NAME

dwarf - access debugging information in object files

LIBRARY

DWARF Access Library (libdwarf, -ldwarf)

SYNOPSIS

#include <libdwarf.h>

DESCRIPTION

The DWARF Access Library (libdwarf, -ldwarf) provides functions that allow an application to read and write debugging information in object files. The format of debugging information accessible through this API is defined by the DWARF standard, see dwarf(4).

The DWARF(3) API has two parts:

- A consumer API set allows applications to read existing debug information in a program object. The functions that comprise the DWARF consumer API are described in the section *DWARF Consumer API* below.

- A producer API set that allows applications to add debug information to a program object. The functions that comprise the DWARF producer API are described in the section *DWARF Producer API* below.

Each function referenced below is further described in its own manual page.

Namespace use

The DWARF library uses the following prefixes:

DWARF_*
: Used for error numbers and constants.

DW_*
: Used for constants.

Dwarf_*
: Used for types.

dwarf_*
: Used for functions and macros that make up the API.

Data Types

The DWARF(3) API uses the following data types:

Dwarf_Abbrev
: Describes DWARF abbreviations.

Dwarf_Addr
: A program address in the target object.

Dwarf_Arange Describes address ranges.

Dwarf_Attribute, Dwarf_P_Attribute
Describes attributes of debugging information entries.

Dwarf_Bool Used for boolean states.

Dwarf_Cie, Dwarf_P_Cie
Describes call information that is common to several frames.

Dwarf_Debug, Dwarf_P_Debug
An opaque type describing a debug context.

Dwarf_Die, Dwarf_P_Die
A debugging information entry.

Dwarf_Fde, Dwarf_P_Fde
A frame descriptor.

Dwarf_Func A descriptor representing a function.

Dwarf_Global A descriptor representing a global name.

Dwarf_Half A 16-bit wide unsigned numeric type.

Dwarf_Handler A pointer to an error handling function.

Dwarf_Line A descriptor for a source line.

Dwarf_Off An unsigned file offset.

Dwarf_P_Expr A descriptor for a location expression.

Dwarf_Ptr A virtual address used by an application.

Dwarf_Signed A 64-bit wide signed numeric type.

Dwarf_Small An 8-bit wide unsigned numeric type.

Dwarf_Type A descriptor representing a user-specified type.

Dwarf_Unsigned A 64-bit wide unsigned numeric type.

Dwarf_Var A descriptor representing a static variable.

Dwarf_Weak A descriptor representing a weak name.

Error Handling

Library functions that encounter an error will return with a value other than DW_DLV_OK.

The DWARF Access Library (libdwarf, -ldwarf) allows applications to specify three levels of error handling:

1. Most library functions take a parameter of type *Dwarf_Error* that specifies a location to store an error descriptor in case of an error. If an error occurs during the execution on an API, and if this parameter is non-NULL, then an error descriptor is written to the location specified.
2. Otherwise, if the error parameter was NULL, but if an error handler was defined for the debug context in use using dwarf_init(3) or dwarf_seterrhand(3), then the library will invoke the specified error handler with an error descriptor as argument.
3. Otherwise, if a library wide error handler was specified using dwarf_seterrhand(3), it is called.

Error descriptors may be used with dwarf_errmsg(3) or dwarf_errno(3).

The DWARF Consumer API

The DWARF consumer API permits applications to read DWARF information in an object file.

The major functional groups of functions in the consumer API are listed below.

Abbreviations

dwarf_get_abbrev()
: Retrieve abbreviation information at a given offset.

dwarf_get_abbrev_children_flag()
: Check if an abbreviation has child elements.

dwarf_get_abbrev_code()
: Retrieve the abbreviation code for an abbreviation entry descriptor.

dwarf_get_abbrev_entry()
: Retrieve abbreviation information for an abbreviation entry descriptor.

dwarf_get_abbrev_tag()
: Retrieve the tag for an abbreviation entry.

