.30	lua_isnoneornil	96
4.13.1.31	lua_istable	96
4.13.1.32	lua_isthread	96
4.13.1.33	LUA_MASKCALL	96
4.13.1.34	LUA_MASKCOUNT	96
4.13.1.35	LUA_MASKLINE	96
4.13.1.36	LUA_MASKRET	96

4.13.1.37	LUA_MINSTACK	96
4.13.1.38	LUA_MULTRET	96
4.13.1.39	lua_newtable	96
4.13.1.40	LUA_NUMTAGS	96
4.13.1.41	LUA_OK	97
4.13.1.42	LUA_OPADD	97
4.13.1.43	LUA_OPBAND	97
4.13.1.44	LUA_OPBNOT	97
4.13.1.45	LUA_OPBOR	97
4.13.1.46	LUA_OPBXOR	97
4.13.1.47	LUA_OPDIV	97
4.13.1.48	LUA_OPEQ	97
4.13.1.49	LUA_OPIDIV	97
4.13.1.50	LUA_OPLE	97
4.13.1.51	LUA_OPLT	97
4.13.1.52	LUA_OPMOD	97
4.13.1.53	LUA_OPMUL	98
4.13.1.54	LUA_OPPOW	98
4.13.1.55	LUA_OPSHL	98
4.13.1.56	LUA_OPSHR	98
4.13.1.57	LUA_OPSUB	98
4.13.1.58	LUA_OPUNM	98
4.13.1.59	lua_pcall	98
4.13.1.60	lua_pop	98
4.13.1.61	lua_pushcfunction	98
4.13.1.62	lua_pushglobaltable	98
4.13.1.63	lua_pushliteral	98
4.13.1.64	lua_register	98
4.13.1.65	LUA_REGISTRYINDEX	99
4.13.1.66	LUA_RELEASE	99
4.13.1.67	lua_remove	99
4.13.1.68	lua_replace	99
4.13.1.69	LUA_RIDX_GLOBALS	99
4.13.1.70	LUA_RIDX_LAST	99
4.13.1.71	LUA_RIDX_MAINTHREAD	99
4.13.1.72	LUA_SIGNATURE	99
4.13.1.73	LUA_TBOOLEAN	99
4.13.1.74	LUA_TFUNCTION	99
4.13.1.75	LUA_TLIGHTUSERDATA	99
4.13.1.76	LUA_TNIL	99

4.13.1.77	LUA_TNONE	100
4.13.1.78	LUA_TNUMBER	100
4.13.1.79	lua_tointeger	100
4.13.1.80	lua_tonumber	100
4.13.1.81	lua_tostring	100
4.13.1.82	LUA_TSTRING	100
4.13.1.83	LUA_TTABLE	100
4.13.1.84	LUA_TTHREAD	100
4.13.1.85	LUA_TUSERDATA	100
4.13.1.86	lua_upvalueindex	100
4.13.1.87	LUA_VERSION	100
4.13.1.88	LUA_VERSION_MAJOR	100
4.13.1.89	LUA_VERSION_MINOR	101
4.13.1.90	LUA_VERSION_NUM	101
4.13.1.91	LUA_VERSION_RELEASE	101
4.13.1.92	LUA_YIELD	101
4.13.1.93	lua_yield	101
4.13.2	Typedef Documentation	101
4.13.2.1	lua_Alloc	101
4.13.2.2	lua_CFunction	101
4.13.2.3	lua_Debug	101
4.13.2.4	lua_Hook	101
4.13.2.5	lua_Integer	101
4.13.2.6	lua_KContext	101
4.13.2.7	lua_KFunction	101
4.13.2.8	lua_Number	102
4.13.2.9	lua_Reader	102
4.13.2.10	lua_State	102
4.13.2.11	lua_Unsigned	102
4.13.2.12	lua_Writer	102
4.13.3	Function Documentation	102
4.13.3.1	lua_absindex	102
4.13.3.2	lua_arith	102
4.13.3.3	lua_atpanic	102
4.13.3.4	lua_callk	102
4.13.3.5	lua_checkstack	102
4.13.3.6	lua_close	102
4.13.3.7	lua_compare	102
4.13.3.8	lua_concat	102
4.13.3.9	lua_copy	102

4.13.3.10 lua_createtable	102
4.13.3.11 lua_dump	102
4.13.3.12 lua_error	102
4.13.3.13 lua_gc	102
4.13.3.14 lua_getallocf	102
4.13.3.15 lua_getfield	102
4.13.3.16 lua_getglobal	103
4.13.3.17 lua_gethook	103
4.13.3.18 lua_gethookcount	103
4.13.3.19 lua_gethookmask	103
4.13.3.20 lua_geti	103
4.13.3.21 lua_getinfo	103
4.13.3.22 lua_getlocal	103
4.13.3.23 lua_getmetatable	103
4.13.3.24 lua_getstack	103
4.13.3.25 lua_gettable	103
4.13.3.26 lua_gettop	103
4.13.3.27 lua_getupvalue	103
4.13.3.28 lua_getuservalue	103
4.13.3.29 lua_iscfunction	103
4.13.3.30 lua_isinteger	103
4.13.3.31 lua_isnumber	103
4.13.3.32 lua_isstring	103
4.13.3.33 lua_isuserdata	103
4.13.3.34 lua_isyieldable	103
4.13.3.35 lua_len	103
4.13.3.36 lua_load	103
4.13.3.37 lua_newstate	103
4.13.3.38 lua_newthread	103
4.13.3.39 lua_newuserdata	103
4.13.3.40 lua_next	103
4.13.3.41 lua_pcallk	103
4.13.3.42 lua_pushboolean	103
4.13.3.43 lua_pushcclosure	104
4.13.3.44 lua_pushfstring	104
4.13.3.45 lua_pushinteger	104
4.13.3.46 lua_pushlightuserdata	104
4.13.3.47 lua_pushlstring	104
4.13.3.48 lua_pushnil	104
4.13.3.49 lua_pushnumber	104

4.13.3.50 lua_pushstring	104
4.13.3.51 lua_pushthread	104
4.13.3.52 lua_pushvalue	104
4.13.3.53 lua_pushvfstring	104
4.13.3.54 lua_rawequal	104
4.13.3.55 lua_rawget	104
4.13.3.56 lua_rawgeti	104
4.13.3.57 lua_rawgetp	104
4.13.3.58 lua_rawlen	104
4.13.3.59 lua_rawset	104
4.13.3.60 lua_rawseti	104
4.13.3.61 lua_rawsetp	104
4.13.3.62 lua_resume	104
4.13.3.63 lua_rotate	104
4.13.3.64 lua_setallocf	104
4.13.3.65 lua_setfield	104
4.13.3.66 lua_setglobal	104
4.13.3.67 lua_sethook	104
4.13.3.68 lua_seti	104
4.13.3.69 lua_setlocal	104
4.13.3.70 lua_setmetatable	104
4.13.3.71 lua_settable	105
4.13.3.72 lua_settop	105
4.13.3.73 lua_setupvalue	105
4.13.3.74 lua_setuservalue	105
4.13.3.75 lua_status	105
4.13.3.76 lua_stringtonumber	105
4.13.3.77 lua_toboolean	105
4.13.3.78 lua_tocfunction	105
4.13.3.79 lua_tointegerx	105
4.13.3.80 lua_tolstring	105
4.13.3.81 lua_tonumberx	105
4.13.3.82 lua_topointer	105
4.13.3.83 lua_tothread	105
4.13.3.84 lua_touserdata	105
4.13.3.85 lua_type	105
4.13.3.86 lua_typename	105
4.13.3.87 lua_upvalueid	105
4.13.3.88 lua_upvaluejoin	105
4.13.3.89 lua_version	105

4.13.3.90 lua_xmove	105
4.13.3.91 lua_yieldk	105
4.13.4 Variable Documentation	105
4.13.4.1 lua_ident	105
4.14 wsh/include/luafunc.h File Reference	105
4.14.1 Macro Definition Documentation	106
4.14.1.1 l_floor	106
4.14.1.2 l_mathlim	107
4.14.1.3 l_mathop	107
4.14.1.4 l_sprintf	107
4.14.1.5 LUA_API	107
4.14.1.6 LUA_CDIRENTRY	107
4.14.1.7 LUA_CPATH_DEFAULT	107
4.14.1.8 LUA_DIRSEP	107
4.14.1.9 LUA_EXTRASPACE	107
4.14.1.10 LUA_FLOAT_DOUBLE	107
4.14.1.11 LUA_FLOAT_FLOAT	107
4.14.1.12 LUA_FLOAT_LONGDOUBLE	107
4.14.1.13 LUA_FLOAT_TYPE	107
4.14.1.14 lua_getlocaledecpoint	108
4.14.1.15 LUA_IDSIZE	108
4.14.1.16 LUA_INT_INT	108
4.14.1.17 LUA_INT_LONG	108
4.14.1.18 LUA_INT_LONGLONG	108
4.14.1.19 LUA_INT_TYPE	108
4.14.1.20 lua_integer2str	108
4.14.1.21 LUA_INTEGER_FMT	108
4.14.1.22 LUA_KCONTEXT	108
4.14.1.23 LUA_LDIR	108
4.14.1.24 LUA_NUMBER	108
4.14.1.25 lua_number2str	108
4.14.1.26 lua_number2strx	109
4.14.1.27 LUA_NUMBER_FMT	109
4.14.1.28 LUA_NUMBER_FRMLEN	109
4.14.1.29 lua_numbertointeger	109
4.14.1.30 LUA_PATH_DEFAULT	109
4.14.1.31 LUA_QL	109
4.14.1.32 LUA_QS	109
4.14.1.33 LUA_ROOT	109
4.14.1.34 lua_str2number	109

4.14.1.35 lua_strx2number	109
4.14.1.36 LUA_UNSIGNED	110
4.14.1.37 LUA_VDIR	110
4.14.1.38 LUAI_BITSINT	110
4.14.1.39 LUAI_DDEC	110
4.14.1.40 LUAI_DDEF	110
4.14.1.41 LUAI_FUNC	110
4.14.1.42 LUAI_MAXSTACK	110
4.14.1.43 LUAI_UACINT	110
4.14.1.44 LUAI_UACNUMBER	110
4.14.1.45 LUAL_BUFFERSIZE	110
4.14.1.46 LUALIB_API	110
4.14.1.47 LUAMOD_API	110
4.15 wsh/include/lualib.h File Reference	111
4.15.1 Macro Definition Documentation	111
4.15.1.1 lua_assert	111
4.15.1.2 LUA_BITLIBNAME	111
4.15.1.3 LUA_COLIBNAME	111
4.15.1.4 LUA_DBLIBNAME	111
4.15.1.5 LUA_IOLIBNAME	112
4.15.1.6 LUA_LOADLIBNAME	112
4.15.1.7 LUA_MATHLIBNAME	112
4.15.1.8 LUA_OSLIBNAME	112
4.15.1.9 LUA_STRLIBNAME	112
4.15.1.10 LUA_TABLIBNAME	112
4.15.1.11 LUA_UTF8LIBNAME	112
4.15.2 Function Documentation	112
4.15.2.1 luaL_openlibs	112
4.15.2.2 luaopen_base	112
4.15.2.3 luaopen_bit32	112
4.15.2.4 luaopen_coroutine	112
4.15.2.5 luaopen_debug	112
4.15.2.6 luaopen_io	112
4.15.2.7 luaopen_math	112
4.15.2.8 luaopen_os	112
4.15.2.9 luaopen_package	112
4.15.2.10 luaopen_string	112
4.15.2.11 luaopen_table	112
4.15.2.12 luaopen_utf8	113
4.16 wsh/wsh.c File Reference	113

4.16.1	Macro Definition Documentation	116
4.16.1.1	CS_MODE	116
4.16.1.2	Elf_Addr	117
4.16.1.3	Elf_Ehdr	117
4.16.1.4	Elf_Off	117
4.16.1.5	Elf_Phdr	117
4.16.1.6	ELF_R_INFO	117
4.16.1.7	ELF_R_SYM	117
4.16.1.8	ELF_R_TYPE	117
4.16.1.9	Elf_Rel	117
4.16.1.10	Elf_Rela	117
4.16.1.11	Elf_Section	117
4.16.1.12	Elf_Shdr	117
4.16.1.13	ELF_ST_BIND	117
4.16.1.14	ELF_ST_TYPE	118
4.16.1.15	Elf_Sword	118
4.16.1.16	Elf_Sym	118
4.16.1.17	Elf_Word	118
4.16.1.18	Elf_Xword	118
4.16.1.19	ELFCLASS	118
4.16.1.20	ELFMACHINE	118
4.16.1.21	REG_RIP	118
4.16.1.22	RELOC_MODE	119
4.16.2	Typedef Documentation	119
4.16.2.1	help_t	119
4.16.2.2	learn_key_t	119
4.16.2.3	learn_t	119
4.16.3	Function Documentation	119
4.16.3.1	_exit	119
4.16.3.2	add_binary_preload	119
4.16.3.3	add_script_arguments	119
4.16.3.4	add_script_exec	119
4.16.3.5	add_symbol	119
4.16.3.6	affinity	119
4.16.3.7	alarmhandler	119
4.16.3.8	alloccharbuf	120
4.16.3.9	bfmap	120
4.16.3.10	breakpoint	120
4.16.3.11	bsspolute	120
4.16.3.12	btr_disable	120

4.16.3.13 btr_enable	120
4.16.3.14 bushandler	120
4.16.3.15 completion	120
4.16.3.16 declare_func	121
4.16.3.17 declare_internals	121
4.16.3.18 declare_num	121
4.16.3.19 decode_flags	121
4.16.3.20 decode_type	121
4.16.3.21 detailed_help	121
4.16.3.22 disable_aslr	121
4.16.3.23 disable_core	121
4.16.3.24 do_loadlib	121
4.16.3.25 empty_eps	122
4.16.3.26 empty_phdrs	122
4.16.3.27 empty_shdrs	122
4.16.3.28 empty_symbols	122
4.16.3.29 enable_aslr	122
4.16.3.30 enable_core	122
4.16.3.31 entry_point_add	122
4.16.3.32 entrypoints	122
4.16.3.33 execlib	122
4.16.3.34 exit	122
4.16.3.35 exit_group	123
4.16.3.36 fatal_error	123
4.16.3.37 gencore	123
4.16.3.38 getcharbuf	123
4.16.3.39 grep	123
4.16.3.40 grepptr	123
4.16.3.41 headers	123
4.16.3.42 help	123
4.16.3.43 hexdump	123
4.16.3.44 hollywood	124
4.16.3.45 info	124
4.16.3.46 info_function	124
4.16.3.47 inthandler	124
4.16.3.48 learn_proto	124
4.16.3.49 libcall	124
4.16.3.50 loadbin	125
4.16.3.51 loadlibrary	125
4.16.3.52 ltrace	125

4.16.3.53 lua_strerror	125
4.16.3.54 man	125
4.16.3.55 map	125
4.16.3.56 mk_backtrace	125
4.16.3.57 parse_dyn	125
4.16.3.58 parse_link_map_dyn	126
4.16.3.59 phdr_callback	126
4.16.3.60 phdr_cmp	126
4.16.3.61 phdrs	126
4.16.3.62 print_backtrace	126
4.16.3.63 print_eps	126
4.16.3.64 print_functions	126
4.16.3.65 print_libs	126
4.16.3.66 print_objects	126
4.16.3.67 print_phdrs	127
4.16.3.68 print_procmap	127
4.16.3.69 print_shdrs	127
4.16.3.70 print_symbols	127
4.16.3.71 printarg	127
4.16.3.72 priv_memcpy	127
4.16.3.73 priv_strcat	127
4.16.3.74 priv_strcpy	127
4.16.3.75 procmap_lua	127
4.16.3.76 prototypes	127
4.16.3.77 ptoh	128
4.16.3.78 ralloc	128
4.16.3.79 rawmemaddr	128
4.16.3.80 rawmemread	128
4.16.3.81 rawmemstr	128
4.16.3.82 rawmemstrlen	128
4.16.3.83 rawmemusage	128
4.16.3.84 rawmemwrite	128
4.16.3.85 rdnum	129
4.16.3.86 rdstr	129
4.16.3.87 read_elf_sig	129
4.16.3.88 reload_elfs	129
4.16.3.89 rescan	129
4.16.3.90 restore_exit	129
4.16.3.91 rtrace	129
4.16.3.92 run_script	129

4.16.3.93	run_shell	129
4.16.3.94	scan_section	130
4.16.3.95	scan_sections	130
4.16.3.96	scan_symbol	130
4.16.3.97	scan_syms	130
4.16.3.98	script	130
4.16.3.99	section_add	130
4.16.3.100	section_from_addr	130
4.16.3.101	segment_add	130
4.16.3.102	segment_from_addr	131
4.16.3.103	set_align_flag	131
4.16.3.104	set_alloc_opt	131
4.16.3.105	set_branch_flag	131
4.16.3.106	set_sighandlers	131
4.16.3.107	set_trace_flag	131
4.16.3.108	setcharbuf	131
4.16.3.109	shdr_callback	131
4.16.3.110	shdr_cmp	131
4.16.3.111	shdrs	131
4.16.3.112	sicode_strerror	131
4.16.3.113	sicodetname	132
4.16.3.114	sighandler	132
4.16.3.115	signaltoname	132
4.16.3.116	singlebranch	132
4.16.3.117	singlestep	132
4.16.3.118	sort_learnt	132
4.16.3.119	symbol_from_addr	132
4.16.3.120	symbol_from_name	132
4.16.3.121	symbol_tobind	132
4.16.3.122	symbol_totype	132
4.16.3.123	systrace	133
4.16.3.124	test_stdin	133
4.16.3.125	traceback	133
4.16.3.126	traceunaligned	133
4.16.3.127	traphandler	133
4.16.3.128	unrtrace	133
4.16.3.129	unset_align_flag	133
4.16.3.130	unset_branch_flag	133
4.16.3.131	unset_trace_flag	133
4.16.3.132	unsinglebranch	133

4.16.3.133	unsinglestep	134
4.16.3.134	unsystrace	134
4.16.3.135	untraceunaligned	134
4.16.3.136	unverbosetrace	134
4.16.3.137	verbose	134
4.16.3.138	verbosetrace	134
4.16.3.139	wsh_getopt	134
4.16.3.140	wsh_init	134
4.16.3.141	wsh_loadlibs	134
4.16.3.142	wsh_print_version	134
4.16.3.143	wsh_run	134
4.16.3.144	wsh_usage	135
4.16.3.145	xalloc	135
4.16.3.146	xfree	135
4.16.4	Variable Documentation	135
4.16.4.1	cmdhelp	135
4.16.4.2	fcnhelp	135
4.16.4.3	protorecords	136
4.16.4.4	wsh	136
4.17	wsh/wshmain.c File Reference	137
4.17.1	Function Documentation	137
4.17.1.1	main	137
4.17.2	Variable Documentation	137
4.17.2.1	wsh	137

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

breakpoint_t	5
ctx_t	5
elfdata_t	9
eps_t	11
gimport_t	11
help_t	12
learn_key_t	13
learn_t	14
linenoiseCompletions	14
lua_Debug	15
luaL_Buffer	17
luaL_Reg	18
luaL_Stream	18
msec_t	19
mseg_t	20
preload_t	21
range_t	22
script_t	23
section	23
sections_t	25
segments_t	26
signame_t	27
symaddr	27
symbols_t	28
tuple_t	29
wsh_t	30

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

wcc/wcc.c	37
wld/wld.c	53
wsh/helper.c	54
wsh/wsh.c	113
wsh/wshmain.c	137
wsh/include/colors.h	56
wsh/include/lauxlib.h	57
wsh/include/linenoise.h	87
wsh/include/longjmp.h	88
wsh/include/lua.h	89
wsh/include/luaconf.h	105
wsh/include/lualib.h	111
wsh/include/libwitch/helper.h	63
wsh/include/libwitch/mylaux.h	63
wsh/include/libwitch/sigs.h	65
wsh/include/libwitch/wsh.h	66
wsh/include/libwitch/wsh_functions.h	86

Chapter 3

Data Structure Documentation

3.1 breakpoint_t Struct Reference

```
#include <wsh.h>
```

Data Fields

- char * [ptr](#)
- char [backup](#)
- unsigned int [weight](#)

3.1.1 Detailed Description

Breakpoint structure

Definition at line 442 of file wsh.h.

3.1.2 Field Documentation

3.1.2.1 char breakpoint_t::backup

Definition at line 444 of file wsh.h.

3.1.2.2 char* breakpoint_t::ptr

Definition at line 443 of file wsh.h.

3.1.2.3 unsigned int breakpoint_t::weight

Definition at line 445 of file wsh.h.

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

- [wsh/include/libwitch/wsh.h](#)

3.2 ctx_t Struct Reference

Data Fields

- char * [binname](#)
- unsigned int [archsz](#)
- unsigned int [shnum](#)
- unsigned int [phnum](#)
- char * [strndx](#)
- unsigned int [strndx_len](#)
- unsigned int [strndx_index](#)
- unsigned int [start_shdrs](#)
- unsigned int [start_phdrs](#)
- int [fdout](#)
- bfd * [abfd](#)
- unsigned int [corefile](#)
- unsigned int [base_address](#)
- [msec_t](#) * [mshdrs](#)
- unsigned int [mshnum](#)
- [mseg_t](#) * [mphdrs](#)
- unsigned int [mphnum](#)
- unsigned int [has_relativerelocations](#)
- char * [opt_binname](#)
- char * [opt_interp](#)
- unsigned int [opt_arch](#)
- unsigned int [opt_static](#)
- unsigned int [opt_reloc](#)
- unsigned int [opt_strip](#)
- unsigned int [opt_sstrip](#)
- unsigned int [opt_exec](#)
- unsigned int [opt_core](#)
- unsigned int [opt_shared](#)
- unsigned int [opt_verbose](#)
- unsigned long int [opt_entrypoint](#)
- unsigned char [opt_poison](#)
- unsigned int [opt_original](#)
- unsigned int [opt_debug](#)
- unsigned int [opt_asmdebug](#)
- unsigned int [opt_flags](#)

3.2.1 Detailed Description

Definition at line 264 of file wcc.c.

3.2.2 Field Documentation

3.2.2.1 bfd* ctx_t::abfd

Definition at line 279 of file wcc.c.

3.2.2.2 unsigned int ctx_t::archsz

Definition at line 270 of file wcc.c.

3.2.2.3 unsigned int ctx_t::base_address

Definition at line 283 of file wcc.c.

3.2.2.4 char* ctx_t::binname

Internal options

Definition at line 269 of file wcc.c.

3.2.2.5 unsigned int ctx_t::corefile

Definition at line 280 of file wcc.c.

3.2.2.6 int ctx_t::fdout

Definition at line 278 of file wcc.c.

3.2.2.7 unsigned int ctx_t::has_relativerelocations

Definition at line 293 of file wcc.c.

3.2.2.8 mseg_t* ctx_t::mphdrs

Definition at line 290 of file wcc.c.

3.2.2.9 unsigned int ctx_t::mphnum

Definition at line 291 of file wcc.c.

3.2.2.10 msec_t* ctx_t::mshdrs

Definition at line 286 of file wcc.c.

3.2.2.11 unsigned int ctx_t::mshnum

Definition at line 287 of file wcc.c.

3.2.2.12 unsigned int ctx_t::opt_arch

Definition at line 299 of file wcc.c.

3.2.2.13 unsigned int ctx_t::opt_asmdebug

Definition at line 312 of file wcc.c.

3.2.2.14 char* ctx_t::opt_binname

User options

Definition at line 297 of file wcc.c.

3.2.2.15 unsigned int ctx_t::opt_core

Definition at line 305 of file wcc.c.

3.2.2.16 unsigned int ctx_t::opt_debug

Definition at line 311 of file wcc.c.

3.2.2.17 unsigned long int ctx_t::opt_entrypoint

Definition at line 308 of file wcc.c.

3.2.2.18 unsigned int ctx_t::opt_exec

Definition at line 304 of file wcc.c.

3.2.2.19 unsigned int ctx_t::opt_flags

Definition at line 313 of file wcc.c.

3.2.2.20 char* ctx_t::opt_interp

Definition at line 298 of file wcc.c.

3.2.2.21 unsigned int ctx_t::opt_original

Definition at line 310 of file wcc.c.

3.2.2.22 unsigned char ctx_t::opt_poison

Definition at line 309 of file wcc.c.

3.2.2.23 unsigned int ctx_t::opt_reloc

Definition at line 301 of file wcc.c.

3.2.2.24 unsigned int ctx_t::opt_shared

Definition at line 306 of file wcc.c.

3.2.2.25 unsigned int ctx_t::opt_sstrip

Definition at line 303 of file wcc.c.

3.2.2.26 unsigned int ctx_t::opt_static

Definition at line 300 of file wcc.c.

3.2.2.27 unsigned int ctx_t::opt_strip

Definition at line 302 of file wcc.c.

3.2.2.28 unsigned int ctx_t::opt_verbose

Definition at line 307 of file wcc.c.

3.2.2.29 unsigned int ctx_t::phnum

Definition at line 272 of file wcc.c.

3.2.2.30 unsigned int ctx_t::shnum

Definition at line 271 of file wcc.c.

3.2.2.31 unsigned int ctx_t::start_phdrs

Definition at line 277 of file wcc.c.

3.2.2.32 unsigned int ctx_t::start_shdrs

Definition at line 276 of file wcc.c.

3.2.2.33 char* ctx_t::strndx

Definition at line 273 of file wcc.c.

3.2.2.34 unsigned int ctx_t::strndx_index

Definition at line 275 of file wcc.c.

3.2.2.35 unsigned int ctx_t::strndx_len

Definition at line 274 of file wcc.c.

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

- [wcc/wcc.c](#)

3.3 elfdata_t Struct Reference

```
#include <wsh.h>
```

Data Fields

- [bool et_dyn](#)
- [Elf_Dyn * dyns](#)
- [Elf_Ehdr * ehdr](#)
- [Elf_Phdr * phdrs](#)

- [uint32_t dyn_index](#)
- [uintptr_t base](#)
- [uintptr_t limit](#)
- [uintptr_t * p_pltgot](#)
- [struct r_debug * r_debug](#)
- [struct link_map * link_map](#)

3.3.1 Detailed Description

Internal representation of an ELF
Definition at line 417 of file wsh.h.

3.3.2 Field Documentation

3.3.2.1 `uintptr_t elfdata_t::base`

Definition at line 423 of file wsh.h.

3.3.2.2 `uint32_t elfdata_t::dyn_index`

Definition at line 422 of file wsh.h.

3.3.2.3 `Elf_Dyn* elfdata_t::dyns`

Definition at line 419 of file wsh.h.

3.3.2.4 `Elf_Ehdr* elfdata_t::ehdr`

Definition at line 420 of file wsh.h.

3.3.2.5 `bool elfdata_t::et_dyn`

Definition at line 418 of file wsh.h.

3.3.2.6 `uintptr_t elfdata_t::limit`

Definition at line 423 of file wsh.h.

3.3.2.7 `struct link_map* elfdata_t::link_map`

Definition at line 426 of file wsh.h.

3.3.2.8 `uintptr_t* elfdata_t::p_pltgot`

Definition at line 424 of file wsh.h.

3.3.2.9 `Elf_Phdr* elfdata_t::phdrs`

Definition at line 421 of file wsh.h.

3.3.2.10 struct r_debug* elfdata_t::r_debug

Definition at line 425 of file wsh.h.

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

- wsh/include/libwitch/wsh.h

3.4 eps_t Struct Reference

```
#include <wsh.h>
```

Data Fields

- unsigned long int [addr](#)
- char * [name](#)
- struct [eps_t](#) * [prev](#)
- struct [eps_t](#) * [next](#)

3.4.1 Detailed Description

Definition at line 520 of file wsh.h.

3.4.2 Field Documentation

3.4.2.1 unsigned long int eps_t::addr

Definition at line 521 of file wsh.h.

3.4.2.2 char* eps_t::name

Definition at line 522 of file wsh.h.

3.4.2.3 struct eps_t* eps_t::next

Definition at line 525 of file wsh.h.

3.4.2.4 struct eps_t* eps_t::prev

Definition at line 524 of file wsh.h.

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

- wsh/include/libwitch/wsh.h

3.5 gimport_t Struct Reference

Data Fields

- char * [sname](#)

- [msec_t](#) * [sec](#)
- [Elf_Rel](#) * [r](#)
- int [rtype](#)
- unsigned int [sindex](#)

3.5.1 Detailed Description

Definition at line 2770 of file [wcc.c](#).

3.5.2 Field Documentation

3.5.2.1 [Elf_Rel](#)* [gimport_t::r](#)

Definition at line 2773 of file [wcc.c](#).

3.5.2.2 int [gimport_t::rtype](#)

Definition at line 2774 of file [wcc.c](#).

3.5.2.3 [msec_t](#)* [gimport_t::sec](#)

Definition at line 2772 of file [wcc.c](#).

3.5.2.4 unsigned int [gimport_t::sindex](#)

Definition at line 2775 of file [wcc.c](#).

3.5.2.5 [char](#)* [gimport_t::sname](#)

Definition at line 2771 of file [wcc.c](#).

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

- [wcc/wcc.c](#)

3.6 [help_t](#) Struct Reference

Data Fields

- [char](#) * [name](#)
- [char](#) * [proto](#)
- [char](#) * [descr](#)
- [char](#) * [protoprefix](#)
- [char](#) * [retval](#)

3.6.1 Detailed Description

Definition at line 489 of file [wsh.c](#).

3.6.2 Field Documentation

3.6.2.1 char* help_t::descr

Definition at line 492 of file wsh.c.

3.6.2.2 char* help_t::name

Definition at line 490 of file wsh.c.

3.6.2.3 char* help_t::proto

Definition at line 491 of file wsh.c.

3.6.2.4 char* help_t::protoprefix

Definition at line 493 of file wsh.c.

3.6.2.5 char* help_t::retval

Definition at line 494 of file wsh.c.

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

- [wsh/wsh.c](#)

3.7 learn_key_t Struct Reference

Data Fields

- char [ttype](#) [10]
- char [tlib](#) [200]
- char [tfunction](#) [200]
- char [targ](#) [20]
- char [tvalue](#) [200]

3.7.1 Detailed Description

Definition at line 1861 of file wsh.c.

3.7.2 Field Documentation

3.7.2.1 char learn_key_t::targ[20]

Definition at line 1866 of file wsh.c.

3.7.2.2 char learn_key_t::tfunction[200]

Definition at line 1865 of file wsh.c.

3.7.2.3 char learn_key_t::tlib[200]

Definition at line 1864 of file wsh.c.

3.7.2.4 char learn_key_t::ttype[10]

Definition at line 1863 of file wsh.c.

3.7.2.5 char learn_key_t::tvalue[200]

Definition at line 1867 of file wsh.c.

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

- [wsh/wsh.c](#)

3.8 learn_t Struct Reference

Data Fields

- [learn_key_t key](#)
- char [toffset](#) [20]
- [UT_hash_handle hh](#)

3.8.1 Detailed Description

Definition at line 1870 of file wsh.c.

3.8.2 Field Documentation

3.8.2.1 UT_hash_handle learn_t::hh

Definition at line 1873 of file wsh.c.

3.8.2.2 learn_key_t learn_t::key

Definition at line 1871 of file wsh.c.

3.8.2.3 char learn_t::toffset[20]

Definition at line 1872 of file wsh.c.

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

- [wsh/wsh.c](#)

3.9 linenoiseCompletions Struct Reference

```
#include <linenoise.h>
```


Data Fields

- size_t [len](#)
- char ** [cvec](#)

3.9.1 Detailed Description

Definition at line 46 of file `linenoise.h`.

3.9.2 Field Documentation

3.9.2.1 char** linenoiseCompletions::cvec

Definition at line 48 of file `linenoise.h`.

3.9.2.2 size_t linenoiseCompletions::len

Definition at line 47 of file `linenoise.h`.

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

- `wsh/include/linenoise.h`

3.10 lua_Debug Struct Reference

```
#include <lua.h>
```

Data Fields

- int [event](#)
- const char * [name](#)
- const char * [namewhat](#)
- const char * [what](#)
- const char * [source](#)
- int [currentline](#)
- int [linedefined](#)
- int [lastlinedefined](#)
- unsigned char [nups](#)
- unsigned char [nparams](#)
- char [isvararg](#)
- char [istailcall](#)
- char [short_src](#) [LUA_IDSIZE]
- struct CallInfo * [i_ci](#)

3.10.1 Detailed Description

Definition at line 441 of file `lua.h`.

3.10.2 Field Documentation

3.10.2.1 int lua_Debug::currentline

Definition at line 447 of file lua.h.

3.10.2.2 int lua_Debug::event

Definition at line 442 of file lua.h.

3.10.2.3 struct CallInfo* lua_Debug::i_ci

Definition at line 456 of file lua.h.

3.10.2.4 char lua_Debug::istailcall

Definition at line 453 of file lua.h.

3.10.2.5 char lua_Debug::isvararg

Definition at line 452 of file lua.h.

3.10.2.6 int lua_Debug::lastlinedefined

Definition at line 449 of file lua.h.

3.10.2.7 int lua_Debug::linedefined

Definition at line 448 of file lua.h.

3.10.2.8 const char* lua_Debug::name

Definition at line 443 of file lua.h.

3.10.2.9 const char* lua_Debug::namewhat

Definition at line 444 of file lua.h.

3.10.2.10 unsigned char lua_Debug::nparams

Definition at line 451 of file lua.h.

3.10.2.11 unsigned char lua_Debug::nups

Definition at line 450 of file lua.h.

3.10.2.12 char lua_Debug::short_src[LUA_IDSIZE]

Definition at line 454 of file lua.h.

3.10.2.13 const char* lua_Debug::source

Definition at line 446 of file lua.h.

3.10.2.14 const char* lua_Debug::what

Definition at line 445 of file lua.h.

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

- [wsh/include/lua.h](#)

3.11 luaL_Buffer Struct Reference

```
#include <lauxlib.h>
```

Data Fields

- char * [b](#)
- size_t [size](#)
- size_t [n](#)
- lua_State * [L](#)
- char [initb](#) [[LUAL_BUFFERSIZE](#)]

3.11.1 Detailed Description

Definition at line 140 of file lauxlib.h.

3.11.2 Field Documentation

3.11.2.1 char* luaL_Buffer::b

Definition at line 141 of file lauxlib.h.

3.11.2.2 char luaL_Buffer::initb[LUAL_BUFFERSIZE]

Definition at line 145 of file lauxlib.h.

3.11.2.3 lua_State* luaL_Buffer::L

Definition at line 144 of file lauxlib.h.

3.11.2.4 size_t luaL_Buffer::n

Definition at line 143 of file lauxlib.h.

3.11.2.5 `size_t luaL_Buffer::size`

Definition at line 142 of file `lauxlib.h`.

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

- [wsh/include/lauxlib.h](#)

3.12 `luaL_Reg` Struct Reference

```
#include <lauxlib.h>
```

Data Fields

- `const char * name`
- `lua_CFunction func`

3.12.1 Detailed Description

Definition at line 23 of file `lauxlib.h`.

3.12.2 Field Documentation

3.12.2.1 `lua_CFunction luaL_Reg::func`

Definition at line 25 of file `lauxlib.h`.

3.12.2.2 `const char* luaL_Reg::name`

Definition at line 24 of file `lauxlib.h`.

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

- [wsh/include/lauxlib.h](#)

3.13 `luaL_Stream` Struct Reference

```
#include <lauxlib.h>
```

Data Fields

- `FILE * f`
- `lua_CFunction closef`

3.13.1 Detailed Description

Definition at line 185 of file `lauxlib.h`.

3.13.2 Field Documentation

3.13.2.1 lua_CFunction lua_Stream::closef

Definition at line 187 of file lauxlib.h.

3.13.2.2 FILE* lua_Stream::f

Definition at line 186 of file lauxlib.h.

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

- [wsh/include/lauxlib.h](#)

3.14 msec_t Struct Reference

Data Fields

- char * [name](#)
- unsigned long int [len](#)
- unsigned char * [data](#)
- char * [outoffset](#)
- unsigned int [flags](#)
- asection * [s_bfd](#)
- [Elf_Shdr](#) * [s_elf](#)
- struct [msec_t](#) * [prev](#)
- struct [msec_t](#) * [next](#)

3.14.1 Detailed Description

Meta section header

Definition at line 229 of file wcc.c.

3.14.2 Field Documentation

3.14.2.1 unsigned char* msec_t::data

Definition at line 232 of file wcc.c.

3.14.2.2 unsigned int msec_t::flags

Definition at line 234 of file wcc.c.

3.14.2.3 unsigned long int msec_t::len

Definition at line 231 of file wcc.c.

3.14.2.4 char* msec_t::name

Definition at line 230 of file wcc.c.

3.14.2.5 struct msec_t* msec_t::next

Definition at line 240 of file wcc.c.

3.14.2.6 char* msec_t::outoffset

Definition at line 233 of file wcc.c.

3.14.2.7 struct msec_t* msec_t::prev

Definition at line 239 of file wcc.c.

3.14.2.8 asection* msec_t::s_bfd

Definition at line 236 of file wcc.c.

3.14.2.9 Elf_Shdr* msec_t::s_elf

Definition at line 237 of file wcc.c.

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

- [wcc/wcc.c](#)

3.15 mseg_t Struct Reference

Data Fields

- [Elf_Word p_type](#)
- [Elf_Word p_flags](#)
- [Elf_Off p_offset](#)
- [Elf_Addr p_vaddr](#)
- [Elf_Addr p_paddr](#)
- [Elf_Xword p_filesz](#)
- [Elf_Xword p_memsz](#)
- [Elf_Xword p_align](#)
- struct msec_t * [prev](#)
- struct msec_t * [next](#)

3.15.1 Detailed Description

Meta segment header

Definition at line 248 of file wcc.c.

3.15.2 Field Documentation

3.15.2.1 struct msec_t* mseg_t::next

Definition at line 259 of file wcc.c.

3.15.2.2 Elf_Xword mseg_t::p_align

Definition at line 256 of file wcc.c.

3.15.2.3 Elf_Xword mseg_t::p_filesz

Definition at line 254 of file wcc.c.

3.15.2.4 Elf_Word mseg_t::p_flags

Definition at line 250 of file wcc.c.

3.15.2.5 Elf_Xword mseg_t::p_memsz

Definition at line 255 of file wcc.c.

3.15.2.6 Elf_Off mseg_t::p_offset

Definition at line 251 of file wcc.c.

3.15.2.7 Elf_Addr mseg_t::p_paddr

Definition at line 253 of file wcc.c.

