1. 简介

1. 简介

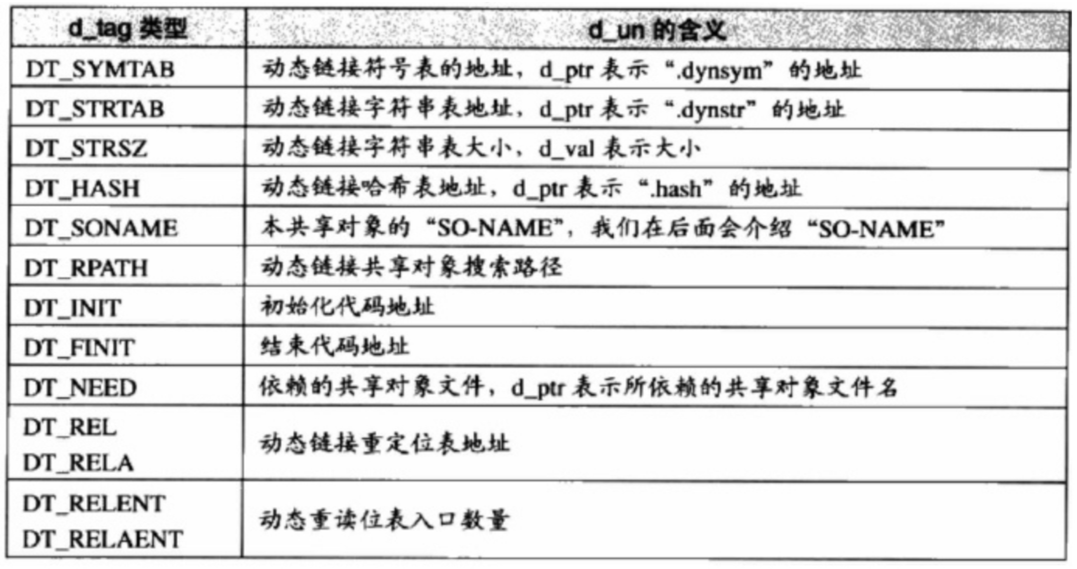
这个段里保存了动态链接器所需要的基本信息，比如依赖哪些共享对象、动态链接符号表的位置、动态链接重定位表的位置、共享对象初始化代码的地址等。

Linker在加载so时会通过.dynamic segment解析所有的sections，具体参见如下

.dynamic段里保存的信息有点像ELF文件头。

.dynamic段的结构是由Elf32_Dyn组成的数组。

Elf32_Dyn结构由一个类型值加上一个附加的数值或指针，对于不同的类型，后面附加的数值或者指针有着不同的含义



d_tag 类型	d_un 的含义
DT_SYMTAB	动态链接符号表的地址，d_ptr 表示 “.dynsym” 的地址
DT_STRTAB	动态链接字符串表地址，d_ptr 表示 “.dynstr” 的地址
DT_STRSZ	动态链接字符串表大小，d_val 表示大小
DT_HASH	动态链接哈希表地址，d_ptr 表示 “.hash” 的地址
DT_SONAME	本共享对象的 “SO-NAME”，我们在后面会介绍 “SO-NAME”
DT_RPATH	动态链接共享对象搜索路径
DT_INIT	初始化代码地址
DT_FINIT	结束代码地址
DT_NEED	依赖的共享对象文件，d_ptr 表示所依赖的共享对象文件名
DT_REL DT_RELA	动态链接重定位表地址
DT_RELENT DT_RELAENT	动态重读位表入口数量

If an object file participates in dynamic linking, its program header table will have an element of type **PT_DYNAMIC**. This segment contains the .dynamic section. A special symbol, _DYNAMIC, labels the section, which contains an array of the following structures. See sys/link.h.

1.1 格式

typedef struct {

```
        Elf32_Sword d_tag;
        union {
                Elf32_Word      d_val;
                Elf32_Addr      d_ptr;
                Elf32_Off       d_off;
        } d_un;
} Elf32_Dyn;

typedef struct {
        Elf64_Xword d_tag;
        union {
                Elf64_Xword     d_val;
                Elf64_Addr      d_ptr;
        } d_un;
} Elf64_Dyn;
```

对每一个有该类型的object，d_tag控制着d_un的解释。

- d_val：那些Elf32_Word object描绘了具有不同解释的整形变量。
- d_ptr：那些Elf32_Word object描绘了程序的虚拟地址。

In general, the value of each dynamic tag determines the interpretation of the d_un union. This convention provides for simpler interpretation of dynamic tags by third party tools. A tag whose value is an even number indicates a dynamic section entry that uses d_ptr. A tag whose value is an odd number indicates a dynamic section entry that uses d_val, or that the tag uses neither d_ptr nor d_val. Tags with values in the following special compatibility ranges do not follow these rules. Third party tools must handle these exception ranges explicitly on an item by item basis.

