The Witchcraft Compiler Collection

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

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

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 263 of file wcc.c.

3.2.2 Field Documentation

3.2.2.1 bfd∗ ctx_t::abfd

Definition at line 278 of file wcc.c.

3.2.2.2 unsigned int ctx_t::archsz

Definition at line 269 of file wcc.c.

3.2.2.3 unsigned int ctx_t::base_address

Definition at line 282 of file wcc.c.

3.2.2.4 char∗ ctx_t::binname

Internal options

Definition at line 268 of file wcc.c.

3.2.2.5 unsigned int ctx_t::corefile

Definition at line 279 of file wcc.c.

3.2.2.6 int ctx_t::fdout

Definition at line 277 of file wcc.c.

3.2.2.7 unsigned int ctx_t::has_relativerelocations

Definition at line 292 of file wcc.c.

3.2.2.8 mseg_t∗ ctx_t::mphdrs

Definition at line 289 of file wcc.c.

3.2.2.9 unsigned int ctx_t::mphnum

Definition at line 290 of file wcc.c.

3.2.2.10 msec_t∗ ctx_t::mshdrs

Definition at line 285 of file wcc.c.

3.2.2.11 unsigned int ctx_t::mshnum

Definition at line 286 of file wcc.c.

3.2.2.12 unsigned int ctx_t::opt_arch

Definition at line 298 of file wcc.c.

3.2.2.13 unsigned int ctx_t::opt_asmdebug

Definition at line 311 of file wcc.c.

3.2.2.14 char∗ ctx_t::opt_binname

User options

Definition at line 296 of file wcc.c.

3.2.2.15 unsigned int ctx_t::opt_core

Definition at line 304 of file wcc.c.

3.2.2.16 unsigned int ctx_t::opt_debug

Definition at line 310 of file wcc.c.

3.2.2.17 unsigned long int ctx_t::opt_entrypoint

Definition at line 307 of file wcc.c.

3.2.2.18 unsigned int ctx_t::opt_exec

Definition at line 303 of file wcc.c.

3.2.2.19 unsigned int ctx_t::opt_flags

Definition at line 312 of file wcc.c.

3.2.2.20 char∗ ctx_t::opt_interp

Definition at line 297 of file wcc.c.

3.2.2.21 unsigned int ctx_t::opt_original

Definition at line 309 of file wcc.c.

3.2.2.22 unsigned char ctx_t::opt_poison

Definition at line 308 of file wcc.c.

3.2.2.23 unsigned int ctx_t::opt_reloc

Definition at line 300 of file wcc.c.

3.2.2.24 unsigned int ctx_t::opt_shared

Definition at line 305 of file wcc.c.

3.2.2.25 unsigned int ctx_t::opt_sstrip

Definition at line 302 of file wcc.c.

3.2.2.26 unsigned int ctx_t::opt_static

Definition at line 299 of file wcc.c.

3.2.2.27 unsigned int ctx_t::opt_strip

Definition at line 301 of file wcc.c.

3.2.2.28 unsigned int ctx_t::opt_verbose

Definition at line 306 of file wcc.c.

3.2.2.29 unsigned int ctx_t::phnum

Definition at line 271 of file wcc.c.

3.2.2.30 unsigned int ctx_t::shnum

Definition at line 270 of file wcc.c.

3.2.2.31 unsigned int ctx_t::start_phdrs

Definition at line 276 of file wcc.c.

3.2.2.32 unsigned int ctx_t::start_shdrs

Definition at line 275 of file wcc.c.

3.2.2.33 char∗ ctx_t::strndx

Definition at line 272 of file wcc.c.

3.2.2.34 unsigned int ctx_t::strndx_index

Definition at line 274 of file wcc.c.

3.2.2.35 unsigned int ctx_t::strndx_len

Definition at line 273 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_Rela ∗ r
- int rtype
- unsigned int sindex

3.5.1 Detailed Description

Definition at line 2769 of file wcc.c.