3.15.2.8 Elf_Word mseg_t::p_type

Definition at line 249 of file wcc.c.

3.15.2.9 Elf_Addr mseg_t::p_vaddr

Definition at line 252 of file wcc.c.

3.15.2.10 struct msec_t* mseg_t::prev

Definition at line 258 of file wcc.c.

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

- [wcc/wcc.c](#)

3.16 preload_t Struct Reference

```
#include <wsh.h>
```

Data Fields

- char * [name](#)
- struct [preload_t](#) * [prev](#)
- struct [preload_t](#) * [next](#)

3.16.1 Detailed Description

Libraries to be preloaded (before shell/script execution)

Definition at line 453 of file wsh.h.

3.16.2 Field Documentation

3.16.2.1 char* preload_t::name

Definition at line 454 of file wsh.h.

3.16.2.2 struct preload_t* preload_t::next

Definition at line 457 of file wsh.h.

3.16.2.3 struct preload_t* preload_t::prev

Definition at line 456 of file wsh.h.

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

- [wsh/include/libwitch/wsh.h](#)

3.17 range_t Struct Reference

```
#include <wsh.h>
```

Data Fields

- unsigned long long int [min](#)
- unsigned long long int [max](#)

3.17.1 Detailed Description

Memory ranges

Definition at line 433 of file wsh.h.

3.17.2 Field Documentation

3.17.2.1 unsigned long long int range_t::max

Definition at line 435 of file wsh.h.

3.17.2.2 unsigned long long int range_t::min

Definition at line 434 of file wsh.h.

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

- [wsh/include/libwitch/wsh.h](#)

3.18 `script_t` Struct Reference

```
#include <wsh.h>
```

Data Fields

- char * `name`
- struct `preload_t` * `prev`
- struct `preload_t` * `next`

3.18.1 Detailed Description

Scripts to be executed

Definition at line 464 of file `wsh.h`.

3.18.2 Field Documentation

3.18.2.1 `char* script_t::name`

Definition at line 465 of file `wsh.h`.

3.18.2.2 `struct preload_t* script_t::next`

Definition at line 468 of file `wsh.h`.

3.18.2.3 `struct preload_t* script_t::prev`

Definition at line 467 of file `wsh.h`.

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

- `wsh/include/libwitch/wsh.h`

3.19 `section` Struct Reference

```
#include <helper.h>
```

Data Fields

- unsigned long long int `init`
- unsigned long long int `end`
- int `size`
- int `perms`
- char `name` [255]
- char `hperms` [10]
- void * `next`
- int `num`
- int `proba`
- int `probableval`

3.19.1 Detailed Description

Definition at line 11 of file helper.h.

3.19.2 Field Documentation

3.19.2.1 unsigned long long int section::end

Definition at line 13 of file helper.h.

3.19.2.2 char section::hperms[10]

Definition at line 17 of file helper.h.

3.19.2.3 unsigned long long int section::init

Definition at line 12 of file helper.h.

3.19.2.4 char section::name[255]

Definition at line 16 of file helper.h.

3.19.2.5 void* section::next

Definition at line 18 of file helper.h.

3.19.2.6 int section::num

Definition at line 20 of file helper.h.

3.19.2.7 int section::perms

Definition at line 15 of file helper.h.

3.19.2.8 int section::proba

Definition at line 21 of file helper.h.

3.19.2.9 int section::probableval

Definition at line 22 of file helper.h.

3.19.2.10 int section::size

Definition at line 14 of file helper.h.

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

- [wsh/include/libwitch/helper.h](#)

3.20 sections_t Struct Reference

```
#include <wsh.h>
```

Data Fields

- unsigned long int [addr](#)
- unsigned long int [size](#)
- char * [libname](#)
- char * [name](#)
- char * [perms](#)
- int [flags](#)
- struct [sections_t](#) * [prev](#)
- struct [sections_t](#) * [next](#)

3.20.1 Detailed Description

Representation of ELF Sections

Definition at line 474 of file wsh.h.

3.20.2 Field Documentation

3.20.2.1 unsigned long int sections_t::addr

Definition at line 475 of file wsh.h.

3.20.2.2 int sections_t::flags

Definition at line 480 of file wsh.h.

3.20.2.3 char* sections_t::libname

Definition at line 477 of file wsh.h.

3.20.2.4 char* sections_t::name

Definition at line 478 of file wsh.h.

3.20.2.5 struct sections_t* sections_t::next

Definition at line 483 of file wsh.h.

3.20.2.6 char* sections_t::perms

Definition at line 479 of file wsh.h.

3.20.2.7 struct sections_t* sections_t::prev

Definition at line 482 of file wsh.h.

3.20.2.8 unsigned long int sections_t::size

Definition at line 476 of file wsh.h.

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

- wsh/include/libwitch/wsh.h

3.21 segments_t Struct Reference

```
#include <wsh.h>
```

Data Fields

- unsigned long int [addr](#)
- unsigned long int [size](#)
- char * [libname](#)
- char * [type](#)
- char * [perms](#)
- int [flags](#)
- struct [segments_t](#) * [prev](#)
- struct [segments_t](#) * [next](#)

3.21.1 Detailed Description

Representation of ELF Segments

Definition at line 490 of file wsh.h.

3.21.2 Field Documentation

3.21.2.1 unsigned long int segments_t::addr

Definition at line 491 of file wsh.h.

3.21.2.2 int segments_t::flags

Definition at line 496 of file wsh.h.

3.21.2.3 char* segments_t::libname

Definition at line 493 of file wsh.h.

3.21.2.4 struct segments_t* segments_t::next

Definition at line 499 of file wsh.h.

3.21.2.5 char* segments_t::perms

Definition at line 495 of file wsh.h.

3.21.2.6 `struct segments_t* segments_t::prev`

Definition at line 498 of file `wsh.h`.

3.21.2.7 `unsigned long int segments_t::size`

Definition at line 492 of file `wsh.h`.

3.21.2.8 `char* segments_t::type`

Definition at line 494 of file `wsh.h`.

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

- `wsh/include/libwitch/wsh.h`

3.22 `signame_t` Struct Reference

```
#include <sigs.h>
```

Data Fields

- `int signal`
- `char * name`

3.22.1 Detailed Description

Definition at line 1 of file `sigs.h`.

3.22.2 Field Documentation

3.22.2.1 `char* signame_t::name`

Definition at line 3 of file `sigs.h`.

3.22.2.2 `int signame_t::signal`

Definition at line 2 of file `sigs.h`.

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

- `wsh/include/libwitch/sigs.h`

3.23 `symaddr` Struct Reference

Data Fields

- `struct symaddr * next`
- `char * name`
- `int addr`

3.23.1 Detailed Description

Definition at line 405 of file wcc.c.

3.23.2 Field Documentation

3.23.2.1 int symaddr::addr

Definition at line 408 of file wcc.c.

3.23.2.2 char* symaddr::name

Definition at line 407 of file wcc.c.

3.23.2.3 struct symaddr* symaddr::next

Definition at line 406 of file wcc.c.

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

- [wcc/wcc.c](#)

3.24 symbols_t Struct Reference

```
#include <wsh.h>
```

Data Fields

- unsigned long int [addr](#)
- unsigned long int [size](#)
- char * [symbol](#)
- char * [libname](#)
- char * [htype](#)
- char * [hbind](#)
- unsigned long int [value](#)
- struct [symbols_t](#) * [prev](#)
- struct [symbols_t](#) * [next](#)

3.24.1 Detailed Description

Representation of ELF Symbols

Definition at line 506 of file wsh.h.

3.24.2 Field Documentation

3.24.2.1 unsigned long int symbols_t::addr

Definition at line 507 of file wsh.h.

3.24.2.2 char* symbols_t::hbind

Definition at line 512 of file wsh.h.

3.24.2.3 char* symbols_t::htype

Definition at line 511 of file wsh.h.

3.24.2.4 char* symbols_t::libname

Definition at line 510 of file wsh.h.

3.24.2.5 struct symbols_t* symbols_t::next

Definition at line 516 of file wsh.h.

3.24.2.6 struct symbols_t* symbols_t::prev

Definition at line 515 of file wsh.h.

3.24.2.7 unsigned long int symbols_t::size

Definition at line 508 of file wsh.h.

3.24.2.8 char* symbols_t::symbol

Definition at line 509 of file wsh.h.

3.24.2.9 unsigned long int symbols_t::value

Definition at line 513 of file wsh.h.

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

- [wsh/include/libwitch/wsh.h](#)

3.25 tuple_t Struct Reference

```
#include <wsh.h>
```

Data Fields

- void * [addr](#)
- char * [name](#)

3.25.1 Detailed Description

Definition at line 610 of file wsh.h.

3.25.2 Field Documentation

3.25.2.1 void* tuple_t::addr

Definition at line 611 of file wsh.h.

3.25.2.2 char* tuple_t::name

Definition at line 612 of file wsh.h.

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

- wsh/include/libwitch/wsh.h

3.26 wsh_t Struct Reference

```
#include <wsh.h>
```

Data Fields

- lua_State * L
- FILE * scriptfile
- char * scriptname
- char * learnlog
- FILE * learnfile
- unsigned int opt_verbose
- unsigned int opt_hollywood
- unsigned int mainhandle
- unsigned int opt_rescan
- unsigned int opt_verbosetrace
- unsigned int firsterrno
- unsigned int firstsicode
- unsigned int firstsignal
- unsigned int tosignals
- unsigned int globalsignals
- unsigned long int faultaddr
- void * firstcontext
- unsigned int reason
- unsigned int is_stdinscript
- unsigned int bp_points
- void * pltgot
- unsigned int pltsz
- ucontext_t * errcontext
- unsigned long int btcaller
- breakpoint_t * bp_array
- unsigned int bp_num
- unsigned int opt_argc
- char * opt_argv
- char ** script_args
- unsigned int script_argnum
- unsigned int trace_unaligned
- unsigned int trace_singlestep
- unsigned int trace_singlebranch

- unsigned int [trace_rtrace](#)
- unsigned int [trace_strace](#)
- unsigned int [singlestep_count](#)
- unsigned int [singlebranch_count](#)
- unsigned int [sigbus_count](#)
- unsigned long long int [singlestep_hash](#)
- unsigned long long int [singlebranch_hash](#)
- unsigned long long int [sigbus_hash](#)
- jmp_buf [longjmp_ptr_high](#)
- jmp_buf [longjmp_ptr](#)
- unsigned int [interrupted](#)
- unsigned int [longjmp_ptr_high_cnt](#)
- struct [sections_t](#) * [shdrs](#)
- struct [segments_t](#) * [phdrs](#)
- struct [symbols_t](#) * [symbols](#)
- struct [eps_t](#) * [eps](#)
- struct [preload_t](#) * [preload](#)
- struct [script_t](#) * [scripts](#)

3.26.1 Detailed Description

wsh context

Definition at line 532 of file wsh.h.

3.26.2 Field Documentation

3.26.2.1 [breakpoint_t*](#) wsh_t::bp_array

Definition at line 570 of file wsh.h.

3.26.2.2 unsigned int wsh_t::bp_num

Definition at line 571 of file wsh.h.

3.26.2.3 unsigned int wsh_t::bp_points

Definition at line 560 of file wsh.h.

3.26.2.4 unsigned long int wsh_t::btcaller

Definition at line 568 of file wsh.h.

3.26.2.5 struct [eps_t](#)* wsh_t::eps

Definition at line 603 of file wsh.h.

3.26.2.6 [ucontext_t](#)* wsh_t::errcontext

Definition at line 565 of file wsh.h.

3.26.2.7 unsigned long int wsh_t::faultaddr

Definition at line 554 of file wsh.h.

3.26.2.8 void* wsh_t::firstcontext

Definition at line 556 of file wsh.h.

3.26.2.9 unsigned int wsh_t::firsterrno

Definition at line 549 of file wsh.h.

3.26.2.10 unsigned int wsh_t::firstsicode

Definition at line 550 of file wsh.h.

3.26.2.11 unsigned int wsh_t::firstsignal

Definition at line 551 of file wsh.h.

3.26.2.12 unsigned int wsh_t::globalsignals

Definition at line 553 of file wsh.h.

3.26.2.13 unsigned int wsh_t::interrupted

Definition at line 597 of file wsh.h.

3.26.2.14 unsigned int wsh_t::is_stdinscript

Definition at line 559 of file wsh.h.

3.26.2.15 lua_State* wsh_t::L

Definition at line 535 of file wsh.h.

3.26.2.16 FILE* wsh_t::learnfile

Definition at line 540 of file wsh.h.

3.26.2.17 char* wsh_t::learnlog

Definition at line 539 of file wsh.h.

3.26.2.18 jmp_buf wsh_t::longjmp_ptr

Definition at line 595 of file wsh.h.

3.26.2.19 jmp_buf wsh_t::longjmp_ptr_high

Definition at line 594 of file wsh.h.

3.26.2.20 unsigned int wsh_t::longjmp_ptr_high_cnt

Definition at line 598 of file wsh.h.

3.26.2.21 unsigned int wsh_t::mainhandle

Definition at line 544 of file wsh.h.

3.26.2.22 unsigned int wsh_t::opt_argc

Definition at line 573 of file wsh.h.

3.26.2.23 char* wsh_t::opt_argv

Definition at line 574 of file wsh.h.

3.26.2.24 unsigned int wsh_t::opt_hollywood

Definition at line 543 of file wsh.h.

3.26.2.25 unsigned int wsh_t::opt_rescan

Definition at line 545 of file wsh.h.

3.26.2.26 unsigned int wsh_t::opt_verbose

Definition at line 542 of file wsh.h.

3.26.2.27 unsigned int wsh_t::opt_verbosetrace

Definition at line 547 of file wsh.h.

3.26.2.28 struct segments_t* wsh_t::phdrs

Definition at line 601 of file wsh.h.

3.26.2.29 void* wsh_t::pltgot

Definition at line 562 of file wsh.h.

3.26.2.30 unsigned int wsh_t::pltz

Definition at line 563 of file wsh.h.

3.26.2.31 struct preload_t* wsh_t::preload

Definition at line 605 of file wsh.h.

3.26.2.32 unsigned int wsh_t::reason

Definition at line 557 of file wsh.h.

3.26.2.33 unsigned int wsh_t::script_argnum

Definition at line 577 of file wsh.h.

3.26.2.34 char wsh_t::script_args**

Definition at line 576 of file wsh.h.

3.26.2.35 FILE* wsh_t::scriptfile

Definition at line 536 of file wsh.h.

3.26.2.36 char* wsh_t::scriptname

Definition at line 537 of file wsh.h.

3.26.2.37 struct script_t* wsh_t::scripts

Definition at line 606 of file wsh.h.

3.26.2.38 struct sections_t* wsh_t::shdrs

Definition at line 600 of file wsh.h.

3.26.2.39 unsigned int wsh_t::sigbus_count

Definition at line 588 of file wsh.h.

3.26.2.40 unsigned long long int wsh_t::sigbus_hash

Definition at line 592 of file wsh.h.

3.26.2.41 unsigned int wsh_t::singlebranch_count

Definition at line 587 of file wsh.h.

3.26.2.42 unsigned long long int wsh_t::singlebranch_hash

Definition at line 591 of file wsh.h.

3.26.2.43 unsigned int wsh_t::singlestep_count

Definition at line 586 of file wsh.h.

3.26.2.44 unsigned long long int wsh_t::singlestep_hash

Definition at line 590 of file wsh.h.

3.26.2.45 struct symbols_t* wsh_t::symbols

Definition at line 602 of file wsh.h.

3.26.2.46 unsigned int wsh_t::totsignals

Definition at line 552 of file wsh.h.

3.26.2.47 unsigned int wsh_t::trace_rtrace

Definition at line 583 of file wsh.h.

3.26.2.48 unsigned int wsh_t::trace_singlebranch

Definition at line 581 of file wsh.h.

3.26.2.49 unsigned int wsh_t::trace_singlestep

Definition at line 580 of file wsh.h.

3.26.2.50 unsigned int wsh_t::trace_strace

Definition at line 584 of file wsh.h.

3.26.2.51 unsigned int wsh_t::trace_unaligned

Definition at line 579 of file wsh.h.

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

- [wsh/include/libwitch/wsh.h](#)

Chapter 4

File Documentation

4.1 wcc/wcc.c File Reference

```
#include <bfd.h>
#include <dlfcn.h>
#include <elf.h>
#include <errno.h>
#include <fcntl.h>
#include <getopt.h>
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <sys/procfs.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ucontext.h>
#include <unistd.h>
#include <utlist.h>
#include <ctype.h>
#include <libelf.h>
#include <gelf.h>
#include <nametotype.h>
#include <nametoalign.h>
#include <nametoentsz.h>
#include <nametolink.h>
#include <nametoinfo.h>
#include <arch.h>
#include <inttypes.h>
#include <config.h>
#include <capstone/capstone.h>
```

Data Structures

- struct [msec_t](#)
- struct [mseg_t](#)
- struct [ctx_t](#)
- struct [symaddr](#)
- struct [gimport_t](#)

Macros

- #define `__USE_GNU`
- #define `_GNU_SOURCE`
- #define `DEFAULT_STRNDX_SIZE` 4096
- #define `FLAG_BSS` 1
- #define `FLAG_NOBIT` 2
- #define `FLAG_NOWRITE` 4
- #define `FLAG_TEXT` 8
- #define `ifis(x)` `if(!strcmp(name, x, strlen(x)))`
- #define `elis(x)` `else if(!strcmp(name, x, strlen(x)))`
- #define `MAXPADLEN` 20
- #define `EXTRA_CREATED_SECTIONS` 4
- #define `RELOC_X86_64` 1
- #define `RELOC_X86_32` 2
- #define `Elf_Ehdr` `Elf32_Ehdr`
- #define `Elf_Shdr` `Elf32_Shdr`
- #define `Elf_Sym` `Elf32_Sym`
- #define `Elf_Addr` `Elf32_Addr`
- #define `Elf_Sword` `Elf64_Sword`
- #define `Elf_Section` `Elf32_Half`
- #define `ELF_ST_BIND` `ELF32_ST_BIND`
- #define `ELF_ST_TYPE` `ELF32_ST_TYPE`
- #define `Elf_Rel` `Elf32_Rel`
- #define `Elf_Rela` `Elf32_Rela`
- #define `ELF_R_SYM` `ELF32_R_SYM`
- #define `ELF_R_TYPE` `ELF32_R_TYPE`
- #define `ELF_R_INFO` `ELF32_R_INFO`
- #define `Elf_Phdr` `Elf32_Phdr`
- #define `Elf_Xword` `Elf32_Xword`
- #define `Elf_Word` `Elf32_Word`
- #define `Elf_Off` `Elf32_Off`
- #define `ELFCLASS` `ELFCLASS32`
- #define `ELFMACHINE` `EM_386`
- #define `CS_MODE` `CS_MODE_32`
- #define `RELOC_MODE` `RELOC_X86_32`
- #define `nullstr` `"\x00"`

Typedefs

- typedef struct `msec_t` `msec_t`
- typedef struct `mseg_t` `mseg_t`
- typedef struct `ctx_t` `ctx_t`
- typedef struct `gimport_t` `gimport_t`

Functions

- int `craft_section` (`ctx_t *ctx`, `msec_t *m`)
- unsigned int `secindex_from_name` (`ctx_t *ctx`, const char *name)
- `msec_t * section_from_name` (`ctx_t *ctx`, char *name)
- `msec_t * section_from_addr` (`ctx_t *ctx`, unsigned long int addr)
- int `print_bfd_sections` (`ctx_t *ctx`)
- `msec_t * section_from_index` (`ctx_t *ctx`, unsigned int index)
- unsigned int `secindex_from_name_after_strip` (`ctx_t *ctx`, const char *name)

- int [analyze_text](#) (ctx_t *ctx, char *data, unsigned int datalen, unsigned long int addr)
- int [save_reloc](#) (ctx_t *ctx, Elf_Rela *r, unsigned int sindex, int has_addend)
- unsigned int [protect_perms](#) (unsigned int perms)
- void [add_symaddr](#) (ctx_t *ctx, const char *name, int addr, char symclass)
- int [add_extra_symbols](#) (ctx_t *ctx)
- int [rd_symbols](#) (ctx_t *ctx)
- int [entszfromname](#) (const char *name)
- unsigned int [max](#) (unsigned int a, unsigned int b)
- char * [sec_name_from_index_after_strip](#) (ctx_t *ctx, unsigned int index)
- int [link_from_name](#) (ctx_t *ctx, const char *name)
- int [info_from_name](#) (ctx_t *ctx, const char *name)
- int [typefromname](#) (const char *name)
- unsigned int [alignfromname](#) (const char *name)
- unsigned int [ptype_from_section](#) (msec_t *ms)
- unsigned int [pflag_from_section](#) (msec_t *ms)
- int [phdr_cmp_premerge](#) (mseg_t *a, mseg_t *b)
- int [phdr_cmp](#) (mseg_t *a, mseg_t *b)
- int [sort_phdrs](#) (ctx_t *ctx)
- int [sort_phdrs_premerge](#) (ctx_t *ctx)
- msec_t * [alloc_phdr](#) (msec_t *ms)
- int [create_phdrs](#) (ctx_t *ctx)
- int [merge_phdrs](#) (ctx_t *ctx)
- int [adjust_baseaddress](#) (ctx_t *ctx)
- msec_t * [mk_section](#) (void)
- char * [reloc_htype_x86_64](#) (int thetype)
- char * [reloc_htype_x86_32](#) (int thetype)
- char * [reloc_htype](#) (int thetype)
- int [fixup_strtab_and_symtab](#) (ctx_t *ctx)
- int [fixup_text](#) (ctx_t *ctx)
- unsigned int [append_sym](#) (Elf_Sym *s)
- unsigned int [append_strtab](#) (char *str)
- void [hexdump](#) (unsigned char *data, size_t size)
- unsigned int [open_best](#) (ctx_t *ctx)
- int [open_target](#) (ctx_t *ctx)
- int [copy_body](#) (ctx_t *ctx)
- int [load_binary](#) (ctx_t *ctx)
- int [flags_from_name](#) (const char *name)
- int [print_msec](#) (ctx_t *ctx)
- int [rd_sections](#) (ctx_t *ctx)
- int [save_dynstr](#) (ctx_t *ctx, GElf_Shdr shdr, char *binary)
- int [save_dynsym](#) (ctx_t *ctx, GElf_Shdr shdr, char *binary)
- int [patch_symbol_index](#) (ctx_t *ctx, Elf_Sym *s)
- int [fixup_symtab_section_index](#) (ctx_t *ctx)
- int [append_reloc](#) (Elf_Rela *r)
- int [save_global_import](#) (ctx_t *ctx, char *sname, msec_t *sec, Elf_Rela *r, unsigned int sindex)
- int [check_global_import](#) (unsigned long int addr)
- int [internal_function_store](#) (ctx_t *ctx, unsigned long long int addr)
- int [rd_symtab](#) (ctx_t *ctx)
- int [rm_section](#) (ctx_t *ctx, char *name)
- int [strip_binary_reloc](#) (ctx_t *ctx)
- unsigned int [libify](#) (ctx_t *ctx)
- int [print_maps](#) (void)
- ctx_t * [ctx_init](#) (void)
- int [usage](#) (char *name)
- int [print_version](#) (void)
- int [desired_arch](#) (ctx_t *ctx, char *name)
- int [ctx_getopt](#) (ctx_t *ctx, int argc, char **argv)
- int [main](#) (int argc, char **argv)

Variables

- unsigned int `maxoldsec` = 0
- unsigned int `maxnewsec` = 0
- unsigned int `deltastrtab` = 0
- char * `allowed_sections` []
- char * `blnames` []
- char * `globalsymtab` = 0
- int `globalsymtablen` = 0
- unsigned int `globalsymtableoffset` = 0
- char * `globalstrtab` = 0
- unsigned int `globalstrtablen` = 0
- unsigned int `globalstrtableoffset` = 0
- unsigned int `globalsymindex` = 0
- char * `globalreloc` = 0
- unsigned int `globalreloclen` = 0
- unsigned int `globalrelocoffset` = 0
- unsigned long int `mintext` = -1
- unsigned long int `maxtext` = 0
- unsigned long int `textvma` = 0
- unsigned long int `mindata` = -1
- unsigned long int `maxdata` = 0
- unsigned long int `datavma` = 0
- unsigned long int `orig_text` = 0
- unsigned long int `orig_sz` = 0
- struct `symaddr` * `symaddrs`
- `gimport_t` ** `gimports` = 0
- unsigned int `gimportslen` = 0

4.1.1 Macro Definition Documentation

4.1.1.1 #define __USE_GNU

Witchcraft Compiler Collection

Author: Jonathan Brossard - endrazine@gmail.com

The MIT License (MIT) Copyright (c) 2016 Jonathan Brossard

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Definition at line 31 of file `wcc.c`.

4.1.1.2 #define _GNU_SOURCE

Definition at line 32 of file `wcc.c`.

4.1.1.3 #define CS_MODE CS_MODE_32

Definition at line 134 of file wcc.c.

4.1.1.4 #define DEFAULT_STRNDX_SIZE 4096

Definition at line 72 of file wcc.c.

4.1.1.5 #define Elf_Addr Elf32_Addr

Definition at line 118 of file wcc.c.

4.1.1.6 #define Elf_Ehdr Elf32_Ehdr

Definition at line 115 of file wcc.c.

4.1.1.7 #define Elf_Off Elf32_Off

Definition at line 131 of file wcc.c.

4.1.1.8 #define Elf_Phdr Elf32_Phdr

Definition at line 128 of file wcc.c.

4.1.1.9 #define ELF_R_INFO ELF32_R_INFO

Definition at line 127 of file wcc.c.

4.1.1.10 #define ELF_R_SYM ELF32_R_SYM

Definition at line 125 of file wcc.c.

4.1.1.11 #define ELF_R_TYPE ELF32_R_TYPE

Definition at line 126 of file wcc.c.

4.1.1.12 #define Elf_Rel Elf32_Rel

Definition at line 123 of file wcc.c.

4.1.1.13 #define Elf_Rela Elf32_Rela

Definition at line 124 of file wcc.c.

4.1.1.14 #define Elf_Section Elf32_Half

Definition at line 120 of file wcc.c.

4.1.1.15 `#define Elf_Shdr Elf32_Shdr`

Definition at line 116 of file wcc.c.

4.1.1.16 `#define ELF_ST_BIND ELF32_ST_BIND`

Definition at line 121 of file wcc.c.

4.1.1.17 `#define ELF_ST_TYPE ELF32_ST_TYPE`

Definition at line 122 of file wcc.c.

4.1.1.18 `#define Elf_Sword Elf64_Sword`

Definition at line 119 of file wcc.c.

4.1.1.19 `#define Elf_Sym Elf32_Sym`

Definition at line 117 of file wcc.c.

4.1.1.20 `#define Elf_Word Elf32_Word`

Definition at line 130 of file wcc.c.

4.1.1.21 `#define Elf_Xword Elf32_Xword`

Definition at line 129 of file wcc.c.

4.1.1.22 `#define ELFCLASS ELFCLASS32`

Definition at line 132 of file wcc.c.

4.1.1.23 `#define ELFMACHINE EM_386`

Definition at line 133 of file wcc.c.

4.1.1.24 `#define elis(x) else if(!strcmp(name, x, strlen(x)))`

Definition at line 81 of file wcc.c.

4.1.1.25 `#define EXTRA_CREATED_SECTIONS 4`

Definition at line 85 of file wcc.c.

4.1.1.26 `#define FLAG_BSS 1`

Definition at line 75 of file wcc.c.

4.1.1.27 #define FLAG_NOBIT 2

Definition at line 76 of file wcc.c.

4.1.1.28 #define FLAG_NOWRITE 4

Definition at line 77 of file wcc.c.

4.1.1.29 #define FLAG_TEXT 8

Definition at line 78 of file wcc.c.

4.1.1.30 #define ifis(x) if(!strcmp(name, x, strlen(x)))

Definition at line 80 of file wcc.c.

4.1.1.31 #define MAXPADLEN 20

Definition at line 83 of file wcc.c.

4.1.1.32 #define nullstr "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"

Definition at line 139 of file wcc.c.

4.1.1.33 #define RELOC_MODE RELOC_X86_32

Definition at line 135 of file wcc.c.

4.1.1.34 #define RELOC_X86_32 2

Definition at line 89 of file wcc.c.

4.1.1.35 #define RELOC_X86_64 1

Definition at line 88 of file wcc.c.

4.1.2 Typedef Documentation**4.1.2.1 typedef struct ctx_t ctx_t****4.1.2.2 typedef struct gimport_t gimport_t****4.1.2.3 typedef struct msec_t msec_t**

Meta section header

4.1.2.4 typedef struct mseg_t mseg_t

Meta segment header

4.1.3 Function Documentation

4.1.3.1 `int add_extra_symbols (ctx_t * ctx)`

Add extra symbols

Definition at line 561 of file wcc.c.

4.1.3.2 `void add_symaddr (ctx_t * ctx, const char * name, int addr, char symclass)`

Append name to global string table

Append symbol to global symbol table

Definition at line 423 of file wcc.c.

4.1.3.3 `int adjust_baseaddress (ctx_t * ctx)`

Definition at line 1108 of file wcc.c.

4.1.3.4 `unsigned int alignfromname (const char * name)`

Return a section alignment from its name

Definition at line 881 of file wcc.c.

4.1.3.5 `mseg_t* alloc_phdr (msec_t * ms)`

Allocate Phdr

Definition at line 1009 of file wcc.c.

4.1.3.6 `int analyze_text (ctx_t * ctx, char * data, unsigned int datalen, unsigned long int addr)`

Definition at line 3395 of file wcc.c.

4.1.3.7 `int append_reloc (Elf_Rel * r)`

Definition at line 2740 of file wcc.c.

4.1.3.8 `unsigned int append_strtab (char * str)`

Append a string to symbol table, reports offset in strtab where this symbol will start

Definition at line 1776 of file wcc.c.

4.1.3.9 `unsigned int append_sym (Elf_Sym * s)`

Append a symbol to global symbol table

Definition at line 1755 of file wcc.c.

4.1.3.10 int check_global_import (unsigned long int *addr*)

Return index in global import matching this address

Definition at line 2818 of file wcc.c.

4.1.3.11 int copy_body (ctx_t * *ctx*)

Write sections to disk

Definition at line 2459 of file wcc.c.

4.1.3.12 int craft_section (ctx_t * *ctx*, msec_t * *m*)

Forwardd prototypes declarations

Craft Section header

Definition at line 2500 of file wcc.c.

4.1.3.13 int create_phdrs (ctx_t * *ctx*)

Create Program Headers based on ELF section headers

Definition at line 1032 of file wcc.c.

4.1.3.14 int ctx_getopt (ctx_t * *ctx*, int *argc*, char ** *argv*)

Definition at line 3847 of file wcc.c.

4.1.3.15 ctx_t* ctx_init (void)

Initialize a reversing context Set default values

Definition at line 3775 of file wcc.c.

4.1.3.16 int desired_arch (ctx_t * *ctx*, char * *name*)

Definition at line 3827 of file wcc.c.

4.1.3.17 int entszfromname (const char * *name*)

Return section entry size from name

Definition at line 682 of file wcc.c.

4.1.3.18 int fixup_strtab_and_syntab (ctx_t * *ctx*)

check if name is in blacklist

Definition at line 1638 of file wcc.c.

4.1.3.19 int fixup_syntab_section_index (ctx_t * *ctx*)

Definition at line 2720 of file wcc.c.

4.1.3.20 int fixup_text (ctx_t * ctx)

Definition at line 1694 of file wcc.c.

4.1.3.21 int flags_from_name (const char * name)

Return section flags from its name

Definition at line 2486 of file wcc.c.

4.1.3.22 void hexdump (unsigned char * data, size_t size)

Simple hexdump routine

Definition at line 2346 of file wcc.c.

4.1.3.23 int info_from_name (ctx_t * ctx, const char * name)

Return a section info from its name

Definition at line 843 of file wcc.c.

4.1.3.24 int internal_function_store (ctx_t * ctx, unsigned long long int addr)

Definition at line 3289 of file wcc.c.

4.1.3.25 unsigned int libify (ctx_t * ctx)

Main routine LOAD OPERATIONS

Load each section of binary using bfd

Print BFD sections

Read .text segment boundaries

Open target binary

Read sections from disk

Read symtab + strtab : BFD doesn't do this

Read symbols

Add extra symbols

Parse relocations

Fix section indexes in symtab

PROCESSING

Copy each section content in output file

Relocation stripping

Create Program Headers

FINAL WRITE OPERATIONS

Write strtab and symtab

Add section headers to output file

Add segment headers to output file

Add ELF Header to output file
Finalize/Close/Cleanup
Definition at line 3597 of file wcc.c.

4.1.3.26 `int link_from_name (ctx_t * ctx, const char * name)`

Return a section link from its name
Definition at line 820 of file wcc.c.

4.1.3.27 `int load_binary (ctx_t * ctx)`

Load a binary using bfd
Definition at line 2472 of file wcc.c.

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

Application Entry Point
Definition at line 4014 of file wcc.c.

4.1.3.29 `unsigned int max (unsigned int a, unsigned int b)`

Return max of two unsigned integers
Definition at line 697 of file wcc.c.

4.1.3.30 `int merge_phdrs (ctx_t * ctx)`

Merge two consecutive Phdrs if:

- their vma ranges overlap
- Permissions match
- Type of segment matches

Note: assume phdrs have been sorted by increasing p_vaddr first
Definition at line 1073 of file wcc.c.

4.1.3.31 `msec_t* mk_section (void)`

Definition at line 1330 of file wcc.c.

4.1.3.32 `unsigned int open_best (ctx_t * ctx)`

Open a binary the best way we can
Definition at line 2373 of file wcc.c.

4.1.3.33 `int open_target (ctx_t * ctx)`

Open destination binary

Definition at line 2405 of file wcc.c.

4.1.3.34 `int patch_symbol_index (ctx_t * ctx, Elf_Sym * s)`

Definition at line 2701 of file wcc.c.

4.1.3.35 `unsigned int pflag_from_section (msec_t * ms)`

Return Segment flags based on a section

Definition at line 943 of file wcc.c.

4.1.3.36 `int phdr_cmp (mseg_t * a, mseg_t * b)`

Helper sort routine for ELF Phdrs

Definition at line 982 of file wcc.c.

4.1.3.37 `int phdr_cmp_premerge (mseg_t * a, mseg_t * b)`

Helper sort routine for ELF Phdrs (pre-merge)

Definition at line 971 of file wcc.c.

4.1.3.38 `int print_bfd_sections (ctx_t * ctx)`

Display BFD memory sections

Definition at line 2288 of file wcc.c.

4.1.3.39 `int print_maps (void)`

Print content of /proc/pid/maps

Definition at line 3763 of file wcc.c.

4.1.3.40 `int print_msec (ctx_t * ctx)`

Display sections

Definition at line 2633 of file wcc.c.

4.1.3.41 `int print_version (void)`

Definition at line 3821 of file wcc.c.

4.1.3.42 `unsigned int protect_perms (unsigned int perms)`

Convert octal permissions into permissions consumable by mprotect()

Definition at line 381 of file wcc.c.

4.1.3.43 unsigned int ptype_from_section (msec_t * ms)

Return Segment ptype

Definition at line 896 of file wcc.c.

4.1.3.44 int rd_sections (ctx_t * ctx)

Read sections from input binary

Definition at line 2650 of file wcc.c.

4.1.3.45 int rd_symbols (ctx_t * ctx)

Read symbol table. This is a two stages process : allocate the table, then read it Process symbol table

Process dynamic symbol table

Definition at line 574 of file wcc.c.

4.1.3.46 int rd_symtab (ctx_t * ctx)

Read original symtab + strtab. BDF doesn't do this

Definition at line 3443 of file wcc.c.

4.1.3.47 char* reloc_htype (int thetype)

Definition at line 1535 of file wcc.c.

4.1.3.48 char* reloc_htype_x86_32 (int thetype)

Definition at line 1474 of file wcc.c.

4.1.3.49 char* reloc_htype_x86_64 (int thetype)

Definition at line 1391 of file wcc.c.

4.1.3.50 int rm_section (ctx_t * ctx, char * name)

Suppress a given section

Definition at line 3533 of file wcc.c.

4.1.3.51 int save_dynstr (ctx_t * ctx, GElf_Shdr shdr, char * binary)

Definition at line 2663 of file wcc.c.

4.1.3.52 int save_dynsym (ctx_t * ctx, GElf_Shdr shdr, char * binary)

Definition at line 2681 of file wcc.c.

4.1.3.53 `int save_global_import (ctx_t * ctx, char * sname, msec_t * sec, Elf_Rel * r, unsigned int sindex)`

Definition at line 2781 of file wcc.c.

4.1.3.54 `int save_reloc (ctx_t * ctx, Elf_Rel * r, unsigned int sindex, int has_addend)`

Convert relocation depending on type and source section

Definition at line 2835 of file wcc.c.

4.1.3.55 `char* sec_name_from_index_after_strip (ctx_t * ctx, unsigned int index)`

Definition at line 791 of file wcc.c.

4.1.3.56 `unsigned int secindex_from_name (ctx_t * ctx, const char * name)`

Return a section index from its name

Definition at line 753 of file wcc.c.

4.1.3.57 `unsigned int secindex_from_name_after_strip (ctx_t * ctx, const char * name)`

Return a section index (after strip) from its name

Definition at line 770 of file wcc.c.

4.1.3.58 `msec_t * section_from_addr (ctx_t * ctx, unsigned long int addr)`

Return a section from its address

Definition at line 720 of file wcc.c.

4.1.3.59 `msec_t * section_from_index (ctx_t * ctx, unsigned int index)`

Return a section from its index

Definition at line 736 of file wcc.c.

4.1.3.60 `msec_t * section_from_name (ctx_t * ctx, char * name)`

Return a section from its name

Definition at line 705 of file wcc.c.

4.1.3.61 `int sort_phdrs (ctx_t * ctx)`

Reorganise Program Headers : sort by p_offset

Definition at line 991 of file wcc.c.

4.1.3.62 `int sort_phdrs_premerge (ctx_t * ctx)`

Helper sort routine for ELF Phdrs

Definition at line 1000 of file wcc.c.

4.1.3.63 `int strip_binary_reloc (ctx_t * ctx)`

Strip binary relocation data

Definition at line 3560 of file wcc.c.

4.1.3.64 `int typefromname (const char * name)`

Return a section type from its name

Definition at line 866 of file wcc.c.

4.1.3.65 `int usage (char * name)`

Definition at line 3795 of file wcc.c.

