NAME

dwarf_child, **dwarf_offdie**, **dwarf_offdie_b**, **dwarf_siblingof**, **dwarf_siblingof_b** - retrieve DWARF Debugging Information Entry descriptors

LIBRARY

DWARF Access Library (libdwarf, -ldwarf)

SYNOPSIS

#include <libdwarf.h>

int
dwarf_child(*Dwarf_Die die, Dwarf_Die *ret_die, Dwarf_Error *err*);

int
dwarf_offdie(*Dwarf_Debug dbg, Dwarf_Off offset, Dwarf_Die *ret_die, Dwarf_Error *err*);

int
dwarf_offdie_b(*Dwarf_Debug dbg, Dwarf_Off offset, Dwarf_Bool is_info, Dwarf_Die *ret_die, Dwarf_Error *err*);

int
dwarf_siblingof(*Dwarf_Debug dbg, Dwarf_Die die, Dwarf_Die *ret_die, Dwarf_Error *err*);

int
dwarf_siblingof_b(*Dwarf_Debug dbg, Dwarf_Die die, Dwarf_Die *ret_die, Dwarf_Bool is_info, Dwarf_Error *err*);

DESCRIPTION

These functions are used to retrieve and traverse DWARF Debugging Information Entry (DIE) descriptors associated with a compilation unit. These descriptors are arranged in the form of a tree, traversable using "child" and "sibling" links; see dwarf(3) for more information. DWARF Debugging Information Entry descriptors are represented by the *Dwarf_Die* opaque type.

Function **dwarf_child**() retrieves the child of descriptor denoted by argument *die*, and stores it in the location pointed to by argument *ret_die*.

Function **dwarf_siblingof**() retrieves the sibling of the descriptor denoted by argument *die*, and stores it in the location pointed to by argument *ret_die*. If argument *die* is NULL, the first debugging information entry descriptor for the current compilation unit will be returned. This function and function **dwarf_child**() may be used together to traverse the tree of debugging information entry

descriptors for a compilation unit.

Function **dwarf_siblingof_b**() is identical to the function **dwarf_siblingof**() except that it can retrieve the sibling descriptor from either the current compilation unit or type unit. If argument *is_info* is non-zero, the function behaves identically to function **dwarf_siblingof**(). If argument *is_info* is zero, the descriptor referred by argument *die* should be associated with a debugging information entry in the type unit. The function will store the sibling of the descriptor in the location pointed to by argument *ret_die*. If argument *is_info* is zero and argument *die* is NULL, the first debugging information entry descriptor for the current type unit will be returned.

Function **dwarf_offdie**() retrieves the debugging information entry descriptor at global offset *offset* in the ".debug_info" section of the object associated with argument *dbg*. The returned descriptor is written to the location pointed to by argument *ret_die*.

Function **dwarf_offdie_b**() is identical to the function **dwarf_offdie**() except that it can retrieve the debugging information entry descriptor at global offset *offset* from either of the ".debug_info" and ".debug_types" sections of the object associated with argument *dbg*. If argument *is_info* is non-zero, the function will retrieve the debugging information entry from the ".debug_info" section, otherwise the function will retrieve the debugging information entry from the ".debug_types" section. The returned descriptor is written to the location pointed to by argument *ret_die*.

Memory Management

The memory area used for the *Dwarf_Die* descriptor returned in argument *ret_die* is allocated by the DWARF Access Library (libdwarf, -ldwarf). Application code should use function **dwarf_dealloc**() with the allocation type DW_DLA_DIE to free the memory area when the *Dwarf_Die* descriptor is no longer needed.

RETURN VALUES

These functions return the following values:

[DW_DLV_OK] The call succeeded.

[DW_DLV_ERROR] The requested operation failed. Additional information about the error encountered will be recorded in argument *err*, if it is not NULL.

[DW_DLV_NO_ENTRY] For functions **dwarf_child**(), **dwarf_siblingof**() and **dwarf_siblingof_b**(), the descriptor denoted by argument *die* did not have a child or sibling.

For functions **dwarf_offdie**() and **dwarf_offdie_b**(), there was no debugging information entry at the offset specified by argument *offset*.

EXAMPLES

To retrieve the first DWARF Debugging Information Entry descriptor for the first compilation unit associated with a *Dwarf_Debug* instance, and to traverse all its children, use:

```
Dwarf_Debug dbg;
Dwarf_Die die, die0;
Dwarf_Error de;

... allocate dbg using dwarf_init() etc ...

if (dwarf_next_cu_header(dbg, NULL, NULL, NULL, NULL, NULL, &de) !=
   DW_DLV_OK)
        errx(EXIT_FAILURE, "dwarf_next_cu_header: %s",
           dwarf_errmsg(de));

/* Get the first DIE for the current compilation unit. */
die = NULL;
if (dwarf_siblingof(dbg, die, &die0, &de) != DW_DLV_OK)
        errx(EXIT_FAILURE, "dwarf_siblingof: %s", dwarf_errmsg(de));

/* Get the first child of this DIE. */
die = die0;
if (dwarf_child(die, &die0, &de) != DW_DLV_OK)
        errx(EXIT_FAILURE, "dwarf_child: %s", dwarf_errmsg(de));

/* Get the rest of children. */
do {
        die = die0;
        if (dwarf_siblingof(dbg, die, &die0, &de) == DW_DLV_ERROR)
                errx(EXIT_FAILURE, "dwarf_siblingof: %s",
                   dwarf_errmsg(de));
} while (die0 != NULL);
```

ERRORS

These functions may fail with the following errors:

[DW_DLE_ARGUMENT] Arguments *dbg*, *die* or *ret_die* were NULL.

[DW_DLE_DIE_NO_CU_CONTEXT] Argument *dbg* was not associated with a compilation unit.

[DW_DLE_NO_ENTRY] The descriptor denoted by argument *die* had no child or sibling, or there was no DWARF debugging information entry at the offset specified by argument *offset*.

SEE ALSO

dwarf(3), dwarf_errmsg(3), dwarf_get_die_infotypes_flag(3), dwarf_next_cu_header(3)