NAME

dwarf_loclist, **dwarf_loclist_n** - retrieve DWARF location expression information

LIBRARY

DWARF Access Library (libdwarf, -ldwarf)

SYNOPSIS

#include <libdwarf.h>

int
dwarf_loclist(*Dwarf_Attribute at, Dwarf_Locdesc **llbuf, Dwarf_Signed *listlen, Dwarf_Error *error*);

int
dwarf_loclist_n(*Dwarf_Attribute at, Dwarf_Locdesc ***llbuf, Dwarf_Signed *listlen, Dwarf_Error *error*);

DESCRIPTION

These functions retrieve the location expressions associated with a DWARF attribute.

Note: function **dwarf_loclist**() is deprecated. New application code should instead use function **dwarf_loclist_n**()

Function **dwarf_loclist_n**() retrieves the list of location expressions associated with a DWARF attribute. Argument *at* should reference a valid DWARF attribute. Argument *llbuf* should point to a location which will hold a returned array of pointers to *Dwarf_Locdesc* descriptors. Argument *listlen* should point to a location which will be set to the number of elements contained in the returned array. If argument *err* is not NULL, it will be used to store error information in case of an error.

Function **dwarf_loclist**() retrieves the first location expression associated with an attribute. Argument *at* should reference a valid DWARF attribute. Argument *llbuf* should point to a location which will hold the returned pointer to a *Dwarf_Locdesc* descriptor. Argument *listlen* should point to a location which will be always set to 1. If argument *err* is not NULL, it will be used to store error information in case of an error.

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

ld_lopc The lowest program counter address covered by the descriptor. This field will be set to 0 if the descriptor is not associated with an address range.

ld_hipc The highest program counter address covered by the descriptor. This field will be set to 0 if the descriptor is not associated with an address range.

ld_cents The number of entries returned in *ld_s* field.
ld_s Pointer to an array of *Dwarf_Loc* descriptors.

Each *Dwarf_Loc* descriptor represents one operation of a location expression. These descriptors are defined in the header file <*libdwarf.h*>, and consist of following fields:

lr_atom The operator name, one of the DW_OP_* constants defined in the header file <*dwarf.h*>.
lr_number The first operand of this operation.
lr_number2 The second operand of this operation.
lr_offset The byte offset of this operation within the containing location expression.

Memory Management

The memory area used for the descriptor array returned in argument *llbuf* is allocated by the DWARF Access Library (libdwarf, -ldwarf). When the descriptor array is no longer needed, application code should use function dwarf_dealloc(3) to free the memory area in the following manner:

1. First, the *ld_s* field of each *Dwarf_Locdesc* descriptor should be deallocated using the allocation type DW_DLA_LOC_BLOCK.

2. Then, the application should free each *Dwarf_Locdesc* descriptor using the allocation type DW_DLA_LOCDESC.

3. Finally, the *llbuf* pointer should be deallocated using the allocation type DW_DLA_LIST.

RETURN VALUES

On success, these functions returns DW_DLV_OK. In case of an error, they return DW_DLV_ERROR and set the argument *err*.

EXAMPLES

To retrieve the location list associated with an attribute, use:

```
Dwarf_Attribute at;
Dwarf_Locdesc **llbuf;
Dwarf_Signed lcnt;
Dwarf_Loc *lr;
Dwarf_Error de;
int i;

if (dwarf_loclist_n(at, &llbuf, &lcnt, &de) != DW_DLV_OK)
        errx(EXIT_FAILURE, "dwarf_loclist_n failed: %s",
```

```
                dwarf_errmsg(de));

        for (i = 0; i < lcnt; i++) {
                /* ... Use llbuf[i] ... */
                for (j = 0; (Dwarf_Half) j < llbuf[i]->ld_cents; j++) {
                        lr = &llbuf[i]->ld_s[j];
                        /* ... Use each Dwarf_Loc descriptor ... */
                }
                dwarf_dealloc(dbg, llbuf[i]->ld_s, DW_DLA_LOC_BLOCK);
                dwarf_dealloc(dbg, llbuf[i], DW_DLA_LOCDESC);
        }
        dwarf_dealloc(dbg, llbuf, DW_DLA_LIST);
```

ERRORS

These functions can fail with:

[DW_DLE_ARGUMENT] One of the arguments *at*, *llbuf* or *listlen* was NULL.

[DW_DLE_ARGUMENT] The attribute provided by argument *at* does not contain a location expression or is not associated with a location expression list.

SEE ALSO

dwarf(3), dwarf_dealloc(3), dwarf_get_loclist_entry(3), dwarf_loclist_from_expr(3), dwarf_loclist_from_expr_a(3)