4.1.4 Variable Documentation

4.1.4.1 `char* allowed_sections[]`

Initial value:

```
= {  
    ".rodata",  
    ".data",  
    ".text",  
    ".load",  
    ".strtab",  
    ".symtab",  
    ".comment",  
    ".note.GNU-stack",  
    ".rsrc",  
    ".bss",  
}
```

Definition at line 143 of file wcc.c.

4.1.4.2 `char* bnames[]`

Definition at line 158 of file wcc.c.

4.1.4.3 `unsigned long int datavma = 0`

Definition at line 358 of file wcc.c.

4.1.4.4 `unsigned int deltastrtab = 0`

Definition at line 141 of file wcc.c.

4.1.4.5 `gimport_t** gimports = 0`

Definition at line 2778 of file wcc.c.

4.1.4.6 `unsigned int gimportslen = 0`

Definition at line 2779 of file wcc.c.

4.1.4.7 char* globalreloc = 0

Definition at line 348 of file wcc.c.

4.1.4.8 unsigned int globalreloclen = 0

Definition at line 349 of file wcc.c.

4.1.4.9 unsigned int globalreloffset = 0

Definition at line 350 of file wcc.c.

4.1.4.10 char* globalstrtab = 0

Definition at line 342 of file wcc.c.

4.1.4.11 unsigned int globalstrtablen = 0

Definition at line 343 of file wcc.c.

4.1.4.12 unsigned int globalstrtableoffset = 0

Definition at line 344 of file wcc.c.

4.1.4.13 unsigned int globalsymindex = 0

Definition at line 346 of file wcc.c.

4.1.4.14 char* globalsymtab = 0

Globals

Definition at line 338 of file wcc.c.

4.1.4.15 int globalsymtablen = 0

Definition at line 339 of file wcc.c.

4.1.4.16 unsigned int globalsymtableoffset = 0

Definition at line 340 of file wcc.c.

4.1.4.17 unsigned long int maxdata = 0

Definition at line 357 of file wcc.c.

4.1.4.18 unsigned int maxnewsec = 0

Definition at line 140 of file wcc.c.

4.1.4.19 unsigned int maxoldsec = 0

Definition at line 140 of file wcc.c.

4.1.4.20 unsigned long int maxtext = 0

Definition at line 353 of file wcc.c.

4.1.4.21 unsigned long int mindata = -1

Definition at line 356 of file wcc.c.

4.1.4.22 unsigned long int mintext = -1

Definition at line 352 of file wcc.c.

4.1.4.23 unsigned long int orig_sz = 0

Definition at line 361 of file wcc.c.

4.1.4.24 unsigned long int orig_text = 0

Definition at line 360 of file wcc.c.

4.1.4.25 struct symaddr * symaddrs

4.1.4.26 unsigned long int textvma = 0

Definition at line 354 of file wcc.c.

4.2 wld/wld.c File Reference

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <unistd.h>
#include <sys/types.h>
#include <string.h>
#include <limits.h>
#include <errno.h>
#include <elf.h>
#include <config.h>
```

Macros

- #define `DEFAULT_NAME` "wld"

Functions

- int `mk_lib` (char *name)
- int `print_version` (void)
- int `main` (int argc, char **argv)

4.2.1 Macro Definition Documentation

4.2.1.1 `#define DEFAULT_NAME "wld"`

Witchcraft Compiler Collection

Author: Jonathan Brossard - endrazine@gmail.com

This code is published under the MIT License.

Definition at line 31 of file `wld.c`.

4.2.2 Function Documentation

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

Definition at line 91 of file `wld.c`.

4.2.2.2 `int mk_lib (char * name)`

Patch ELF `ehdr->e_type` to `ET_DYN`

Definition at line 36 of file `wld.c`.

4.2.2.3 `int print_version (void)`

Definition at line 85 of file `wld.c`.

4.3 `wsh/helper.c` File Reference

```
#include <math.h>
#include <ctype.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <limits.h>
#include <regex.h>
#include <sys/ptrace.h>
#include <signal.h>
#include <string.h>
#include <libwitch/helper.h>
```


Macros

- #define `_XOPEN_SOURCE` 500
- #define `_FILE_OFFSET_BITS` 64
- #define `HAS_ZFIRST` 1

Functions

- int `is_mapped` (unsigned long int *addr*)
- int `read_maps` (int *pid*)

Variables

- unsigned int `lastsignal`
- struct `section` * `zfirst` = 0
- int `nsections` =0

4.3.1 Macro Definition Documentation

4.3.1.1 #define `_FILE_OFFSET_BITS` 64

Definition at line 25 of file helper.c.

4.3.1.2 #define `_XOPEN_SOURCE` 500

Definition at line 24 of file helper.c.

4.3.1.3 #define `HAS_ZFIRST` 1

Definition at line 45 of file helper.c.

4.3.2 Function Documentation

4.3.2.1 int `is_mapped` (unsigned long int *addr*)

Definition at line 56 of file helper.c.

4.3.2.2 int `read_maps` (int *pid*)

Definition at line 72 of file helper.c.

4.3.3 Variable Documentation

4.3.3.1 unsigned int `lastsignal`

4.3.3.2 int `nsections` =0

Definition at line 47 of file helper.c.

4.3.3.3 struct section* zfirst = 0

Definition at line 46 of file helper.c.

4.4 wsh/include/colors.h File Reference

Macros

- #define RED "\033[1;31m"
- #define CYAN "\033[1;36m"
- #define GREEN "\033[1;32m"
- #define BLUE "\033[1;34m"
- #define BLACK "\033[1;30m"
- #define BROWN "\033[1;33m"
- #define MAGENTA "\033[1;35m"
- #define GRAY "\033[1;37m"
- #define DARKGRAY "\033[1;30m"
- #define YELLOW "\033[1;33m"
- #define NORMAL "\033[0m" /* flush the previous properties */
- #define CLEAR "\033[2J"

4.4.1 Macro Definition Documentation

4.4.1.1 #define BLACK "\033[1;30m"

Definition at line 6 of file colors.h.

4.4.1.2 #define BLUE "\033[1;34m"

Definition at line 5 of file colors.h.

4.4.1.3 #define BROWN "\033[1;33m"

Definition at line 7 of file colors.h.

4.4.1.4 #define CLEAR "\033[2J"

Definition at line 17 of file colors.h.

4.4.1.5 #define CYAN "\033[1;36m"

Definition at line 3 of file colors.h.

4.4.1.6 #define DARKGRAY "\033[1;30m"

Definition at line 10 of file colors.h.

4.4.1.7 #define GRAY "\033[1;37m"

Definition at line 9 of file colors.h.

4.4.1.8 `#define GREEN "\033[1;32m"`

Definition at line 4 of file colors.h.

4.4.1.9 `#define MAGENTA "\033[1;35m"`

Definition at line 8 of file colors.h.

4.4.1.10 `#define NORMAL "\033[0m" /* flush the previous properties */`

Definition at line 14 of file colors.h.

4.4.1.11 `#define RED "\033[1;31m"`

Definition at line 2 of file colors.h.

4.4.1.12 `#define YELLOW "\033[1;33m"`

Definition at line 11 of file colors.h.

4.5 wsh/include/lauxlib.h File Reference

```
#include <stddef.h>
#include <stdio.h>
#include "lua.h"
```

Data Structures

- struct [luaL_Reg](#)
- struct [luaL_Buffer](#)
- struct [luaL_Stream](#)

Macros

- `#define LUA_ERRFILE (LUA_ERRERR+1)`
- `#define LUAL_NUMSIZES (sizeof(lua_Integer)*16 + sizeof(lua_Number))`
- `#define luaL_checkversion(L) luaL_checkversion_(L, LUA_VERSION_NUM, LUAL_NUMSIZES)`
- `#define LUA_NOREF (-2)`
- `#define LUA_REFNIL (-1)`
- `#define luaL_loadfile(L, f) luaL_loadfilex(L, f, NULL)`
- `#define luaL_newlibtable(L, l) luaL_createtable(L, 0, sizeof(l)/sizeof((l)[0]) - 1)`
- `#define luaL_newlib(L, l) (luaL_checkversion(L), luaL_newlibtable(L, l), luaL_setfuncs(L, l, 0))`
- `#define luaL_argcheck(L, cond, arg, extrams) ((void)((cond) || luaL_argerror(L, (arg), (extrams))))`
- `#define luaL_checkstring(L, n) (luaL_checklstring(L, (n), NULL))`
- `#define luaL_optstring(L, n, d) (luaL_optlstring(L, (n), (d), NULL))`
- `#define luaL_typename(L, i) luaL_typename(L, lua_type(L, (i)))`
- `#define luaL_dofile(L, fn) (luaL_loadfile(L, fn) || lua_pcall(L, 0, LUA_MULTRET, 0))`
- `#define luaL_dostring(L, s) (luaL_loadstring(L, s) || lua_pcall(L, 0, LUA_MULTRET, 0))`
- `#define luaL_getmetatable(L, n) (lua_getfield(L, LUA_REGISTRYINDEX, (n)))`

- #define `luaL_opt(L, f, n, d) (lua_isnoneornil(L,(n)) ? (d) : f(L,(n)))`
- #define `luaL_loadbuffer(L, s, sz, n) luaL_loadbufferx(L,s,sz,n,NULL)`
- #define `luaL_addchar(B, c)`
- #define `luaL_addsize(B, s) ((B)->n += (s))`
- #define `luaL_prepbuffer(B) luaL_prepbuffsize(B, LUAL_BUFFERSIZE)`
- #define `LUA_FILEHANDLE "FILE*"`
- #define `lua_writestring(s, l) fwrite((s), sizeof(char), (l), stdout)`
- #define `lua_writeline() (lua_writestring("\n", 1), fflush(stdout))`
- #define `lua_writestringerror(s, p) (fprintf(stderr, (s), (p)), fflush(stderr))`

Typedefs

- typedef struct `luaL_Reg` `luaL_Reg`
- typedef struct `luaL_Buffer` `luaL_Buffer`
- typedef struct `luaL_Stream` `luaL_Stream`

Functions

- `LUALIB_API void() luaL_checkversion_ (lua_State *L, lua_Number ver, size_t sz)`
- `LUALIB_API int() luaL_getmetafield (lua_State *L, int obj, const char *e)`
- `LUALIB_API int() luaL_callmeta (lua_State *L, int obj, const char *e)`
- `LUALIB_API const char *() luaL_tolstring (lua_State *L, int idx, size_t *len)`
- `LUALIB_API int() luaL_argerror (lua_State *L, int arg, const char *extrams)`
- `LUALIB_API const char *() luaL_checklstring (lua_State *L, int arg, size_t *l)`
- `LUALIB_API const char *() luaL_optlstring (lua_State *L, int arg, const char *def, size_t *l)`
- `LUALIB_API lua_Number() luaL_checknumber (lua_State *L, int arg)`
- `LUALIB_API lua_Number() luaL_optnumber (lua_State *L, int arg, lua_Number def)`
- `LUALIB_API lua_Integer() luaL_checkinteger (lua_State *L, int arg)`
- `LUALIB_API lua_Integer() luaL_optinteger (lua_State *L, int arg, lua_Integer def)`
- `LUALIB_API void() luaL_checkstack (lua_State *L, int sz, const char *msg)`
- `LUALIB_API void() luaL_checktype (lua_State *L, int arg, int t)`
- `LUALIB_API void() luaL_checkany (lua_State *L, int arg)`
- `LUALIB_API int() luaL_newmetatable (lua_State *L, const char *tname)`
- `LUALIB_API void() luaL_setmetatable (lua_State *L, const char *tname)`
- `LUALIB_API void *() luaL_testudata (lua_State *L, int ud, const char *tname)`
- `LUALIB_API void *() luaL_checkudata (lua_State *L, int ud, const char *tname)`
- `LUALIB_API void() luaL_where (lua_State *L, int lvl)`
- `LUALIB_API int() luaL_error (lua_State *L, const char *fmt,...)`
- `LUALIB_API int() luaL_checkoption (lua_State *L, int arg, const char *def, const char *const lst[])`
- `LUALIB_API int() luaL_fileresult (lua_State *L, int stat, const char *fname)`
- `LUALIB_API int() luaL_execresult (lua_State *L, int stat)`
- `LUALIB_API int() luaL_ref (lua_State *L, int t)`
- `LUALIB_API void() luaL_unref (lua_State *L, int t, int ref)`
- `LUALIB_API int() luaL_loadfilex (lua_State *L, const char *filename, const char *mode)`
- `LUALIB_API int() luaL_loadbufferx (lua_State *L, const char *buff, size_t sz, const char *name, const char *mode)`
- `LUALIB_API int() luaL_loadstring (lua_State *L, const char *s)`
- `LUALIB_API lua_State *() luaL_newstate (void)`
- `LUALIB_API lua_Integer() luaL_len (lua_State *L, int idx)`
- `LUALIB_API const char *() luaL_gsub (lua_State *L, const char *s, const char *p, const char *r)`
- `LUALIB_API void() luaL_setfuncs (lua_State *L, const luaL_Reg *l, int nup)`
- `LUALIB_API int() luaL_getsubtable (lua_State *L, int idx, const char *fname)`
- `LUALIB_API void() luaL_traceback (lua_State *L, lua_State *L1, const char *msg, int level)`
- `LUALIB_API void() luaL_requiref (lua_State *L, const char *modname, lua_CFunction openf, int glb)`

- LUALIB_API void() luaL_buffinit (lua_State *L, luaL_Buffer *B)
- LUALIB_API char *() luaL_prepbuffsize (luaL_Buffer *B, size_t sz)
- LUALIB_API void() luaL_addlstring (luaL_Buffer *B, const char *s, size_t l)
- LUALIB_API void() luaL_addstring (luaL_Buffer *B, const char *s)
- LUALIB_API void() luaL_addvalue (luaL_Buffer *B)
- LUALIB_API void() luaL_pushresult (luaL_Buffer *B)
- LUALIB_API void() luaL_pushresultsize (luaL_Buffer *B, size_t sz)
- LUALIB_API char *() luaL_buffinitsize (lua_State *L, luaL_Buffer *B, size_t sz)

4.5.1 Macro Definition Documentation

4.5.1.1 #define LUA_ERRFILE (LUA_ERRERR+1)

Definition at line 20 of file lauxlib.h.

4.5.1.2 #define LUA_FILEHANDLE "FILE*"

Definition at line 182 of file lauxlib.h.

4.5.1.3 #define LUA_NOREF (-2)

Definition at line 69 of file lauxlib.h.

4.5.1.4 #define LUA_REFNIL (-1)

Definition at line 70 of file lauxlib.h.

4.5.1.5 #define lua_writeline() (lua_writestring("\n", 1), fflush(stdout))

Definition at line 220 of file lauxlib.h.

4.5.1.6 #define lua_writestring(s, l) fwrite((s), sizeof(char), (l), stdout)

Definition at line 215 of file lauxlib.h.

4.5.1.7 #define lua_writestringerror(s, p) (fprintf(stderr, (s), (p)), fflush(stderr))

Definition at line 225 of file lauxlib.h.

4.5.1.8 #define luaL_addchar(B, c)

Value:

```
((void)((B)->n < (B)->size || luaL_prepbuffsize((B), 1)), \
  ((B)->b[(B)->n++] = (c))
```

Definition at line 149 of file lauxlib.h.

4.5.1.9 #define luaL_addsize(B, s) ((B)->n += (s))

Definition at line 153 of file lauxlib.h.

4.5.1.10 `#define luaL_argcheck(L, cond, arg, extramsng) ((void)((cond) || luaL_argerror(L, (arg), (extramsng))))`

Definition at line 114 of file lauxlib.h.

4.5.1.11 `#define luaL_checkstring(L, n) (luaL_checklstring(L, (n), NULL))`

Definition at line 116 of file lauxlib.h.

4.5.1.12 `#define luaL_checkversion(L) luaL_checkversion_(L, LUA_VERSION_NUM, LUAL_NUMSIZES)`

Definition at line 32 of file lauxlib.h.

4.5.1.13 `#define luaL_dofile(L, fn) (luaL_loadfile(L, fn) || lua_pcall(L, 0, LUA_MULTRET, 0))`

Definition at line 121 of file lauxlib.h.

4.5.1.14 `#define luaL_dostring(L, s) (luaL_loadstring(L, s) || lua_pcall(L, 0, LUA_MULTRET, 0))`

Definition at line 124 of file lauxlib.h.

4.5.1.15 `#define luaL_getmetatable(L, n) (lua_getfield(L, LUA_REGISTRYINDEX, (n)))`

Definition at line 127 of file lauxlib.h.

4.5.1.16 `#define luaL_loadbuffer(L, s, sz, n) luaL_loadbufferx(L,s,sz,n,NULL)`

Definition at line 131 of file lauxlib.h.

4.5.1.17 `#define luaL_loadfile(L, f) luaL_loadfilex(L,f,NULL)`

Definition at line 78 of file lauxlib.h.

4.5.1.18 `#define luaL_newlib(L, l) (luaL_checkversion(L), luaL_newlibtable(L,l), luaL_setfuncs(L,l,0))`

Definition at line 111 of file lauxlib.h.

4.5.1.19 `#define luaL_newlibtable(L, l) lua_createtable(L, 0, sizeof(l)/sizeof((l)[0]) - 1)`

Definition at line 108 of file lauxlib.h.

4.5.1.20 `#define LUAL_NUMSIZES (sizeof(lua_Integer)*16 + sizeof(lua_Number))`

Definition at line 29 of file lauxlib.h.

4.5.1.21 `#define luaL_opt(L, f, n, d) (lua_isnoneornil(L,(n)) ? (d) : f(L,(n)))`

Definition at line 129 of file lauxlib.h.

4.5.1.22 `#define luaL_optstring(L, n, d)(luaL_optlstring(L, (n), (d), NULL))`

Definition at line 117 of file lauxlib.h.

4.5.1.23 `#define luaL_prepbuffer(B) luaL_prepbuffsize(B, LUAL_BUFFERSIZE)`

Definition at line 164 of file lauxlib.h.

4.5.1.24 `#define luaL_typename(L, i) lua_typename(L, lua_type(L,(i)))`

Definition at line 119 of file lauxlib.h.

4.5.2 Typedef Documentation

4.5.2.1 `typedef struct luaL_Buffer luaL_Buffer`

4.5.2.2 `typedef struct luaL_Reg luaL_Reg`

4.5.2.3 `typedef struct luaL_Stream luaL_Stream`

4.5.3 Function Documentation

4.5.3.1 `LUALIB_API void() luaL_addlstring (luaL_Buffer * B, const char * s, size_t l)`

4.5.3.2 `LUALIB_API void() luaL_addstring (luaL_Buffer * B, const char * s)`

4.5.3.3 `LUALIB_API void() luaL_addvalue (luaL_Buffer * B)`

4.5.3.4 `LUALIB_API int() luaL_argerror (lua_State * L, int arg, const char * extrams)`

4.5.3.5 `LUALIB_API void() luaL_buffinit (lua_State * L, luaL_Buffer * B)`

4.5.3.6 `LUALIB_API char*() luaL_buffinitsize (lua_State * L, luaL_Buffer * B, size_t sz)`

4.5.3.7 `LUALIB_API int() luaL_callmeta (lua_State * L, int obj, const char * e)`

4.5.3.8 `LUALIB_API void() luaL_checkany (lua_State * L, int arg)`

4.5.3.9 `LUALIB_API lua_Integer() luaL_checkinteger (lua_State * L, int arg)`

4.5.3.10 `LUALIB_API const char*() luaL_checklstring (lua_State * L, int arg, size_t * l)`

4.5.3.11 `LUALIB_API lua_Number() luaL_checknumber (lua_State * L, int arg)`

4.5.3.12 `LUALIB_API int() luaL_checkoption (lua_State * L, int arg, const char * def, const char *const lst[])`

4.5.3.13 `LUALIB_API void() luaL_checkstack (lua_State * L, int sz, const char * msg)`

4.5.3.14 `LUALIB_API void() luaL_checktype (lua_State * L, int arg, int t)`

4.5.3.15 `LUALIB_API void*() luaL_checkudata (lua_State * L, int ud, const char * tname)`

4.5.3.16 `LUALIB_API void() luaL_checkversion_ (lua_State * L, lua_Number ver, size_t sz)`

- 4.5.3.17 LUALIB_API int() luaL_error (lua_State * L, const char * *fmt*, ...)
- 4.5.3.18 LUALIB_API int() luaL_execresult (lua_State * L, int *stat*)
- 4.5.3.19 LUALIB_API int() luaL_fileresult (lua_State * L, int *stat*, const char * *fname*)
- 4.5.3.20 LUALIB_API int() luaL_getmetafield (lua_State * L, int *obj*, const char * *e*)
- 4.5.3.21 LUALIB_API int() luaL_getsubtable (lua_State * L, int *idx*, const char * *fname*)
- 4.5.3.22 LUALIB_API const char*() luaL_gsub (lua_State * L, const char * *s*, const char * *p*, const char * *r*)
- 4.5.3.23 LUALIB_API lua_Integer() luaL_len (lua_State * L, int *idx*)
- 4.5.3.24 LUALIB_API int() luaL_loadbufferx (lua_State * L, const char * *buff*, size_t *sz*, const char * *name*, const char * *mode*)
- 4.5.3.25 LUALIB_API int() luaL_loadfilex (lua_State * L, const char * *filename*, const char * *mode*)
- 4.5.3.26 LUALIB_API int() luaL_loadstring (lua_State * L, const char * *s*)
- 4.5.3.27 LUALIB_API int() luaL_newmetatable (lua_State * L, const char * *tname*)
- 4.5.3.28 LUALIB_API lua_State*() luaL_newstate (void)
- 4.5.3.29 LUALIB_API lua_Integer() luaL_optinteger (lua_State * L, int *arg*, lua_Integer *def*)
- 4.5.3.30 LUALIB_API const char*() luaL_optlstring (lua_State * L, int *arg*, const char * *def*, size_t * *l*)
- 4.5.3.31 LUALIB_API lua_Number() luaL_optnumber (lua_State * L, int *arg*, lua_Number *def*)
- 4.5.3.32 LUALIB_API char*() luaL_prepbuffsize (luaL_Buffer * *B*, size_t *sz*)
- 4.5.3.33 LUALIB_API void() luaL_pushresult (luaL_Buffer * *B*)
- 4.5.3.34 LUALIB_API void() luaL_pushresultsize (luaL_Buffer * *B*, size_t *sz*)
- 4.5.3.35 LUALIB_API int() luaL_ref (lua_State * L, int *t*)
- 4.5.3.36 LUALIB_API void() luaL_requiref (lua_State * L, const char * *modname*, lua_CFunction *openf*, int *glb*)
- 4.5.3.37 LUALIB_API void() luaL_setfuncs (lua_State * L, const luaL_Reg * *l*, int *nup*)
- 4.5.3.38 LUALIB_API void() luaL_setmetatable (lua_State * L, const char * *tname*)
- 4.5.3.39 LUALIB_API void*() luaL_testudata (lua_State * L, int *ud*, const char * *tname*)
- 4.5.3.40 LUALIB_API const char*() luaL_tolstring (lua_State * L, int *idx*, size_t * *len*)
- 4.5.3.41 LUALIB_API void() luaL_traceback (lua_State * L, lua_State * *L1*, const char * *msg*, int *level*)
- 4.5.3.42 LUALIB_API void() luaL_unref (lua_State * L, int *t*, int *ref*)
- 4.5.3.43 LUALIB_API void() luaL_where (lua_State * L, int *lvl*)

4.6 wsh/include/libwitch/helper.h File Reference

Data Structures

- struct [section](#)

Functions

- int [read_maps](#) (int pid)
- int [is_mapped](#) (unsigned long int addr)

Variables

- struct [section](#) * [zfirst](#)
- int [nsections](#)

4.6.1 Function Documentation

4.6.1.1 int [is_mapped](#) (unsigned long int *addr*)

Definition at line 56 of file helper.c.

4.6.1.2 int [read_maps](#) (int *pid*)

Definition at line 72 of file helper.c.

4.6.2 Variable Documentation

4.6.2.1 int [nsections](#)

Definition at line 47 of file helper.c.

4.6.2.2 struct [section](#)* [zfirst](#)

Definition at line 46 of file helper.c.

4.7 wsh/include/libwitch/mylaux.h File Reference

Macros

- #define [luaL_newlibtable](#)(L, l) [lua_createtable](#)(L, 0, sizeof(l)/sizeof((l)[0]) - 1)
- #define [luaL_newlib](#)(L, l) ([luaL_checkversion](#)(L), [luaL_newlibtable](#)(L,l), [luaL_setfuncs](#)(L,l,0))
- #define [luaL_argcheck](#)(L, cond, arg, extrams) ((void)((cond) || [luaL_argerror](#)(L, (arg), (extrams))))
- #define [luaL_checkstring](#)(L, n) ([luaL_checklstring](#)(L, (n), NULL))
- #define [luaL_optstring](#)(L, n, d) ([luaL_optlstring](#)(L, (n), (d), NULL))
- #define [luaL_typename](#)(L, i) [lua_typename](#)(L, [lua_type](#)(L,(i)))
- #define [luaL_dofile](#)(L, fn) ([luaL_loadfile](#)(L, fn) || [lua_pcall](#)(L, 0, [LUA_MULTRET](#), 0))
- #define [luaL_dostring](#)(L, s) ([luaL_loadstring](#)(L, s) || [lua_pcall](#)(L, 0, [LUA_MULTRET](#), 0))
- #define [luaL_getmetatable](#)(L, n) ([lua_getfield](#)(L, [LUA_REGISTRYINDEX](#), (n)))
- #define [luaL_opt](#)(L, f, n, d) ([lua_isnoneornil](#)(L,(n)) ? (d) : [f](#)(L,(n)))
- #define [luaL_loadbuffer](#)(L, s, sz, n) [luaL_loadbufferx](#)(L,s,sz,n,NULL)

4.7.1 Macro Definition Documentation

4.7.1.1 `#define luaL_argcheck(L, cond, arg, extramsng) ((void)((cond) || luaL_argerror(L, (arg), (extramsng))))`

Definition at line 15 of file mylaux.h.

4.7.1.2 `#define luaL_checkstring(L, n) (luaL_checklstring(L, (n), NULL))`

Definition at line 17 of file mylaux.h.

4.7.1.3 `#define luaL_dofile(L, fn) (luaL_loadfile(L, fn) || lua_pcall(L, 0, LUA_MULTRET, 0))`

Definition at line 22 of file mylaux.h.

4.7.1.4 `#define luaL_dostring(L, s) (luaL_loadstring(L, s) || lua_pcall(L, 0, LUA_MULTRET, 0))`

Definition at line 25 of file mylaux.h.

4.7.1.5 `#define luaL_getmetatable(L, n) (lua_getfield(L, LUA_REGISTRYINDEX, (n)))`

Definition at line 28 of file mylaux.h.

4.7.1.6 `#define luaL_loadbuffer(L, s, sz, n) luaL_loadbufferx(L,s,sz,n,NULL)`

Definition at line 32 of file mylaux.h.

4.7.1.7 `#define luaL_newlib(L, l) (luaL_checkversion(L), luaL_newlibtable(L,l), luaL_setfuncs(L,l,0))`

Definition at line 12 of file mylaux.h.

4.7.1.8 `#define luaL_newlibtable(L, l) lua_createtable(L, 0, sizeof(l)/sizeof((l)[0]) - 1)`

Definition at line 9 of file mylaux.h.

4.7.1.9 `#define luaL_opt(L, f, n, d) (lua_isnoneornil(L,(n)) ? (d) : f(L,(n)))`

Definition at line 30 of file mylaux.h.

4.7.1.10 `#define luaL_optstring(L, n, d) (luaL_optlstring(L, (n), (d), NULL))`

Definition at line 18 of file mylaux.h.

4.7.1.11 `#define luaL_typename(L, i) lua_typename(L, lua_type(L,(i)))`

Definition at line 20 of file mylaux.h.

4.8 wsh/include/libwitch/sigs.h File Reference

Data Structures

- struct [signame_t](#)

Typedefs

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

Variables

- [signame_t](#) [signames](#) []

4.8.1 Typedef Documentation

4.8.1.1 typedef struct [signame_t](#) [signame_t](#)

4.8.2 Variable Documentation

4.8.2.1 [signame_t](#) [signames](#) []

Definition at line 6 of file sigs.h.

4.9 wsh/include/libwitch/wsh.h File Reference

```
#include <sys/prctl.h>
#include <setjmp.h>
#include <link.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <getopt.h>
#include <dlfcn.h>
#include <string.h>
#include <unistd.h>
#include <limits.h>
#include <errno.h>
#include <stdbool.h>
#include <sys/wait.h>
#include <poll.h>
#include <stropts.h>
#include <signal.h>
#include <malloc.h>
#include <sys/mman.h>
#include <ucontext.h>
#include <ctype.h>
#include <execinfo.h>
#include <pthread.h>
#include <sys/resource.h>
#include <sys/ptrace.h>
#include <longjmp.h>
#include <lua.h>
#include <luauxlib.h>
#include <lualib.h>
#include <linenoise.h>
#include "helper.h"
#include <colors.h>
#include <config.h>
#include <utlist.h>
```

Data Structures

- struct [elfdata_t](#)
- struct [range_t](#)
- struct [breakpoint_t](#)
- struct [preload_t](#)
- struct [script_t](#)
- struct [sections_t](#)
- struct [segments_t](#)
- struct [symbols_t](#)
- struct [eps_t](#)
- struct [wsh_t](#)
- struct [tuple_t](#)

Macros

- #define `_GNU_SOURCE`
- #define `USE_LUA` 1
- #define `DEFAULT_SCRIPT` `"/usr/share/wcc/scripts/debug"`
- #define `DEFAULT_SCRIPT_INDEX` `"/usr/share/wcc/scripts/INDEX"`
- #define `PROC_ASLR_PATH` `"/proc/sys/kernel/randomize_va_space"`
- #define `DEFAULT_LEARN_FILE` `"/.learnwitch.log"`
- #define `MAX_SIGNALS` 2000000
- #define `MY_CPU` 1
- #define `BIND_FLAGS` `RTLD_NOW`
- #define `DMGL_PARAMS` `(1 << 0)`
- #define `DMGL_ANSI` `(1 << 1)`
- #define `DMGL_ARM` `(1 << 11)`
- #define `Elf_Dyn` `Elf32_Dyn`
- #define `Elf_Ehdr` `Elf32_Ehdr`
- #define `Elf_Phdr` `Elf32_Phdr`
- #define `Elf_Shdr` `Elf32_Shdr`
- #define `Elf_Sym` `Elf32_Sym`
- #define `HPERMSEX` 5
- #define `ELF32_ST_BIND(val)` `((unsigned char) (val)) >> 4`
- #define `ELF32_ST_TYPE(val)` `((val) & 0xf)`
- #define `ELF32_ST_INFO(bind, type)` `((bind) << 4) + ((type) & 0xf)`
- #define `ELF64_ST_BIND(val)` `ELF32_ST_BIND (val)`
- #define `ELF64_ST_TYPE(val)` `ELF32_ST_TYPE (val)`
- #define `ELF64_ST_INFO(bind, type)` `ELF32_ST_INFO ((bind), (type))`
- #define `STB_LOCAL` 0
- #define `STB_GLOBAL` 1
- #define `STB_WEAK` 2
- #define `STB_GNU_UNIQUE` 10
- #define `STB_GNU_SECONDARY` 11
- #define `STT_NOTYPE` 0
- #define `STT_OBJECT` 1
- #define `STT_FUNC` 2
- #define `STT_SECTION` 3
- #define `STT_FILE` 4
- #define `STT_COMMON` 5
- #define `STT_TLS` 6
- #define `LINES_MAX` 50
- #define `read_arg1(arg1)`
- #define `read_arg2(arg2)`
- #define `read_arg3(arg3)`
- #define `read_arg4(arg4)`
- #define `read_arg(arg, j)`
- #define `SHELL_HISTORY_NAME` `"/.wsh_history"`
- #define `luaL_reg` `luaL_Reg`
- #define `MIN_BIN_SIZE` 10
- #define `FAULT_READ` 1
- #define `FAULT_WRITE` 2
- #define `FAULT_EXEC` 4
- #define `default_poison` `0x61`
- #define `SKIP_INIT` 3
- #define `SKIP_BOTTOM` 13

Typedefs

- typedef struct [range_t](#) [range_t](#)
- typedef struct [breakpoint_t](#) [breakpoint_t](#)
- typedef struct [preload_t](#) [preload_t](#)
- typedef struct [script_t](#) [script_t](#)
- typedef struct [sections_t](#) [sections_t](#)
- typedef struct [segments_t](#) [segments_t](#)
- typedef struct [symbols_t](#) [symbols_t](#)
- typedef struct [eps_t](#) [eps_t](#)
- typedef struct [wsh_t](#) [wsh_t](#)
- typedef struct [tuple_t](#) [tuple_t](#)

Functions

- char * [cplus_demangle](#) (const char *mangled, int options)
- int [do_loadlib](#) (char *libname)
- int [empty_phdrs](#) (void)
- int [empty_shdrs](#) (void)
- int [getsize](#) (lua_State *L)
- int [newarray](#) (lua_State *L)
- int [print_functions](#) (lua_State *L)
- int [print_libs](#) (lua_State *L)
- int [print_objects](#) (lua_State *L)
- int [print_phdrs](#) (void)
- int [print_shdrs](#) (void)
- int [entrypoints](#) (lua_State *L)
- int [print_symbols](#) (lua_State *L)
- int [print_version](#) (void)
- int [setarray](#) (lua_State *L)
- int [usage](#) (char *name)
- void [set_align_flag](#) (void)
- void [set_branch_flag](#) (void)
- void [set_trace_flag](#) (void)
- void [singlebranch](#) (lua_State *L)
- void [singlestep](#) (lua_State *L)
- void [traceunaligned](#) (lua_State *L)
- void [unset_align_flag](#) (void)
- void [unset_branch_flag](#) (void)
- void [unset_trace_flag](#) (void)
- void [unsinglebranch](#) (lua_State *L)
- void [unsinglestep](#) (lua_State *L)
- void [untraceunaligned](#) (lua_State *L)
- void [unverbostrace](#) (lua_State *L)
- void [verbostrace](#) (lua_State *L)
- void [xfree](#) (lua_State *L)
- void [systrace](#) (lua_State *L)
- void [rtrace](#) (lua_State *L)
- void [unsystrace](#) (lua_State *L)
- void [unrtrace](#) (lua_State *L)
- int [add_symbol](#) (char *symbol, char *libname, char *htype, char *hbind, unsigned long value, unsigned int size, unsigned long int addr)
- void [segment_add](#) (unsigned long int addr, unsigned long int size, char *perms, char *fname, char *ptype, int flags)

- int [alloccharbuf](#) (lua_State *L)
- int [bfmap](#) (lua_State *L)
- int [breakpoint](#) (lua_State *L)
- int [execlib](#) (lua_State *L)
- int [getcharbuf](#) (lua_State *L)
- int [grep](#) (lua_State *L)
- int [grepptr](#) (lua_State *L)
- int [help](#) (lua_State *L)
- int [hollywood](#) (lua_State *L)
- int [info](#) (lua_State *L)
- int [libcall](#) (lua_State *L)
- int [loadbin](#) (lua_State *L)
- int [man](#) (lua_State *L)
- int [map](#) (lua_State *L)
- int [phdrs](#) (lua_State *L)
- int [priv_memcpy](#) (lua_State *L)
- int [priv_strcat](#) (lua_State *L)
- int [priv_strcpy](#) (lua_State *L)
- int [rdnum](#) (lua_State *L)
- int [rdstr](#) (lua_State *L)
- int [setcharbuf](#) (lua_State *L)
- int [shdrs](#) (lua_State *L)
- int [verbose](#) (lua_State *L)
- int [xalloc](#) (lua_State *L)
- int [ralloc](#) (lua_State *L)
- int [headers](#) (lua_State *L)
- int [prototypes](#) (lua_State *L)
- int [bsspolute](#) (lua_State *L)
- unsigned int [ltrace](#) (void)
- int [procmmap_lua](#) (void)
- void [rescan](#) (void)
- void [hexdump](#) (uint8_t *data, size_t size, size_t colorstart, size_t color_len)
- int [disable_aslr](#) (void)
- int [enable_aslr](#) (void)
- void [script](#) (char *path)
- int [enable_core](#) (lua_State *L)
- int [disable_core](#) (lua_State *L)
- int [gencore](#) (lua_State *L)
- char * [signaltoname](#) (int signal)
- char * [sicode_strerror](#) (int signal, siginfo_t *s)
- int [rawmemread](#) (lua_State *L)
- int [rawmemwrite](#) (lua_State *L)
- int [rawmemstr](#) (lua_State *L)
- int [rawmemusage](#) (lua_State *L)
- int [rawmemaddr](#) (lua_State *L)
- int [rawmemstrlen](#) (lua_State *L)
- int [wsh_init](#) (void)
- int [wsh_getopt](#) (wsh_t *wsh1, int argc, char **argv)
- int [wsh_loadlibs](#) (void)
- int [reload_elfs](#) (void)
- int [wsh_run](#) (void)

Variables

- char * [__progname_full](#)

4.9.1 Macro Definition Documentation

4.9.1.1 #define _GNU_SOURCE

Definition at line 1 of file wsh.h.

4.9.1.2 #define BIND_FLAGS RTLD_NOW

Definition at line 113 of file wsh.h.

4.9.1.3 #define DEFAULT_LEARN_FILE "./learnwitch.log"

Definition at line 107 of file wsh.h.

4.9.1.4 #define default_poison 0x61

Definition at line 287 of file wsh.h.

4.9.1.5 #define DEFAULT_SCRIPT "/usr/share/wcc/scripts/debug"

Definition at line 103 of file wsh.h.

4.9.1.6 #define DEFAULT_SCRIPT_INDEX "/usr/share/wcc/scripts/INDEX"

Definition at line 104 of file wsh.h.

4.9.1.7 #define DMGL_ANSI (1 << 1)

Definition at line 123 of file wsh.h.

4.9.1.8 #define DMGL_ARM (1 << 11)

Definition at line 124 of file wsh.h.

4.9.1.9 #define DMGL_PARAMS (1 << 0)

Definition at line 122 of file wsh.h.

4.9.1.10 #define ELF32_ST_BIND(val) (((unsigned char) (val)) >> 4)

Definition at line 142 of file wsh.h.

4.9.1.11 #define ELF32_ST_INFO(bind, type) (((bind) << 4) + ((type) & 0xf))

Definition at line 144 of file wsh.h.

4.9.1.12 #define ELF32_ST_TYPE(val) ((val) & 0xf)

Definition at line 143 of file wsh.h.

4.9.1.13 `#define ELF64_ST_BIND(val) ELF32_ST_BIND(val)`

Definition at line 146 of file wsh.h.

4.9.1.14 `#define ELF64_ST_INFO(bind, type) ELF32_ST_INFO((bind), (type))`

Definition at line 148 of file wsh.h.

4.9.1.15 `#define ELF64_ST_TYPE(val) ELF32_ST_TYPE(val)`

Definition at line 147 of file wsh.h.

4.9.1.16 `#define Elf_Dyn Elf32_Dyn`

Definition at line 133 of file wsh.h.

4.9.1.17 `#define Elf_Ehdr Elf32_Ehdr`

Definition at line 134 of file wsh.h.

4.9.1.18 `#define Elf_Phdr Elf32_Phdr`

Definition at line 135 of file wsh.h.

4.9.1.19 `#define Elf_Shdr Elf32_Shdr`

Definition at line 136 of file wsh.h.

4.9.1.20 `#define Elf_Sym Elf32_Sym`

Definition at line 137 of file wsh.h.

4.9.1.21 `#define FAULT_EXEC 4`

Definition at line 285 of file wsh.h.

4.9.1.22 `#define FAULT_READ 1`

Definition at line 283 of file wsh.h.

4.9.1.23 `#define FAULT_WRITE 2`

Definition at line 284 of file wsh.h.

4.9.1.24 `#define HPERMSMAX 5`

Definition at line 140 of file wsh.h.

4.9.1.25 #define LINES_MAX 50

Definition at line 165 of file wsh.h.

4.9.1.26 #define luaL_reg luaL_Reg

Definition at line 279 of file wsh.h.

4.9.1.27 #define MAX_SIGNALS 2000000

Definition at line 109 of file wsh.h.

4.9.1.28 #define MIN_BIN_SIZE 10

Definition at line 281 of file wsh.h.

4.9.1.29 #define MY_CPU 1

Definition at line 111 of file wsh.h.

4.9.1.30 #define PROC_ASLR_PATH "/proc/sys/kernel/randomize_va_space"

Definition at line 105 of file wsh.h.

4.9.1.31 #define read_arg(arg, j)

Value:

```
{ \
    if (lua_isnil(L, j)) { \
        arg = 0; \
    } else if (lua_isnumber(L, j)) { \
        arg = (unsigned long) lua_tonumber(L, j); \
    } else if (lua_isstring(L, j)) { \
        arg = luaL_checkstring(L, j); \
    } else if (lua_istable(L, j)) { \
    } else if (lua_isfunction(L, j)) { \
        arg = lua_tocfunction(L, j); \
    } else if (lua_iscfunction(L, j)) { \
        arg = lua_touserdata(L, j); \
    } else if (lua_isuserdata(L, j)) { \
        arg = lua_touserdata(L, j); \
    } else { \
        arg = 0; \
    } \
}
```

Read argument number j

Definition at line 259 of file wsh.h.

4.9.1.32 #define read_arg1(arg1)

Value:

```
{ \
    if (lua_isnil(L, 1)) { \
        arg1 = 0; \
    } else if (lua_isnumber(L, 1)) { \
        arg1 = (unsigned long) lua_tonumber(L, 1); \
    }
```

```

    } else if (lua_isstring(L, 1)) { \
        arg1 = luaL_checkstring(L, 1); \
    } else if (lua_istable(L, 1)) { \
    } else if (lua_isfunction(L, 1)) { \
        arg1 = lua_tocfunction(L, 1); \
    } else if (lua_iscfunction(L, 1)) { \
        arg1 = lua_touserdata(L, 1); \
    } else if (lua_isuserdata(L, 1)) { \
        arg1 = lua_touserdata(L, 1); \
    } else { \
        arg1 = 0; \
    } \
}

```

Read arg1

Definition at line 171 of file wsh.h.

4.9.1.33 #define read_arg2(arg2)

Value:

```

{ \
    if (lua_isnil(L, 2)) { \
        arg2 = 0; \
    } else if (lua_isnumber(L, 2)) { \
        arg2 = (unsigned long) lua_tonumber(L, 2); \
    } else if (lua_isstring(L, 2)) { \
        arg2 = luaL_checkstring(L, 2); \
    } else if (lua_istable(L, 2)) { \
    } else if (lua_isfunction(L, 2)) { \
        arg2 = lua_tocfunction(L, 2); \
    } else if (lua_iscfunction(L, 2)) { \
        arg2 = lua_touserdata(L, 2); \
    } else if (lua_isuserdata(L, 2)) { \
        arg2 = lua_touserdata(L, 2); \
    } else { \
        arg2 = 0; \
    } \
}

```

Read arg2

Definition at line 193 of file wsh.h.

4.9.1.34 #define read_arg3(arg3)

Value:

```

{ \
    if (lua_isnil(L, 3)) { \
        arg3 = 0; \
    } else if (lua_isnumber(L, 3)) { \
        arg3 = (unsigned long) lua_tonumber(L, 3); \
    } else if (lua_isstring(L, 3)) { \
        arg3 = luaL_checkstring(L, 3); \
    } else if (lua_istable(L, 3)) { \
    } else if (lua_isfunction(L, 3)) { \
        arg3 = lua_tocfunction(L, 3); \
    } else if (lua_iscfunction(L, 3)) { \
        arg3 = lua_touserdata(L, 3); \
    } else if (lua_isuserdata(L, 3)) { \
        arg3 = lua_touserdata(L, 3); \
    } else { \
        arg3 = 0; \
    } \
}

```

Read arg3

Definition at line 215 of file wsh.h.

4.9.1.35 #define read_arg4(arg4)**Value:**

```
{ \
    if (lua_isnil(L, 4)) { \
        arg4 = 0; \
    } else if (lua_isnumber(L, 4)) { \
        arg4 = (unsigned long) lua_tonumber(L, 4); \
    } else if (lua_isstring(L, 4)) { \
        arg4 = luaL_checkstring(L, 4); \
    } else if (lua_istable(L, 4)) { \
    } else if (lua_isfunction(L, 4)) { \
        arg4 = lua_tocfunction(L, 4); \
    } else if (lua_iscfunction(L, 4)) { \
        arg4 = lua_touserdata(L, 4); \
    } else if (lua_isuserdata(L, 4)) { \
        arg4 = lua_touserdata(L, 4); \
    } else { \
        arg4 = 0; \
    } \
}
```

Read arg4

Definition at line 237 of file wsh.h.

4.9.1.36 #define SHELL_HISTORY_NAME ".wsh_history"

Definition at line 278 of file wsh.h.

4.9.1.37 #define SKIP_BOTTOM 13

Definition at line 297 of file wsh.h.

4.9.1.38 #define SKIP_INIT 3

Backtrace parameters

Definition at line 296 of file wsh.h.

4.9.1.39 #define STB_GLOBAL 1

Definition at line 151 of file wsh.h.

4.9.1.40 #define STB_GNU_SECONDARY 11

Definition at line 154 of file wsh.h.

4.9.1.41 #define STB_GNU_UNIQUE 10

Definition at line 153 of file wsh.h.

4.9.1.42 #define STB_LOCAL 0

Definition at line 150 of file wsh.h.

4.9.1.43 `#define STB_WEAK 2`

Definition at line 152 of file wsh.h.

4.9.1.44 `#define STT_COMMON 5`

Definition at line 161 of file wsh.h.

4.9.1.45 `#define STT_FILE 4`

Definition at line 160 of file wsh.h.

4.9.1.46 `#define STT_FUNC 2`

Definition at line 158 of file wsh.h.

4.9.1.47 `#define STT_NOTYPE 0`

Definition at line 156 of file wsh.h.

4.9.1.48 `#define STT_OBJECT 1`

Definition at line 157 of file wsh.h.

4.9.1.49 `#define STT_SECTION 3`

Definition at line 159 of file wsh.h.

4.9.1.50 `#define STT_TLS 6`

Definition at line 162 of file wsh.h.

4.9.1.51 `#define USE_LUA 1`

Definition at line 71 of file wsh.h.

4.9.2 Typedef Documentation

4.9.2.1 `typedef struct breakpoint_t breakpoint_t`

Breakpoint structure

4.9.2.2 `typedef struct eps_t eps_t`

4.9.2.3 `typedef struct preload_t preload_t`

Libraries to be preloaded (before shell/script execution)

4.9.2.4 typedef struct range_t range_t

Memory ranges

4.9.2.5 typedef struct script_t script_t

Scripts to be executed

4.9.2.6 typedef struct sections_t sections_t

Representation of ELF Sections

4.9.2.7 typedef struct segments_t segments_t

Representation of ELF Segments

4.9.2.8 typedef struct symbols_t symbols_t

Representation of ELF Symbols

4.9.2.9 typedef struct tuple_t tuple_t

4.9.2.10 typedef struct wsh_t wsh_t

wsh context

4.9.3 Function Documentation

4.9.3.1 int add_symbol (char * *symbol*, char * *libname*, char * *htype*, char * *hbind*, unsigned long *value*, unsigned int *size*, unsigned long int *addr*)

Add a symbol to linked list

Definition at line 719 of file wsh.c.

4.9.3.2 int alloccharbuf (lua_State * *L*)

Buffer management subroutines

Definition at line 1590 of file wsh.c.

4.9.3.3 int bfmap (lua_State * *L*)

Bruteforce valid memory mapping ranges

Definition at line 100 of file wsh.c.

4.9.3.4 int breakpoint (lua_State * *L*)

Set a breakpoint Make sure destination address is mapped

Change memory protections to RWX on destination's page

Backup byte at destination

Write Breakpoint

Save breakpoint informations

Definition at line 4218 of file wsh.c.

4.9.3.5 int bsspollute (lua_State * L)

Pollute .bss sections

Definition at line 3712 of file wsh.c.

4.9.3.6 char* cplus_demangle (const char * *mangled*, int *options*)

Imported declarations prototypes

4.9.3.7 int disable_aslr (void)

Disable ASLR

Definition at line 455 of file wsh.c.

4.9.3.8 int disable_core (lua_State * L)

Disable core files generation

Definition at line 4351 of file wsh.c.

4.9.3.9 int do_loadlib (char * *libname*)

Forward prototypes declarations

Do load a shared binary into the address space

Definition at line 4581 of file wsh.c.

4.9.3.10 int empty_phdrs (void)

Empty linked list of segments

Definition at line 999 of file wsh.c.

4.9.3.11 int empty_shdrs (void)

Empty linked list of sections

Definition at line 1018 of file wsh.c.

4.9.3.12 int enable_aslr (void)

Enable ASLR

Definition at line 473 of file wsh.c.

4.9.3.13 int enable_core (lua_State * L)

Enable core files generation

Definition at line 4359 of file wsh.c.

4.9.3.14 int entrypoints (lua_State * L)

Display ELF Entry points

Definition at line 1469 of file wsh.c.

4.9.3.15 int execlib (lua_State * L)

Definition at line 2792 of file wsh.c.

4.9.3.16 int gencore (lua_State * L)

Generate a core file

Definition at line 4340 of file wsh.c.

4.9.3.17 int getcharbuf (lua_State * L)

Definition at line 1657 of file wsh.c.

4.9.3.18 int getsize (lua_State * L)**4.9.3.19 int grep (lua_State * L)**

search a pattern over all sections mapped in memory

Definition at line 4069 of file wsh.c.

4.9.3.20 int grepptr (lua_State * L)

Search a given value in memory

grepptr(Pattern, patternlen, hexadumplen, nbytesbeforematch)

Definition at line 3979 of file wsh.c.

4.9.3.21 int headers (lua_State * L)

Generate headers generate headers for imported objects

generate forward prototypes for imported functions

Definition at line 931 of file wsh.c.

4.9.3.22 int help (lua_State * L)

Display help

Definition at line 574 of file wsh.c.

4.9.3.23 void hexdump (uint8_t * data, size_t size, size_t colorstart, size_t color_len)

Simple hexdump routine

Definition at line 184 of file wsh.c.

4.9.3.24 int hollywood (lua_State * L)

Definition at line 3632 of file wsh.c.

4.9.3.25 int info (lua_State * L)

Display information on an object/memory address Address is mapped

Search corresponding symbols

Search corresponding section

Search corresponding segment

Search corresponding symbols

Resolve symbol...

Definition at line 1495 of file wsh.c.

4.9.3.26 int libcall (lua_State * L)

Main wrapper around a library call. This function returns 9 values: ret (returned by library call), errno, firstsignal, total number of signals, firstsicode, firsterrno, faultaddr, reason, context Handle (reverse-) system calls tracing

Make the library call

Analyse return value

Learn prototypes

Create output execution context table

Push errno to lua table

Push strerror(errno) to lua table

Push first signal

Push first signal name

Push total of signals emitted during this libcall

Push first errno

Push first sicode

Push first sicode name

Address of last caller in backtrace

Push fault address

Push reason

Push mode

Push errctx

Push pointer to ucontext

Push arguments as a new table

Push number of non NULL arguments

Push retval

Push libcall/libname

Invoke store running function on context

Definition at line 2087 of file wsh.c.

4.9.3.27 int loadbin (lua_State * L)

Load a binary into the address space

Definition at line 4054 of file wsh.c.

4.9.3.28 unsigned int ltrace (void)

Definition at line 328 of file wsh.c.

4.9.3.29 int man (lua_State * L)

Open a manual page

Definition at line 1478 of file wsh.c.

4.9.3.30 int map (lua_State * L)

Display mapped sections

Definition at line 3658 of file wsh.c.

4.9.3.31 int newarray (lua_State * L)

4.9.3.32 int phdrs (lua_State * L)

Display Program headers (ELF Segments)

Definition at line 859 of file wsh.c.

4.9.3.33 int print_functions (lua_State * L)

Display functions

Definition at line 1176 of file wsh.c.

4.9.3.34 int print_libs (lua_State * L)

Display mapped libraries, return a list of library names

Definition at line 1308 of file wsh.c.

4.9.3.35 int print_objects (lua_State * L)

Display objects (typically globals)

Definition at line 1255 of file wsh.c.

4.9.3.36 int print_phdrs (void)

Display program headers (ELF Segments)

Definition at line 1052 of file wsh.c.

4.9.3.37 int print_shdrs (void)

Display ELF sections

Definition at line 1344 of file wsh.c.

4.9.3.38 int print_symbols (lua_State * L)

Display symbols

Definition at line 1108 of file wsh.c.

4.9.3.39 int print_version (void)

Definition at line 3821 of file wcc.c.

4.9.3.40 int priv_memcpy (lua_State * L)

Our own version of memcpy callable from LUA

Definition at line 4154 of file wsh.c.

4.9.3.41 int priv_strcat (lua_State * L)

Our own version of strcat callable from LUA

Definition at line 4197 of file wsh.c.

4.9.3.42 int priv_strcpy (lua_State * L)

Our own version of strcpy callable from LUA

Definition at line 4176 of file wsh.c.

4.9.3.43 int procmmap_lua (void)

Definition at line 2787 of file wsh.c.

4.9.3.44 int prototypes (lua_State * L)

Display learned prototypes Read all the lines to learnt data structure

Sort learnt data structures

Definition at line 1885 of file wsh.c.

4.9.3.45 int ralloc (lua_State * L)

ralloc(unsigned int size, unsigned char poison); allocate 1 page set to 0x00, set size bytes to poison, remap the page R only

Definition at line 3755 of file wsh.c.

4.9.3.46 int rawmemaddr (lua_State * L)

int addr rawmemaddr(obj)

Return the address in memory of the object passed as argument. Or returns an address itself if an address is given as argument.

Definition at line 4833 of file wsh.c.

4.9.3.47 int rawmemread (lua_State * L)

string res rawmemread(addr, len)

Read len bytes at address addr and return them as a lua string.

Definition at line 4759 of file wsh.c.

4.9.3.48 int rawmemstr (lua_State * L)

Returns a string, from an address passed as argument.

Definition at line 4797 of file wsh.c.

4.9.3.49 int rawmemstrlen (lua_State * L)

int rawmemstrlen(addr) Returns the length of a string passed as argument

Definition at line 4845 of file wsh.c.

4.9.3.50 int rawmemusage (lua_State * L)

Display memory usage.

Definition at line 4811 of file wsh.c.

4.9.3.51 int rawmemwrite (lua_State * L)

int written rawmemwrite(addr, data, len)

Raw write to addr of len bytes of data returns number of bytes written.

Definition at line 4778 of file wsh.c.

4.9.3.52 int rdnum (lua_State * L)

Read a number (to a LUA number)

Definition at line 1642 of file wsh.c.

4.9.3.53 int rdstr (lua_State * L)

Read a string (to a LUA string)

Definition at line 1621 of file wsh.c.

4.9.3.54 int reload_elfs (void)

Reload linked lists from ELF binaries

Definition at line 1441 of file wsh.c.

4.9.3.55 void rescan (void)

Rescan address space

Definition at line 2752 of file wsh.c.

4.9.3.56 void rtrace (lua_State * L)

Definition at line 3921 of file wsh.c.

4.9.3.57 void script (char * path)

Run a script

Definition at line 166 of file wsh.c.

4.9.3.58 void segment_add (unsigned long int *addr*, unsigned long int *size*, char * *perms*, char * *fname*, char * *pctype*, int *flags*)

Add a segment to linked list

Definition at line 769 of file wsh.c.

4.9.3.59 void set_align_flag (void) [inline]

Definition at line 2904 of file wsh.c.

4.9.3.60 void set_branch_flag (void) [inline]

Definition at line 2999 of file wsh.c.

4.9.3.61 void set_trace_flag (void) [inline]

Definition at line 2931 of file wsh.c.

4.9.3.62 int setarray (lua_State * L)**4.9.3.63 int setcharbuf (lua_State * L)**

Definition at line 1603 of file wsh.c.

4.9.3.64 int shdrs (lua_State * L)

Display section headers (ELF Sections)

Definition at line 1459 of file wsh.c.

4.9.3.65 char* sicode_strerror (int signal, siginfo_t * s)

Definition at line 3340 of file wsh.c.

4.9.3.66 char* signaltoname (int signal)

Definition at line 2878 of file wsh.c.

4.9.3.67 void singlebranch (lua_State * L)

Definition at line 3945 of file wsh.c.

4.9.3.68 void singlestep (lua_State * L)

Definition at line 3903 of file wsh.c.

4.9.3.69 void systrace (lua_State * L)

Definition at line 3916 of file wsh.c.

4.9.3.70 void traceunaligned (lua_State * L)

Resize a xallocated memory zone

Definition at line 3891 of file wsh.c.

4.9.3.71 void untrace (lua_State * L)

Definition at line 3931 of file wsh.c.

4.9.3.72 void unset_align_flag (void) [inline]

Definition at line 2890 of file wsh.c.

4.9.3.73 void unset_branch_flag (void) [inline]

Definition at line 3022 of file wsh.c.

4.9.3.74 void unset_trace_flag (void) [inline]

Definition at line 2917 of file wsh.c.

4.9.3.75 void unsinglebranch (lua_State * L)

Definition at line 3967 of file wsh.c.

4.9.3.76 void `unsinglestep (lua_State * L)`

Definition at line 3909 of file `wsh.c`.

4.9.3.77 void `unsystrace (lua_State * L)`

Definition at line 3926 of file `wsh.c`.

4.9.3.78 void `untraceunaligned (lua_State * L)`

Definition at line 3897 of file `wsh.c`.

4.9.3.79 void `unverbosetrace (lua_State * L)`

Definition at line 3941 of file `wsh.c`.

4.9.3.80 int `usage (char * name)`

Definition at line 3795 of file `wcc.c`.

4.9.3.81 int `verbose (lua_State * L)`

Definition at line 3618 of file `wsh.c`.

4.9.3.82 void `verbosetrace (lua_State * L)`

Definition at line 3937 of file `wsh.c`.

4.9.3.83 int `wsh_getopt (wsh_t * wsh1, int argc, char ** argv)`

Parse command line

Definition at line 4629 of file `wsh.c`.

4.9.3.84 int `wsh_init (void)`

Definition at line 4364 of file `wsh.c`.

4.9.3.85 int `wsh_loadlibs (void)`

Load all preload libraries

Definition at line 4608 of file `wsh.c`.

4.9.3.86 int `wsh_run (void)`

Run a lua shell/script Run all the scripts specified in the command line

Run a lua shell

Definition at line 4475 of file `wsh.c`.

4.9.3.87 int xalloc (lua_State * L)

xalloc(unsigned int size, unsigned char poison, unsigned int perms); Allocate size bytes (% getpagesize())

The mapping auto-references itself, unless a poison byte is given

[page unmaped] [mapped][OURPTR, size] [page unmaped]

Definition at line 3807 of file wsh.c.

4.9.3.88 void xfree (lua_State * L)

Release a bloc allocated via [xalloc\(\)](#)

Definition at line 3868 of file wsh.c.

4.9.4 Variable Documentation**4.9.4.1 char* __progrname_full**

Imported globals

4.10 wsh/include/libwitch/wsh_functions.h File Reference**Variables**

- char * [default_options](#) []
- char * [lua_default_functions](#) []
- char * [lua_blacklist](#) []
- [tuple_t](#) [exposed](#) []
- [range_t](#) [ranges](#) []
- unsigned int [global_xalloc](#) = 0

4.10.1 Variable Documentation**4.10.1.1 char* default_options[]**

Definition at line 6 of file wsh_functions.h.

4.10.1.2 tuple_t exposed[]

Definition at line 277 of file wsh_functions.h.

4.10.1.3 unsigned int global_xalloc = 0

Definition at line 352 of file wsh_functions.h.

4.10.1.4 char* lua_blacklist[]

Initial value:


```

= {
  "and",
  "break",
  "do",
  "else",
  "elseif",
  "end",
  "false",
  "for",
  "function",
  "if",
  "in",
  "local",
  "nil",
  "not",
  "or",
  "repeat",
  "return",
  "then",
  "true",
  "until",
  "while"
}

```

Definition at line 253 of file wsh_functions.h.

4.10.1.5 char* lua_default_functions[]

Definition at line 89 of file wsh_functions.h.

4.10.1.6 range_t ranges[]

Initial value:

```

= {
    {0x00000000, 0x100000000},
}

```

Definition at line 343 of file wsh_functions.h.

4.11 wsh/include/linenoise.h File Reference

Data Structures

- struct [linenoiseCompletions](#)

Typedefs

- typedef struct [linenoiseCompletions](#) [linenoiseCompletions](#)
- typedef void([linenoiseCompletionCallback](#))(const char *, [linenoiseCompletions](#) *)

Functions

- void [linenoiseSetCompletionCallback](#) ([linenoiseCompletionCallback](#) *)
- void [linenoiseAddCompletion](#) ([linenoiseCompletions](#) *, const char *)
- char * [linenoise](#) (const char *prompt)
- int [linenoiseHistoryAdd](#) (const char *line)

- int [linenoiseHistorySetMaxLen](#) (int len)
- int [linenoiseHistorySave](#) (const char *filename)
- int [linenoiseHistoryLoad](#) (const char *filename)
- void [linenoiseClearScreen](#) (void)
- void [linenoiseSetMultiLine](#) (int ml)
- void [linenoisePrintKeyCodes](#) (void)

4.11.1 Typedef Documentation

4.11.1.1 typedef void(linenoiseCompletionCallback)(const char *, linenoiseCompletions *)

Definition at line 51 of file linenoise.h.

4.11.1.2 typedef struct linenoiseCompletions linenoiseCompletions

4.11.2 Function Documentation

4.11.2.1 char* linenoise (const char * *prompt*)

4.11.2.2 void linenoiseAddCompletion (linenoiseCompletions *, const char *)

4.11.2.3 void linenoiseClearScreen (void)

4.11.2.4 int linenoiseHistoryAdd (const char * *line*)

4.11.2.5 int linenoiseHistoryLoad (const char * *filename*)

4.11.2.6 int linenoiseHistorySave (const char * *filename*)

4.11.2.7 int linenoiseHistorySetMaxLen (int *len*)

4.11.2.8 void linenoisePrintKeyCodes (void)

4.11.2.9 void linenoiseSetCompletionCallback (linenoiseCompletionCallback *)

4.11.2.10 void linenoiseSetMultiLine (int *ml*)

4.12 wsh/include/longjmp.h File Reference

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

Macros

- #define [TRY](#) do { jmp_buf ex_buf__; switch(setjmp(ex_buf__)) { case 0: while(1) {
- #define [CATCH](#)(x) break; case x:
- #define [FINALLY](#) break; } default: {
- #define [ETRY](#) break; } } while(0)
- #define [THROW](#)(x) longjmp(ex_buf__, x)

4.12.1 Macro Definition Documentation

4.12.1.1 #define CATCH(x) break; case x:

Definition at line 40 of file longjmp.h.

4.12.1.2 #define ETRY break; } }while(0)

Definition at line 42 of file longjmp.h.

4.12.1.3 #define FINALLY break; } default: {

Definition at line 41 of file longjmp.h.

4.12.1.4 #define THROW(x) longjmp(ex_buf_, x)

Definition at line 43 of file longjmp.h.

4.12.1.5 #define TRY do { jmp_buf ex_buf_; switch(setjmp(ex_buf_)) { case 0: while(1) {

This code taken from http://www.di.unipi.it/~nids/docs/longjump_try_throw_catch.-html Licensed under MIT License

Definition at line 39 of file longjmp.h.

4.13 wsh/include/lua.h File Reference