Addresses

dwarf_get_address_size()
: Return the number of bytes needed to represent an address.

dwarf_get_arange()
: Search for an address range descriptor covering an address.

dwarf_get_arange_cu_header_offset()
: Retrieve the offsets associated with an address range descriptor.

dwarf_get_arange_info()
: Extract address range information from a descriptor.

dwarf_get_aranges()
: Retrieve program address space mappings.

dwarf_get_cu_die_offset()
: Retrieve the offset associated with a compilation unit for an address range descriptor.

dwarf_get_ranges(), **dwarf_get_ranges_a**()
: Retrieve information about non-contiguous address ranges for a debugging information entry.

Attributes

dwarf_arrayorder()
: Retrieve the value of a DW_AT_ordering attribute.

dwarf_attr()
: Retrieve an attribute descriptor.

dwarf_attrlist()

Retrieve attribute descriptors for a debugging information entry.

dwarf_attroffset()
Retrieve the section-relative offset of an attribute descriptor.

dwarf_attrval_flag()
Retrieve a DW_AT_FORM_flag value.

dwarf_attrval_signed()
Retrieve an attribute's value as a signed integral quantity.

dwarf_attrval_string()
Retrieve an attribute's value as a NUL-terminated string.

dwarf_attrval_unsigned()
Retrieve an attribute's value as an unsigned integral quantity.

dwarf_bitoffset(),
Retrieve the value of a DW_AT_bit_offset attribute.

dwarf_bitsize(),
Retrieve the value of a DW_AT_bit_size attribute.

dwarf_bytesize()
Retrieve the value of a DW_AT_byte_size attribute.

dwarf_formaddr()
Return the value of an ADDRESS-class attribute.

dwarf_formblock()
Return the value of a BLOCK-class attribute

dwarf_formexprloc()
Return information about a location expression.

dwarf_formflag()
Retrieve information about a BOOLEAN-class attribute.

dwarf_formref(), **dwarf_global_formref**()
Retrieve offsets for REFERENCE-class attributes.

dwarf_formsdata(), **dwarf_formudata**()
Retrieve the value of a CONSTANT-class attribute.

dwarf_formsig8()
Return the type signature for a DWARF type.

dwarf_formstring()
Retrieve information about a STRING-class attribute.

dwarf_get_form_class()
Retrieve the form class for an attribute.

dwarf_hasattr()
Check for the presence of an attribute.

dwarf_hasform()
Check if an attribute has the given form.

dwarf_whatattr()

Retrieve the attribute code for an attribute.

dwarf_whatform(), **dwarf_whatform_direct**()

Retrieve the form of an attribute.

Call Information Entries and Frame Descriptor Entries

dwarf_get_cie_index()

Retrieve the index for a CIE descriptor.

dwarf_get_cie_info()

Retrieve information from a CIE descriptor.

dwarf_get_cie_of_fde()

Retrieve a CIE descriptor.

dwarf_get_fde_at_pc()

Retrieve an FDE descriptor for an address.

dwarf_get_fde_info_for_all_regs()

Retrieve register rule row.

dwarf_get_fde_info_for_all_regs3()

Retrieve register rule row (revised API).

dwarf_get_fde_info_for_cfa_reg3()

Retrieve a CFA register rule.

dwarf_get_fde_info_for_reg()

Retrieve a register rule.

dwarf_get_fde_info_for_reg3()

Retrieve a register rule (revised API).

dwarf_get_fde_instr_bytes()

Retrieve instructions from an FDE descriptor.

dwarf_get_fde_list(), **dwarf_get_fde_list_eh**()

Retrieve frame information.

dwarf_get_fde_n()

Retrieve an FDE descriptor.

dwarf_get_fde_range()

Retrieve range information from an FDE descriptor.

Compilation Units

dwarf_get_cu_die_offset_given_cu_header_offset(),
dwarf_get_cu_die_offset_given_cu_header_offset_b()

Retrieve the offset of the debugging information entry for a compilation or type unit.

dwarf_next_cu_header(), **dwarf_next_cu_header_b**(), **dwarf_next_cu_header_c**()

Step through compilation units in a debug context.