3.5.2 Field Documentation

3.5.2.1 Elf_Rela∗ gimport_t::r

Definition at line 2772 of file wcc.c.

3.5.2.2 int gimport_t::rtype

Definition at line 2773 of file wcc.c.

3.5.2.3 msec_t∗ gimport_t::sec

Definition at line 2771 of file wcc.c.

3.5.2.4 unsigned int gimport_t::sindex

Definition at line 2774 of file wcc.c.

3.5.2.5 char∗ gimport_t::sname

Definition at line 2770 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 luaL_Stream::closef

Definition at line 187 of file lauxlib.h.

3.13.2.2 FILE∗ luaL_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 228 of file wcc.c.

3.14.2 Field Documentation

3.14.2.1 unsigned char∗ msec_t::data

Definition at line 231 of file wcc.c.

3.14.2.2 unsigned int msec_t::flags

Definition at line 233 of file wcc.c.

3.14.2.3 unsigned long int msec_t::len

Definition at line 230 of file wcc.c.

3.14.2.4 char∗ msec_t::name

Definition at line 229 of file wcc.c.

3.14.2.5 struct msec_t∗ msec_t::next

Definition at line 239 of file wcc.c.

3.14.2.6 char∗ msec_t::outoffset

Definition at line 232 of file wcc.c.

3.14.2.7 struct msec_t∗ msec_t::prev

Definition at line 238 of file wcc.c.

3.14.2.8 asection∗ msec_t::s_bfd

Definition at line 235 of file wcc.c.

3.14.2.9 Elf_Shdr∗ msec_t::s_elf

Definition at line 236 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 247 of file wcc.c.

3.15.2 Field Documentation

3.15.2.1 struct msec_t∗ mseg_t::next

Definition at line 258 of file wcc.c.

3.15.2.2 Elf_Xword mseg_t::p_align

Definition at line 255 of file wcc.c.

3.15.2.3 Elf_Xword mseg_t::p_filesz

Definition at line 253 of file wcc.c.

3.15.2.4 Elf_Word mseg_t::p_flags

Definition at line 249 of file wcc.c.

3.15.2.5 Elf_Xword mseg_t::p_memsz

Definition at line 254 of file wcc.c.

3.15.2.6 Elf_Off mseg_t::p_offset

Definition at line 250 of file wcc.c.

3.15.2.7 Elf_Addr mseg_t::p_paddr

Definition at line 252 of file wcc.c.

3.15.2.8 Elf_Word mseg_t::p_type

Definition at line 248 of file wcc.c.

3.15.2.9 Elf_Addr mseg_t::p_vaddr

Definition at line 251 of file wcc.c.

3.15.2.10 struct msec_t∗ mseg_t::prev

Definition at line 257 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 404 of file wcc.c.

3.23.2 Field Documentation

3.23.2.1 int symaddr::addr

Definition at line 407 of file wcc.c.

3.23.2.2 char∗ symaddr::name

Definition at line 406 of file wcc.c.

3.23.2.3 struct **symaddr**∗ symaddr::next

Definition at line 405 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 totsignals
- 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::pltsz

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 <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(!strncmp(name, x, strlen(x)))
- #define elis(x) else if(!strncmp(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\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\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)
- mseg_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 globalrecloclen = 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`

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 133 of file wcc.c.

4.1.1.4 #define DEFAULT_STRNDX_SIZE 4096

Definition at line 71 of file wcc.c.

4.1.1.5 #define Elf_Addr Elf32_Addr

Definition at line 117 of file wcc.c.

4.1.1.6 #define Elf_Ehdr Elf32_Ehdr

Definition at line 114 of file wcc.c.

4.1.1.7 #define Elf_Off Elf32_Off

Definition at line 130 of file wcc.c.

4.1.1.8 #define Elf_Phdr Elf32_Phdr

Definition at line 127 of file wcc.c.

4.1.1.9 #define ELF_R_INFO ELF32_R_INFO

Definition at line 126 of file wcc.c.