```
#include <stdarg.h>
#include <stddef.h>
#include "luaconf.h"
```

Data Structures

- struct [lua_Debug](#)

Macros

- #define [LUA_VERSION_MAJOR](#) "5"
- #define [LUA_VERSION_MINOR](#) "3"
- #define [LUA_VERSION_NUM](#) 503
- #define [LUA_VERSION_RELEASE](#) "2"
- #define [LUA_VERSION](#) "Lua " [LUA_VERSION_MAJOR](#) "." [LUA_VERSION_MINOR](#)
- #define [LUA_RELEASE](#) [LUA_VERSION](#) " " [LUA_VERSION_RELEASE](#)
- #define [LUA_COPYRIGHT](#) [LUA_RELEASE](#) " Copyright (C) 1994-2015 Lua.org, PUC-Rio"
- #define [LUA_AUTHORS](#) "R. Ierusalimschy, L. H. de Figueiredo, W. Celes"
- #define [LUA_SIGNATURE](#) "\x1bLua"
- #define [LUA_MULTRET](#) (-1)
- #define [LUA_REGISTRYINDEX](#) (-[LUA_MAXSTACK](#) - 1000)
- #define [lua_upvalueindex](#)(i) ([LUA_REGISTRYINDEX](#) - (i))
- #define [LUA_OK](#) 0

- #define [LUA_YIELD](#) 1
- #define [LUA_ERRRUN](#) 2
- #define [LUA_ERRSYNTAX](#) 3
- #define [LUA_ERRMEM](#) 4
- #define [LUA_ERRGCMM](#) 5
- #define [LUA_ERRERR](#) 6
- #define [LUA_TNONE](#) (-1)
- #define [LUA_TNIL](#) 0
- #define [LUA_TBOOLEAN](#) 1
- #define [LUA_TLIGHTUSERDATA](#) 2
- #define [LUA_TNUMBER](#) 3
- #define [LUA_TSTRING](#) 4
- #define [LUA_TTABLE](#) 5
- #define [LUA_TFUNCTION](#) 6
- #define [LUA_TUSERDATA](#) 7
- #define [LUA_TTHREAD](#) 8
- #define [LUA_NUMTAGS](#) 9
- #define [LUA_MINSTACK](#) 20
- #define [LUA_RIDX_MAINTHREAD](#) 1
- #define [LUA_RIDX_GLOBALS](#) 2
- #define [LUA_RIDX_LAST](#) [LUA_RIDX_GLOBALS](#)
- #define [LUA_OPADD](#) 0 /* ORDER TM, ORDER OP */
- #define [LUA_OPSUB](#) 1
- #define [LUA_OPMUL](#) 2
- #define [LUA_OPMOD](#) 3
- #define [LUA_OPPOW](#) 4
- #define [LUA_OPDIV](#) 5
- #define [LUA_OPIDIV](#) 6
- #define [LUA_OPBAND](#) 7
- #define [LUA_OPBOR](#) 8
- #define [LUA_OPBXOR](#) 9
- #define [LUA_OPSHL](#) 10
- #define [LUA_OPSHR](#) 11
- #define [LUA_OPUNM](#) 12
- #define [LUA_OPBNOT](#) 13
- #define [LUA_OPEQ](#) 0
- #define [LUA_OPLT](#) 1
- #define [LUA_OPLE](#) 2
- #define [lua_call](#)(L, n, r) [lua_callk](#)(L, (n), (r), 0, NULL)
- #define [lua_pcall](#)(L, n, r, f) [lua_pcallk](#)(L, (n), (r), (f), 0, NULL)
- #define [lua_yield](#)(L, n) [lua_yieldk](#)(L, (n), 0, NULL)
- #define [LUA_GCSTOP](#) 0
- #define [LUA_GCRESTART](#) 1
- #define [LUA_GCCOLLECT](#) 2
- #define [LUA_GCCOUNT](#) 3
- #define [LUA_GCCOUNTB](#) 4
- #define [LUA_GCSTEP](#) 5
- #define [LUA_GCSETPAUSE](#) 6
- #define [LUA_GCSETSTEPMUL](#) 7
- #define [LUA_GCISRUNNING](#) 9
- #define [lua_getextraspace](#)(L) ((void *)((char *) (L) - [LUA_EXTRASPACE](#)))
- #define [lua_tonumber](#)(L, i) [lua_tonumberx](#)(L, (i), NULL)
- #define [lua_tointeger](#)(L, i) [lua_tointegerx](#)(L, (i), NULL)
- #define [lua_pop](#)(L, n) [lua_settop](#)(L, -(n)-1)
- #define [lua_newtable](#)(L) [lua_createtable](#)(L, 0, 0)

- #define `lua_register(L, n, f)` (`lua_pushcfunction(L, (f)), lua_setglobal(L, (n))`)
- #define `lua_pushcfunction(L, f)` `lua_pushcclosure(L, (f), 0)`
- #define `lua_istfunction(L, n)` (`lua_type(L, (n)) == LUA_TFUNCTION`)
- #define `lua_istable(L, n)` (`lua_type(L, (n)) == LUA_TTABLE`)
- #define `lua_istlightuserdata(L, n)` (`lua_type(L, (n)) == LUA_TLIGHTUSERDATA`)
- #define `lua_isnil(L, n)` (`lua_type(L, (n)) == LUA_TNIL`)
- #define `lua_isboolean(L, n)` (`lua_type(L, (n)) == LUA_TBOOLEAN`)
- #define `lua_isthread(L, n)` (`lua_type(L, (n)) == LUA_TTHREAD`)
- #define `lua_isnone(L, n)` (`lua_type(L, (n)) == LUA_TNONE`)
- #define `lua_isnoneornil(L, n)` (`lua_type(L, (n)) <= 0`)
- #define `lua_pushliteral(L, s)` `lua_pushstring(L, "" s)`
- #define `lua_pushglobaltable(L)` `lua_rawgeti(L, LUA_REGISTRYINDEX, LUA_RIDX_GLOBALS)`
- #define `lua_tostring(L, i)` `lua_tolstring(L, (i), NULL)`
- #define `lua_insert(L, idx)` `lua_rotate(L, (idx), 1)`
- #define `lua_remove(L, idx)` (`lua_rotate(L, (idx), -1), lua_pop(L, 1)`)
- #define `lua_replace(L, idx)` (`lua_copy(L, -1, (idx)), lua_pop(L, 1)`)
- #define `LUA_HOOKCALL` 0
- #define `LUA_HOOKRET` 1
- #define `LUA_HOOKLINE` 2
- #define `LUA_HOOKCOUNT` 3
- #define `LUA_HOOKTAILCALL` 4
- #define `LUA_MASKCALL` (`1 << LUA_HOOKCALL`)
- #define `LUA_MASKRET` (`1 << LUA_HOOKRET`)
- #define `LUA_MASKLINE` (`1 << LUA_HOOKLINE`)
- #define `LUA_MASKCOUNT` (`1 << LUA_HOOKCOUNT`)

Typedefs

- typedef struct `lua_State` `lua_State`
- typedef `LUA_NUMBER` `lua_Number`
- typedef `LUA_INTEGER` `lua_Integer`
- typedef `LUA_UNSIGNED` `lua_Unsigned`
- typedef `LUA_KCONTEXT` `lua_KContext`
- typedef int(* `lua_CFunction`)(`lua_State *L`)
- typedef int(* `lua_KFunction`)(`lua_State *L`, int status, `lua_KContext` ctx)
- typedef const char *(`lua_Reader`)(`lua_State *L`, void *ud, size_t *sz)
- typedef int(* `lua_Writer`)(`lua_State *L`, const void *p, size_t sz, void *ud)
- typedef void *(`lua_Alloc`)(void *ud, void *ptr, size_t osize, size_t nsize)
- typedef struct `lua_Debug` `lua_Debug`
- typedef void(* `lua_Hook`)(`lua_State *L`, `lua_Debug` *ar)

Functions

- `LUA_API` `lua_State *`(`lua_Alloc` f, void *ud)
- `LUA_API` void(`lua_State *L`) `lua_close`
- `LUA_API` `lua_State *`(`lua_State *L`) `lua_newthread`
- `LUA_API` `lua_CFunction`(`lua_State *L`) `lua_atpanic` (`lua_CFunction` panicf)
- `LUA_API` const `lua_Number *`(`lua_State *L`) `lua_version`
- `LUA_API` int(`lua_State *L`, int idx) `lua_absindex`
- `LUA_API` int(`lua_State *L`) `lua_gettop`
- `LUA_API` void(`lua_State *L`, int idx) `lua_settop`
- `LUA_API` void(`lua_State *L`, int idx) `lua_pushvalue`
- `LUA_API` void(`lua_State *L`, int idx, int n) `lua_rotate`
- `LUA_API` void(`lua_State *L`, int fromidx, int toidx) `lua_copy`

- [LUA_API](#) int() [lua_checkstack](#) ([lua_State](#) *L, int n)
- [LUA_API](#) void() [lua_xmove](#) ([lua_State](#) *from, [lua_State](#) *to, int n)
- [LUA_API](#) int() [lua_isnumber](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) int() [lua_isstring](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) int() [lua_iscfunction](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) int() [lua_isinteger](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) int() [lua_isuserdata](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) int() [lua_type](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) const char *() [lua_typename](#) ([lua_State](#) *L, int tp)
- [LUA_API](#) [lua_Number](#)() [lua_tonumberx](#) ([lua_State](#) *L, int idx, int *isnum)
- [LUA_API](#) [lua_Integer](#)() [lua_tointegerx](#) ([lua_State](#) *L, int idx, int *isnum)
- [LUA_API](#) int() [lua_toboolean](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) const char *() [lua_tolstring](#) ([lua_State](#) *L, int idx, size_t *len)
- [LUA_API](#) size_t() [lua_rawlen](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) [lua_CFunction](#)() [lua_tocfunction](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) void *() [lua_touserdata](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) [lua_State](#) *() [lua_tothread](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) const void *() [lua_topointer](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) void() [lua_arith](#) ([lua_State](#) *L, int op)
- [LUA_API](#) int() [lua_rawequal](#) ([lua_State](#) *L, int idx1, int idx2)
- [LUA_API](#) int() [lua_compare](#) ([lua_State](#) *L, int idx1, int idx2, int op)
- [LUA_API](#) void() [lua_pushnil](#) ([lua_State](#) *L)
- [LUA_API](#) void() [lua_pushnumber](#) ([lua_State](#) *L, [lua_Number](#) n)
- [LUA_API](#) void() [lua_pushinteger](#) ([lua_State](#) *L, [lua_Integer](#) n)
- [LUA_API](#) const char *() [lua_pushlstring](#) ([lua_State](#) *L, const char *s, size_t len)
- [LUA_API](#) const char *() [lua_pushstring](#) ([lua_State](#) *L, const char *s)
- [LUA_API](#) const char *() [lua_pushvfstring](#) ([lua_State](#) *L, const char *fmt, va_list argp)
- [LUA_API](#) const char *() [lua_pushfstring](#) ([lua_State](#) *L, const char *fmt,...)
- [LUA_API](#) void() [lua_pushcclosure](#) ([lua_State](#) *L, [lua_CFunction](#) fn, int n)
- [LUA_API](#) void() [lua_pushboolean](#) ([lua_State](#) *L, int b)
- [LUA_API](#) void() [lua_pushlightuserdata](#) ([lua_State](#) *L, void *p)
- [LUA_API](#) int() [lua_pushthread](#) ([lua_State](#) *L)
- [LUA_API](#) int() [lua_getglobal](#) ([lua_State](#) *L, const char *name)
- [LUA_API](#) int() [lua_gettable](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) int() [lua_getfield](#) ([lua_State](#) *L, int idx, const char *k)
- [LUA_API](#) int() [lua_geti](#) ([lua_State](#) *L, int idx, [lua_Integer](#) n)
- [LUA_API](#) int() [lua_rawget](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) int() [lua_rawgeti](#) ([lua_State](#) *L, int idx, [lua_Integer](#) n)
- [LUA_API](#) int() [lua_rawgetp](#) ([lua_State](#) *L, int idx, const void *p)
- [LUA_API](#) void() [lua_createtable](#) ([lua_State](#) *L, int narr, int nrec)
- [LUA_API](#) void *() [lua_newuserdata](#) ([lua_State](#) *L, size_t sz)
- [LUA_API](#) int() [lua_getmetatable](#) ([lua_State](#) *L, int objindex)
- [LUA_API](#) int() [lua_getuservalue](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) void() [lua_setglobal](#) ([lua_State](#) *L, const char *name)
- [LUA_API](#) void() [lua_settable](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) void() [lua_setfield](#) ([lua_State](#) *L, int idx, const char *k)
- [LUA_API](#) void() [lua_seti](#) ([lua_State](#) *L, int idx, [lua_Integer](#) n)
- [LUA_API](#) void() [lua_rawset](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) void() [lua_rawseti](#) ([lua_State](#) *L, int idx, [lua_Integer](#) n)
- [LUA_API](#) void() [lua_rawsetp](#) ([lua_State](#) *L, int idx, const void *p)
- [LUA_API](#) int() [lua_setmetatable](#) ([lua_State](#) *L, int objindex)
- [LUA_API](#) void() [lua_setuservalue](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) void() [lua_callk](#) ([lua_State](#) *L, int nargs, int nresults, [lua_KContext](#) ctx, [lua_KFunction](#) k)
- [LUA_API](#) int() [lua_pcallk](#) ([lua_State](#) *L, int nargs, int nresults, int errfunc, [lua_KContext](#) ctx, [lua_KFunction](#) k)

- [LUA_API](#) int() [lua_load](#) ([lua_State](#) *L, [lua_Reader](#) reader, void *dt, const char *chunkname, const char *mode)
- [LUA_API](#) int() [lua_dump](#) ([lua_State](#) *L, [lua_Writer](#) writer, void *data, int strip)
- [LUA_API](#) int() [lua_yieldk](#) ([lua_State](#) *L, int nresults, [lua_KContext](#) ctx, [lua_KFunction](#) k)
- [LUA_API](#) int() [lua_resume](#) ([lua_State](#) *L, [lua_State](#) *from, int narg)
- [LUA_API](#) int() [lua_status](#) ([lua_State](#) *L)
- [LUA_API](#) int() [lua_isyieldable](#) ([lua_State](#) *L)
- [LUA_API](#) int() [lua_gc](#) ([lua_State](#) *L, int what, int data)
- [LUA_API](#) int() [lua_error](#) ([lua_State](#) *L)
- [LUA_API](#) int() [lua_next](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) void() [lua_concat](#) ([lua_State](#) *L, int n)
- [LUA_API](#) void() [lua_len](#) ([lua_State](#) *L, int idx)
- [LUA_API](#) size_t() [lua_stringtonumber](#) ([lua_State](#) *L, const char *s)
- [LUA_API](#) [lua_Alloc](#)() [lua_getallocf](#) ([lua_State](#) *L, void **ud)
- [LUA_API](#) void() [lua_setallocf](#) ([lua_State](#) *L, [lua_Alloc](#) f, void *ud)
- [LUA_API](#) int() [lua_getstack](#) ([lua_State](#) *L, int level, [lua_Debug](#) *ar)
- [LUA_API](#) int() [lua_getinfo](#) ([lua_State](#) *L, const char *what, [lua_Debug](#) *ar)
- [LUA_API](#) const char *() [lua_getlocal](#) ([lua_State](#) *L, const [lua_Debug](#) *ar, int n)
- [LUA_API](#) const char *() [lua_setlocal](#) ([lua_State](#) *L, const [lua_Debug](#) *ar, int n)
- [LUA_API](#) const char *() [lua_getupvalue](#) ([lua_State](#) *L, int funcindex, int n)
- [LUA_API](#) const char *() [lua_setupvalue](#) ([lua_State](#) *L, int funcindex, int n)
- [LUA_API](#) void *() [lua_upvalueid](#) ([lua_State](#) *L, int fidx, int n)
- [LUA_API](#) void() [lua_upvaluejoin](#) ([lua_State](#) *L, int fidx1, int n1, int fidx2, int n2)
- [LUA_API](#) void() [lua_sethook](#) ([lua_State](#) *L, [lua_Hook](#) func, int mask, int count)
- [LUA_API](#) [lua_Hook](#)() [lua_gethook](#) ([lua_State](#) *L)
- [LUA_API](#) int() [lua_gethookmask](#) ([lua_State](#) *L)
- [LUA_API](#) int() [lua_gethookcount](#) ([lua_State](#) *L)

Variables

- const char [lua_ident](#) []

4.13.1 Macro Definition Documentation

4.13.1.1 `#define LUA_AUTHORS "R. Ierusalimschy, L. H. de Figueiredo, W. Celes"`

Definition at line 27 of file lua.h.

4.13.1.2 `#define lua_call(L, n, r) lua_callk(L, (n), (r), 0, NULL)`

Definition at line 274 of file lua.h.

4.13.1.3 `#define LUA_COPYRIGHT LUA_RELEASE " Copyright (C) 1994-2015 Lua.org, PUC-Rio"`

Definition at line 26 of file lua.h.

4.13.1.4 `#define LUA_ERRERR 6`

Definition at line 53 of file lua.h.

4.13.1.5 #define LUA_ERRGCMM 5

Definition at line 52 of file lua.h.

4.13.1.6 #define LUA_ERRMEM 4

Definition at line 51 of file lua.h.

4.13.1.7 #define LUA_ERRRUN 2

Definition at line 49 of file lua.h.

4.13.1.8 #define LUA_ERRSYNTAX 3

Definition at line 50 of file lua.h.

4.13.1.9 #define LUA_GCCOLLECT 2

Definition at line 304 of file lua.h.

4.13.1.10 #define LUA_GCCOUNT 3

Definition at line 305 of file lua.h.

4.13.1.11 #define LUA_GCCOUNTB 4

Definition at line 306 of file lua.h.

4.13.1.12 #define LUA_GCISRUNNING 9

Definition at line 310 of file lua.h.

4.13.1.13 #define LUA_GCRESTART 1

Definition at line 303 of file lua.h.

4.13.1.14 #define LUA_GCSETPAUSE 6

Definition at line 308 of file lua.h.

4.13.1.15 #define LUA_GCSETSTEMUL 7

Definition at line 309 of file lua.h.

4.13.1.16 #define LUA_GCSTEP 5

Definition at line 307 of file lua.h.

4.13.1.17 `#define LUA_GCSTOP 0`

Definition at line 302 of file lua.h.

4.13.1.18 `#define lua_getextraspac(L) ((void *)((char *)L - LUA_EXTRASPACE))`

Definition at line 339 of file lua.h.

4.13.1.19 `#define LUA_HOOKCALL 0`

Definition at line 402 of file lua.h.

4.13.1.20 `#define LUA_HOOKCOUNT 3`

Definition at line 405 of file lua.h.

4.13.1.21 `#define LUA_HOOKLINE 2`

Definition at line 404 of file lua.h.

4.13.1.22 `#define LUA_HOOKRET 1`

Definition at line 403 of file lua.h.

4.13.1.23 `#define LUA_HOOKTAILCALL 4`

Definition at line 406 of file lua.h.

4.13.1.24 `#define lua_insert(L, idx) lua_rotate(L, (idx), 1)`

Definition at line 369 of file lua.h.

4.13.1.25 `#define lua_isboolean(L, n) (lua_type(L, (n)) == LUA_TBOOLEAN)`

Definition at line 356 of file lua.h.

4.13.1.26 `#define lua_isfunction(L, n) (lua_type(L, (n)) == LUA_TFUNCTION)`

Definition at line 352 of file lua.h.

4.13.1.27 `#define lua_isthread(L, n) (lua_type(L, (n)) == LUA_TTHREAD)`

Definition at line 354 of file lua.h.

4.13.1.28 `#define lua_isnil(L, n) (lua_type(L, (n)) == LUA_TNIL)`

Definition at line 355 of file lua.h.

4.13.1.29 `#define lua_isnone(L, n) (lua_type(L, n) == LUA_TNONE)`

Definition at line 358 of file lua.h.

4.13.1.30 `#define lua_isnoneornil(L, n) (lua_type(L, n) <= 0)`

Definition at line 359 of file lua.h.

4.13.1.31 `#define lua_istable(L, n) (lua_type(L, n) == LUA_TTABLE)`

Definition at line 353 of file lua.h.

4.13.1.32 `#define lua_isthread(L, n) (lua_type(L, n) == LUA_TTHREAD)`

Definition at line 357 of file lua.h.

4.13.1.33 `#define LUA_MASKCALL (1 << LUA_HOOKCALL)`

Definition at line 412 of file lua.h.

4.13.1.34 `#define LUA_MASKCOUNT (1 << LUA_HOOKCOUNT)`

Definition at line 415 of file lua.h.

4.13.1.35 `#define LUA_MASKLINE (1 << LUA_HOOKLINE)`

Definition at line 414 of file lua.h.

4.13.1.36 `#define LUA_MASKRET (1 << LUA_HOOKRET)`

Definition at line 413 of file lua.h.

4.13.1.37 `#define LUA_MINSTACK 20`

Definition at line 79 of file lua.h.

4.13.1.38 `#define LUA_MULTRET (-1)`

Definition at line 34 of file lua.h.

4.13.1.39 `#define lua_newtable(L) lua_createtable(L, 0, 0)`

Definition at line 346 of file lua.h.

4.13.1.40 `#define LUA_NUMTAGS 9`

Definition at line 74 of file lua.h.

4.13.1.41 `#define LUA_OK 0`

Definition at line 47 of file lua.h.

4.13.1.42 `#define LUA_OPADD 0 /* ORDER TM, ORDER OP */`

Definition at line 196 of file lua.h.

4.13.1.43 `#define LUA_OPBAND 7`

Definition at line 203 of file lua.h.

4.13.1.44 `#define LUA_OPBNOT 13`

Definition at line 209 of file lua.h.

4.13.1.45 `#define LUA_OPBOR 8`

Definition at line 204 of file lua.h.

4.13.1.46 `#define LUA_OPBXOR 9`

Definition at line 205 of file lua.h.

4.13.1.47 `#define LUA_OPDIV 5`

Definition at line 201 of file lua.h.

4.13.1.48 `#define LUA_OPEQ 0`

Definition at line 213 of file lua.h.

4.13.1.49 `#define LUA_OPIDIV 6`

Definition at line 202 of file lua.h.

4.13.1.50 `#define LUA_OPLE 2`

Definition at line 215 of file lua.h.

4.13.1.51 `#define LUA_OPLT 1`

Definition at line 214 of file lua.h.

4.13.1.52 `#define LUA_OPMOD 3`

Definition at line 199 of file lua.h.

4.13.1.53 `#define LUA_OPMUL 2`

Definition at line 198 of file lua.h.

4.13.1.54 `#define LUA_OPPOW 4`

Definition at line 200 of file lua.h.

4.13.1.55 `#define LUA_OPSHL 10`

Definition at line 206 of file lua.h.

4.13.1.56 `#define LUA_OPSHR 11`

Definition at line 207 of file lua.h.

4.13.1.57 `#define LUA_OPSUB 1`

Definition at line 197 of file lua.h.

4.13.1.58 `#define LUA_OPUNM 12`

Definition at line 208 of file lua.h.

4.13.1.59 `#define lua_pcall(L, n, r, f) lua_pcallk(L, (n), (r), (f), 0, NULL)`

Definition at line 278 of file lua.h.

4.13.1.60 `#define lua_pop(L, n) lua_settop(L, -(n)-1)`

Definition at line 344 of file lua.h.

4.13.1.61 `#define lua_pushcfunction(L, f) lua_pushcclosure(L, (f), 0)`

Definition at line 350 of file lua.h.

4.13.1.62 `#define lua_pushglobaltable(L) lua_rawgeti(L, LUA_REGISTRYINDEX, LUA_RIDX_GLOBALS)`

Definition at line 363 of file lua.h.

4.13.1.63 `#define lua_pushliteral(L, s) lua_pushstring(L, "" s)`

Definition at line 361 of file lua.h.

4.13.1.64 `#define lua_register(L, n, f)(lua_pushcfunction(L, (f)), lua_setglobal(L, (n)))`

Definition at line 348 of file lua.h.

4.13.1.65 `#define LUA_REGISTRYINDEX (-LUA_MAXSTACK - 1000)`

Definition at line 42 of file lua.h.

4.13.1.66 `#define LUA_RELEASE LUA_VERSION "." LUA_VERSION_RELEASE`

Definition at line 25 of file lua.h.

4.13.1.67 `#define lua_remove(L, idx) (lua_rotate(L, (idx), -1), lua_pop(L, 1))`

Definition at line 371 of file lua.h.

4.13.1.68 `#define lua_replace(L, idx) (lua_copy(L, -1, (idx)), lua_pop(L, 1))`

Definition at line 373 of file lua.h.

4.13.1.69 `#define LUA_RIDX_GLOBALS 2`

Definition at line 84 of file lua.h.

4.13.1.70 `#define LUA_RIDX_LAST LUA_RIDX_GLOBALS`

Definition at line 85 of file lua.h.

4.13.1.71 `#define LUA_RIDX_MAINTHREAD 1`

Definition at line 83 of file lua.h.

4.13.1.72 `#define LUA_SIGNATURE "\x1bLua"`

Definition at line 31 of file lua.h.

4.13.1.73 `#define LUA_TBOOLEAN 1`

Definition at line 65 of file lua.h.

4.13.1.74 `#define LUA_TFUNCTION 6`

Definition at line 70 of file lua.h.

4.13.1.75 `#define LUA_TLIGHTUSERDATA 2`

Definition at line 66 of file lua.h.

4.13.1.76 `#define LUA_TNIL 0`

Definition at line 64 of file lua.h.

4.13.1.77 `#define LUA_TNONE (-1)`

Definition at line 62 of file lua.h.

4.13.1.78 `#define LUA_TNUMBER 3`

Definition at line 67 of file lua.h.

4.13.1.79 `#define lua_tointeger(L, i) lua_tointegerx(L,(i),NULL)`

Definition at line 342 of file lua.h.

4.13.1.80 `#define lua_tonumber(L, i) lua_tonumberx(L,(i),NULL)`

Definition at line 341 of file lua.h.

4.13.1.81 `#define lua_tostring(L, i) lua_tolstring(L, (i), NULL)`

Definition at line 366 of file lua.h.

4.13.1.82 `#define LUA_TSTRING 4`

Definition at line 68 of file lua.h.

4.13.1.83 `#define LUA_TTABLE 5`

Definition at line 69 of file lua.h.

4.13.1.84 `#define LUA_TTHREAD 8`

Definition at line 72 of file lua.h.

4.13.1.85 `#define LUA_TUSERDATA 7`

Definition at line 71 of file lua.h.

4.13.1.86 `#define lua_upvalueindex(i) (LUA_REGISTRYINDEX - (i))`

Definition at line 43 of file lua.h.

4.13.1.87 `#define LUA_VERSION "Lua " LUA_VERSION_MAJOR "." LUA_VERSION_MINOR`

Definition at line 24 of file lua.h.

4.13.1.88 `#define LUA_VERSION_MAJOR "5"`

Definition at line 19 of file lua.h.

4.13.1.89 `#define LUA_VERSION_MINOR "3"`

Definition at line 20 of file lua.h.

4.13.1.90 `#define LUA_VERSION_NUM 503`

Definition at line 21 of file lua.h.

4.13.1.91 `#define LUA_VERSION_RELEASE "2"`

Definition at line 22 of file lua.h.

4.13.1.92 `#define LUA_YIELD 1`

Definition at line 48 of file lua.h.

4.13.1.93 `#define lua_yield(L, n) lua_yieldk(L, (n), 0, NULL)`

Definition at line 295 of file lua.h.

4.13.2 Typedef Documentation

4.13.2.1 `typedef void>(* lua_Alloc)(void *ud, void *ptr, size_t osize, size_t nsize)`

Definition at line 124 of file lua.h.

4.13.2.2 `typedef int(* lua_CFunction)(lua_State *L)`

Definition at line 105 of file lua.h.

4.13.2.3 `typedef struct lua_Debug lua_Debug`

Definition at line 417 of file lua.h.

4.13.2.4 `typedef void(* lua_Hook)(lua_State *L, lua_Debug *ar)`

Definition at line 421 of file lua.h.

4.13.2.5 `typedef LUA_INTEGER lua_Integer`

Definition at line 93 of file lua.h.

4.13.2.6 `typedef LUA_KCONTEXT lua_KContext`

Definition at line 99 of file lua.h.

4.13.2.7 `typedef int(* lua_KFunction)(lua_State *L, int status, lua_KContext ctx)`

Definition at line 110 of file lua.h.

4.13.2.8 typedef LUA_NUMBER lua_Number

Definition at line 89 of file lua.h.

4.13.2.9 typedef const char*(** lua_Reader*)(lua_State *L, void *ud, size_t *sz)

Definition at line 116 of file lua.h.

4.13.2.10 typedef struct lua_State lua_State

Definition at line 56 of file lua.h.

4.13.2.11 typedef LUA_UNSIGNED lua_Unsigned

Definition at line 96 of file lua.h.

4.13.2.12 typedef int(** lua_Writer*)(lua_State *L, const void *p, size_t sz, void *ud)

Definition at line 118 of file lua.h.

4.13.3 Function Documentation

4.13.3.1 LUA_API int() lua_absindex (lua_State * L, int *idx*)

4.13.3.2 LUA_API void() lua_arith (lua_State * L, int *op*)

4.13.3.3 LUA_API lua_CFunction() lua_atpanic (lua_State * L, lua_CFunction *panicf*)

4.13.3.4 LUA_API void() lua_callk (lua_State * L, int *nargs*, int *nresults*, lua_KContext *ctx*, lua_KFunction *k*)

4.13.3.5 LUA_API int() lua_checkstack (lua_State * L, int *n*)

4.13.3.6 LUA_API void() lua_close (lua_State * L)

4.13.3.7 LUA_API int() lua_compare (lua_State * L, int *idx1*, int *idx2*, int *op*)

4.13.3.8 LUA_API void() lua_concat (lua_State * L, int *n*)

4.13.3.9 LUA_API void() lua_copy (lua_State * L, int *fromidx*, int *toidx*)

4.13.3.10 LUA_API void() lua_createtable (lua_State * L, int *narr*, int *nrec*)

4.13.3.11 LUA_API int() lua_dump (lua_State * L, lua_Writer *writer*, void * *data*, int *strip*)

4.13.3.12 LUA_API int() lua_error (lua_State * L)

4.13.3.13 LUA_API int() lua_gc (lua_State * L, int *what*, int *data*)

4.13.3.14 LUA_API lua_Alloc() lua_getallocf (lua_State * L, void ** *ud*)

4.13.3.15 LUA_API int() lua_getfield (lua_State * L, int *idx*, const char * *k*)

- 4.13.3.16 `LUA_API int() lua_getglobal (lua_State * L, const char * name)`
- 4.13.3.17 `LUA_API lua_Hook() lua_gethook (lua_State * L)`
- 4.13.3.18 `LUA_API int() lua_gethookcount (lua_State * L)`
- 4.13.3.19 `LUA_API int() lua_gethookmask (lua_State * L)`
- 4.13.3.20 `LUA_API int() lua_geti (lua_State * L, int idx, lua_Integer n)`
- 4.13.3.21 `LUA_API int() lua_getinfo (lua_State * L, const char * what, lua_Debug * ar)`
- 4.13.3.22 `LUA_API const char*() lua_getlocal (lua_State * L, const lua_Debug * ar, int n)`
- 4.13.3.23 `LUA_API int() lua_getmetatable (lua_State * L, int objindex)`
- 4.13.3.24 `LUA_API int() lua_getstack (lua_State * L, int level, lua_Debug * ar)`
- 4.13.3.25 `LUA_API int() lua_gettable (lua_State * L, int idx)`
- 4.13.3.26 `LUA_API int() lua_gettop (lua_State * L)`
- 4.13.3.27 `LUA_API const char*() lua_getupvalue (lua_State * L, int funcindex, int n)`
- 4.13.3.28 `LUA_API int() lua_getuservalue (lua_State * L, int idx)`
- 4.13.3.29 `LUA_API int() lua_isfunction (lua_State * L, int idx)`
- 4.13.3.30 `LUA_API int() lua_isinteger (lua_State * L, int idx)`
- 4.13.3.31 `LUA_API int() lua_isnumber (lua_State * L, int idx)`
- 4.13.3.32 `LUA_API int() lua_isstring (lua_State * L, int idx)`
- 4.13.3.33 `LUA_API int() lua_isuserdata (lua_State * L, int idx)`
- 4.13.3.34 `LUA_API int() lua_isyieldable (lua_State * L)`
- 4.13.3.35 `LUA_API void() lua_len (lua_State * L, int idx)`
- 4.13.3.36 `LUA_API int() lua_load (lua_State * L, lua_Reader reader, void * dt, const char * chunkname, const char * mode)`
- 4.13.3.37 `LUA_API lua_State*() lua_newstate (lua_Alloc f, void * ud)`
- 4.13.3.38 `LUA_API lua_State*() lua_newthread (lua_State * L)`
- 4.13.3.39 `LUA_API void*() lua_newuserdata (lua_State * L, size_t sz)`
- 4.13.3.40 `LUA_API int() lua_next (lua_State * L, int idx)`
- 4.13.3.41 `LUA_API int() lua_pcallk (lua_State * L, int nargs, int nresults, int errfunc, lua_KContext ctx, lua_KFunction k)`
- 4.13.3.42 `LUA_API void() lua_pushboolean (lua_State * L, int b)`

- 4.13.3.43 `LUA_API void() lua_pushcclosure (lua_State * L, lua_CFunction fn, int n)`
- 4.13.3.44 `LUA_API const char*() lua_pushfstring (lua_State * L, const char * fmt, ...)`
- 4.13.3.45 `LUA_API void() lua_pushinteger (lua_State * L, lua_Integer n)`
- 4.13.3.46 `LUA_API void() lua_pushlightuserdata (lua_State * L, void * p)`
- 4.13.3.47 `LUA_API const char*() lua_pushlstring (lua_State * L, const char * s, size_t len)`
- 4.13.3.48 `LUA_API void() lua_pushnil (lua_State * L)`
- 4.13.3.49 `LUA_API void() lua_pushnumber (lua_State * L, lua_Number n)`
- 4.13.3.50 `LUA_API const char*() lua_pushstring (lua_State * L, const char * s)`
- 4.13.3.51 `LUA_API int() lua_pushthread (lua_State * L)`
- 4.13.3.52 `LUA_API void() lua_pushvalue (lua_State * L, int idx)`
- 4.13.3.53 `LUA_API const char*() lua_pushvfstring (lua_State * L, const char * fmt, va_list argp)`
- 4.13.3.54 `LUA_API int() lua_rawequal (lua_State * L, int idx1, int idx2)`
- 4.13.3.55 `LUA_API int() lua_rawget (lua_State * L, int idx)`
- 4.13.3.56 `LUA_API int() lua_rawgeti (lua_State * L, int idx, lua_Integer n)`
- 4.13.3.57 `LUA_API int() lua_rawgetp (lua_State * L, int idx, const void * p)`
- 4.13.3.58 `LUA_API size_t() lua_rawlen (lua_State * L, int idx)`
- 4.13.3.59 `LUA_API void() lua_rawset (lua_State * L, int idx)`
- 4.13.3.60 `LUA_API void() lua_rawseti (lua_State * L, int idx, lua_Integer n)`
- 4.13.3.61 `LUA_API void() lua_rawsetp (lua_State * L, int idx, const void * p)`
- 4.13.3.62 `LUA_API int() lua_resume (lua_State * L, lua_State * from, int narg)`
- 4.13.3.63 `LUA_API void() lua_rotate (lua_State * L, int idx, int n)`
- 4.13.3.64 `LUA_API void() lua_setallocf (lua_State * L, lua_Alloc f, void * ud)`
- 4.13.3.65 `LUA_API void() lua_setfield (lua_State * L, int idx, const char * k)`
- 4.13.3.66 `LUA_API void() lua_setglobal (lua_State * L, const char * name)`
- 4.13.3.67 `LUA_API void() lua_sethook (lua_State * L, lua_Hook func, int mask, int count)`
- 4.13.3.68 `LUA_API void() lua_seti (lua_State * L, int idx, lua_Integer n)`
- 4.13.3.69 `LUA_API const char*() lua_setlocal (lua_State * L, const lua_Debug * ar, int n)`
- 4.13.3.70 `LUA_API int() lua_setmetatable (lua_State * L, int objindex)`

- 4.13.3.71 **LUA_API** void() lua_settable (lua_State * L, int idx)
- 4.13.3.72 **LUA_API** void() lua_settop (lua_State * L, int idx)
- 4.13.3.73 **LUA_API** const char*() lua_setupvalue (lua_State * L, int funcindex, int n)
- 4.13.3.74 **LUA_API** void() lua_setuservalue (lua_State * L, int idx)
- 4.13.3.75 **LUA_API** int() lua_status (lua_State * L)
- 4.13.3.76 **LUA_API** size_t() lua_stringtonumber (lua_State * L, const char * s)
- 4.13.3.77 **LUA_API** int() lua_toboolean (lua_State * L, int idx)
- 4.13.3.78 **LUA_API** lua_CFunction() lua_tocfunction (lua_State * L, int idx)
- 4.13.3.79 **LUA_API** lua_Integer() lua_tointegerx (lua_State * L, int idx, int * isnum)
- 4.13.3.80 **LUA_API** const char*() lua_tolstring (lua_State * L, int idx, size_t * len)
- 4.13.3.81 **LUA_API** lua_Number() lua_tonumberx (lua_State * L, int idx, int * isnum)
- 4.13.3.82 **LUA_API** const void*() lua_topointer (lua_State * L, int idx)
- 4.13.3.83 **LUA_API** lua_State*() lua_tothread (lua_State * L, int idx)
- 4.13.3.84 **LUA_API** void*() lua_touserdata (lua_State * L, int idx)
- 4.13.3.85 **LUA_API** int() lua_type (lua_State * L, int idx)
- 4.13.3.86 **LUA_API** const char*() lua_typename (lua_State * L, int tp)
- 4.13.3.87 **LUA_API** void*() lua_upvalueid (lua_State * L, int fidx, int n)
- 4.13.3.88 **LUA_API** void() lua_upvaluejoin (lua_State * L, int fidx1, int n1, int fidx2, int n2)
- 4.13.3.89 **LUA_API** const lua_Number*() lua_version (lua_State * L)
- 4.13.3.90 **LUA_API** void() lua_xmove (lua_State * from, lua_State * to, int n)
- 4.13.3.91 **LUA_API** int() lua_yieldk (lua_State * L, int nresults, lua_KContext ctx, lua_KFunction k)

4.13.4 Variable Documentation

- 4.13.4.1 const char lua_ident[]

4.14 wsh/include/luacnf.h File Reference

```
#include <limits.h>
#include <stddef.h>
```

Macros

- #define [LUA_BITSINT](#) 16

- #define `LUA_INT_INT` 1
- #define `LUA_INT_LONG` 2
- #define `LUA_INT_LONGLONG` 3
- #define `LUA_FLOAT_FLOAT` 1
- #define `LUA_FLOAT_DOUBLE` 2
- #define `LUA_FLOAT_LONGDOUBLE` 3
- #define `LUA_INT_TYPE` `LUA_INT_LONGLONG`
- #define `LUA_FLOAT_TYPE` `LUA_FLOAT_DOUBLE`
- #define `LUA_VDIR` `LUA_VERSION_MAJOR` "." `LUA_VERSION_MINOR`
- #define `LUA_ROOT` `"/usr/local/"`
- #define `LUA_LDIR` `LUA_ROOT` `"share/lua/"` `LUA_VDIR` `"/"`
- #define `LUA_CDIR` `LUA_ROOT` `"lib/lua/"` `LUA_VDIR` `"/"`
- #define `LUA_PATH_DEFAULT`
- #define `LUA_CPATH_DEFAULT` `LUA_CDIR` `"?.so;"` `LUA_CDIR` `"loadall.so;"` `"/?.so"`
- #define `LUA_DIRSEP` `"/"`
- #define `LUA_API` `extern`
- #define `LUALIB_API` `LUA_API`
- #define `LUAMOD_API` `LUALIB_API`
- #define `LUAI_FUNC` `extern`
- #define `LUAI_DDEC` `LUAI_FUNC`
- #define `LUAI_DDEF` `/* empty */`
- #define `l_floor(x)` `(l_mathop(floor))(x)`
- #define `lua_number2str(s, sz, n)` `l_sprintf((s), sz, LUA_NUMBER_FMT, (n))`
- #define `lua_numbertointeger(n, p)`
- #define `LUA_NUMBER` `double`
- #define `l_mathlim(n)` `(DBL_##n)`
- #define `LUAI_UACNUMBER` `double`
- #define `LUA_NUMBER_FRMLEN` `""`
- #define `LUA_NUMBER_FMT` `"%.14g"`
- #define `l_mathop(op)` `op`
- #define `lua_str2number(s, p)` `strtod((s), (p))`
- #define `LUA_INTEGER_FMT` `"%"` `LUA_INTEGER_FRMLEN` `"d"`
- #define `lua_integer2str(s, sz, n)` `l_sprintf((s), sz, LUA_INTEGER_FMT, (n))`
- #define `LUAI_UACINT` `LUA_INTEGER`
- #define `LUA_UNSIGNED` `unsigned` `LUAI_UACINT`
- #define `l_sprintf(s, sz, f, i)` `snprintf(s, sz, f, i)`
- #define `lua_strx2number(s, p)` `lua_str2number(s, p)`
- #define `lua_number2strx(L, b, sz, f, n)` `l_sprintf(b, sz, f, n)`
- #define `LUA_KCONTEXT` `ptrdiff_t`
- #define `lua_getlocaledecpoint()` `(localeconv()->decimal_point[0])`
- #define `LUAI_MAXSTACK` `15000`
- #define `LUA_EXTRSPACE` `(sizeof(void*))`
- #define `LUA_IDSIZE` `60`
- #define `LUAL_BUFFERSIZE` `8192`
- #define `LUA_QL(x)` `"" x ""`
- #define `LUA_QS` `LUA_QL("%s")`

4.14.1 Macro Definition Documentation

4.14.1.1 #define `l_floor(x)` `(l_mathop(floor))(x)`

Definition at line 422 of file `luaconf.h`.

4.14.1.2 `#define l_mathlim(n)(DBL_##n)`

Definition at line 477 of file luacnf.h.

4.14.1.3 `#define l_mathop(op) op`

Definition at line 484 of file luacnf.h.

4.14.1.4 `#define l_sprintf(s, sz, f, i) snprintf(s,sz,f,i)`

Definition at line 591 of file luacnf.h.

4.14.1.5 `#define LUA_API extern`

Definition at line 242 of file luacnf.h.

4.14.1.6 `#define LUA_CDIR LUA_ROOT "lib/luac/" LUA_VDIR "/"`

Definition at line 193 of file luacnf.h.

4.14.1.7 `#define LUA_CPATH_DEFAULT LUA_CDIR"?so;" LUA_CDIR"loadall.so;" "./?.so"`

Definition at line 198 of file luacnf.h.

4.14.1.8 `#define LUA_DIRSEP "/"`

Definition at line 211 of file luacnf.h.

4.14.1.9 `#define LUA_EXTRASPACE (sizeof(void *))`

Definition at line 717 of file luacnf.h.

4.14.1.10 `#define LUA_FLOAT_DOUBLE 2`

Definition at line 115 of file luacnf.h.

4.14.1.11 `#define LUA_FLOAT_FLOAT 1`

Definition at line 114 of file luacnf.h.

4.14.1.12 `#define LUA_FLOAT_LONGDOUBLE 3`

Definition at line 116 of file luacnf.h.

4.14.1.13 `#define LUA_FLOAT_TYPE LUA_FLOAT_DOUBLE`

Definition at line 147 of file luacnf.h.

4.14.1.14 `#define lua_getlocaledecpoint() (localeconv()->decimal_point[0])`

Definition at line 657 of file luaconf.h.

4.14.1.15 `#define LUA_IDSIZE 60`

Definition at line 725 of file luaconf.h.

4.14.1.16 `#define LUA_INT_INT 1`

Definition at line 109 of file luaconf.h.

4.14.1.17 `#define LUA_INT_LONG 2`

Definition at line 110 of file luaconf.h.

4.14.1.18 `#define LUA_INT_LONGLONG 3`

Definition at line 111 of file luaconf.h.

4.14.1.19 `#define LUA_INT_TYPE LUA_INT_LONGLONG`

Definition at line 143 of file luaconf.h.

4.14.1.20 `#define lua_integer2str(s, sz, n) l_sprintf((s), sz, LUA_INTEGER_FMT, (n))`

Definition at line 514 of file luaconf.h.

4.14.1.21 `#define LUA_INTEGER_FMT "%" LUA_INTEGER_FRMLEN "d"`

Definition at line 513 of file luaconf.h.

4.14.1.22 `#define LUA_KCONTEXT ptrdiff_t`

Definition at line 639 of file luaconf.h.

4.14.1.23 `#define LUA_LDIR LUA_ROOT "share/lua/" LUA_VDIR "/"`

Definition at line 192 of file luaconf.h.

4.14.1.24 `#define LUA_NUMBER double`

Definition at line 475 of file luaconf.h.

4.14.1.25 `#define lua_number2str(s, sz, n) l_sprintf((s), sz, LUA_NUMBER_FMT, (n))`

Definition at line 424 of file luaconf.h.

4.14.1.26 `#define lua_number2strx(L, b, sz, f, n) l_sprintf(b,sz,f,n)`

Definition at line 615 of file luacnf.h.

4.14.1.27 `#define LUA_NUMBER_FMT "%.14g"`

Definition at line 482 of file luacnf.h.

4.14.1.28 `#define LUA_NUMBER_FRMLEN ""`

Definition at line 481 of file luacnf.h.

4.14.1.29 `#define lua_numbertointeger(n, p)`

Value:

```
((n) >= (LUA_NUMBER) (LUA_MININTEGER) && \
  (n) < -(LUA_NUMBER) (LUA_MININTEGER) && \
  (* (p) = (LUA_INTEGER) (n), 1))
```

Definition at line 434 of file luacnf.h.

4.14.1.30 `#define LUA_PATH_DEFAULT`

Value:

```
LUA_LDIR"?.lua;" LUA_LDIR"?/init.lua;" \
  LUA_CDIR"?.lua;" LUA_CDIR"?/init.lua;" \
  "./?.lua;" "./?/init.lua"
```

Definition at line 194 of file luacnf.h.

4.14.1.31 `#define LUA_QL(x) "" x ""`

Definition at line 749 of file luacnf.h.

4.14.1.32 `#define LUA_QS LUA_QL("%s")`

Definition at line 750 of file luacnf.h.

4.14.1.33 `#define LUA_ROOT "/usr/local/"`

Definition at line 191 of file luacnf.h.

4.14.1.34 `#define lua_str2number(s, p) strtod((s), (p))`

Definition at line 486 of file luacnf.h.

4.14.1.35 `#define lua_strx2number(s, p) lua_str2number(s,p)`

Definition at line 604 of file luacnf.h.

4.14.1.36 **#define LUA_UNSIGNED unsigned LUAI_UACINT**

Definition at line 522 of file luaconf.h.

4.14.1.37 **#define LUA_VDIR LUA_VERSION_MAJOR "." LUA_VERSION_MINOR**

Definition at line 170 of file luaconf.h.

4.14.1.38 **#define LUAI_BITSINT 16**

Definition at line 94 of file luaconf.h.

4.14.1.39 **#define LUAI_DDEC LUAI_FUNC**

Definition at line 273 of file luaconf.h.

4.14.1.40 **#define LUAI_DDEF /* empty */**

Definition at line 274 of file luaconf.h.

4.14.1.41 **#define LUAI_FUNC extern**

Definition at line 270 of file luaconf.h.

4.14.1.42 **#define LUAI_MAXSTACK 15000**

Definition at line 708 of file luaconf.h.

4.14.1.43 **#define LUAI_UACINT LUA_INTEGER**

Definition at line 516 of file luaconf.h.

4.14.1.44 **#define LUAI_UACNUMBER double**

Definition at line 479 of file luaconf.h.

4.14.1.45 **#define LUAL_BUFFERSIZE 8192**

Definition at line 736 of file luaconf.h.

4.14.1.46 **#define LUALIB_API LUA_API**

Definition at line 248 of file luaconf.h.

4.14.1.47 **#define LUAMOD_API LUALIB_API**

Definition at line 249 of file luaconf.h.

4.15 wsh/include/lualib.h File Reference

```
#include "lua.h"
```

Macros

- #define [LUA_COLIBNAME](#) "coroutine"
- #define [LUA_TABLIBNAME](#) "table"
- #define [LUA_IOLIBNAME](#) "io"
- #define [LUA_OSLIBNAME](#) "os"
- #define [LUA_STRLIBNAME](#) "string"
- #define [LUA_UTF8LIBNAME](#) "utf8"
- #define [LUA_BITLIBNAME](#) "bit32"
- #define [LUA_MATHLIBNAME](#) "math"
- #define [LUA_DBLIBNAME](#) "debug"
- #define [LUA_LOADLIBNAME](#) "package"
- #define [lua_assert\(x\)](#) ((void)0)

Functions

- [LUAMOD_API int\(\)](#) [luaopen_base](#) ([lua_State *L](#))
- [LUAMOD_API int\(\)](#) [luaopen_coroutine](#) ([lua_State *L](#))
- [LUAMOD_API int\(\)](#) [luaopen_table](#) ([lua_State *L](#))
- [LUAMOD_API int\(\)](#) [luaopen_io](#) ([lua_State *L](#))
- [LUAMOD_API int\(\)](#) [luaopen_os](#) ([lua_State *L](#))
- [LUAMOD_API int\(\)](#) [luaopen_string](#) ([lua_State *L](#))
- [LUAMOD_API int\(\)](#) [luaopen_utf8](#) ([lua_State *L](#))
- [LUAMOD_API int\(\)](#) [luaopen_bit32](#) ([lua_State *L](#))
- [LUAMOD_API int\(\)](#) [luaopen_math](#) ([lua_State *L](#))
- [LUAMOD_API int\(\)](#) [luaopen_debug](#) ([lua_State *L](#))
- [LUAMOD_API int\(\)](#) [luaopen_package](#) ([lua_State *L](#))
- [LUALIB_API void\(\)](#) [luaL_openlibs](#) ([lua_State *L](#))

4.15.1 Macro Definition Documentation

4.15.1.1 #define [lua_assert\(x \)](#) ((void)0)

Definition at line 54 of file lualib.h.

4.15.1.2 #define [LUA_BITLIBNAME](#) "bit32"

Definition at line 35 of file lualib.h.

4.15.1.3 #define [LUA_COLIBNAME](#) "coroutine"

Definition at line 17 of file lualib.h.

4.15.1.4 #define [LUA_DBLIBNAME](#) "debug"

Definition at line 41 of file lualib.h.

4.15.1.5 `#define LUA_IOLIBNAME "io"`

Definition at line 23 of file lualib.h.

4.15.1.6 `#define LUA_LOADLIBNAME "package"`

Definition at line 44 of file lualib.h.

4.15.1.7 `#define LUA_MATHLIBNAME "math"`

Definition at line 38 of file lualib.h.

4.15.1.8 `#define LUA_OSLIBNAME "os"`

Definition at line 26 of file lualib.h.

4.15.1.9 `#define LUA_STRLIBNAME "string"`

Definition at line 29 of file lualib.h.

4.15.1.10 `#define LUA_TABLIBNAME "table"`

Definition at line 20 of file lualib.h.

4.15.1.11 `#define LUA_UTF8LIBNAME "utf8"`

Definition at line 32 of file lualib.h.

4.15.2 Function Documentation

4.15.2.1 `LUALIB_API void() luaL_openlibs (lua_State * L)`

4.15.2.2 `LUAMOD_API int() luaopen_base (lua_State * L)`

4.15.2.3 `LUAMOD_API int() luaopen_bit32 (lua_State * L)`

4.15.2.4 `LUAMOD_API int() luaopen_coroutine (lua_State * L)`

4.15.2.5 `LUAMOD_API int() luaopen_debug (lua_State * L)`

4.15.2.6 `LUAMOD_API int() luaopen_io (lua_State * L)`

4.15.2.7 `LUAMOD_API int() luaopen_math (lua_State * L)`

4.15.2.8 `LUAMOD_API int() luaopen_os (lua_State * L)`

4.15.2.9 `LUAMOD_API int() luaopen_package (lua_State * L)`

4.15.2.10 `LUAMOD_API int() luaopen_string (lua_State * L)`

4.15.2.11 `LUAMOD_API int() luaopen_table (lua_State * L)`

4.15.2.12 LUAMOD_API int() luaopen_utf8 (lua_State * L)

4.16 wsh/wsh.c File Reference