Debugging Information Entries

dwarf_child()

Returns the child of a debugging information entry.

dwarf_die_abbrev_code()

Returns the abbreviation code for a debugging information entry.

dwarf_die_CU_offset(), **dwarf_die_CU_offset_range**()
Retrieve offsets and lengths for a compilation unit.

dwarf_diename()
Returns the DW_AT_name attribute for a debugging information entry.

dwarf_dieoffset()
Retrieves the offset for a debugging information entry.

dwarf_get_die_infotypes_flag()
Indicate the originating section for a debugging information entry.

dwarf_highpc(), **dwarf_highpc_b**()
Return the highest PC value for a debugging information entry.

dwarf_lowpc()
Return the lowest PC value for a debugging information entry.

dwarf_offdie(), **dwarf_offdie_b**()
Retrieve a debugging information entry given an offset.

dwarf_siblingof(), **dwarf_siblingof_b**()
Retrieve the sibling descriptor for a debugging information entry.

dwarf_srclang()
Retrieve the source language attribute for a debugging information entry.

dwarf_tag()
Retrieve the tag for a debugging information entry.

Functions

dwarf_func_cu_offset()
Retrieves the offset for the compilation unit for a function.

dwarf_func_die_offset()
Retrieves the offset for the debugging information entry for a function.

dwarf_funcname()
Retrieves the name of a function.

dwarf_func_name_offsets()
Retrieve both the name and offsets for a function.

dwarf_get_funcs()
Retrieve information about static functions.

Globals

dwarf_get_globals()
Retrieve a list of globals.

dwarf_global_cu_offset()
Return the offset for compilation unit for a global.

dwarf_global_die_offset()
Return the offset for the debugging information entry for a global.

dwarf_global_name_offsets()

Return the name and offsets for a global.

dwarf_globname()

Return the name for a global.

Initialization and Finalization

Functions **dwarf_elf_init**() and **dwarf_init**() may be used for initialization. The function **dwarf_finish**() may be used to release resources.

The functions **dwarf_object_init**() and **dwarf_object_finish**() allow an application to specify alternate low-level file access routines.

Line Numbers

dwarf_lineaddr()

Retrieve the program address for a source line.

dwarf_linebeginstatement()

Check if a source line corresponds to the beginning of a statement.

dwarf_lineblock()

Check if a source line corresponds to the start of a basic block.

dwarf_lineendsequence()

Check if the source line corresponds to the end of a sequence of instructions.

dwarf_lineno()

Retrieve the line number for a line descriptor.

dwarf_lineoff()

Retrieve the column number for a line descriptor.

dwarf_linesrc()

Retrieve the source file for a line descriptor.

dwarf_line_srcfileno()

Retrieve the index of the source file for a line descriptor.

dwarf_srcfiles()

Retrieve source files for a compilation unit.

dwarf_srclines()

Return line number information for a compilation unit.

Location Lists

dwarf_get_loclist_entry()

Retrieve a location list entry.

dwarf_loclist(), **dwarf_loclist_n**()

Retrieve location expressions.

dwarf_loclist_from_expr(), **dwarf_loclist_from_expr_a**(), **dwarf_loclist_from_expr_b**()

Translate a location expression into a location descriptor.

Error Handling

dwarf_errmsg()

Retrieve a human-readable error message.

dwarf_errno()
Retrieve an error number from an error descriptor.
dwarf_seterrarg()
Set the argument passed to a callback error handler.
dwarf_seterrhand()
Set the callback handler to be called in case of an error.

Frame Handling

dwarf_expand_frame_instructions()
Translate frame instruction bytes.
dwarf_set_frame_cfa_value()
Set the CFA parameter for the internal register rule table.
dwarf_set_frame_rule_initial_value()
Set the initial value of the register rules in the internal register rule table.
dwarf_set_frame_rule_table_size()
Set the maximum number of columns in the register rule table.
dwarf_set_frame_same_value()
Set the register number representing the "same value" rule.
dwarf_set_frame_undefined_value()
Set the register number representing the "undefined" rule.

Macros

dwarf_find_macro_value_start()
Return the macro value part of a macro string.
dwarf_get_macro_details()
Retrieve macro information.

Memory Management

In the DWARF consumer API, the rules for memory management differ between functions. In some cases, the memory areas returned to the application by the library are freed by calling specific API functions. In others, the deallocation function **dwarf_dealloc**() suffices. The individual manual pages for the API's functions document the specific memory management rules to be followed.