4.1.1.10 #define ELF_R_SYM ELF32_R_SYM

Definition at line 124 of file wcc.c.

4.1.1.11 #define ELF_R_TYPE ELF32_R_TYPE

Definition at line 125 of file wcc.c.

4.1.1.12 #define Elf_Rel Elf32_Rel

Definition at line 122 of file wcc.c.

4.1.1.13 #define Elf_Rela Elf32_Rela

Definition at line 123 of file wcc.c.

4.1.1.14 #define Elf_Section Elf32_Half

Definition at line 119 of file wcc.c.

4.1.1.15 #define Elf_Shdr Elf32_Shdr

Definition at line 115 of file wcc.c.

4.1.1.16 #define ELF_ST_BIND ELF32_ST_BIND

Definition at line 120 of file wcc.c.

4.1.1.17 #define ELF_ST_TYPE ELF32_ST_TYPE

Definition at line 121 of file wcc.c.

4.1.1.18 #define Elf_Sword Elf64_Sword

Definition at line 118 of file wcc.c.

4.1.1.19 #define Elf_Sym Elf32_Sym

Definition at line 116 of file wcc.c.

4.1.1.20 #define Elf_Word Elf32_Word

Definition at line 129 of file wcc.c.

4.1.1.21 #define Elf_Xword Elf32_Xword

Definition at line 128 of file wcc.c.

4.1.1.22 #define ELFCLASS ELFCLASS32

Definition at line 131 of file wcc.c.

4.1.1.23 #define ELFMACHINE EM_386

Definition at line 132 of file wcc.c.

4.1.1.24 #define elis(*x*) else if(!strncmp(name, x, strlen(x)))

Definition at line 80 of file wcc.c.

4.1.1.25 #define EXTRA_CREATED_SECTIONS 4

Definition at line 84 of file wcc.c.

4.1.1.26 #define FLAG_BSS 1

Definition at line 74 of file wcc.c.

4.1.1.27 #define FLAG_NOBIT 2

Definition at line 75 of file wcc.c.

4.1.1.28 #define FLAG_NOWRITE 4

Definition at line 76 of file wcc.c.

4.1.1.29 #define FLAG_TEXT 8

Definition at line 77 of file wcc.c.

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

Definition at line 79 of file wcc.c.

4.1.1.31 #define MAXPADLEN 20

Definition at line 82 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\x00\x00"

Definition at line 138 of file wcc.c.

4.1.1.33 #define RELOC_MODE RELOC_X86_32

Definition at line 134 of file wcc.c.

4.1.1.34 #define RELOC_X86_32 2

Definition at line 88 of file wcc.c.

4.1.1.35 #define RELOC_X86_64 1

Definition at line 87 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 560 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 422 of file wcc.c.

4.1.3.3 int adjust_baseaddress (ctx_t ∗ *ctx*)

Definition at line 1107 of file wcc.c.

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

Return a section alignment from its name

Definition at line 880 of file wcc.c.

4.1.3.5 mseg_t∗ alloc_phdr (msec_t ∗ *ms*)

Allocate Phdr

Definition at line 1008 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 3394 of file wcc.c.

4.1.3.7 int append_reloc (Elf_Rela ∗ *r*)

Definition at line 2739 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 1775 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 1754 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 2817 of file wcc.c.

4.1.3.11 int copy_body (ctx_t ∗ *ctx*)

Write sections to disk

Definition at line 2458 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 2499 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 1031 of file wcc.c.

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

Definition at line 3846 of file wcc.c.

4.1.3.15 ctx_t∗ ctx_init (void)

Initialize a reversing context Set default values

Definition at line 3774 of file wcc.c.

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

Definition at line 3826 of file wcc.c.

4.1.3.17 int entszfromname (const char ∗ *name*)

Return section entry size from name

Definition at line 681 of file wcc.c.

4.1.3.18 int fixup_strtab_and_symtab (ctx_t ∗ *ctx*)

check if name is in blacklist

Definition at line 1637 of file wcc.c.