tag属性将决定**d_un** union。此约定提供了第三方工具对动态标记的简单解释。**值为偶数的标记表示使用d_ptr的动态节条目。值为奇数的标记表示使用d_val的动态节条目，或者该标记既不使用d_ptr也不使用d_val**。具有以下特殊兼容性范围值的标记不遵循这些规则。第三方工具必须逐项显式地处理这些异常范围。

1. Tags whose values are less than the special value **DT_ENCODING**.

2. Tags with values that fall between **DT_LOOS** and **DT_SUNW_ENCODING**.
3. Tags with values that fall between **DT_HIOS** and **DT_LOPROC**.

1.2 映射关系

下表总结了可执行和共享对象文件的标记要求。如果标记被标记为强制，则动态链接数组必须具有该类型的条目。同样，可选意味着标签的条目可以出现，但不是必需的

ELF Dynamic Array Tags

Name	Value	d_un	Exec
DT_NULL	0	Ignored	强制
DT_NEEDED	1	d_val	Optic
DT_PLTRELSZ	2	d_val	Optic
DT_PLTGOT	3	d_ptr	Optic
DT_HASH	4	d_ptr	强制
DT_STRTAB	5	d_ptr	强制
DT_SYMTAB	6	d_ptr	强制
DT_RELA	7	d_ptr	强制
DT_RELASZ	8	d_val	强制
DT_RELAENT	9	d_val	强制
DT_STRSZ	10	d_val	强制
DT_SYMENT	11	d_val	强制
DT_INIT	12	d_ptr	Optic
DT_FINI	13	d_ptr	Optic

DT_FINI	13	d_ptr	Optic
DT_SONAME	14	d_val	Ignor
DT_RPATH	15	d_val	Optic
DT_SYMBOLIC	16	Ignored	Ignor
DT_REL	17	d_ptr	强制
DT_RELSZ	18	d_val	强制
DT_RELENT	19	d_val	强制
DT_PLTREL	20	d_val	Optic
DT_DEBUG	21	d_ptr	Optic
DT_TEXTREL	22	Ignored	Optic
DT_JMPREL	23	d_ptr	Optic
DT_BIND_NOW	24	Ignored	Optic
DT_INIT_ARRAY	25	d_ptr	Optic
DT_FINI_ARRAY	26	d_ptr	Optic
DT_INIT_ARRAYSZ	27	d_val	Optic
DT_FINI_ARRAYSZ	28	d_val	Optic
DT_RUNPATH	29	d_val	Optic
DT_FLAGS	30	d_val	Optic
DT_ENCODING	32	Unspecified	Unsp
DT_PREINIT_ARRAY	32	d_ptr	Optic
DT_PREINIT_ARRAYSZ	33	d_val	Optic

DT_MAXPOSTAGS	34	Unspecified	Unsp
DT_LOOS	0x6000000d	Unspecified	Unsp
DT_SUNW_AUXILIARY	0x6000000d	d_ptr	Unsp
DT_SUNW_RTLDINF	0x6000000e	d_ptr	Optic
DT_SUNW_FILTER	0x6000000e	d_ptr	Unsp
DT_SUNW_CAP	0x60000010	d_ptr	Optic
DT_SUNW_SYMTAB	0x60000011	d_ptr	Optic
DT_SUNW_SYMSZ	0x60000012	d_val	Optic
DT_SUNW_ENCODING	0x60000013	Unspecified	Unsp
DT_SUNW_SORTENT	0x60000013	d_val	Optic
DT_SUNW_SYMSORT	0x60000014	d_ptr	Optic
DT_SUNW_SYMSORTSZ	0x60000015	d_val	Optic
DT_SUNW_TLSSORT	0x60000016	d_ptr	Optic
DT_SUNW_TLSSORTSZ	0x60000017	d_val	Optic
DT_SUNW_CAPINFO	0x60000018	d_ptr	Optic
DT_SUNW_STRPAD	0x60000019	d_val	Optic
DT_SUNW_CAPCHAIN	0x6000001a	d_ptr	Optic
DT_SUNW_LDMACH	0x6000001b	d_val	Optic
DT_SUNW_CAPCHAINENT	0x6000001d	d_val	Optic
DT_SUNW_CAPCHAINSZ	0x6000001f	d_val	Optic
DT_HIOS	0x6ffff000	Unspecified	Unsp