The function **dwarf_dealloc**() is used to mark memory arenas as unused. Additionally, the following functions release specific types of DWARF resources: **dwarf_fde_cie_list_dealloc**(), **dwarf_funcs_dealloc**(), **dwarf_globals_dealloc**(), **dwarf_pubtypes_dealloc**(), **dwarf_ranges_dealloc**(), **dwarf_srclines_dealloc**(), **dwarf_types_dealloc**(), **dwarf_vars_dealloc**(), and **dwarf_weaks_dealloc**().

Symbol Constants

The following functions may be used to return symbolic names for DWARF constants: **dwarf_get_ACCESS_name**(), **dwarf_get_AT_name**(), **dwarf_get_ATE_name**(), **dwarf_get_CC_name**(), **dwarf_get_CFA_name**(), **dwarf_get_CHILDREN_name**(),

dwarf_get_DS_name(), **dwarf_get_DSC_name**(), **dwarf_get_EH_name**(), **dwarf_get_END_name**(), **dwarf_get_FORM_name**(), **dwarf_get_ID_name**(), **dwarf_get_INL_name**(), **dwarf_get_LANG_name**(), **dwarf_get_LNE_name**(), **dwarf_get_LNS_name**(), **dwarf_get_MACINFO_name**(), **dwarf_get_OP_name**(), **dwarf_get_ORD_name**(), **dwarf_get_TAG_name**(), **dwarf_get_VIRTUALITY_name**(), and **dwarf_get_VIS_name**().

Types

dwarf_get_pubtypes(), **dwarf_get_types**()
: Retrieve descriptors for user-defined types.

dwarf_next_types_section()
: Step through ".debug_types" sections in a debug context.

dwarf_pubtype_cu_offset(), **dwarf_type_cu_offset**()
: Return the offset for the compilation unit for a type.

dwarf_pubtype_die_offset(), **dwarf_type_die_offset**()
: Return the offset for the debugging information entry for a type.

dwarf_pubtypename(), **dwarf_typename**()
: Retrieve the name of a type.

dwarf_pubtype_name_offsets(), **dwarf_type_name_offsets**()
: Retrieve the name and offsets for a type.

Variables

dwarf_get_vars()
: Retrieve descriptors for static variables.

dwarf_var_cu_offset()
: Return the offset for the compilation unit for a variable.

dwarf_var_die_offset()
: Return the offset for the debugging information entry for a variable.

dwarf_varname()
: Retrieve the name of a variable.

dwarf_var_name_offsets()
: Retrieve the name and offsets for a variable.

Weak Symbols

dwarf_get_weaks()
: Retrieve information about weak symbols.

dwarf_weak_cu_offset()
: Return the offset for the compilation unit for a weak symbol.

dwarf_weak_die_offset()
: Return the offset for the debugging information entry for a weak symbol.

dwarf_weakname()
: Retrieve the name of a weak symbol.

dwarf_weak_name_offsets()

Retrieve the name and offsets for a weak symbol.

Miscellaneous

dwarf_get_elf()
Retrieve the ELF descriptor for a debug context, see elf(3).

dwarf_get_str()
Retrieve a NUL-terminated string from the DWARF string section.

dwarf_set_reloc_application()
Control whether relocations are to be handled by DWARF Access Library (libdwarf, -ldwarf).

The DWARF Producer API

The DWARF producer API permits applications to add DWARF information to an object file.

The major functional groups of functions in the producer API are listed below.

Attribute Management

The following functions are used to attach attributes to a debugging information entry: **dwarf_add_AT_comp_dir**(), **dwarf_add_AT_const_value_signedint**(), **dwarf_add_AT_const_value_string**(), **dwarf_add_AT_const_value_unsignedint**(), **dwarf_add_AT_dataref**(), **dwarf_add_AT_flag**(), **dwarf_add_AT_location_expr**(), **dwarf_add_AT_name**(), **dwarf_add_AT_producer**(), **dwarf_add_AT_ref_address**(), **dwarf_add_AT_reference**(), **dwarf_add_AT_signed_const**(), **dwarf_add_AT_string**(), **dwarf_add_AT_targ_address**(), **dwarf_add_AT_targ_address_b**() and **dwarf_add_AT_unsigned_const**().