4.1.3.19 int fixup_symtab_section_index (ctx_t ∗ *ctx*)

Definition at line 2719 of file wcc.c.

4.1.3.20 int fixup_text (ctx_t * *ctx*)

Definition at line 1693 of file wcc.c.

4.1.3.21 int flags_from_name (const char * *name*)

Return section flags from its name

Definition at line 2485 of file wcc.c.

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

Simple hexdump routine

Definition at line 2345 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 842 of file wcc.c.

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

Definition at line 3288 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 3596 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 819 of file wcc.c.

4.1.3.27 int load_binary (ctx_t * *ctx*)

Load a binary using bfd

Definition at line 2471 of file wcc.c.

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

Application Entry Point

Definition at line 4013 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 696 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 1072 of file wcc.c.

4.1.3.31 msec_t* mk_section (void)

Definition at line 1329 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 2372 of file wcc.c.

4.1.3.33 int open_target (ctx_t ∗ *ctx*)

Open destination binary

Definition at line 2404 of file wcc.c.

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

Definition at line 2700 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 942 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 981 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 970 of file wcc.c.

4.1.3.38 int print_bfd_sections (ctx_t ∗ *ctx*)

Display BFD memory sections

Definition at line 2287 of file wcc.c.

4.1.3.39 int print_maps (void)

Print content of /proc/pid/maps

Definition at line 3762 of file wcc.c.

4.1.3.40 int print_msec (ctx_t ∗ *ctx*)

Display sections

Definition at line 2632 of file wcc.c.

4.1.3.41 int print_version (void)

Definition at line 3820 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 380 of file wcc.c.

4.1.3.43 unsigned int ptype_from_section (msec_t * *ms*)

Return Segment ptype

Definition at line 895 of file wcc.c.

4.1.3.44 int rd_sections (ctx_t * *ctx*)

Read sections from input binary

Definition at line 2649 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 573 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 3442 of file wcc.c.

4.1.3.47 char* reloc_htype (int *thetype*)

Definition at line 1534 of file wcc.c.

4.1.3.48 char* reloc_htype_x86_32 (int *thetype*)

Definition at line 1473 of file wcc.c.

4.1.3.49 char* reloc_htype_x86_64 (int *thetype*)

Definition at line 1390 of file wcc.c.

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

Suppress a given section

Definition at line 3532 of file wcc.c.

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

Definition at line 2662 of file wcc.c.

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

Definition at line 2680 of file wcc.c.

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

Definition at line 2780 of file wcc.c.

4.1.3.54 int save_reloc (ctx_t ∗ *ctx*, Elf_Rela ∗ *r*, unsigned int *sindex*, int *has_addend*)

Convert relocation depending on type and source section

Definition at line 2834 of file wcc.c.

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

Definition at line 790 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 752 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 769 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 719 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 735 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 704 of file wcc.c.

4.1.3.61 int sort_phdrs (ctx_t ∗ *ctx*)

Reorganise Program Headers : sort by p_offset

Definition at line 990 of file wcc.c.

4.1.3.62 int sort_phdrs_premerge (ctx_t ∗ *ctx*)

Helper sort routine for ELF Phdrs

Definition at line 999 of file wcc.c.

4.1.3.63 int strip_binary_reloc (ctx_t * *ctx*)

Strip binary relocation data

Definition at line 3559 of file wcc.c.

4.1.3.64 int typefromname (const char * *name*)

Return a section type from its name

Definition at line 865 of file wcc.c.

4.1.3.65 int usage (char * *name*)

Definition at line 3794 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 142 of file wcc.c.

4.1.4.2 char* blnames[ ]

Definition at line 157 of file wcc.c.

4.1.4.3 unsigned long int datavma = 0

Definition at line 357 of file wcc.c.

4.1.4.4 unsigned int deltastrtab = 0

Definition at line 140 of file wcc.c.

4.1.4.5 gimport_t** gimports = 0

Definition at line 2777 of file wcc.c.