DT_VALRNGLO	0x6ffffd00	Unspecified	Unsp
DT_CHECKSUM	0x6ffffdf8	d_val	Optic
DT_PLTPADSZ	0x6ffffdf9	d_val	Optic
DT_MOVEENT	0x6ffffdfa	d_val	Optic
DT_MOVESZ	0x6ffffdfb	d_val	Optic
DT_POSFLAG_1	0x6ffffdfd	d_val	Optic
DT_SYMINSZ	0x6ffffdfe	d_val	Optic
DT_SYMINENT	0x6ffffdff	d_val	Optic
DT_VALRNGHI	0x6ffffdff	Unspecified	Unsp
DT_ADDRRNGLO	0x6ffffe00	Unspecified	Unsp
DT_CONFIG	0x6ffffefa	d_ptr	Optic
DT_DEPAUDIT	0x6ffffefb	d_ptr	Optic
DT_AUDIT	0x6ffffefc	d_ptr	Optic
DT_PLTPAD	0x6ffffefd	d_ptr	Optic
DT_MOVETAB	0x6ffffefe	d_ptr	Optic
DT_SYMINFO	0x6ffffeff	d_ptr	Optic
DT_ADDRRNGHI	0x6ffffeff	Unspecified	Unsp
DT_RELACOUNT	0x6ffffff9	d_val	Optic
DT_RELCOUNT	0x6ffffffa	d_val	Optic
DT_FLAGS_1	0x6ffffffb	d_val	Optic

DT_VERDEF	0x6ffffffc	d_ptr	Optic
DT_VERDEFNUM	0x6ffffffd	d_val	Optic
DT_VERNEED	0x6ffffffe	d_ptr	Optic
DT_VERNEEDNUM	0x6fffffff	d_val	Optic
DT_LOPROC	0x70000000	Unspecified	Unsp
DT_SPARC_REGISTER	0x70000001	d_val	Optic
DT_AUXILIARY	0x7ffffffd	d_val	Unsp
DT_USED	0x7ffffffe	d_val	Optic
DT_FILTER	0x7fffffff	d_val	Unsp
DT_HIPROC	0x7fffffff	Unspecified	Unsp

1.3 关键tag

`DT_NULL`

Marks the end of the `_DYNAMIC` array.

`DT_NEEDED`

The `DT_STRTAB` string table offset of a null-terminated string, giving the name of a needed dependency. The dynamic array can contain multiple entries of this type. The relative order of these entries is significant, though their relation to entries of other types is not. See Shared Object Dependencies.

`DT_PLTRELSZ`

The total size, **in bytes**, of the relocation entries associated with the procedure linkage table. See Procedure Linkage Table (Processor-Specific).

`DT_PLTGOT`

An address associated with the procedure linkage table or the global offset table. See Procedure Linkage Table (Processor-Specific) and Global Offset Table (Processor-Specific).

`DT_HASH`

The address of the symbol hash table. This table refers to the symbol table indicated by the `DT_SYMTAB` element. See Hash Table Section.

`DT_STRTAB`

The address of the string table. Symbol names, dependency names, and other strings required by the runtime linker reside in this table. See String Table Section.

DT_SYMTAB

The address of the symbol table. See Symbol Table Section.

DT_RELA

The address of a relocation table. See Relocation Sections.

An object file can have multiple relocation sections. When creating the relocation table for an executable or shared object file, the link-editor catenates those sections to form a single table. Although the sections can remain independent in the object file, the runtime linker sees a single table. When the runtime linker creates the process image for an executable file or adds a shared object to the process image, the runtime linker reads the relocation table and performs the associated actions.

This element requires the `DT_RELASZ` and `DT_RELAENT` elements also be present. When relocation is mandatory for a file, either `DT_RELA` or `DT_REL` can occur.

DT_RELASZ

The total size, in bytes, of the `DT_RELA` relocation table.

DT_RELAENT

The size, in bytes, of the `DT_RELA` relocation entry.

DT_STRSZ

The total size, in bytes, of the `DT_STRTAB` string table.

DT_SYMENT

The size, in bytes, of the `DT_SYMTAB` symbol entry.

DT_INIT

The address of an initialization function. See Initialization and Termination Sections.

DT_FINI

The address of a termination function. See Initialization and Termination Sections.

DT_SONAME

The `DT_STRTAB` string table offset of a null-terminated string, identifying the name of the shared object. See Recording a Shared Object Name.

DT_RPATH

The `DT_STRTAB` string table offset of a null-terminated library search path string. This element's use has been superseded by `DT_RUNPATH`. See Directories Searched by the Runtime Linker.

DT_SYMBOLIC

Indicates the object contains symbolic bindings that were applied during its link-edit. This elements use has been superseded by the `DF_SYMBOLIC` flag. See Using the `-B symbolic` Option.

DT_REL

Similar to `DT_RELA`, except its table has implicit addends. This element requires that the `DT_RELSZ` and `DT_RELENT` elements also be present.

`DT_RELSZ`

The total size, in bytes, of the `DT_REL` relocation table.