Debugging Information Entry Management

dwarf_add_die_to_debug()
Set the root debugging information entry for a DWARF producer instance.

dwarf_die_link()
Links debugging information entries.

dwarf_new_die()
Allocate a new debugging information entry.

Initialization and Finalization

The functions **dwarf_producer_init**() and **dwarf_producer_init_b**() are used to initialize a producer instance.

When done, applications release resources using the function **dwarf_producer_finish**().

Relocations and Sections

dwarf_get_relocation_info()
Retrieve a relocation array from a producer instance.
dwarf_get_relocation_info_count()
Return the number of relocation arrays for a producer instance.
dwarf_get_section_bytes()
Retrieve the ELF byte stream for a section.
dwarf_reset_section_bytes()
Reset internal state for a producer instance.
dwarf_transform_to_disk_form()
Prepare byte streams for writing out.

Macros

dwarf_def_macro()
Add a macro definition.
dwarf_end_macro_file(), **dwarf_start_macro_file**()
Record macro file related information.
dwarf_undef_macro()
Note the removal of a macro definition.
dwarf_vendor_ext()
Enables storing macro information as specified in the DWARF standard.

Symbols, Expressions, Addresses and Offsets

dwarf_add_arange(), **dwarf_add_arange_b**()
Add address range information.
dwarf_add_directory_decl()
Add information about an include directory to a producer instance.
dwarf_add_fde_inst()
Add an operation to a frame descriptor entry.
dwarf_add_file_decl()
Add information about a source file to a producer instance.
dwarf_add_frame_cie()
Add call information to a frame descriptor.
dwarf_add_frame_fde(), **dwarf_add_frame_fde_b**()
Link a frame descriptor to a producer instance.
dwarf_add_funcname()
Add information about a function to a producer instance.
dwarf_add_line_entry()
Record mapping information between machine addresses and a source line.
dwarf_add_expr_addr(), **dwarf_add_expr_addr_b**()
Add a DW_OP_addr opcode to a location expression.

dwarf_add_expr_gen()
Add an operator to a location expression.
dwarf_add_pubname()
Add information about a global name to a producer instance.
dwarf_add_typename()
Add information about a type to a producer instance.
dwarf_add_varname()
Add information about a static variable to a producer instance.
dwarf_add_weakname()
Add information about a weak symbol to a producer instance.
dwarf_expr_current_offset()
Retrieve the current size of a location expression.
dwarf_expr_into_block()
Convert a location expression into a byte stream.
dwarf_fde_cfa_offset()
Append a DW_CFA_offset operation to a frame descriptor.
dwarf_lne_end_sequence(), **dwarf_lne_set_address**()
Note address ranges for source lines.
dwarf_new_expr()
Allocate a location expression descriptor.
dwarf_new_fde()
Allocate a frame descriptor.

Miscellaneous
The function **dwarf_producer_set_isa**() sets the instruction set architecture for the producer instance.

COMPATIBILITY

This implementation is believed to be source compatible with the SGI/GNU DWARF(3) library, version 20110113.

Known differences with the SGI/GNU library include:

- The memory management scheme used differs, in a backward-compatible way. See *Memory Management* above, for coding guidelines for portable applications.
- There is provision for setting a library-wide error handler in addition to the per-debug context handlers supported by the SGI/GNU API, see the subsection *Error Handling* above.

Extensions

The following APIs are extensions specific to this implementation:

- **dwarf_attroffset**()

- **dwarf_next_types_section**()
- **dwarf_producer_set_isa**()

SEE ALSO

elf(3)

STANDARDS

The DWARF standard is defined by *The DWARF Debugging Information Format*, Version 4, http://www.dwarfstd.org/.

HISTORY

The DWARF(3) API originated at Silicon Graphics Inc.

A BSD-licensed implementation of a subset of the API was written by John Birrell <*jb@FreeBSD.org*> for the FreeBSD project. The implementation was subsequently revised and completed by Kai Wang <*kaiwang27@users.sourceforge.net*>.

Manual pages for this implementation were written by Joseph Koshy <*jkoshy@users.sourceforge.net*> and Kai Wang <*kaiwang27@users.sourceforge.net*>.