```
#include <libwitch/wsh.h>
#include <libwitch/wsh_functions.h>
#include <libwitch/sigs.h>
#include <uthash.h>
```

Data Structures

- struct [help_t](#)
- struct [learn_key_t](#)
- struct [learn_t](#)

Macros

- #define [REG_RIP](#) 16
- #define [Elf_Ehdr](#) Elf32_Ehdr
- #define [Elf_Shdr](#) Elf32_Shdr
- #define [Elf_Sym](#) Elf32_Sym
- #define [Elf_Addr](#) Elf32_Addr
- #define [Elf_Sword](#) Elf64_Sword
- #define [Elf_Section](#) Elf32_Half
- #define [ELF_ST_BIND](#) ELF32_ST_BIND
- #define [ELF_ST_TYPE](#) ELF32_ST_TYPE
- #define [Elf_Rel](#) Elf32_Rel
- #define [Elf_Rela](#) Elf32_Rela
- #define [ELF_R_SYM](#) ELF32_R_SYM
- #define [ELF_R_TYPE](#) ELF32_R_TYPE
- #define [ELF_R_INFO](#) ELF32_R_INFO
- #define [Elf_Phdr](#) Elf32_Phdr
- #define [Elf_Xword](#) Elf32_Xword
- #define [Elf_Word](#) Elf32_Word
- #define [Elf_Off](#) Elf32_Off
- #define [ELFCLASS](#) ELFCLASS32
- #define [ELFMACHINE](#) EM_386
- #define [CS_MODE](#) CS_MODE_32
- #define [RELOC_MODE](#) RELOC_X86_32

Typedefs

- typedef struct [help_t](#) [help_t](#)
- typedef struct [learn_key_t](#) [learn_key_t](#)
- typedef struct [learn_t](#) [learn_t](#)

Functions

- int [bfmap](#) (lua_State *L)
- int [ptoh](#) (int perms, char hperms[])
- void [info_function](#) (void *addr)
- void [fatal_error](#) (lua_State *L, char *msg)
- void [script](#) (char *path)
- void [hexdump](#) (uint8_t *data, size_t size, size_t colorstart, size_t color_len)
- char * [symbol_tobind](#) (int n)
- char * [symbol_totype](#) (int n)
- unsigned int [ltrace](#) (void)
- int [scan_symbol](#) (char *symbol, char *libname)
- void [completion](#) (const char *buf, [linenoiseCompletions](#) *lc)
- int [disable_aslr](#) (void)
- int [enable_aslr](#) (void)
- int [detailed_help](#) (char *name)
- int [help](#) (lua_State *L)
- char * [decode_flags](#) (unsigned int flags)
- char * [decode_type](#) (unsigned int type)
- int [phdr_callback](#) (struct dl_phdr_info *info, size_t size, void *data)
- int [add_symbol](#) (char *symbol, char *libname, char *htype, char *hbind, unsigned long value, unsigned int size, unsigned long int addr)
- void [section_add](#) (unsigned long int addr, unsigned long int size, char *libname, char *name, char *perms, int flags)
- void [segment_add](#) (unsigned long int addr, unsigned long int size, char *perms, char *fname, char *ptype, int flags)
- void [entry_point_add](#) (unsigned long int addr, char *fname)
- void [scan_section](#) (Elf_Shdr *shdr, char *strTab, int shnum, char *fname, unsigned long int baseaddr)
- int [scan_sections](#) (char *fname, unsigned long int baseaddr)
- int [shdr_callback](#) (struct dl_phdr_info *info, size_t size, void *data)
- int [phdrs](#) (lua_State *L)
- [sections_t](#) * [section_from_addr](#) (unsigned long int addr)
- [segments_t](#) * [segment_from_addr](#) (unsigned long int addr)
- [sections_t](#) * [symbol_from_addr](#) (unsigned long int addr)
- [sections_t](#) * [symbol_from_name](#) (char *fname)
- int [headers](#) (lua_State *L)
- int [empty_symbols](#) (void)
- int [empty_phdrs](#) (void)
- int [empty_shdrs](#) (void)
- int [empty_eps](#) (void)
- int [print_phdrs](#) (void)
- int [print_symbols](#) (lua_State *L)
- int [print_functions](#) (lua_State *L)
- int [print_objects](#) (lua_State *L)
- int [print_libs](#) (lua_State *L)
- int [print_shdrs](#) (void)
- int [print_eps](#) (void)
- int [shdr_cmp](#) ([sections_t](#) *a, [sections_t](#) *b)
- int [phdr_cmp](#) ([segments_t](#) *a, [segments_t](#) *b)
- int [reload_elfs](#) (void)
- int [shdrs](#) (lua_State *L)
- int [entrypoints](#) (lua_State *L)
- int [man](#) (lua_State *L)
- int [info](#) (lua_State *L)
- int [alloccharbuf](#) (lua_State *L)

- int [setcharbuf](#) (lua_State *L)
- int [rdstr](#) (lua_State *L)
- int [rdnum](#) (lua_State *L)
- int [getcharbuf](#) (lua_State *L)
- int [run_shell](#) (lua_State *L)
- int [learn_proto](#) (unsigned long *arg, unsigned long int faultaddr, int reason)
- int [sort_learnt](#) (learn_t *a, learn_t *b)
- int [prototypes](#) (lua_State *L)
- int [libcall](#) (lua_State *L)
- void [scan_syms](#) (char *dynstr, Elf_Sym *sym, unsigned long int sz, char *libname)
- void [parse_dyn](#) (struct link_map *map)
- void [parse_link_map_dyn](#) (struct link_map *map)
- void [rescan](#) (void)
- int [print_procmmap](#) (unsigned int pid)
- int [procmmap_lua](#) (void)
- int [execlib](#) (lua_State *L)
- int [traceback](#) (lua_State *L)
- void [print_backtrace](#) (void)
- char * [sicode_toname](#) (int code)
- char * [signaltoname](#) (int signal)
- void [unset_align_flag](#) (void)
- void [set_align_flag](#) (void)
- void [unset_trace_flag](#) (void)
- void [set_trace_flag](#) (void)
- void [affinity](#) (int procnum)
- void [btr_enable](#) (int procnum)
- void [btr_disable](#) (int procnum)
- void [set_branch_flag](#) (void)
- void [unset_branch_flag](#) (void)
- void [bushandler](#) (int signal, siginfo_t *s, void *ptr)
- void [alarmhandler](#) (int signal, siginfo_t *s, void *u)
- void [inthandler](#) (int signal, siginfo_t *s, void *u)
- int [mk_backtrace](#) (void)
- void [restore_exit](#) (void)
- void [exit](#) (int status)
- void [_exit](#) (int status)
- void [exit_group](#) (int status)
- int [printarg](#) (unsigned long int val)
- void [traphandler](#) (int signal, siginfo_t *s, void *ptr)
- char * [sicode_strerror](#) (int signal, siginfo_t *s)
- void [sighandler](#) (int signal, siginfo_t *s, void *ptr)
- int [set_sighandlers](#) (void)
- int [test_stdin](#) (void)
- int [verbose](#) (lua_State *L)
- int [hollywood](#) (lua_State *L)
- int [map](#) (lua_State *L)
- int [bsspolute](#) (lua_State *L)
- int [ralloc](#) (lua_State *L)
- int [xalloc](#) (lua_State *L)
- void [xfree](#) (lua_State *L)
- void [traceunaligned](#) (lua_State *L)
- void [untraceunaligned](#) (lua_State *L)
- void [singlestep](#) (lua_State *L)
- void [unsinglestep](#) (lua_State *L)
- void [systrace](#) (lua_State *L)

- void `rtrace` (`lua_State *L`)
- void `unsystrace` (`lua_State *L`)
- void `unrtrace` (`lua_State *L`)
- void `verbostrace` (`lua_State *L`)
- void `unverbostrace` (`lua_State *L`)
- void `singlebranch` (`lua_State *L`)
- void `unsinglebranch` (`lua_State *L`)
- int `grep_ptr` (`lua_State *L`)
- int `loadbin` (`lua_State *L`)
- int `grep` (`lua_State *L`)
- int `priv_memcpy` (`lua_State *L`)
- int `priv_strcpy` (`lua_State *L`)
- int `priv_strcat` (`lua_State *L`)
- int `breakpoint` (`lua_State *L`)
- void `declare_func` (`void *addr, char *name`)
- void `declare_num` (`int val, char *name`)
- void `declare_internals` (`void`)
- struct `link_map * loadlibrary` (`char *libname`)
- int `set_alloc_opt` (`void`)
- int `gencore` (`lua_State *L`)
- int `disable_core` (`lua_State *L`)
- int `enable_core` (`lua_State *L`)
- int `wsh_init` (`void`)
- int `lua_strerror` (`int err`)
- int `run_script` (`char *name`)
- unsigned int `read_elf_sig` (`char *fname, struct stat *sb`)
- int `wsh_run` (`void`)
- int `add_script_arguments` (`int argc, char **argv, unsigned int i`)
- int `add_script_exec` (`char *name`)
- int `add_binary_preload` (`char *name`)
- int `do_loadlib` (`char *libname`)
- int `wsh_loadlibs` (`void`)
- int `wsh_getopt` (`wsh_t *wsh1, int argc, char **argv`)
- int `wsh_print_version` (`void`)
- int `wsh_usage` (`char *name`)
- int `rawmemread` (`lua_State *L`)
- int `rawmemwrite` (`lua_State *L`)
- int `rawmemstr` (`lua_State *L`)
- int `rawmemusage` (`lua_State *L`)
- int `rawmemaddr` (`lua_State *L`)
- int `rawmemstrlen` (`lua_State *L`)

Variables

- `wsh_t * wsh`
- `help_t cmdhelp []`
- `help_t fcnhelp []`
- `learn_t * protorecords = NULL`

4.16.1 Macro Definition Documentation

4.16.1.1 `#define CS_MODE CS_MODE_32`

Definition at line 88 of file `wsh.c`.

4.16.1.2 #define Elf_Addr Elf32_Addr

Definition at line 72 of file wsh.c.

4.16.1.3 #define Elf_Ehdr Elf32_Ehdr

Definition at line 69 of file wsh.c.

4.16.1.4 #define Elf_Off Elf32_Off

Definition at line 85 of file wsh.c.

4.16.1.5 #define Elf_Phdr Elf32_Phdr

Definition at line 82 of file wsh.c.

4.16.1.6 #define ELF_R_INFO ELF32_R_INFO

Definition at line 81 of file wsh.c.

4.16.1.7 #define ELF_R_SYM ELF32_R_SYM

Definition at line 79 of file wsh.c.

4.16.1.8 #define ELF_R_TYPE ELF32_R_TYPE

Definition at line 80 of file wsh.c.

4.16.1.9 #define Elf_Rel Elf32_Rel

Definition at line 77 of file wsh.c.

4.16.1.10 #define Elf_Rela Elf32_Rela

Definition at line 78 of file wsh.c.

4.16.1.11 #define Elf_Section Elf32_Half

Definition at line 74 of file wsh.c.

4.16.1.12 #define Elf_Shdr Elf32_Shdr

Definition at line 70 of file wsh.c.

4.16.1.13 #define ELF_ST_BIND ELF32_ST_BIND

Definition at line 75 of file wsh.c.

4.16.1.14 #define ELF_ST_TYPE ELF32_ST_TYPE

Definition at line 76 of file wsh.c.

4.16.1.15 #define Elf_Sword Elf64_Sword

Definition at line 73 of file wsh.c.

4.16.1.16 #define Elf_Sym Elf32_Sym

Definition at line 71 of file wsh.c.

4.16.1.17 #define Elf_Word Elf32_Word

Definition at line 84 of file wsh.c.

4.16.1.18 #define Elf_Xword Elf32_Xword

Definition at line 83 of file wsh.c.

4.16.1.19 #define ELFCLASS ELFCLASS32

Definition at line 86 of file wsh.c.

4.16.1.20 #define ELF_MACHINE EM_386

Definition at line 87 of file wsh.c.

4.16.1.21 #define REG_RIP 16

Witchcraft Compiler Collection

Author: Jonathan Brossard - endrazine@gmail.com

The MIT License (MIT) Copyright (c) 2016 Jonathan Brossard

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Definition at line 38 of file wsh.c.

4.16.1.22 #define RELOC_MODE RELOC_X86_32

Definition at line 89 of file wsh.c.

4.16.2 Typedef Documentation

4.16.2.1 typedef struct help_t help_t

4.16.2.2 typedef struct learn_key_t learn_key_t

4.16.2.3 typedef struct learn_t learn_t

4.16.3 Function Documentation

4.16.3.1 void _exit (int *status*)

Definition at line 3143 of file wsh.c.

4.16.3.2 int add_binary_preload (char * *name*)

Add a binary to the list of binaries to preload

Definition at line 4566 of file wsh.c.

4.16.3.3 int add_script_arguments (int *argc*, char ** *argv*, unsigned int *i*)

Definition at line 4530 of file wsh.c.

4.16.3.4 int add_script_exec (char * *name*)

Add a script to the execution queue

Definition at line 4552 of file wsh.c.

4.16.3.5 int add_symbol (char * *symbol*, char * *libname*, char * *htype*, char * *hbind*, unsigned long *value*, unsigned int *size*, unsigned long int *addr*)

Add a symbol to linked list

Definition at line 719 of file wsh.c.

4.16.3.6 void affinity (int *procnum*)

Set affinity of a thread to a given CPU

Definition at line 2947 of file wsh.c.

4.16.3.7 void alarmhandler (int *signal*, siginfo_t * *s*, void * *u*)

Definition at line 3083 of file wsh.c.

4.16.3.8 int alloccharbuf (lua_State * L)

Buffer management subroutines

Definition at line 1590 of file wsh.c.

4.16.3.9 int bfmap (lua_State * L)

Bruteforce valid memory mapping ranges

Definition at line 100 of file wsh.c.

4.16.3.10 int breakpoint (lua_State * L)

Set a breakpoint Make sure destination address is mapped

Change memory protections to RWX on destination's page

Backup byte at destination

Write Breakpoint

Save breakpoint informations

Definition at line 4218 of file wsh.c.

4.16.3.11 int bsspollute (lua_State * L)

Pollute .bss sections

Definition at line 3712 of file wsh.c.

4.16.3.12 void btr_disable (int procnum)

Disable Branch Tracing

Definition at line 2981 of file wsh.c.

4.16.3.13 void btr_enable (int procnum)

Enable Branch Tracing

Definition at line 2961 of file wsh.c.

4.16.3.14 void bushandler (int signal, siginfo_t * s, void * ptr)

SIGBUS handler

Definition at line 3031 of file wsh.c.

4.16.3.15 void completion (const char * buf, linenoiseCompletions * lc)

Shell autocompletion routine We want to add the next word upon 'tab' completion, exposing all the internally available keywords dynamically

Definition at line 377 of file wsh.c.

4.16.3.16 void declare_func (void * *addr*, char * *name*)

Definition at line 4269 of file wsh.c.

4.16.3.17 void declare_internals (void)

Export functions to lua Create definitions for internal functions

Create a wrapper functions for other internal functions

Definition at line 4282 of file wsh.c.

4.16.3.18 void declare_num (int *val*, char * *name*)

Definition at line 4274 of file wsh.c.

4.16.3.19 char* decode_flags (unsigned int *flags*)

Decode Segment flags

Definition at line 602 of file wsh.c.

4.16.3.20 char* decode_type (unsigned int *type*)

Decode Segment type

Definition at line 631 of file wsh.c.

4.16.3.21 int detailed_help (char * *name*)

Display detailed help Search command

Search function

Definition at line 541 of file wsh.c.

4.16.3.22 int disable_aslr (void)

Disable ASLR

Definition at line 455 of file wsh.c.

4.16.3.23 int disable_core (lua_State * *L*)

Disable core files generation

Definition at line 4351 of file wsh.c.

4.16.3.24 int do_loadlib (char * *libname*)

Do load a shared binary into the address space

Definition at line 4581 of file wsh.c.

4.16.3.25 int empty_eps (void)

Empty linked list of entry points

Definition at line 1036 of file wsh.c.

4.16.3.26 int empty_phdrs (void)

Empty linked list of segments

Definition at line 999 of file wsh.c.

4.16.3.27 int empty_shdrs (void)

Empty linked list of sections

Definition at line 1018 of file wsh.c.

4.16.3.28 int empty_symbols (void)

Empty linked list of symbols

Definition at line 980 of file wsh.c.

4.16.3.29 int enable_aslr (void)

Enable ASLR

Definition at line 473 of file wsh.c.

4.16.3.30 int enable_core (lua_State * L)

Enable core files generation

Definition at line 4359 of file wsh.c.

4.16.3.31 void entry_point_add (unsigned long int addr, char * fname)

Add an entry point to linked list

Definition at line 789 of file wsh.c.

4.16.3.32 int entrypoints (lua_State * L)

Display ELF Entry points

Definition at line 1469 of file wsh.c.

4.16.3.33 int execlib (lua_State * L)

Definition at line 2792 of file wsh.c.

4.16.3.34 void exit (int status)

Definition at line 3137 of file wsh.c.

4.16.3.35 void exit_group (int status)

Definition at line 3149 of file wsh.c.

4.16.3.36 void fatal_error (lua_State * L, char * msg)

Fatal error : print an error message and exit with error

Definition at line 157 of file wsh.c.

4.16.3.37 int gencore (lua_State * L)

Generate a core file

Definition at line 4340 of file wsh.c.

4.16.3.38 int getcharbuf (lua_State * L)

Definition at line 1657 of file wsh.c.

4.16.3.39 int grep (lua_State * L)

search a pattern over all sections mapped in memory

Definition at line 4069 of file wsh.c.

4.16.3.40 int grepptr (lua_State * L)

Search a given value in memory

grepptr(Pattern, patternlen, hexadumplen, nbytesbeforematch)

Definition at line 3979 of file wsh.c.

4.16.3.41 int headers (lua_State * L)

Generate headers generate headers for imported objects

generate forward prototypes for imported functions

Definition at line 931 of file wsh.c.

4.16.3.42 int help (lua_State * L)

Display help

Definition at line 574 of file wsh.c.

4.16.3.43 void hexdump (uint8_t * data, size_t size, size_t colorstart, size_t color_len)

Simple hexdump routine

Definition at line 184 of file wsh.c.

4.16.3.44 int hollywood (lua_State * L)

Definition at line 3632 of file wsh.c.

4.16.3.45 int info (lua_State * L)

Display information on an object/memory address Address is mapped

Search corresponding symbols

Search corresponding section

Search corresponding segment

Search corresponding symbols

Resolve symbol...

Definition at line 1495 of file wsh.c.

4.16.3.46 void info_function (void * addr)

Print information on a given function

Definition at line 147 of file wsh.c.

4.16.3.47 void inthandler (int signal, siginfo_t * s, void * u)

Definition at line 3094 of file wsh.c.

4.16.3.48 int learn_proto (unsigned long * arg, unsigned long int faultaddr, int reason)

Definition at line 1801 of file wsh.c.

4.16.3.49 int libcall (lua_State * L)

Main wrapper around a library call. This function returns 9 values: ret (returned by library call), errno, firstsignal, total number of signals, firstsicode, firsterrno, faultaddr, reason, context Handle (reverse-) system calls tracing

Make the library call

Analyse return value

Learn prototypes

Create output execution context table

Push errno to lua table

Push strerror(errno) to lua table

Push first signal

Push first signal name

Push total of signals emitted during this libcall

Push first errno

Push first sicode

Push first sicode name

Address of last caller in backtrace

Push fault address

Push reason

Push mode

Push errctx

Push pointer to ucontext

Push arguments as a new table

Push number of non NULL arguments

Push retval

Push libcall/libname

Invoke store running function on context

Definition at line 2087 of file wsh.c.

4.16.3.50 int loadbin (lua_State * L)

Load a binary into the address space

Definition at line 4054 of file wsh.c.

4.16.3.51 struct link_map* loadlibrary (char * libname)

Definition at line 4311 of file wsh.c.

4.16.3.52 unsigned int ltrace (void)

Definition at line 328 of file wsh.c.

4.16.3.53 int lua_strerror (int err)

Definition at line 4395 of file wsh.c.

4.16.3.54 int man (lua_State * L)

Open a manual page

Definition at line 1478 of file wsh.c.

4.16.3.55 int map (lua_State * L)

Display mapped sections

Definition at line 3658 of file wsh.c.

4.16.3.56 int mk_backtrace (void)

Definition at line 3110 of file wsh.c.

4.16.3.57 void parse_dyn (struct link_map * map)

Walk the array of ELF_Dyn once looking for critical sections

Definition at line 2625 of file wsh.c.

4.16.3.58 void parse_link_map_dyn (struct link_map * map)

Definition at line 2724 of file wsh.c.

4.16.3.59 int phdr_callback (struct dl_phdr_info * info, size_t size, void * data)

Callback function to parse Program headers (ELF Segments)

Definition at line 683 of file wsh.c.

4.16.3.60 int phdr_cmp (segments_t * a, segments_t * b)

Sort function helper for segments

Definition at line 1434 of file wsh.c.

4.16.3.61 int phdrs (lua_State * L)

Display Program headers (ELF Segments)

Definition at line 859 of file wsh.c.

4.16.3.62 void print_backtrace (void)

Definition at line 2847 of file wsh.c.

4.16.3.63 int print_eps (void)

Display Entry points

Definition at line 1409 of file wsh.c.

4.16.3.64 int print_functions (lua_State * L)

Display functions

Definition at line 1176 of file wsh.c.

4.16.3.65 int print_libs (lua_State * L)

Display mapped libraries, return a list of library names

Definition at line 1308 of file wsh.c.

4.16.3.66 int print_objects (lua_State * L)

Display objects (typically globals)

Definition at line 1255 of file wsh.c.

4.16.3.67 int print_phdrs (void)

Display program headers (ELF Segments)

Definition at line 1052 of file wsh.c.

4.16.3.68 int print_procmmap (unsigned int *pid*)

Display content of /proc/self/maps

Definition at line 2765 of file wsh.c.

4.16.3.69 int print_shdrs (void)

Display ELF sections

Definition at line 1344 of file wsh.c.

4.16.3.70 int print_symbols (lua_State * *L*)

Display symbols

Definition at line 1108 of file wsh.c.

4.16.3.71 int printarg (unsigned long int *val*)

Definition at line 3155 of file wsh.c.

4.16.3.72 int priv_memcpy (lua_State * *L*)

Our own version of memcpy callable from LUA

Definition at line 4154 of file wsh.c.

4.16.3.73 int priv_strcat (lua_State * *L*)

Our own version of strcat callable from LUA

Definition at line 4197 of file wsh.c.

4.16.3.74 int priv_strcpy (lua_State * *L*)

Our own version of strcpy callable from LUA

Definition at line 4176 of file wsh.c.

4.16.3.75 int procmmap_lua (void)

Definition at line 2787 of file wsh.c.

4.16.3.76 int prototypes (lua_State * *L*)

Display learned prototypes Read all the lines to learnt data structure

Sort learnt data structures

Definition at line 1885 of file wsh.c.

4.16.3.77 int ptoh (int perms, char hperms[])

Get permissions in human readable format

Definition at line 138 of file wsh.c.

4.16.3.78 int ralloc (lua_State * L)

ralloc(unsigned int size, unsigned char poison); allocate 1 page set to 0x00, set size bytes to poison, remap the page R only

Definition at line 3755 of file wsh.c.

4.16.3.79 int rawmemaddr (lua_State * L)

int addr rawmemaddr(obj)

Return the address in memory of the object passed as argument. Or returns an address itself if an address is given as argument.

Definition at line 4833 of file wsh.c.

4.16.3.80 int rawmemread (lua_State * L)

string res rawmemread(addr, len)

Read len bytes at address addr and return them as a lua string.

Definition at line 4759 of file wsh.c.

4.16.3.81 int rawmemstr (lua_State * L)

Returns a string, from an address passed as argument.

Definition at line 4797 of file wsh.c.

4.16.3.82 int rawmemstrlen (lua_State * L)

int rawmemstrlen(addr) Returns the length of a string passed as argument

Definition at line 4845 of file wsh.c.

4.16.3.83 int rawmemusage (lua_State * L)

Display memory usage.

Definition at line 4811 of file wsh.c.

4.16.3.84 int rawmemwrite (lua_State * L)

int written rawmemwrite(addr, data, len)

Raw write to addr of len bytes of data returns number of bytes written.

Definition at line 4778 of file wsh.c.

4.16.3.85 int rdnum (lua_State * L)

Read a number (to a LUA number)

Definition at line 1642 of file wsh.c.

4.16.3.86 int rdstr (lua_State * L)

Read a string (to a LUA string)

Definition at line 1621 of file wsh.c.

4.16.3.87 unsigned int read_elf_sig (char * fname, struct stat * sb)

Verify ELF signature in a binary

Definition at line 4452 of file wsh.c.

4.16.3.88 int reload_elfs (void)

Reload linked lists from ELF binaries

Definition at line 1441 of file wsh.c.

4.16.3.89 void rescan (void)

Rescan address space

Definition at line 2752 of file wsh.c.

4.16.3.90 void restore_exit (void)

generic function to restore from [exit\(\)](#)

Definition at line 3132 of file wsh.c.

4.16.3.91 void rtrace (lua_State * L)

Definition at line 3921 of file wsh.c.

4.16.3.92 int run_script (char * name)

Run a lua script

Definition at line 4418 of file wsh.c.

4.16.3.93 int run_shell (lua_State * L)

Run minimal LUA shell Set handlers for tab completion

Prepare history full log name

Load shell history

Main loop

Command analysis/execution

Definition at line 1689 of file wsh.c.

4.16.3.94 void scan_section (Elf_Shdr * shdr, char * strTab, int shnum, char * fname, unsigned long int baseaddr)

Parse a section from an ELF

Definition at line 803 of file wsh.c.

4.16.3.95 int scan_sections (char * fname, unsigned long int baseaddr)

Parse all sections from an ELF

Definition at line 821 of file wsh.c.

4.16.3.96 int scan_symbol (char * symbol, char * libname)

Scan a symbol, save it to linked list

Definition at line 338 of file wsh.c.

4.16.3.97 void scan_syms (char * dynstr, Elf_Sym * sym, unsigned long int sz, char * libname)

Walk symbol table

If function name is blacklisted, skip...

Add function/object to linked list

Add function/object to linked list

Definition at line 2507 of file wsh.c.

4.16.3.98 void script (char * path)

Run a script

Definition at line 166 of file wsh.c.

4.16.3.99 void section_add (unsigned long int addr, unsigned long int size, char * libname, char * name, char * perms, int flags)

Add a section to linked list

Definition at line 751 of file wsh.c.

4.16.3.100 sections_t* section_from_addr (unsigned long int addr)

Find section from address

Definition at line 869 of file wsh.c.

4.16.3.101 void segment_add (unsigned long int addr, unsigned long int size, char * perms, char * fname, char * ptype, int flags)

Add a segment to linked list

Definition at line 769 of file wsh.c.

4.16.3.102 `segments_t* segment_from_addr (unsigned long int addr)`

Find segment from address

Definition at line 884 of file wsh.c.

4.16.3.103 `void set_align_flag (void) [inline]`

Definition at line 2904 of file wsh.c.

4.16.3.104 `int set_alloc_opt (void)`

Definition at line 4331 of file wsh.c.

4.16.3.105 `void set_branch_flag (void) [inline]`

Definition at line 2999 of file wsh.c.

4.16.3.106 `int set_sighandlers (void)`

Set all signal handlers

Definition at line 3542 of file wsh.c.

4.16.3.107 `void set_trace_flag (void) [inline]`

Definition at line 2931 of file wsh.c.

4.16.3.108 `int setcharbuf (lua_State * L)`

Definition at line 1603 of file wsh.c.

4.16.3.109 `int shdr_callback (struct dl_phdr_info * info, size_t size, void * data)`

Callback function to parse Section headers (ELF Sections)

Definition at line 846 of file wsh.c.

4.16.3.110 `int shdr_cmp (sections_t * a, sections_t * b)`

Sort function helper for sections

Definition at line 1427 of file wsh.c.

4.16.3.111 `int shdrs (lua_State * L)`

Display section headers (ELF Sections)

Definition at line 1459 of file wsh.c.

4.16.3.112 `char* sicode_strerror (int signal, siginfo_t * s)`

Definition at line 3340 of file wsh.c.

4.16.3.113 `char* sicondtoname (int code)`

Definition at line 2872 of file wsh.c.

4.16.3.114 `void sighandler (int signal, siginfo_t * s, void * ptr)`

Get access type

Get signal name

Get signal code

Restore execution from known good point

Definition at line 3454 of file wsh.c.

4.16.3.115 `char* signaltoname (int signal)`

Definition at line 2878 of file wsh.c.

4.16.3.116 `void singlebranch (lua_State * L)`

Definition at line 3945 of file wsh.c.

4.16.3.117 `void singlestep (lua_State * L)`

Definition at line 3903 of file wsh.c.

4.16.3.118 `int sort_learnt (learn_t * a, learn_t * b)`

Definition at line 1878 of file wsh.c.

4.16.3.119 `sections_t* symbol_from_addr (unsigned long int addr)`

Return a symbol from an address

Definition at line 899 of file wsh.c.

4.16.3.120 `sections_t* symbol_from_name (char * fname)`

Return a symbol from its name

Definition at line 914 of file wsh.c.

4.16.3.121 `char* symbol_tobind (int n)`

Return symbol binding type in human readable format

Definition at line 279 of file wsh.c.

4.16.3.122 `char* symbol_totype (int n)`

Return symbol type in human readable format

Definition at line 303 of file wsh.c.

4.16.3.123 void systrace (lua_State * L)

Definition at line 3916 of file wsh.c.

4.16.3.124 int test_stdin (void)

Set global variable is_stdinscript to 1 if there is data on stdin

Definition at line 3599 of file wsh.c.

4.16.3.125 int traceback (lua_State * L)

Definition at line 2836 of file wsh.c.

4.16.3.126 void traceunaligned (lua_State * L)

Resize a xallocated memory zone

Definition at line 3891 of file wsh.c.

4.16.3.127 void traphandler (int *signal*, siginfo_t * *s*, void * *ptr*)

Search corresponding Breakpoint

This is a breakpoint

We are single branching

We are single stepping

We are tracing unaligned access via SIGBUS, single step once

This is an unhandled exception : exit

Definition at line 3175 of file wsh.c.

4.16.3.128 void unrtrace (lua_State * L)

Definition at line 3931 of file wsh.c.

4.16.3.129 void unset_align_flag (void) [inline]

Definition at line 2890 of file wsh.c.

4.16.3.130 void unset_branch_flag (void) [inline]

Definition at line 3022 of file wsh.c.

4.16.3.131 void unset_trace_flag (void) [inline]

Definition at line 2917 of file wsh.c.

4.16.3.132 void unsinglebranch (lua_State * L)

Definition at line 3967 of file wsh.c.

4.16.3.133 void unsinglestep (lua_State * L)

Definition at line 3909 of file wsh.c.

4.16.3.134 void unsystrace (lua_State * L)

Definition at line 3926 of file wsh.c.

4.16.3.135 void untraceunaligned (lua_State * L)

Definition at line 3897 of file wsh.c.

4.16.3.136 void unverbosetrace (lua_State * L)

Definition at line 3941 of file wsh.c.

4.16.3.137 int verbose (lua_State * L)

Definition at line 3618 of file wsh.c.

4.16.3.138 void verbosetrace (lua_State * L)

Definition at line 3937 of file wsh.c.

4.16.3.139 int wsh_getopt (wsh_t * wsh1, int argc, char ** argv)

Parse command line

Definition at line 4629 of file wsh.c.

4.16.3.140 int wsh_init (void)

Definition at line 4364 of file wsh.c.

4.16.3.141 int wsh_loadlibs (void)

Load all preload libraries

Definition at line 4608 of file wsh.c.

4.16.3.142 int wsh_print_version (void)

Print software version

Definition at line 4720 of file wsh.c.

4.16.3.143 int wsh_run (void)

Run a lua shell/script Run all the scripts specified in the command line

Run a lua shell

Definition at line 4475 of file wsh.c.

4.16.3.144 `int wsh_usage (char * name)`

Print usage

Definition at line 4729 of file wsh.c.

4.16.3.145 `int xalloc (lua_State * L)`

`xalloc(unsigned int size, unsigned char poison, unsigned int perms);` Allocate size bytes (% `getpagesize()`)

The mapping auto-references itself, unless a poison byte is given

[page unmapped] [mapped][OURPTR, size] [page unmapped]

Definition at line 3807 of file wsh.c.

4.16.3.146 `void xfree (lua_State * L)`

Release a bloc allocated via `xalloc()`

Definition at line 3868 of file wsh.c.

4.16.4 Variable Documentation

4.16.4.1 `help_t cmdhelp[]`

Initial value:

```
={
    {"quit", "", "Exit wsh.", "", "Does not return : exit wsh\n"},
    {"exit", "", "Exit wsh.", "", "Does not return : exit wsh\n"},
    {"shell", "[command]", "Run a /bin/sh shell.", "", "None. Returns upon shell termination."},
    {"exec", "<command>", "Run <command> via the system() library call.", "", "None. Returns upon <command> termination."},
    {"clear", "", "Clear terminal.", "", "None."},
}
```

Definition at line 497 of file wsh.c.

4.16.4.2 `help_t fcnhelp[]`

Initial value:

```
={
    {"help", "[topic]", "Display help on [topic]. If [topic] is omitted, display general help.", "", "None"},
    {"man", "[page]", "Display system manual page for [page].", "", "None"},
    {"hexdump", "<address>, <num>", "Display <num> bytes from memory <address> in enhanced hexadecimal form.", "", "None"},
    {"hex", "<object>", "Display lua <object> in enhanced hexadecimal form.", "", "None"},
    {"phdrs", "", "Display ELF program headers from all binaries loaded in address space.", "", "None"},
    {"shdrs", "", "Display ELF section headers from all binaries loaded in address space.", "", "None"},
    {"map", "", "Display a table of all the memory ranges mapped in memory in the address space.", "", "None"},
    {"procmmap", "", "Display a table of all the memory ranges mapped in memory in the address space as displayed in /proc/<pid>/maps.", "", "None"},
    {"bfmap", "", "Bruteforce valid mapped memory ranges in address space.", "", "None"},
    {"symbols", "[sympattern], [libpattern], [mode]", "Display all the symbols in memory matching [sympattern], from library [libpattern]. If [mode] is set to 1 or 2, do not wait user input between paggers. [mode] = 2 provides a shorter output.", "", "None"},
    {"functions", "[sympattern], [libpattern], [mode]", "Display all the functions in memory matching [sympattern], from library [libpattern]. If [mode] is set to 1 or 2, do not wait user input between paggers. [mode] = 2 provides a shorter output.", "table func = ", "Return 1 lua table _func_ whose keys are valid function names in address space, and values are pointers to them in memory."},
    {"objects", "[pattern]", "Display all the functions in memory matching [sympattern]", "", "None"},
}
```

```

{"info", "[address] | [name]", "Display various informations about the [address] or [name] provided
: if it is mapped, and if so from which library and in which section if available.", "", "None"},
{"search", "<pattern>", "Search all object names matching <pattern> in address space.", "", "None"}
,
{"headers", "", "Display C headers suitable for linking against the API loaded in address space.",
"", "None"},
{"grep", "<pattern>, [patternlen], [dumplen], [before]", "Search <pattern> in all ELF sections in
memory. Match [patternlen] bytes, then display [dumplen] bytes, optionally including [before] bytes before the
match. Results are displayed in enhanced decimal form", "table match = ", "Returns 1 lua table containing
matching memory addresses."},
{"grepptr", "<pattern>, [patternlen], [dumplen], [before]", "Search pointer <pattern> in all ELF
sections in memory. Match [patternlen] bytes, then display [dumplen] bytes, optionally including [before] bytes
before the match. Results are displayed in enhanced decimal form", "table match = ", "Returns 1 lua table
containing matching memory addresses."},
{"loadbin", "<pathname>", "Load binary to memory from <pathname>.", "", "None"},
{"libs", "", "Display all libraries loaded in address space.", "table libraries = ", "Returns 1
value: a lua table _libraries_ whose values contain valid binary names (executable/libraries) mapped in memory.
"},
{"entrypoints", "", "Display entry points for each binary loaded in address space.", "", "None"},
{"rescan", "", "Re-perform address space scan.", "", "None"},
{"libcall", "<function>, [arg1], [arg2], ... arg[6]", "Call binary <function> with provided
arguments.", "void *ret, table ctx = ", "Returns 2 return values: _ret_ is the return value of the binary function
(nill if none), _ctx_ a lua table representing the execution context of the library call.\n"},
{"enableaslr", "", "Enable Address Space Layout Randomization (requires root privileges).", "", "
None"},
{"disableaslr", "", "Disable Address Space Layout Randomization (requires root privileges).", "", "
None"},
{"verbose", "<verbosity>", "Change verbosity setting to <verbosity>.", "", "None"},
{"breakpoint", "<address>, [weight]", "Set a breakpoint at memory <address>. Optionally add a
<weight> to breakpoint score if hit.", "", "None"},
{"bp", "<address>, [weight]", "Set a breakpoint at memory <address>. Optionally add a <weight> to
breakpoint score if hit. Alias for breakpoint() function.", "", "None"},
{"hollywood", "<level>", "Change hollywood (fun) display setting to <level>, impacting color
display (enable/disable).", "", "None"},
}

```

Definition at line 506 of file wsh.c.

4.16.4.3 learn_t* protorecords = NULL

Definition at line 1876 of file wsh.c.

4.16.4.4 wsh_t* wsh

Main wsh context

Witchcraft Compiler Collection

Author: Jonathan Brossard - endrazine@gmail.com

The MIT License (MIT) Copyright (c) 2016 Jonathan Brossard

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. Main wsh context

Definition at line 37 of file wshmain.c.

4.17 wsh/wshmain.c File Reference