4.1.4.6 unsigned int gimportslen = 0

Definition at line 2778 of file wcc.c.

4.1.4.7 char∗ globalreloc = 0

Definition at line 347 of file wcc.c.

4.1.4.8 unsigned int globalreloclen = 0

Definition at line 348 of file wcc.c.

4.1.4.9 unsigned int globalrelocoffset = 0

Definition at line 349 of file wcc.c.

4.1.4.10 char∗ globalstrtab = 0

Definition at line 341 of file wcc.c.

4.1.4.11 unsigned int globalstrtablen = 0

Definition at line 342 of file wcc.c.

4.1.4.12 unsigned int globalstrtableoffset = 0

Definition at line 343 of file wcc.c.

4.1.4.13 unsigned int globalsymindex = 0

Definition at line 345 of file wcc.c.

4.1.4.14 char∗ globalsymtab = 0

Globals

Definition at line 337 of file wcc.c.

4.1.4.15 int globalsymtablen = 0

Definition at line 338 of file wcc.c.

4.1.4.16 unsigned int globalsymtableoffset = 0

Definition at line 339 of file wcc.c.

4.1.4.17 unsigned long int maxdata = 0

Definition at line 356 of file wcc.c.

4.1.4.18 unsigned int maxnewsec = 0

Definition at line 139 of file wcc.c.

4.1.4.19 unsigned int maxoldsec = 0

Definition at line 139 of file wcc.c.

4.1.4.20 unsigned long int maxtext = 0

Definition at line 352 of file wcc.c.

4.1.4.21 unsigned long int mindata = -1

Definition at line 355 of file wcc.c.

4.1.4.22 unsigned long int mintext = -1

Definition at line 351 of file wcc.c.

4.1.4.23 unsigned long int orig_sz = 0

Definition at line 360 of file wcc.c.

4.1.4.24 unsigned long int orig_text = 0

Definition at line 359 of file wcc.c.

4.1.4.25 struct symaddr * symaddrs

4.1.4.26 unsigned long int textvma = 0

Definition at line 353 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) 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, extramsg) ((void)((cond) || luaL_argerror(L, (arg), (extramsg))))
- #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)
- #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 ∗extramsg)
- 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, extramsg*) ((void)((cond) || luaL_argerror(L, (arg), (extramsg))))

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 ∗ *extramsg*)

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, extramsg) ((void)((cond) || luaL_argerror(L, (arg), (extramsg))))
- #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*, *extramsg*) ((void)((cond) || **luaL_argerror**(L, (arg), (extramsg))))

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 <lauxlib.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 HPERMSMAX 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 unverbosetrace (lua_State ∗L)
- void verbosetrace (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 allocharbuf (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 procmap_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 destionation’s page

Backup byte at destination

Write Breakpoint

Save breakpoint informations

Definition at line 4218 of file wsh.c.

4.9.3.5 int bsspolute (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 emmited 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 librairies, 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 3820 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 procmap_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 4827 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 4753 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 4791 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 4839 of file wsh.c.

4.9.3.50 int rawmemusage (lua_State ∗ *L*)

Display memory usage.

Definition at line 4805 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 4772 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 ELFs 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 ∗ *ptype,* 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 unrtrace (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 3794 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∗ __progname_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_trow_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 (-LUAI_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_isfunction(L, n) (lua_type(L, (n)) == LUA_TFUNCTION)
- #define lua_istable(L, n) (lua_type(L, (n)) == LUA_TTABLE)
- #define lua_islightuserdata(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_newstate (lua_Alloc f, void ∗ud)
- LUA_API void() lua_close (lua_State ∗L)
- LUA_API lua_State ∗() lua_newthread (lua_State ∗L)
- LUA_API lua_CFunction() lua_atpanic (lua_State ∗L, lua_CFunction panicf)
- LUA_API const lua_Number ∗() lua_version (lua_State ∗L)
- LUA_API int() lua_absindex (lua_State ∗L, int idx)
- LUA_API int() lua_gettop (lua_State ∗L)
- LUA_API void() lua_settop (lua_State ∗L, int idx)
- LUA_API void() lua_pushvalue (lua_State ∗L, int idx)
- LUA_API void() lua_rotate (lua_State ∗L, int idx, int n)
- LUA_API void() lua_copy (lua_State ∗L, int fromidx, int toidx)

