NAME

dwarf_get_ranges - retrieve non-contiguous address ranges

LIBRARY

DWARF Access Library (libdwarf, -ldwarf)

SYNOPSIS

#include <libdwarf.h>

int
dwarf_get_ranges(*Dwarf_Debug dbg*, *Dwarf_Off offset*, *Dwarf_Ranges **ranges*, *Dwarf_Signed *cnt*, *Dwarf_Unsigned *byte_cnt*, *Dwarf_Error *err*);

int
dwarf_get_ranges_a(*Dwarf_Debug dbg*, *Dwarf_Off offset*, *Dwarf_Die die*, *Dwarf_Ranges **ranges*, *Dwarf_Signed *cnt*, *Dwarf_Unsigned *byte_cnt*, *Dwarf_Error *err*);

DESCRIPTION

Function **dwarf_get_ranges**() retrieves information about the non-contiguous address ranges associated with a DWARF debugging information entry. Information about address ranges is returned as an array of descriptors of type *Dwarf_Ranges*, with each *Dwarf_Ranges* descriptor describing one address range entry.

Argument *dbg* should reference a DWARF debug context allocated using dwarf_init(3).

Argument *offset* is an offset, relative to the ".debug_ranges" section, to the start of the desired list of address ranges. The offset of an address ranges list is indicated by the DW_AT_ranges attribute of a debugging information entry.

Argument *die* (function **dwarf_get_ranges_a**() only) is ignored in this implementation; see the section *Compatibility Notes* below.

Argument *ranges* should point to a location that will be set to a pointer to an array of *Dwarf_Ranges* descriptors.

Argument *cnt* should point to a location that will be set to the number of entries returned. If argument *byte_cnt* is not NULL, it will be set to the number of bytes occupied by the returned entries in the ".debug_ranges" section.

If argument *err* is not NULL, it will be used to store error information in case of an error.

Dwarf_Ranges descriptors are defined in the header file <*libdwarf.h*>, and consists of the following fields:

dwr_addr1 The first address offset, whose meaning depends on the type of the entry.

dwr_addr2 The second address offset, whose meaning depends on the type of the entry.

dwr_type The type of this address range entry:

DW_RANGES_ENTRY	A range list entry. For this type of entry, the fields *dwr_addr1* and *dwr_addr2* hold the beginning and ending offsets of the address range, respectively.
DW_RANGES_ADDRESS_SELECTION	A base address selection entry. For this type of entry, the field *dwr_addr1* is the value of the largest representable address offset, and *dwr_addr2* is a base address for the beginning and ending address offsets of subsequent address range entries in the list.
DW_RANGES_END	An end of list mark. Both *dwr_addr1* and *dwr_addr2* are set to 0.

Memory Management

The memory area used for the array of *Dwarf_Ranges* descriptors returned in argument *ranges* is owned by the DWARF Access Library (libdwarf, -ldwarf). The application should not attempt to directly free this pointer. Portable code should instead use **dwarf_ranges_dealloc**() to indicate that the memory may be freed.

RETURN VALUES

These functions return DW_DLV_OK when they succeed. They return DW_DLV_NO_ENTRY if there is no address range list at the specified offset *offset*. In case of an error, they return DW_DLV_ERROR and set the argument *err*.

EXAMPLES

To retrieve the address range list associated with a debugging information entry, use:

```
Dwarf_Debug dbg;
Dwarf_Die die;
Dwarf_Error de;
Dwarf_Addr base;
Dwarf_Attribute *attr_list;
Dwarf_Ranges *ranges;
Dwarf_Signed cnt;
Dwarf_Unsigned off, attr_count, bytecnt;
```

```
        int i, j;

        if ((ret = dwarf_attrlist(die, &attr_list, &attr_count, &de)) !=
            DW_DLV_OK)
                errx(EXIT_FAILURE, "dwarf_attrlist failed: %s",
                    dwarf_errmsg(de));

        for (i = 0; (Dwarf_Unsigned) i < attr_count; i++) {
                if (dwarf_whatattr(attr_list[i], &attr, &de) != DW_DLV_OK) {
                        warnx("dwarf_whatattr failed: %s",
                            dwarf_errmsg(de));
                        continue;
                }
                if (attr != DW_AT_ranges)
                        continue;
                if (dwarf_formudata(attr_list[i], &off, &de) != DW_DLV_OK) {
                        warnx("dwarf_formudata failed: %s",
                            dwarf_errmsg(de));
                        continue;
                }
                if (dwarf_get_ranges(dbg, (Dwarf_Off) off, &ranges, &cnt,
                    &bytecnt, &de) != DW_DLV_OK)
                        continue;
                for (j = 0; j < cnt; j++) {
                        if (ranges[j].dwr_type == DW_RANGES_END)
                                break;
                        else if (ranges[j].dwr_type ==
                            DW_RANGES_ADDRESS_SELECTION)
                                base = ranges[j].dwr_addr2;
                        else {
                                /*
                                 * DW_RANGES_ENTRY entry.
                                 * .. Use dwr_addr1 and dwr_addr2 ..
                                 */
                        }
                }
        }
```

COMPATIBILITY

Function **dwarf_get_ranges_a**() is identical to **dwarf_get_ranges**(), except that it requires one additional argument *die* denoting the debugging information entry associated with the address range list. In this implementation of the DWARF Access Library (libdwarf, -ldwarf), the argument *die* is ignored, and function **dwarf_get_ranges_a**() is only provided for compatibility with other implementations of the DWARF(3) API.

ERRORS

These function can fail with:

[DW_DLE_ARGUMENT]	One of the arguments *dbg*, *ranges* or *cnt* was NULL.
[DW_DLE_NO_ENTRY]	There is no address range list at the specified offset *offset*.

SEE ALSO

dwarf(3), dwarf_ranges_dealloc(3)