```
#include <libwitch/wsh.h>
```

Functions

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

Variables

- [wsh_t*](#) [wsh](#)

4.17.1 Function Documentation

4.17.1.1 int main (int argc, char ** argv, char ** envp)

Application entry point

Definition at line 42 of file wshmain.c.

4.17.2 Variable Documentation

4.17.2.1 wsh_t* wsh

Witchcraft Compiler Collection

Author: Jonathan Brossard - endrazine@gmail.com

The MIT License (MIT) Copyright (c) 2016 Jonathan Brossard

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. Main wsh context

Definition at line 37 of file wshmain.c.

Index

- `_FILE_OFFSET_BITS`
 - helper.c, 55
- `_GNU_SOURCE`
 - wcc.c, 40
 - wsh.h, 70
- `_XOPEN_SOURCE`
 - helper.c, 55
- `__USE_GNU`
 - wcc.c, 40
- `__progname_full`
 - wsh.h, 86
- `_exit`
 - wsh.c, 119
- abfd
 - ctx_t, 6
- add_binary_preload
 - wsh.c, 119
- add_extra_symbols
 - wcc.c, 44
- add_script_arguments
 - wsh.c, 119
- add_script_exec
 - wsh.c, 119
- add_symaddr
 - wcc.c, 44
- add_symbol
 - wsh.c, 119
 - wsh.h, 76
- addr
 - eps_t, 11
 - sections_t, 25
 - segments_t, 26
 - symaddr, 28
 - symbols_t, 28
 - tuple_t, 30
- adjust_baseaddress
 - wcc.c, 44
- affinity
 - wsh.c, 119
- alarmhandler
 - wsh.c, 119
- alignfromname
 - wcc.c, 44
- alloc_phdr
 - wcc.c, 44
- alloccharbuf
 - wsh.c, 119
 - wsh.h, 76
- allowed_sections
 - wcc.c, 51
- analyze_text
 - wcc.c, 44
- append_reloc
 - wcc.c, 44
- append_strtab
 - wcc.c, 44
- append_sym
 - wcc.c, 44
- archsz
 - ctx_t, 6
- b
 - luaL_Buffer, 17
- BIND_FLAGS
 - wsh.h, 70
- BLACK
 - colors.h, 56
- BLUE
 - colors.h, 56
- BROWN
 - colors.h, 56
- backup
 - breakpoint_t, 5
- base
 - elfdata_t, 10
- base_address
 - ctx_t, 6
- bfmap
 - wsh.c, 120
 - wsh.h, 76
- binname
 - ctx_t, 7
- blnames
 - wcc.c, 51
- bp_array
 - wsh_t, 31
- bp_num
 - wsh_t, 31
- bp_points
 - wsh_t, 31
- breakpoint
 - wsh.c, 120
 - wsh.h, 76
- breakpoint_t, 5
 - backup, 5
 - ptr, 5
 - weight, 5
 - wsh.h, 75
- bsspolute

- wsh.c, 120
 - wsh.h, 77
- btcaller
 - wsh_t, 31
- btr_disable
 - wsh.c, 120
- btr_enable
 - wsh.c, 120
- bushandler
 - wsh.c, 120
- CATCH
 - longjmp.h, 89
- CLEAR
 - colors.h, 56
- CS_MODE
 - wcc.c, 40
 - wsh.c, 116
- CYAN
 - colors.h, 56
- check_global_import
 - wcc.c, 44
- closef
 - luaL_Stream, 19
- cmdhelp
 - wsh.c, 135
- colors.h
 - BLACK, 56
 - BLUE, 56
 - BROWN, 56
 - CLEAR, 56
 - CYAN, 56
 - DARKGRAY, 56
 - GRAY, 56
 - GREEN, 56
 - MAGENTA, 57
 - NORMAL, 57
 - RED, 57
 - YELLOW, 57
- completion
 - wsh.c, 120
- copy_body
 - wcc.c, 45
- corefile
 - ctx_t, 7
- cplus_demangle
 - wsh.h, 77
- craft_section
 - wcc.c, 45
- create_phdrs
 - wcc.c, 45
- ctx_getopt
 - wcc.c, 45
- ctx_init
 - wcc.c, 45
- ctx_t, 5
 - abfd, 6
 - archsz, 6
 - base_address, 6
 - binname, 7
 - corefile, 7
 - fdout, 7
 - has_relativerelocations, 7
 - mphdrs, 7
 - mphnum, 7
 - mshdrs, 7
 - mshnum, 7
 - opt_arch, 7
 - opt_asmdebug, 7
 - opt_binname, 7
 - opt_core, 7
 - opt_debug, 8
 - opt_entrypoint, 8
 - opt_exec, 8
 - opt_flags, 8
 - opt_interp, 8
 - opt_original, 8
 - opt_poison, 8
 - opt_reloc, 8
 - opt_shared, 8
 - opt_sstrip, 8
 - opt_static, 8
 - opt_strip, 8
 - opt_verbose, 9
 - phnum, 9
 - shnum, 9
 - start_phdrs, 9
 - start_shdrs, 9
 - strndx, 9
 - strndx_index, 9
 - strndx_len, 9
 - wcc.c, 43
- currentline
 - lua_Debug, 16
- cvec
 - linenoiseCompletions, 15
- DARKGRAY
 - colors.h, 56
- DEFAULT_LEARN_FILE
 - wsh.h, 70
- DEFAULT_NAME
 - wld.c, 54
- DEFAULT_SCRIPT
 - wsh.h, 70
- DEFAULT_SCRIPT_INDEX
 - wsh.h, 70
- DEFAULT_STRNDX_SIZE
 - wcc.c, 41
- DMGL_ANSI
 - wsh.h, 70
- DMGL_ARM
 - wsh.h, 70
- DMGL_PARAMS
 - wsh.h, 70
- data
 - msec_t, 19
- datavma

- wcc.c, 51
- declare_func
 - wsh.c, 120
- declare_internals
 - wsh.c, 121
- declare_num
 - wsh.c, 121
- decode_flags
 - wsh.c, 121
- decode_type
 - wsh.c, 121
- default_options
 - wsh_functions.h, 86
- default_poison
 - wsh.h, 70
- deltastrtab
 - wcc.c, 51
- descr
 - help_t, 13
- desired_arch
 - wcc.c, 45
- detailed_help
 - wsh.c, 121
- disable_aslr
 - wsh.c, 121
 - wsh.h, 77
- disable_core
 - wsh.c, 121
 - wsh.h, 77
- do_loadlib
 - wsh.c, 121
 - wsh.h, 77
- dyn_index
 - elfdata_t, 10
- dyns
 - elfdata_t, 10
- ELF32_ST_BIND
 - wsh.h, 70
- ELF32_ST_INFO
 - wsh.h, 70
- ELF32_ST_TYPE
 - wsh.h, 70
- ELF64_ST_BIND
 - wsh.h, 70
- ELF64_ST_INFO
 - wsh.h, 71
- ELF64_ST_TYPE
 - wsh.h, 71
- ELF_R_INFO
 - wcc.c, 41
 - wsh.c, 117
- ELF_R_SYM
 - wcc.c, 41
 - wsh.c, 117
- ELF_R_TYPE
 - wcc.c, 41
 - wsh.c, 117
- ELF_ST_BIND
 - wcc.c, 42
 - wsh.c, 117
- ELF_ST_TYPE
 - wcc.c, 42
 - wsh.c, 117
- ELFCLASS
 - wcc.c, 42
 - wsh.c, 118
- ELFMACHINE
 - wcc.c, 42
 - wsh.c, 118
- ENTRY
 - longjmp.h, 89
- ehdr
 - elfdata_t, 10
- Elf_Addr
 - wcc.c, 41
 - wsh.c, 116
- Elf_Dyn
 - wsh.h, 71
- Elf_Ehdr
 - wcc.c, 41
 - wsh.c, 117
 - wsh.h, 71
- Elf_Off
 - wcc.c, 41
 - wsh.c, 117
- Elf_Phdr
 - wcc.c, 41
 - wsh.c, 117
 - wsh.h, 71
- Elf_Rel
 - wcc.c, 41
 - wsh.c, 117
- Elf_Rela
 - wcc.c, 41
 - wsh.c, 117
- Elf_Section
 - wcc.c, 41
 - wsh.c, 117
- Elf_Shdr
 - wcc.c, 41
 - wsh.c, 117
 - wsh.h, 71
- Elf_Sword
 - wcc.c, 42
 - wsh.c, 118
- Elf_Sym
 - wcc.c, 42
 - wsh.c, 118
 - wsh.h, 71
- Elf_Word
 - wcc.c, 42
 - wsh.c, 118
- Elf_Xword
 - wcc.c, 42
 - wsh.c, 118
- elfdata_t, 9

- base, 10
- dyn_index, 10
- dyns, 10
- ehdr, 10
- et_dyn, 10
- limit, 10
- link_map, 10
- p_pltgot, 10
- phdrs, 10
- r_debug, 10
- elis
 - wcc.c, 42
- empty_eps
 - wsh.c, 121
- empty_phdrs
 - wsh.c, 122
 - wsh.h, 77
- empty_shdrs
 - wsh.c, 122
 - wsh.h, 77
- empty_symbols
 - wsh.c, 122
- enable_aslr
 - wsh.c, 122
 - wsh.h, 77
- enable_core
 - wsh.c, 122
 - wsh.h, 77
- end
 - section, 24
- entry_point_add
 - wsh.c, 122
- entrypoints
 - wsh.c, 122
 - wsh.h, 78
- entszfromname
 - wcc.c, 45
- eps
 - wsh_t, 31
- eps_t, 11
 - addr, 11
 - name, 11
 - next, 11
 - prev, 11
 - wsh.h, 75
- errcontext
 - wsh_t, 31
- et_dyn
 - elfdata_t, 10
- event
 - lua_Debug, 16
- execlib
 - wsh.c, 122
 - wsh.h, 78
- exit
 - wsh.c, 122
- exit_group
 - wsh.c, 122
- exposed
 - wsh_functions.h, 86
- f
 - luaL_Stream, 19
- FAULT_EXEC
 - wsh.h, 71
- FAULT_READ
 - wsh.h, 71
- FAULT_WRITE
 - wsh.h, 71
- FINALLY
 - longjmp.h, 89
- FLAG_BSS
 - wcc.c, 42
- FLAG_NOBIT
 - wcc.c, 42
- FLAG_NOWRITE
 - wcc.c, 43
- FLAG_TEXT
 - wcc.c, 43
- fatal_error
 - wsh.c, 123
- faultaddr
 - wsh_t, 31
- fcnhelp
 - wsh.c, 135
- fdout
 - ctx_t, 7
- firstcontext
 - wsh_t, 32
- firsterrno
 - wsh_t, 32
- firstsicode
 - wsh_t, 32
- firstsignal
 - wsh_t, 32
- fixup_strtab_and_symtab
 - wcc.c, 45
- fixup_symtab_section_index
 - wcc.c, 45
- fixup_text
 - wcc.c, 45
- flags
 - msec_t, 19
 - sections_t, 25
 - segments_t, 26
- flags_from_name
 - wcc.c, 46
- func
 - luaL_Reg, 18
- GRAY
 - colors.h, 56
- GREEN
 - colors.h, 56
- gencore
 - wsh.c, 123
 - wsh.h, 78

- getcharbuf
 - wsh.c, 123
 - wsh.h, 78
- getsize
 - wsh.h, 78
- gimport_t, 11
 - r, 12
 - rtype, 12
 - sec, 12
 - sindex, 12
 - sname, 12
 - wcc.c, 43
- gimports
 - wcc.c, 51
- gimportslen
 - wcc.c, 51
- global_xalloc
 - wsh_functions.h, 86
- globalreloc
 - wcc.c, 51
- globalreloclen
 - wcc.c, 52
- globalrelocoffset
 - wcc.c, 52
- globalsignals
 - wsh_t, 32
- globalstrtab
 - wcc.c, 52
- globalstrtablen
 - wcc.c, 52
- globalstrtableoffset
 - wcc.c, 52
- globalsymindex
 - wcc.c, 52
- globalsymtab
 - wcc.c, 52
- globalsymtablen
 - wcc.c, 52
- globalsymtableoffset
 - wcc.c, 52
- grep
 - wsh.c, 123
 - wsh.h, 78
- grepptr
 - wsh.c, 123
 - wsh.h, 78
- HAS_ZFIRST
 - helper.c, 55
- HPERMSMAX
 - wsh.h, 71
- has_relativerelocations
 - ctx_t, 7
- hbind
 - symbols_t, 28
- headers
 - wsh.c, 123
 - wsh.h, 78
- help
 - wsh.c, 123
 - wsh.h, 78
- help_t, 12
 - descr, 13
 - name, 13
 - proto, 13
 - protoprefix, 13
 - retval, 13
 - wsh.c, 119
- helper.c
 - _FILE_OFFSET_BITS, 55
 - _XOPEN_SOURCE, 55
 - HAS_ZFIRST, 55
 - is_mapped, 55
 - lastsignal, 55
 - nsections, 55
 - read_maps, 55
 - zfirst, 55
- helper.h
 - is_mapped, 63
 - nsections, 63
 - read_maps, 63
 - zfirst, 63
- hexdump
 - wcc.c, 46
 - wsh.c, 123
 - wsh.h, 78
- hh
 - learn_t, 14
- hollywood
 - wsh.c, 123
 - wsh.h, 79
- hperms
 - section, 24
- htype
 - symbols_t, 29
- i_ci
 - lua_Debug, 16
- ifis
 - wcc.c, 43
- info
 - wsh.c, 124
 - wsh.h, 79
- info_from_name
 - wcc.c, 46
- info_function
 - wsh.c, 124
- init
 - section, 24
- initb
 - luaL_Buffer, 17
- internal_function_store
 - wcc.c, 46
- interrupted
 - wsh_t, 32
- inthandler
 - wsh.c, 124
- is_mapped

- helper.c, 55
- helper.h, 63
- is_stdinscript
 - wsh_t, 32
- istailcall
 - lua_Debug, 16
- isvararg
 - lua_Debug, 16
- key
 - learn_t, 14
- L
 - luaL_Buffer, 17
 - wsh_t, 32
- l_floor
 - luaconf.h, 106
- l_mathlim
 - luaconf.h, 106
- l_mathop
 - luaconf.h, 107
- l_sprintf
 - luaconf.h, 107
- LINES_MAX
 - wsh.h, 71
- LUA_API
 - luaconf.h, 107
- LUA_AUTHORS
 - lua.h, 93
- LUA_BITLIBNAME
 - lua.h, 111
- LUA_CDIRENTRY
 - luaconf.h, 107
- LUA_COLIBNAME
 - lua.h, 111
- LUA_COPYRIGHT
 - lua.h, 93
- LUA_CPATH_DEFAULT
 - luaconf.h, 107
- LUA_DBLIBNAME
 - lua.h, 111
- LUA_DIRSEP
 - luaconf.h, 107
- LUA_ERRERR
 - lua.h, 93
- LUA_ERRFILE
 - lua.h, 93
- LUA_ERRGCMM
 - lua.h, 93
- LUA_ERRMEM
 - lua.h, 94
- LUA_ERRRUN
 - lua.h, 94
- LUA_ERRSYNTAX
 - lua.h, 94
- LUA_EXTRASPACE
 - luaconf.h, 107
- LUA_FILEHANDLE
 - lua.h, 96
- LUA_FLOAT_DOUBLE
 - luaconf.h, 107
- LUA_FLOAT_FLOAT
 - luaconf.h, 107
- LUA_FLOAT_TYPE
 - luaconf.h, 107
- LUA_GCCOLLECT
 - lua.h, 94
- LUA_GCCOUNT
 - lua.h, 94
- LUA_GCCOUNTB
 - lua.h, 94
- LUA_GCISRUNNING
 - lua.h, 94
- LUA_GCRESTART
 - lua.h, 94
- LUA_GCSETPAUSE
 - lua.h, 94
- LUA_GCSETSTEPMUL
 - lua.h, 94
- LUA_GCSTEP
 - lua.h, 94
- LUA_GCSTOP
 - lua.h, 94
- LUA_HOOKCALL
 - lua.h, 95
- LUA_HOOKCOUNT
 - lua.h, 95
- LUA_HOOKLINE
 - lua.h, 95
- LUA_HOOKRET
 - lua.h, 95
- LUA_HOOKTAILCALL
 - lua.h, 95
- LUA_IDSIZE
 - luaconf.h, 108
- LUA_INT_INT
 - luaconf.h, 108
- LUA_INT_LONG
 - luaconf.h, 108
- LUA_INT_LONGLONG
 - luaconf.h, 108
- LUA_INT_TYPE
 - luaconf.h, 108
- LUA_INTEGER_FMT
 - luaconf.h, 108
- LUA_IOLIBNAME
 - lua.h, 111
- LUA_KCONTEXT
 - luaconf.h, 108
- LUA_LDIR
 - luaconf.h, 108
- LUA_LOADLIBNAME
 - lua.h, 112
- LUA_MASKCALL
 - lua.h, 96
- LUA_MASKCOUNT
 - lua.h, 96

LUA_MASKLINE
lua.h, 96

LUA_MASKRET
lua.h, 96

LUA_MATHLIBNAME
lua.h, 112

LUA_MINSTACK
lua.h, 96

LUA_MULTRET
lua.h, 96

LUA_NOREF
lua.h, 59

LUA_NUMBER
luaconf.h, 108

LUA_NUMBER_FMT
luaconf.h, 109

LUA_NUMBER_FRMLEN
luaconf.h, 109

LUA_NUMTAGS
lua.h, 96

LUA_OK
lua.h, 96

LUA_OPADD
lua.h, 97

LUA_OPBAND
lua.h, 97

LUA_OPBNOT
lua.h, 97

LUA_OPBOR
lua.h, 97

LUA_OPBXOR
lua.h, 97

LUA_OPDIV
lua.h, 97

LUA_OPEQ
lua.h, 97

LUA_OPIDIV
lua.h, 97

LUA_OPLE
lua.h, 97

LUA_OPLT
lua.h, 97

LUA_OPMOD
lua.h, 97

LUA_OPMUL
lua.h, 97

LUA_OPPOW
lua.h, 98

LUA_OPSHL
lua.h, 98

LUA_OPSHR
lua.h, 98

LUA_OPSUB
lua.h, 98

LUA_OPUNM
lua.h, 98

LUA_OSLIBNAME
lua.h, 112

LUA_PATH_DEFAULT
luaconf.h, 109

LUA_QL
luaconf.h, 109

LUA_QS
luaconf.h, 109

LUA_REFNIL
lua.h, 59

LUA_REGISTRYINDEX
lua.h, 98

LUA_RELEASE
lua.h, 99

LUA_RIDX_GLOBALS
lua.h, 99

LUA_RIDX_LAST
lua.h, 99

LUA_RIDX_MAINTHREAD
lua.h, 99

LUA_ROOT
luaconf.h, 109

LUA_SIGNATURE
lua.h, 99

LUA_STRLIBNAME
lua.h, 112

LUA_TABLIBNAME
lua.h, 112

LUA_TBOOLEAN
lua.h, 99

LUA_TFUNCTION
lua.h, 99

LUA_TLIGHTUSERDATA
lua.h, 99

LUA_TNIL
lua.h, 99

LUA_TNONE
lua.h, 99

LUA_TNUMBER
lua.h, 100

LUA_TSTRING
lua.h, 100

LUA_TTABLE
lua.h, 100

LUA_TTHREAD
lua.h, 100

LUA_TUSERDATA
lua.h, 100

LUA_UNSIGNED
luaconf.h, 109

LUA_UTF8LIBNAME
lua.h, 112

LUA_VDIR
luaconf.h, 110

LUA_VERSION
lua.h, 100

LUA_VERSION_MAJOR
lua.h, 100

LUA_VERSION_MINOR
lua.h, 100

- LUA_VERSION_NUM
 - lua.h, 101
- LUA_VERSION_RELEASE
 - lua.h, 101
- LUA_YIELD
 - lua.h, 101
- LUAI_BITSINT
 - luaconf.h, 110
- LUAI_DDEC
 - luaconf.h, 110
- LUAI_DDEF
 - luaconf.h, 110
- LUAI_FUNC
 - luaconf.h, 110
- LUAI_MAXSTACK
 - luaconf.h, 110
- LUAI_UACINT
 - luaconf.h, 110
- LUAI_UACNUMBER
 - luaconf.h, 110
- LUAL_BUFFERSIZE
 - luaconf.h, 110
- LUAL_NUMSIZES
 - lauxlib.h, 60
- LUALIB_API
 - luaconf.h, 110
- LUAMOD_API
 - luaconf.h, 110
- lastlinedefined
 - lua_Debug, 16
- lastsignal
 - helper.c, 55
- lauxlib.h
 - LUA_ERRFILE, 59
 - LUA_FILEHANDLE, 59
 - LUA_NOREF, 59
 - LUA_REFNIL, 59
 - LUAL_NUMSIZES, 60
 - lua_writeline, 59
 - lua_writestring, 59
 - lua_writestringerror, 59
 - luaL_Buffer, 61
 - luaL_Reg, 61
 - luaL_Stream, 61
 - luaL_addchar, 59
 - luaL_addlstring, 61
 - luaL_addsize, 59
 - luaL_addstring, 61
 - luaL_addvalue, 61
 - luaL_argcheck, 59
 - luaL_argerror, 61
 - luaL_buffinit, 61
 - luaL_buffinitsize, 61
 - luaL_callmeta, 61
 - luaL_checkany, 61
 - luaL_checkinteger, 61
 - luaL_checklstring, 61
 - luaL_checknumber, 61
 - luaL_checkoption, 61
 - luaL_checkstack, 61
 - luaL_checkstring, 60
 - luaL_checktype, 61
 - luaL_checkudata, 61
 - luaL_checkversion, 60
 - luaL_checkversion_, 61
 - luaL_dofile, 60
 - luaL_dostring, 60
 - luaL_error, 61
 - luaL_execresult, 62
 - luaL_fileresult, 62
 - luaL_getmetafield, 62
 - luaL_getmetatable, 60
 - luaL_getsubtable, 62
 - luaL_gsub, 62
 - luaL_len, 62
 - luaL_loadbuffer, 60
 - luaL_loadbufferx, 62
 - luaL_loadfile, 60
 - luaL_loadfilex, 62
 - luaL_loadstring, 62
 - luaL_newlib, 60
 - luaL_newlibtable, 60
 - luaL_newmetatable, 62
 - luaL_newstate, 62
 - luaL_opt, 60
 - luaL_optinteger, 62
 - luaL_optlstring, 62
 - luaL_optnumber, 62
 - luaL_optstring, 60
 - luaL_prepbuffer, 61
 - luaL_prepbuffsize, 62
 - luaL_pushresult, 62
 - luaL_pushresultsizes, 62
 - luaL_ref, 62
 - luaL_requiref, 62
 - luaL_setfuncs, 62
 - luaL_setmetatable, 62
 - luaL_testudata, 62
 - luaL_tolstring, 62
 - luaL_traceback, 62
 - luaL_typename, 61
 - luaL_unref, 62
 - luaL_where, 62
- learn_key_t, 13
 - targ, 13
 - tfunction, 13
 - tlib, 13
 - ttype, 14
 - tvalue, 14
 - wsh.c, 119
- learn_proto
 - wsh.c, 124
- learn_t, 14
 - hh, 14
 - key, 14
 - toffset, 14

- wsh.c, 119
- learnfile
 - wsh_t, 32
- learnlog
 - wsh_t, 32
- len
 - linenoiseCompletions, 15
 - msec_t, 19
- libcall
 - wsh.c, 124
 - wsh.h, 79
- libify
 - wcc.c, 46
- libname
 - sections_t, 25
 - segments_t, 26
 - symbols_t, 29
- limit
 - elfdata_t, 10
- linedefined
 - lua_Debug, 16
- linenoise
 - linenoise.h, 88
- linenoise.h
 - linenoise, 88
 - linenoiseAddCompletion, 88
 - linenoiseClearScreen, 88
 - linenoiseCompletionCallback, 88
 - linenoiseCompletions, 88
 - linenoiseHistoryAdd, 88
 - linenoiseHistoryLoad, 88
 - linenoiseHistorySave, 88
 - linenoiseHistorySetMaxLen, 88
 - linenoisePrintKeyCodes, 88
 - linenoiseSetCompletionCallback, 88
 - linenoiseSetMultiLine, 88
- linenoiseAddCompletion
 - linenoise.h, 88
- linenoiseClearScreen
 - linenoise.h, 88
- linenoiseCompletionCallback
 - linenoise.h, 88
- linenoiseCompletions, 14
 - cvec, 15
 - len, 15
 - linenoise.h, 88
- linenoiseHistoryAdd
 - linenoise.h, 88
- linenoiseHistoryLoad
 - linenoise.h, 88
- linenoiseHistorySave
 - linenoise.h, 88
- linenoiseHistorySetMaxLen
 - linenoise.h, 88
- linenoisePrintKeyCodes
 - linenoise.h, 88
- linenoiseSetCompletionCallback
 - linenoise.h, 88
- linenoiseSetMultiLine
 - linenoise.h, 88
- link_from_name
 - wcc.c, 47
- link_map
 - elfdata_t, 10
- load_binary
 - wcc.c, 47
- loadbin
 - wsh.c, 125
 - wsh.h, 80
- loadlibrary
 - wsh.c, 125
- longjmp.h
 - CATCH, 89
 - ETRY, 89
 - FINALLY, 89
 - THROW, 89
 - TRY, 89
- longjmp_ptr
 - wsh_t, 32
- longjmp_ptr_high
 - wsh_t, 32
- longjmp_ptr_high_cnt
 - wsh_t, 33
- ltrace
 - wsh.c, 125
 - wsh.h, 80
- lua.h
 - LUA_AUTHORS, 93
 - LUA_COPYRIGHT, 93
 - LUA_ERRERR, 93
 - LUA_ERRGCOMM, 93
 - LUA_ERRMEM, 94
 - LUA_ERRRUN, 94
 - LUA_ERRSYNTAX, 94
 - LUA_GCCOLLECT, 94
 - LUA_GCCOUNT, 94
 - LUA_GCCOUNTB, 94
 - LUA_GCISRUNNING, 94
 - LUA_GCRESTART, 94
 - LUA_GCSETPAUSE, 94
 - LUA_GCSETSTEPMUL, 94
 - LUA_GCSTEP, 94
 - LUA_GCSTOP, 94
 - LUA_HOOKCALL, 95
 - LUA_HOOKCOUNT, 95
 - LUA_HOOKLINE, 95
 - LUA_HOOKRET, 95
 - LUA_HOOKTAILCALL, 95
 - LUA_MASKCALL, 96
 - LUA_MASKCOUNT, 96
 - LUA_MASKLINE, 96
 - LUA_MASKRET, 96
 - LUA_MINSTACK, 96
 - LUA_MULTRET, 96
 - LUA_NUMTAGS, 96
 - LUA_OK, 96

LUA_OPADD, 97
LUA_OPBAND, 97
LUA_OPBNOT, 97
LUA_OPBOR, 97
LUA_OPBXOR, 97
LUA_OPDIV, 97
LUA_OPEQ, 97
LUA_OPIDIV, 97
LUA_OPLE, 97
LUA_OPLT, 97
LUA_OPMOD, 97
LUA_OPMUL, 97
LUA_OPPOW, 98
LUA_OPSHL, 98
LUA_OPSHR, 98
LUA_OPSUB, 98
LUA_OPUNM, 98
LUA_REGISTRYINDEX, 98
LUA_RELEASE, 99
LUA_RIDX_GLOBALS, 99
LUA_RIDX_LAST, 99
LUA_RIDX_MAINTHREAD, 99
LUA_SIGNATURE, 99
LUA_TBOOLEAN, 99
LUA_TFUNCTION, 99
LUA_TLIGHTUSERDATA, 99
LUA_TNIL, 99
LUA_TNONE, 99
LUA_TNUMBER, 100
LUA_TSTRING, 100
LUA_TTABLE, 100
LUA_TTHREAD, 100
LUA_TUSERDATA, 100
LUA_VERSION, 100
LUA_VERSION_MAJOR, 100
LUA_VERSION_MINOR, 100
LUA_VERSION_NUM, 101
LUA_VERSION_RELEASE, 101
LUA_YIELD, 101
lua_Alloc, 101
lua_CFunction, 101
lua_Debug, 101
lua_Hook, 101
lua_Integer, 101
lua_KContext, 101
lua_KFunction, 101
lua_Number, 101
lua_Reader, 102
lua_State, 102
lua_Unsigned, 102
lua_Writer, 102
lua_absindex, 102
lua_arith, 102
lua_atpanic, 102
lua_call, 93
lua_callk, 102
lua_checkstack, 102
lua_close, 102
lua_compare, 102
lua_concat, 102
lua_copy, 102
lua_createtable, 102
lua_dump, 102
lua_error, 102
lua_gc, 102
lua_getallocf, 102
lua_getextraspace, 95
lua_getfield, 102
lua_getglobal, 102
lua_gethook, 103
lua_gethookcount, 103
lua_gethookmask, 103
lua_geti, 103
lua_getinfo, 103
lua_getlocal, 103
lua_getmetatable, 103
lua_getstack, 103
lua_gettable, 103
lua_gettop, 103
lua_getupvalue, 103
lua_getuservalue, 103
lua_ident, 105
lua_insert, 95
lua_isboolean, 95
lua_iscfunction, 103
lua_isfunction, 95
lua_isinteger, 103
lua_islighuserdata, 95
lua_isnil, 95
lua_isnone, 95
lua_isnoneornil, 96
lua_isnumber, 103
lua_isstring, 103
lua_istable, 96
lua_isthread, 96
lua_isuserdata, 103
lua_isyieldable, 103
lua_len, 103
lua_load, 103
lua_newstate, 103
lua_newtable, 96
lua_newthread, 103
lua_newuserdata, 103
lua_next, 103
lua_pcall, 98
lua_pcallk, 103
lua_pop, 98
lua_pushboolean, 103
lua_pushcclosure, 103
lua_pushcfunction, 98
lua_pushfstring, 104
lua_pushglobaltable, 98
lua_pushinteger, 104
lua_pushlightuserdata, 104
lua_pushliteral, 98
lua_pushlstring, 104

- lua_pushnil, 104
- lua_pushnumber, 104
- lua_pushstring, 104
- lua_pushthread, 104
- lua_pushvalue, 104
- lua_pushvfstring, 104
- lua_rawequal, 104
- lua_rawget, 104
- lua_rawgeti, 104
- lua_rawgetp, 104
- lua_rawlen, 104
- lua_rawset, 104
- lua_rawseti, 104
- lua_rawsetp, 104
- lua_register, 98
- lua_remove, 99
- lua_replace, 99
- lua_resume, 104
- lua_rotate, 104
- lua_setallocf, 104
- lua_setfield, 104
- lua_setglobal, 104
- lua_sethook, 104
- lua_seti, 104
- lua_setlocal, 104
- lua_setmetatable, 104
- lua_settable, 104
- lua_settop, 105
- lua_setupvalue, 105
- lua_setuservalue, 105
- lua_status, 105
- lua_stringtonumber, 105
- lua_toboolean, 105
- lua_tocfunction, 105
- lua_tointeger, 100
- lua_tointegerx, 105
- lua_tolstring, 105
- lua_tonumber, 100
- lua_tonumberx, 105
- lua_topointer, 105
- lua_tostring, 100
- lua_tothread, 105
- lua_touserdata, 105
- lua_type, 105
- lua_typename, 105
- lua_upvalueid, 105
- lua_upvalueindex, 100
- lua_upvaluejoin, 105
- lua_version, 105
- lua_xmove, 105
- lua_yield, 101
- lua_yieldk, 105
- lua_Alloc
 - lua.h, 101
- lua_CFunction
 - lua.h, 101
- lua_Debug, 15
 - currentline, 16
 - event, 16
 - i_ci, 16
 - istailcall, 16
 - isvararg, 16
 - lastlinedefined, 16
 - linedefined, 16
 - lua.h, 101
 - name, 16
 - namewhat, 16
 - nparams, 16
 - nups, 16
 - short_src, 16
 - source, 16
 - what, 17
- lua_Hook
 - lua.h, 101
- lua_Integer
 - lua.h, 101
- lua_KContext
 - lua.h, 101
- lua_KFunction
 - lua.h, 101
- lua_Number
 - lua.h, 101
- lua_Reader
 - lua.h, 102
- lua_State
 - lua.h, 102
- lua_Unsigned
 - lua.h, 102
- lua_Writer
 - lua.h, 102
- lua_absindex
 - lua.h, 102
- lua_arith
 - lua.h, 102
- lua_assert
 - lua.h, 111
- lua_atpanic
 - lua.h, 102
- lua_blacklist
 - wsh_functions.h, 86
- lua_call
 - lua.h, 93
- lua_callk
 - lua.h, 102
- lua_checkstack
 - lua.h, 102
- lua_close
 - lua.h, 102
- lua_compare
 - lua.h, 102
- lua_concat
 - lua.h, 102
- lua_copy
 - lua.h, 102
- lua_createtable
 - lua.h, 102

- lua_default_functions
 - wsh_functions.h, 87
- lua_dump
 - lua.h, 102
- lua_error
 - lua.h, 102
- lua_gc
 - lua.h, 102
- lua_getallocf
 - lua.h, 102
- lua_getextraspace
 - lua.h, 95
- lua_getfield
 - lua.h, 102
- lua_getglobal
 - lua.h, 102
- lua_gethook
 - lua.h, 103
- lua_gethookcount
 - lua.h, 103
- lua_gethookmask
 - lua.h, 103
- lua_geti
 - lua.h, 103
- lua_getinfo
 - lua.h, 103
- lua_getlocal
 - lua.h, 103
- lua_getlocaledecpoint
 - luaconf.h, 107
- lua_getmetatable
 - lua.h, 103
- lua_getstack
 - lua.h, 103
- lua_gettable
 - lua.h, 103
- lua_gettop
 - lua.h, 103
- lua_getupvalue
 - lua.h, 103
- lua_getuservalue
 - lua.h, 103
- lua_ident
 - lua.h, 105
- lua_insert
 - lua.h, 95
- lua_integer2str
 - luaconf.h, 108
- lua_isboolean
 - lua.h, 95
- lua_iscfunctor
 - lua.h, 103
- lua_isfunction
 - lua.h, 95
- lua_isinteger
 - lua.h, 103
- lua_isthread
 - lua.h, 96
- lua_islightuserdata
 - lua.h, 95
- lua_isnil
 - lua.h, 95
- lua_isnone
 - lua.h, 95
- lua_isnoneornil
 - lua.h, 96
- lua_isnumber
 - lua.h, 103
- lua_isstring
 - lua.h, 103
- lua_istable
 - lua.h, 96
- lua_isthread
 - lua.h, 96
- lua_isuserdata
 - lua.h, 103
- lua_isyieldable
 - lua.h, 103
- lua_len
 - lua.h, 103
- lua_load
 - lua.h, 103
- lua_newstate
 - lua.h, 103
- lua_newtable
 - lua.h, 96
- lua_newthread
 - lua.h, 103
- lua_newuserdata
 - lua.h, 103
- lua_next
 - lua.h, 103
- lua_number2str
 - luaconf.h, 108
- lua_number2strx
 - luaconf.h, 108
- lua_numbertointeger
 - luaconf.h, 109
- lua_pcall
 - lua.h, 98
- lua_pcallk
 - lua.h, 103
- lua_pop
 - lua.h, 98
- lua_pushboolean
 - lua.h, 103
- lua_pushcclosure
 - lua.h, 103
- lua_pushcfunctor
 - lua.h, 98
- lua_pushstring
 - lua.h, 104
- lua_pushglobaltable
 - lua.h, 98
- lua_pushinteger
 - lua.h, 104
- lua_pushlightuserdata
 - lua.h, 104

lua_pushliteral
lua.h, 98

lua_pushlstring
lua.h, 104

lua_pushnil
lua.h, 104

lua_pushnumber
lua.h, 104

lua_pushstring
lua.h, 104

lua_pushthread
lua.h, 104

lua_pushvalue
lua.h, 104

lua_pushvfstring
lua.h, 104

lua_rawequal
lua.h, 104

lua_rawget
lua.h, 104

lua_rawgeti
lua.h, 104

lua_rawgetp
lua.h, 104

lua_rawlen
lua.h, 104

lua_rawset
lua.h, 104

lua_rawseti
lua.h, 104

lua_rawsetp
lua.h, 104

lua_register
lua.h, 98

lua_remove
lua.h, 99

lua_replace
lua.h, 99

lua_resume
lua.h, 104

lua_rotate
lua.h, 104

lua_setallocf
lua.h, 104

lua_setfield
lua.h, 104

lua_setglobal
lua.h, 104

lua_sethook
lua.h, 104

lua_seti
lua.h, 104

lua_setlocal
lua.h, 104

lua_setmetatable
lua.h, 104

lua_settable
lua.h, 104

lua_settop
lua.h, 105

lua_setupvalue
lua.h, 105

lua_setuservalue
lua.h, 105

lua_status
lua.h, 105

lua_str2number
luaconf.h, 109

lua_strerror
wsh.c, 125

lua_stringtonumber
lua.h, 105

lua_strx2number
luaconf.h, 109

lua_toboolean
lua.h, 105

lua_tocfunction
lua.h, 105

lua_tointeger
lua.h, 100

lua_tointegerx
lua.h, 105

lua_tolstring
lua.h, 105

lua_tonumber
lua.h, 100

lua_tonumberx
lua.h, 105

lua_topointer
lua.h, 105

lua_tostring
lua.h, 100

lua_tothread
lua.h, 105

lua_touserdata
lua.h, 105

lua_type
lua.h, 105

lua_typename
lua.h, 105

lua_upvalueid
lua.h, 105

lua_upvalueindex
lua.h, 100

lua_upvaluejoin
lua.h, 105

lua_version
lua.h, 105

lua_writeline
lua.h, 59

lua_writestring
lua.h, 59

lua_writestringerror
lua.h, 59

lua_xmove
lua.h, 105

lua_yield
 lua.h, 101

lua_yieldk
 lua.h, 105

luaL_Buffer, 17
 b, 17
 initb, 17
 L, 17
 lauxlib.h, 61
 n, 17
 size, 17

luaL_Reg, 18
 func, 18
 lauxlib.h, 61
 name, 18

luaL_Stream, 18
 closef, 19
 f, 19
 lauxlib.h, 61

luaL_addchar
 lauxlib.h, 59

luaL_addlstring
 lauxlib.h, 61

luaL_addsize
 lauxlib.h, 59

luaL_addstring
 lauxlib.h, 61

luaL_addvalue
 lauxlib.h, 61

luaL_argcheck
 lauxlib.h, 59
 mylaux.h, 64

luaL_argerror
 lauxlib.h, 61

luaL_buffinit
 lauxlib.h, 61

luaL_buffinitsize
 lauxlib.h, 61

luaL_callmeta
 lauxlib.h, 61

luaL_checkany
 lauxlib.h, 61

luaL_checkinteger
 lauxlib.h, 61

luaL_checklstring
 lauxlib.h, 61

luaL_checknumber
 lauxlib.h, 61

luaL_checkoption