- 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_GCSETSTEPMUL 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_getextraspace(*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_islightuserdata(*L, n*) (lua_type(L, (n)) == LUA_TLIGHTUSERDATA)

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 (-LUAI_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_iscfunction (**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/luaconf.h File Reference

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

Macros

- #define LUAI_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_EXTRASPACE (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 luaconf.h.

4.14.1.3 #define l_mathop(*op*) op

Definition at line 484 of file luaconf.h.

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

Definition at line 591 of file luaconf.h.

4.14.1.5 #define LUA_API extern

Definition at line 242 of file luaconf.h.

4.14.1.6 #define LUA_CDIR LUA_ROOT "lib/lua/" LUA_VDIR "/"

Definition at line 193 of file luaconf.h.

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

Definition at line 198 of file luaconf.h.

4.14.1.8 #define LUA_DIRSEP "/"

Definition at line 211 of file luaconf.h.

4.14.1.9 #define LUA_EXTRASPACE (sizeof(void ∗))

Definition at line 717 of file luaconf.h.

4.14.1.10 #define LUA_FLOAT_DOUBLE 2

Definition at line 115 of file luaconf.h.

4.14.1.11 #define LUA_FLOAT_FLOAT 1

Definition at line 114 of file luaconf.h.

4.14.1.12 #define LUA_FLOAT_LONGDOUBLE 3

Definition at line 116 of file luaconf.h.

4.14.1.13 #define LUA_FLOAT_TYPE LUA_FLOAT_DOUBLE

Definition at line 147 of file luaconf.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 luaconf.h.

4.14.1.27 #define LUA_NUMBER_FMT "%.14g"

Definition at line 482 of file luaconf.h.

4.14.1.28 #define LUA_NUMBER_FRMLEN ""

Definition at line 481 of file luaconf.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 luaconf.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 luaconf.h.

4.14.1.31 #define LUA_QL(*x*) "'" x "'"

Definition at line 749 of file luaconf.h.

4.14.1.32 #define LUA_QS LUA_QL("%s")

Definition at line 750 of file luaconf.h.

4.14.1.33 #define LUA_ROOT "/usr/local/"

Definition at line 191 of file luaconf.h.

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

Definition at line 486 of file luaconf.h.

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

Definition at line 604 of file luaconf.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_procmap (unsigned int pid)
- int procmap_lua (void)
- int execlib (lua_State ∗L)
- int traceback (lua_State ∗L)
- void print_backtrace (void)
- char ∗ sicodetoname (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 verbosetrace (lua_State ∗L)
- void unverbosetrace (lua_State ∗L)
- void singlebranch (lua_State ∗L)
- void unsinglebranch (lua_State ∗L)
- int grepptr (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 ELFMACHINE 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

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 destionation's page

Backup byte at destination

Write Breakpoint

Save breakpoint informations

Definition at line 4218 of file wsh.c.

4.16.3.11 int bsspolute (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 uppon '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 emmited 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 librairies, 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_procmap (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 procmap_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 4827 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 4753 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 4791 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 4839 of file wsh.c.

4.16.3.83 int rawmemusage (lua_State ∗ *L*)

Display memory usage.

Definition at line 4805 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 4772 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 ELFs 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 blackslisted, 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∗ sicodetoname (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 4714 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 4723 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 unmaped] [mapped][OURPTR, size] [page unmaped]

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 uppon shell termination."},
        {"exec", "<command>", "Run <command> via the system() library call.", "", "None. Returns uppon
       <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 ommitted, 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"},
        {"procmap", "", "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 pagers.
       [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 pagers.
       [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

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

Main wsh context

Definition at line 37 of file wshmain.c.

Index