 lauxlib.h, 61

luaL_checkstack
 lauxlib.h, 61

luaL_checkstring
 lauxlib.h, 60
 mylaux.h, 64

luaL_checktype
 lauxlib.h, 61

luaL_checkudata
 lauxlib.h, 61

luaL_checkversion
 lauxlib.h, 60

luaL_checkversion_
 lauxlib.h, 61

luaL_dofile
 lauxlib.h, 60
 mylaux.h, 64

luaL_dostring
 lauxlib.h, 60
 mylaux.h, 64

luaL_error
 lauxlib.h, 61

luaL_execresult
 lauxlib.h, 62

luaL_fileresult
 lauxlib.h, 62

luaL_getmetafield
 lauxlib.h, 62

luaL_getmetatable
 lauxlib.h, 60
 mylaux.h, 64

luaL_getsubtable
 lauxlib.h, 62

luaL_gsub
 lauxlib.h, 62

luaL_len
 lauxlib.h, 62

luaL_loadbuffer
 lauxlib.h, 60
 mylaux.h, 64

luaL_loadbufferx
 lauxlib.h, 62

luaL_loadfile
 lauxlib.h, 60

luaL_loadfilex
 lauxlib.h, 62

luaL_loadstring
 lauxlib.h, 62

luaL_newlib
 lauxlib.h, 60
 mylaux.h, 64

luaL_newlibtable
 lauxlib.h, 60
 mylaux.h, 64

luaL_newmetatable
 lauxlib.h, 62

luaL_newstate
 lauxlib.h, 62

luaL_openlibs
 lua.h, 112

luaL_opt
 lauxlib.h, 60
 mylaux.h, 64

luaL_optinteger
 lauxlib.h, 62

luaL_optlstring
 lauxlib.h, 62

- luaL_optnumber
 - lauxlib.h, 62
- luaL_optstring
 - lauxlib.h, 60
 - mylaux.h, 64
- luaL_prepbuffer
 - lauxlib.h, 61
- luaL_prepbuffsize
 - lauxlib.h, 62
- luaL_pushresult
 - lauxlib.h, 62
- luaL_pushresults
 - lauxlib.h, 62
- luaL_ref
 - lauxlib.h, 62
- luaL_reg
 - wsh.h, 72
- luaL_requiref
 - lauxlib.h, 62
- luaL_setfuncs
 - lauxlib.h, 62
- luaL_setmetatable
 - lauxlib.h, 62
- luaL_testudata
 - lauxlib.h, 62
- luaL_tolstring
 - lauxlib.h, 62
- luaL_traceback
 - lauxlib.h, 62
- luaL_typename
 - lauxlib.h, 61
 - mylaux.h, 64
- luaL_unref
 - lauxlib.h, 62
- luaL_where
 - lauxlib.h, 62
- luaconf.h
 - l_floor, 106
 - l_mathlim, 106
 - l_mathop, 107
 - l_sprintf, 107
 - LUA_API, 107
 - LUA_CDIR, 107
 - LUA_CPATH_DEFAULT, 107
 - LUA_DIRSEP, 107
 - LUA_EXTRASPACE, 107
 - LUA_FLOAT_DOUBLE, 107
 - LUA_FLOAT_FLOAT, 107
 - LUA_FLOAT_TYPE, 107
 - LUA_IDSIZE, 108
 - LUA_INT_INT, 108
 - LUA_INT_LONG, 108
 - LUA_INT_LONGLONG, 108
 - LUA_INT_TYPE, 108
 - LUA_INTEGER_FMT, 108
 - LUA_KCONTEXT, 108
 - LUA_LDIR, 108
 - LUA_NUMBER, 108
 - LUA_NUMBER_FMT, 109
 - LUA_NUMBER_FRMLEN, 109
 - LUA_PATH_DEFAULT, 109
 - LUA_QL, 109
 - LUA_QS, 109
 - LUA_ROOT, 109
 - LUA_UNSIGNED, 109
 - LUA_VDIR, 110
 - LUA_BITSINT, 110
 - LUA_DDEC, 110
 - LUA_DDEF, 110
 - LUA_FUNC, 110
 - LUA_MAXSTACK, 110
 - LUA_UACINT, 110
 - LUA_UACNUMBER, 110
 - LUAL_BUFFERSIZE, 110
 - LUALIB_API, 110
 - LUAMOD_API, 110
 - lua_getlocaledecpoint, 107
 - lua_integer2str, 108
 - lua_number2str, 108
 - lua_number2strx, 108
 - lua_numbertointeger, 109
 - lua_str2number, 109
 - lua_strx2number, 109
- luaolib.h
 - LUA_BITLIBNAME, 111
 - LUA_COLIBNAME, 111
 - LUA_DBLIBNAME, 111
 - LUA_IOLIBNAME, 111
 - LUA_LOADLIBNAME, 112
 - LUA_MATHLIBNAME, 112
 - LUA_OSLIBNAME, 112
 - LUA_STRLIBNAME, 112
 - LUA_TABLIBNAME, 112
 - LUA_UTF8LIBNAME, 112
 - lua_assert, 111
 - luaL_openlibs, 112
 - luaopen_base, 112
 - luaopen_bit32, 112
 - luaopen_coroutine, 112
 - luaopen_debug, 112
 - luaopen_io, 112
 - luaopen_math, 112
 - luaopen_os, 112
 - luaopen_package, 112
 - luaopen_string, 112
 - luaopen_table, 112
 - luaopen_utf8, 112
- luaopen_base
 - luaolib.h, 112
- luaopen_bit32
 - luaolib.h, 112
- luaopen_coroutine
 - luaolib.h, 112
- luaopen_debug
 - luaolib.h, 112
- luaopen_io

- luaolib.h, 112
- luaopen_math
 - luaolib.h, 112
- luaopen_os
 - luaolib.h, 112
- luaopen_package
 - luaolib.h, 112
- luaopen_string
 - luaolib.h, 112
- luaopen_table
 - luaolib.h, 112
- luaopen_utf8
 - luaolib.h, 112
- MAGENTA
 - colors.h, 57
- MAX_SIGNALS
 - wsh.h, 72
- MAXPADLEN
 - wcc.c, 43
- MIN_BIN_SIZE
 - wsh.h, 72
- MY_CPU
 - wsh.h, 72
- main
 - wcc.c, 47
 - wld.c, 54
 - wshmain.c, 137
- mainhandle
 - wsh_t, 33
- man
 - wsh.c, 125
 - wsh.h, 80
- map
 - wsh.c, 125
 - wsh.h, 80
- max
 - range_t, 22
 - wcc.c, 47
- maxdata
 - wcc.c, 52
- maxnewsec
 - wcc.c, 52
- maxoldsec
 - wcc.c, 52
- maxtext
 - wcc.c, 53
- merge_phdrs
 - wcc.c, 47
- min
 - range_t, 22
- mindata
 - wcc.c, 53
- mintext
 - wcc.c, 53
- mk_backtrace
 - wsh.c, 125
- mk_lib
 - wld.c, 54
- mk_section
 - wcc.c, 47
- mphdrs
 - ctx_t, 7
- mphnum
 - ctx_t, 7
- msec_t, 19
 - data, 19
 - flags, 19
 - len, 19
 - name, 19
 - next, 19
 - outoffset, 20
 - prev, 20
 - s_bfd, 20
 - s_elf, 20
 - wcc.c, 43
- mseg_t, 20
 - next, 20
 - p_align, 20
 - p_filesz, 21
 - p_flags, 21
 - p_memsz, 21
 - p_offset, 21
 - p_paddr, 21
 - p_type, 21
 - p_vaddr, 21
 - prev, 21
 - wcc.c, 43
- mshdrs
 - ctx_t, 7
- mshnum
 - ctx_t, 7
- mylaux.h
 - luaL_argcheck, 64
 - luaL_checkstring, 64
 - luaL_dofile, 64
 - luaL_dostring, 64
 - luaL_getmetatable, 64
 - luaL_loadbuffer, 64
 - luaL_newlib, 64
 - luaL_newlibtable, 64
 - luaL_opt, 64
 - luaL_optstring, 64
 - luaL_typename, 64
- n
 - luaL_Buffer, 17
- NORMAL
 - colors.h, 57
- name
 - eps_t, 11
 - help_t, 13
 - lua_Debug, 16
 - lua_Reg, 18
 - msec_t, 19
 - preload_t, 22
 - script_t, 23
 - section, 24

- sections_t, 25
- signame_t, 27
- symaddr, 28
- tuple_t, 30
- namewhat
 - lua_Debug, 16
- newarray
 - wsh.h, 80
- next
 - eps_t, 11
 - msec_t, 19
 - mseg_t, 20
 - preload_t, 22
 - script_t, 23
 - section, 24
 - sections_t, 25
 - segments_t, 26
 - symaddr, 28
 - symbols_t, 29
- nparams
 - lua_Debug, 16
- nsections
 - helper.c, 55
 - helper.h, 63
- nullstr
 - wcc.c, 43
- num
 - section, 24
- nups
 - lua_Debug, 16
- open_best
 - wcc.c, 47
- open_target
 - wcc.c, 47
- opt_arch
 - ctx_t, 7
- opt_argc
 - wsh_t, 33
- opt_argv
 - wsh_t, 33
- opt_asmdebug
 - ctx_t, 7
- opt_binname
 - ctx_t, 7
- opt_core
 - ctx_t, 7
- opt_debug
 - ctx_t, 8
- opt_entrypoint
 - ctx_t, 8
- opt_exec
 - ctx_t, 8
- opt_flags
 - ctx_t, 8
- opt_hollywood
 - wsh_t, 33
- opt_interp
 - ctx_t, 8
- opt_original
 - ctx_t, 8
- opt_poison
 - ctx_t, 8
- opt_reloc
 - ctx_t, 8
- opt_rescan
 - wsh_t, 33
- opt_shared
 - ctx_t, 8
- opt_sstrip
 - ctx_t, 8
- opt_static
 - ctx_t, 8
- opt_strip
 - ctx_t, 8
- opt_verbose
 - ctx_t, 9
 - wsh_t, 33
- opt_verbosetrace
 - wsh_t, 33
- orig_sz
 - wcc.c, 53
- orig_text
 - wcc.c, 53
- outoffset
 - msec_t, 20
- p_align
 - mseg_t, 20
- p_filesz
 - mseg_t, 21
- p_flags
 - mseg_t, 21
- p_memsz
 - mseg_t, 21
- p_offset
 - mseg_t, 21
- p_paddr
 - mseg_t, 21
- p_pltgot
 - elfdata_t, 10
- p_type
 - mseg_t, 21
- p_vaddr
 - mseg_t, 21
- PROC_ASLR_PATH
 - wsh.h, 72
- parse_dyn
 - wsh.c, 125
- parse_link_map_dyn
 - wsh.c, 126
- patch_symbol_index
 - wcc.c, 48
- perms
 - section, 24
 - sections_t, 25
 - segments_t, 26
- pflag_from_section

- wcc.c, 48
- phdr_callback
 - wsh.c, 126
- phdr_cmp
 - wcc.c, 48
 - wsh.c, 126
- phdr_cmp_premerge
 - wcc.c, 48
- phdrs
 - elfdata_t, 10
 - wsh.c, 126
 - wsh.h, 80
 - wsh_t, 33
- phnum
 - ctx_t, 9
- pltgot
 - wsh_t, 33
- pltz
 - wsh_t, 33
- preload
 - wsh_t, 33
- preload_t, 21
 - name, 22
 - next, 22
 - prev, 22
 - wsh.h, 75
- prev
 - eps_t, 11
 - msec_t, 20
 - mseg_t, 21
 - preload_t, 22
 - script_t, 23
 - sections_t, 25
 - segments_t, 26
 - symbols_t, 29
- print_backtrace
 - wsh.c, 126
- print_bfd_sections
 - wcc.c, 48
- print_eps
 - wsh.c, 126
- print_functions
 - wsh.c, 126
 - wsh.h, 80
- print_libs
 - wsh.c, 126
 - wsh.h, 80
- print_maps
 - wcc.c, 48
- print_msec
 - wcc.c, 48
- print_objects
 - wsh.c, 126
 - wsh.h, 80
- print_phdrs
 - wsh.c, 126
 - wsh.h, 80
- print_procmmap
 - wsh.c, 127
- print_shdrs
 - wsh.c, 127
 - wsh.h, 81
- print_symbols
 - wsh.c, 127
 - wsh.h, 81
- print_version
 - wcc.c, 48
 - wld.c, 54
 - wsh.h, 81
- printarg
 - wsh.c, 127
- priv_memcpy
 - wsh.c, 127
 - wsh.h, 81
- priv_strcat
 - wsh.c, 127
 - wsh.h, 81
- priv_strcpy
 - wsh.c, 127
 - wsh.h, 81
- proba
 - section, 24
- probableval
 - section, 24
- procmmap_lua
 - wsh.c, 127
 - wsh.h, 81
- protect_perms
 - wcc.c, 48
- proto
 - help_t, 13
- protoprefix
 - help_t, 13
- protorecords
 - wsh.c, 136
- prototypes
 - wsh.c, 127
 - wsh.h, 81
- ptoh
 - wsh.c, 128
- ptr
 - breakpoint_t, 5
- pctype_from_section
 - wcc.c, 48
- r
 - gimport_t, 12
- r_debug
 - elfdata_t, 10
- RED
 - colors.h, 57
- REG_RIP
 - wsh.c, 118
- RELOC_MODE
 - wcc.c, 43
 - wsh.c, 118
- RELOC_X86_32

- wcc.c, 43
- RELOC_X86_64
 - wcc.c, 43
- ralloc
 - wsh.c, 128
 - wsh.h, 81
- range_t, 22
 - max, 22
 - min, 22
 - wsh.h, 75
- ranges
 - wsh_functions.h, 87
- rawmemaddr
 - wsh.c, 128
 - wsh.h, 82
- rawmemread
 - wsh.c, 128
 - wsh.h, 82
- rawmemstr
 - wsh.c, 128
 - wsh.h, 82
- rawmemstrlen
 - wsh.c, 128
 - wsh.h, 82
- rawmemusage
 - wsh.c, 128
 - wsh.h, 82
- rawmemwrite
 - wsh.c, 128
 - wsh.h, 82
- rd_sections
 - wcc.c, 49
- rd_symbols
 - wcc.c, 49
- rd_syntab
 - wcc.c, 49
- rdnum
 - wsh.c, 128
 - wsh.h, 82
- rdstr
 - wsh.c, 129
 - wsh.h, 82
- read_arg
 - wsh.h, 72
- read_arg1
 - wsh.h, 72
- read_arg2
 - wsh.h, 73
- read_arg3
 - wsh.h, 73
- read_arg4
 - wsh.h, 73
- read_elf_sig
 - wsh.c, 129
- read_maps
 - helper.c, 55
 - helper.h, 63
- reason
 - wsh_t, 34
- reload_elfs
 - wsh.c, 129
 - wsh.h, 83
- reloc_htype
 - wcc.c, 49
- reloc_htype_x86_32
 - wcc.c, 49
- reloc_htype_x86_64
 - wcc.c, 49
- rescan
 - wsh.c, 129
 - wsh.h, 83
- restore_exit
 - wsh.c, 129
- retval
 - help_t, 13
- rm_section
 - wcc.c, 49
- rtrace
 - wsh.c, 129
 - wsh.h, 83
- rtype
 - gimport_t, 12
- run_script
 - wsh.c, 129
- run_shell
 - wsh.c, 129
- s_bfd
 - msec_t, 20
- s_elf
 - msec_t, 20
- SHELL_HISTORY_NAME
 - wsh.h, 74
- SKIP_BOTTOM
 - wsh.h, 74
- SKIP_INIT
 - wsh.h, 74
- STB_GLOBAL
 - wsh.h, 74
- STB_GNU_SECONDARY
 - wsh.h, 74
- STB_GNU_UNIQUE
 - wsh.h, 74
- STB_LOCAL
 - wsh.h, 74
- STB_WEAK
 - wsh.h, 74
- STT_COMMON
 - wsh.h, 75
- STT_FILE
 - wsh.h, 75
- STT_FUNC
 - wsh.h, 75
- STT_NOTYPE
 - wsh.h, 75
- STT_OBJECT
 - wsh.h, 75

STT_SECTION
 wsh.h, 75
 STT_TLS
 wsh.h, 75
 save_dynstr
 wcc.c, 49
 save_dynsym
 wcc.c, 49
 save_global_import
 wcc.c, 49
 save_reloc
 wcc.c, 50
 scan_section
 wsh.c, 129
 scan_sections
 wsh.c, 130
 scan_symbol
 wsh.c, 130
 scan_syms
 wsh.c, 130
 script
 wsh.c, 130
 wsh.h, 83
 script_argnum
 wsh_t, 34
 script_args
 wsh_t, 34
 script_t, 23
 name, 23
 next, 23
 prev, 23
 wsh.h, 76
 scriptfile
 wsh_t, 34
 scriptname
 wsh_t, 34
 scripts
 wsh_t, 34
 sec
 gimport_t, 12
 sec_name_from_index_after_strip
 wcc.c, 50
 secindex_from_name
 wcc.c, 50
 secindex_from_name_after_strip
 wcc.c, 50
 section, 23
 end, 24
 hperms, 24
 init, 24
 name, 24
 next, 24
 num, 24
 perms, 24
 proba, 24
 probableval, 24
 size, 24
 section_add
 wsh.c, 130
 section_from_addr
 wcc.c, 50
 wsh.c, 130
 section_from_index
 wcc.c, 50
 section_from_name
 wcc.c, 50
 sections_t, 25
 addr, 25
 flags, 25
 libname, 25
 name, 25
 next, 25
 perms, 25
 prev, 25
 size, 25
 wsh.h, 76
 segment_add
 wsh.c, 130
 wsh.h, 83
 segment_from_addr
 wsh.c, 130
 segments_t, 26
 addr, 26
 flags, 26
 libname, 26
 next, 26
 perms, 26
 prev, 26
 size, 27
 type, 27
 wsh.h, 76
 set_align_flag
 wsh.c, 131
 wsh.h, 83
 set_alloc_opt
 wsh.c, 131
 set_branch_flag
 wsh.c, 131
 wsh.h, 83
 set_sighandlers
 wsh.c, 131
 set_trace_flag
 wsh.c, 131
 wsh.h, 83
 setarray
 wsh.h, 83
 setcharbuf
 wsh.c, 131
 wsh.h, 83
 shdr_callback
 wsh.c, 131
 shdr_cmp
 wsh.c, 131
 shdrs
 wsh.c, 131
 wsh.h, 83

- wsh_t, 34
- shnum
 - ctx_t, 9
- short_src
 - lua_Debug, 16
- sicode_strerror
 - wsh.c, 131
 - wsh.h, 84
- sicodetname
 - wsh.c, 131
- sigbus_count
 - wsh_t, 34
- sigbus_hash
 - wsh_t, 34
- sighandler
 - wsh.c, 132
- signal
 - signame_t, 27
- signaltoname
 - wsh.c, 132
 - wsh.h, 84
- signame_t, 27
 - name, 27
 - signal, 27
 - sigs.h, 65
- signames
 - sigs.h, 65
- sigs.h
 - signame_t, 65
 - signames, 65
- sindex
 - gimport_t, 12
- singlebranch
 - wsh.c, 132
 - wsh.h, 84
- singlebranch_count
 - wsh_t, 34
- singlebranch_hash
 - wsh_t, 34
- singlestep
 - wsh.c, 132
 - wsh.h, 84
- singlestep_count
 - wsh_t, 34
- singlestep_hash
 - wsh_t, 35
- size
 - luaL_Buffer, 17
 - section, 24
 - sections_t, 25
 - segments_t, 27
 - symbols_t, 29
- sname
 - gimport_t, 12
- sort_learnt
 - wsh.c, 132
- sort_phdrs
 - wcc.c, 50
- sort_phdrs_premerge
 - wcc.c, 50
- source
 - lua_Debug, 16
- start_phdrs
 - ctx_t, 9
- start_shdrs
 - ctx_t, 9
- strip_binary_reloc
 - wcc.c, 50
- strndx
 - ctx_t, 9
- strndx_index
 - ctx_t, 9
- strndx_len
 - ctx_t, 9
- symaddr, 27
 - addr, 28
 - name, 28
 - next, 28
- symaddrs
 - wcc.c, 53
- symbol
 - symbols_t, 29
- symbol_from_addr
 - wsh.c, 132
- symbol_from_name
 - wsh.c, 132
- symbol_tobind
 - wsh.c, 132
- symbol_totype
 - wsh.c, 132
- symbols
 - wsh_t, 35
- symbols_t, 28
 - addr, 28
 - hbind, 28
 - htype, 29
 - libname, 29
 - next, 29
 - prev, 29
 - size, 29
 - symbol, 29
 - value, 29
 - wsh.h, 76
- systrace
 - wsh.c, 132
 - wsh.h, 84
- THROW
 - longjmp.h, 89
- TRY
 - longjmp.h, 89
- targ
 - learn_key_t, 13
- test_stdin
 - wsh.c, 133
- textvma
 - wcc.c, 53

- tfunction
 - learn_key_t, 13
- tlib
 - learn_key_t, 13
- toffset
 - learn_t, 14
- tosignals
 - wsh_t, 35
- trace_rtrace
 - wsh_t, 35
- trace_singlebranch
 - wsh_t, 35
- trace_singlestep
 - wsh_t, 35
- trace_strace
 - wsh_t, 35
- trace_unaligned
 - wsh_t, 35
- traceback
 - wsh.c, 133
- traceunaligned
 - wsh.c, 133
 - wsh.h, 84
- traphandler
 - wsh.c, 133
- ttype
 - learn_key_t, 14
- tuple_t, 29
 - addr, 30
 - name, 30
 - wsh.h, 76
- tvalue
 - learn_key_t, 14
- type
 - segments_t, 27
- typefromname
 - wcc.c, 51
- USE_LUA
 - wsh.h, 75
- unrtrace
 - wsh.c, 133
 - wsh.h, 84
- unset_align_flag
 - wsh.c, 133
 - wsh.h, 84
- unset_branch_flag
 - wsh.c, 133
 - wsh.h, 84
- unset_trace_flag
 - wsh.c, 133
 - wsh.h, 84
- unsinglebranch
 - wsh.c, 133
 - wsh.h, 84
- unsinglestep
 - wsh.c, 133
 - wsh.h, 84
- unsystrace
 - wsh.c, 134
 - wsh.h, 85
- untraceunaligned
 - wsh.c, 134
 - wsh.h, 85
- unverbosetrace
 - wsh.c, 134
 - wsh.h, 85
- usage
 - wcc.c, 51
 - wsh.h, 85
- value
 - symbols_t, 29
- verbose
 - wsh.c, 134
 - wsh.h, 85
- verbosetrace
 - wsh.c, 134
 - wsh.h, 85
- wcc.c
 - _GNU_SOURCE, 40
 - __USE_GNU, 40
 - add_extra_symbols, 44
 - add_symaddr, 44
 - adjust_baseaddress, 44
 - alignfromname, 44
 - alloc_phdr, 44
 - allowed_sections, 51
 - analyze_text, 44
 - append_reloc, 44
 - append_strtab, 44
 - append_sym, 44
 - blnames, 51
 - CS_MODE, 40
 - check_global_import, 44
 - copy_body, 45
 - craft_section, 45
 - create_phdrs, 45
 - ctx_getopt, 45
 - ctx_init, 45
 - ctx_t, 43
 - DEFAULT_STRNDX_SIZE, 41
 - datavma, 51
 - deltastrtab, 51
 - desired_arch, 45
 - ELF_R_INFO, 41
 - ELF_R_SYM, 41
 - ELF_R_TYPE, 41
 - ELF_ST_BIND, 42
 - ELF_ST_TYPE, 42
 - ELFCLASS, 42
 - ELFMACHINE, 42
 - Elf_Addr, 41
 - Elf_Ehdr, 41
 - Elf_Off, 41
 - Elf_Phdr, 41
 - Elf_Rel, 41

- Elf_Rel, 41
- Elf_Section, 41
- Elf_Shdr, 41
- Elf_Sword, 42
- Elf_Sym, 42
- Elf_Word, 42
- Elf_Xword, 42
- elis, 42
- entszfromname, 45
- FLAG_BSS, 42
- FLAG_NOBIT, 42
- FLAG_NOWRITE, 43
- FLAG_TEXT, 43
- fixup_strtab_and_syntab, 45
- fixup_syntab_section_index, 45
- fixup_text, 45
- flags_from_name, 46
- gimport_t, 43
- gimports, 51
- gimportslen, 51
- globalreloc, 51
- globalreloclen, 52
- globalreloffset, 52
- globalstrtab, 52
- globalstrtablen, 52
- globalstrtableoffset, 52
- globalsymindex, 52
- globalsymtab, 52
- globalsymtablen, 52
- globalsymtableoffset, 52
- hexdump, 46
- ifis, 43
- info_from_name, 46
- internal_function_store, 46
- libify, 46
- link_from_name, 47
- load_binary, 47
- MAXPADLEN, 43
- main, 47
- max, 47
- maxdata, 52
- maxnewsec, 52
- maxoldsec, 52
- maxtext, 53
- merge_phdrs, 47
- mindata, 53
- mintext, 53
- mk_section, 47
- msec_t, 43
- mseg_t, 43
- nullstr, 43
- open_best, 47
- open_target, 47
- orig_sz, 53
- orig_text, 53
- patch_symbol_index, 48
- pflag_from_section, 48
- phdr_cmp, 48
- phdr_cmp_premerge, 48
- print_bfd_sections, 48
- print_maps, 48
- print_msec, 48
- print_version, 48
- protect_perms, 48
- ptype_from_section, 48
- RELOC_MODE, 43
- RELOC_X86_32, 43
- RELOC_X86_64, 43
- rd_sections, 49
- rd_symbols, 49
- rd_syntab, 49
- reloc_hdtype, 49
- reloc_hdtype_x86_32, 49
- reloc_hdtype_x86_64, 49
- rm_section, 49
- save_dynstr, 49
- save_dynsym, 49
- save_global_import, 49
- save_reloc, 50
- sec_name_from_index_after_strip, 50
- secindex_from_name, 50
- secindex_from_name_after_strip, 50
- section_from_addr, 50
- section_from_index, 50
- section_from_name, 50
- sort_phdrs, 50
- sort_phdrs_premerge, 50
- strip_binary_reloc, 50
- symaddrs, 53
- textvma, 53
- typefromname, 51
- usage, 51
- wcc/wcc.c, 37
- weight
 - breakpoint_t, 5
- what
 - lua_Debug, 17
- wld.c
 - DEFAULT_NAME, 54
 - main, 54
 - mk_lib, 54
 - print_version, 54
- wld/wld.c, 53
- wsh
 - wsh.c, 136
 - wshmain.c, 137
- wsh.c
 - _exit, 119
 - add_binary_preload, 119
 - add_script_arguments, 119
 - add_script_exec, 119
 - add_symbol, 119
 - affinity, 119
 - alarmhandler, 119
 - alloccharbuf, 119
 - bfmap, 120

breakpoint, 120
bsspolute, 120
btr_disable, 120
btr_enable, 120
bushandler, 120
CS_MODE, 116
cmdhelp, 135
completion, 120
declare_func, 120
declare_internals, 121
declare_num, 121
decode_flags, 121
decode_type, 121
detailed_help, 121
disable_aslr, 121
disable_core, 121
do_loadlib, 121
ELF_R_INFO, 117
ELF_R_SYM, 117
ELF_R_TYPE, 117
ELF_ST_BIND, 117
ELF_ST_TYPE, 117
ELFCLASS, 118
ELFMACHINE, 118
Elf_Addr, 116
Elf_Ehdr, 117
Elf_Off, 117
Elf_Phdr, 117
Elf_Rel, 117
Elf_Rela, 117
Elf_Section, 117
Elf_Shdr, 117
Elf_Sword, 118
Elf_Sym, 118
Elf_Word, 118
Elf_Xword, 118
empty_eps, 121
empty_phdrs, 122
empty_shdrs, 122
empty_symbols, 122
enable_aslr, 122
enable_core, 122
entry_point_add, 122
entrypoints, 122
execlib, 122
exit, 122
exit_group, 122
fatal_error, 123
fcnhelp, 135
gencore, 123
getcharbuf, 123
grep, 123
grepptr, 123
headers, 123
help, 123
help_t, 119
hexdump, 123
hollywood, 123
info, 124
info_function, 124
inthandler, 124
learn_key_t, 119
learn_proto, 124
learn_t, 119
libcall, 124
loadbin, 125
loadlibrary, 125
ltrace, 125
lua_strerror, 125
man, 125
map, 125
mk_backtrace, 125
parse_dyn, 125
parse_link_map_dyn, 126
phdr_callback, 126
phdr_cmp, 126
phdrs, 126
print_backtrace, 126
print_eps, 126
print_functions, 126
print_libs, 126
print_objects, 126
print_phdrs, 126
print_procmmap, 127
print_shdrs, 127
print_symbols, 127
printarg, 127
priv_memcpy, 127
priv_strcat, 127
priv_strcpy, 127
procmmap_lua, 127
protorecords, 136
prototypes, 127
ptoh, 128
REG_RIP, 118
RELOC_MODE, 118
ralloc, 128
rawmemaddr, 128
rawmemread, 128
rawmemstr, 128
rawmemstrlen, 128
rawmemusage, 128
rawmemwrite, 128
rdnum, 128
rdstr, 129
read_elf_sig, 129
reload_elfs, 129
rescan, 129
restore_exit, 129
rtrace, 129
run_script, 129
run_shell, 129
scan_section, 129
scan_sections, 130
scan_symbol, 130
scan_syms, 130

- script, 130
- section_add, 130
- section_from_addr, 130
- segment_add, 130
- segment_from_addr, 130
- set_align_flag, 131
- set_alloc_opt, 131
- set_branch_flag, 131
- set_sighandlers, 131
- set_trace_flag, 131
- setcharbuf, 131
- shdr_callback, 131
- shdr_cmp, 131
- shdrs, 131
- sicode_strerror, 131
- sicodetname, 131
- sighandler, 132
- signaltoname, 132
- singlebranch, 132
- singlestep, 132
- sort_learnt, 132
- symbol_from_addr, 132
- symbol_from_name, 132
- symbol_tobind, 132
- symbol_totype, 132
- systrace, 132
- test_stdin, 133
- traceback, 133
- traceunaligned, 133
- traphandler, 133
- unrtrace, 133
- unset_align_flag, 133
- unset_branch_flag, 133
- unset_trace_flag, 133
- unsinglebranch, 133
- unsinglestep, 133
- unsystrace, 134
- untraceunaligned, 134
- unverbosetrace, 134
- verbose, 134
- verbosetrace, 134
- wsh, 136
- wsh_getopt, 134
- wsh_init, 134
- wsh_loadlibs, 134
- wsh_print_version, 134
- wsh_run, 134
- wsh_usage, 134
- xalloc, 135
- xfree, 135
- wsh.h
 - _GNU_SOURCE, 70
 - __progname_full, 86
 - add_symbol, 76
 - alloccharbuf, 76
 - BIND_FLAGS, 70
 - bfmap, 76
 - breakpoint, 76
 - breakpoint_t, 75
 - bsspolute, 77
 - cplus_demangle, 77
 - DEFAULT_LEARN_FILE, 70
 - DEFAULT_SCRIPT, 70
 - DMGL_ANSI, 70
 - DMGL_ARM, 70
 - DMGL_PARAMS, 70
 - default_poison, 70
 - disable_aslr, 77
 - disable_core, 77
 - do_loadlib, 77
 - ELF32_ST_BIND, 70
 - ELF32_ST_INFO, 70
 - ELF32_ST_TYPE, 70
 - ELF64_ST_BIND, 70
 - ELF64_ST_INFO, 71
 - ELF64_ST_TYPE, 71
 - Elf_Dyn, 71
 - Elf_Ehdr, 71
 - Elf_Phdr, 71
 - Elf_Shdr, 71
 - Elf_Sym, 71
 - empty_phdrs, 77
 - empty_shdrs, 77
 - enable_aslr, 77
 - enable_core, 77
 - entrypoints, 78
 - eps_t, 75
 - execlib, 78
 - FAULT_EXEC, 71
 - FAULT_READ, 71
 - FAULT_WRITE, 71
 - gencore, 78
 - getcharbuf, 78
 - getsize, 78
 - grep, 78
 - grepptr, 78
 - HPERMSMAX, 71
 - headers, 78
 - help, 78
 - hexdump, 78
 - hollywood, 79
 - info, 79
 - LINES_MAX, 71
 - libcall, 79
 - loadbin, 80
 - ltrace, 80
 - luaL_reg, 72
 - MAX_SIGNALS, 72
 - MIN_BIN_SIZE, 72
 - MY_CPU, 72
 - man, 80
 - map, 80
 - newarray, 80
 - PROC_ASLR_PATH, 72
 - phdrs, 80
 - preload_t, 75

- print_functions, 80
- print_libs, 80
- print_objects, 80
- print_phdrs, 80
- print_shdrs, 81
- print_symbols, 81
- print_version, 81
- priv_memcpy, 81
- priv_strcat, 81
- priv_strcpy, 81
- procmap_lua, 81
- prototypes, 81
- ralloc, 81
- range_t, 75
- rawmemaddr, 82
- rawmemread, 82
- rawmemstr, 82
- rawmemstrlen, 82
- rawmemusage, 82
- rawmemwrite, 82
- rdnum, 82
- rdstr, 82
- read_arg, 72
- read_arg1, 72
- read_arg2, 73
- read_arg3, 73
- read_arg4, 73
- reload_elfs, 83
- rescan, 83
- rtrace, 83
- SHELL_HISTORY_NAME, 74
- SKIP_BOTTOM, 74
- SKIP_INIT, 74
- STB_GLOBAL, 74
- STB_GNU_SECONDARY, 74
- STB_GNU_UNIQUE, 74
- STB_LOCAL, 74
- STB_WEAK, 74
- STT_COMMON, 75
- STT_FILE, 75
- STT_FUNC, 75
- STT_NOTYPE, 75
- STT_OBJECT, 75
- STT_SECTION, 75
- STT_TLS, 75
- script, 83
- script_t, 76
- sections_t, 76
- segment_add, 83
- segments_t, 76
- set_align_flag, 83
- set_branch_flag, 83
- set_trace_flag, 83
- setarray, 83
- setcharbuf, 83
- shdrs, 83
- sicode_strerror, 84
- signaltoname, 84
- singlebranch, 84
- singlestep, 84
- symbols_t, 76
- systrace, 84
- traceunaligned, 84
- tuple_t, 76
- USE_LUA, 75
- unrtrace, 84
- unset_align_flag, 84
- unset_branch_flag, 84
- unset_trace_flag, 84
- unsinglebranch, 84
- unsinglestep, 84
- unsystrace, 85
- untraceunaligned, 85
- unverbosetrace, 85
- usage, 85
- verbose, 85
- verbosetrace, 85
- wsh_getopt, 85
- wsh_init, 85
- wsh_loadlibs, 85
- wsh_run, 85
- wsh_t, 76
- xalloc, 85
- xfree, 86
- wsh/helper.c, 54
- wsh/include/colors.h, 56
- wsh/include/lauxlib.h, 57
- wsh/include/libwitch/helper.h, 63
- wsh/include/libwitch/mylaux.h, 63
- wsh/include/libwitch/sigs.h, 65
- wsh/include/libwitch/wsh.h, 66
- wsh/include/libwitch/wsh_functions.h, 86
- wsh/include/linenoise.h, 87
- wsh/include/longjmp.h, 88
- wsh/include/lua.h, 89
- wsh/include/luacnf.h, 105
- wsh/include/lualib.h, 111
- wsh/wsh.c, 113
- wsh/wshmain.c, 137
- wsh_functions.h
 - default_options, 86
 - exposed, 86
 - global_xalloc, 86
 - lua_blacklist, 86
 - lua_default_functions, 87
 - ranges, 87
- wsh_getopt
 - wsh.c, 134
 - wsh.h, 85
- wsh_init
 - wsh.c, 134
 - wsh.h, 85
- wsh_loadlibs
 - wsh.c, 134
 - wsh.h, 85
- wsh_print_version

- wsh.c, 134
- wsh_run
 - wsh.c, 134
 - wsh.h, 85
- wsh_t, 30
 - bp_array, 31
 - bp_num, 31
 - bp_points, 31
 - btcaller, 31
 - eps, 31
 - errcontext, 31
 - faultaddr, 31
 - firstcontext, 32
 - firsterrno, 32
 - firstsicode, 32
 - firstsignal, 32
 - globalsignals, 32
 - interrupted, 32
 - is_stdinscript, 32
 - L, 32
 - learnfile, 32
 - learnlog, 32
 - longjmp_ptr, 32
 - longjmp_ptr_high, 32
 - longjmp_ptr_high_cnt, 33
 - mainhandle, 33
 - opt_argc, 33
 - opt_argv, 33
 - opt_hollywood, 33
 - opt_rescan, 33
 - opt_verbose, 33
 - opt_verbosetrace, 33
 - phdrs, 33
 - pltgot, 33
 - pltz, 33
 - preload, 33
 - reason, 34
 - script_argnum, 34
 - script_args, 34
 - scriptfile, 34
 - scriptname, 34
 - scripts, 34
 - shdrs, 34
 - sigbus_count, 34
 - sigbus_hash, 34
 - singlebranch_count, 34
 - singlebranch_hash, 34
 - singlestep_count, 34
 - singlestep_hash, 35
 - symbols, 35
 - totsignals, 35
 - trace_rtrace, 35
 - trace_singlebranch, 35
 - trace_singlestep, 35
 - trace_strace, 35
 - trace_unaligned, 35
 - wsh.h, 76
- wsh_usage
 - wsh.c, 134
- wshmain.c
 - main, 137
 - wsh, 137
- xalloc
 - wsh.c, 135
 - wsh.h, 85
- xfree
 - wsh.c, 135
 - wsh.h, 86
- YELLOW
 - colors.h, 57
- zfirst
 - helper.c, 55
 - helper.h, 63