`DT_RELENT`

The size, in bytes, of the `DT_REL` relocation entry.

`DT_PLTREL`

Indicates the type of relocation entry to which the procedure linkage table refers, either `DT_REL` or `DT_RELA`. All relocations in a procedure linkage table must use the same relocation. See Procedure Linkage Table (Processor-Specific). This element requires a `DT_JMPREL` element also be present.

`DT_DEBUG`

Used for debugging.

`DT_TEXTREL`

Indicates that one or more relocation entries might request modifications to a non-writable segment, and the runtime linker can prepare accordingly. This element's use has been superseded by the `DF_TEXTREL` flag. See Position-Independent Code.

`DT_JMPREL`

The address of relocation entries that are associated solely with the procedure linkage table. See Procedure Linkage Table (Processor-Specific). The separation of these relocation entries enables the runtime linker to ignore these entries when the object is loaded with lazy binding enabled. This element requires the `DT_PLTRELSZ` and `DT_PLTREL` elements also be present.

`DT_POSFLAG_1`

Various state flags which are applied to the `DT_` element immediately following. See Table 13-11.

`DT_BIND_NOW`

Indicates that all relocations for this object must be processed before returning control to the program. The presence of this entry takes precedence over a directive to use lazy binding when specified through the environment or by means of `dlopen`(3C). This element's use has been superseded by the `DF_BIND_NOW` flag. See When Relocations Are Performed.

`DT_INIT_ARRAY`

The address of an array of pointers to initialization functions. This element requires that a `DT_INIT_ARRAYSZ` element also be present. See Initialization and Termination Sections.

`DT_FINI_ARRAY`

The address of an array of pointers to termination functions. This element requires that a `DT_FINI_ARRAYSZ` element also be present. See Initialization and Termination Sections.

`DT_INIT_ARRAYSZ`

The total size, in bytes, of the `DT_INIT_ARRAY` array.

DT_FINI_ARRAYSZ

The total size, in bytes, of the `DT_FINI_ARRAY` array.

DT_RUNPATH

The `DT_STRTAB` string table offset of a null-terminated library search path string. See Directories Searched by the Runtime Linker.

DT_FLAGS

Flag values specific to this object. See Table 13-9.

DT_ENCODING

Dynamic tag values that are greater than or equal to `DT_ENCODING`, and less than or equal to `DT_LOOS`, follow the rules for the interpretation of the `d_un` union.

DT_PREINIT_ARRAY

The address of an array of pointers to pre-initialization functions. This element requires that a `DT_PREINIT_ARRAYSZ` element also be present. This array is processed only in an executable file. This array is ignored if contained in a shared object. See Initialization and Termination Sections.

DT_PREINIT_ARRAYSZ

The total size, in bytes, of the `DT_PREINIT_ARRAY` array.

DT_MAXPOSTAGS

The number of positive dynamic array tag values.

DT_LOOS - DT_HIOS

Values in this inclusive range are reserved for operating system-specific semantics. All such values follow the rules for the interpretation of the `d_un` union.

DT_SUNW_AUXILIARY

The `DT_STRTAB` string table offset of a null-terminated string that names one or more per-symbol, auxiliary filtees. See Generating Auxiliary Filters.

DT_SUNW_RTLDINF

Reserved for internal use by the runtime-linker.

DT_SUNW_FILTER

The `DT_STRTAB` string table offset of a null-terminated string that names one or more per-symbol, standard filtees. See Generating Standard Filters.

DT_SUNW_CAP

The address of the capabilities section. See Capabilities Section.

DT_SUNW_SYMTAB

The address of the symbol table containing local function symbols that augment the symbols provided by `DT_SYMTAB`. These symbols are always adjacent to, and immediately precede the symbols provided by `DT_SYMTAB`. See Symbol Table Section.

DT_SUNW_SYMSZ

The combined size of the symbol tables given by `DT_SUNW_SYMTAB` and `DT_SYMTAB`.

DT_SUNW_ENCODING

Dynamic tag values that are greater than or equal to `DT_SUNW_ENCODING`, and less than or equal to `DT_HIOS`, follow the rules for the interpretation of the `d_un` union.

DT_SUNW_SORTENT

The size, in bytes, of the `DT_SUNW_SYMSORT` and `DT_SUNW_TLSSORT` symbol sort entries.

DT_SUNW_SYMSORT

The address of the array of symbol table indices that provide sorted access to function and variable symbols in the symbol table referenced by `DT_SUNW_SYMTAB`. See Symbol Sort Sections.

DT_SUNW_SYMSORTSZ

The total size, in bytes, of the `DT_SUNW_SYMSORT` array.

DT_SUNW_TLSSORT

The address of the array of symbol table indices that provide sorted access to thread local symbols in the symbol table referenced by `DT_SUNW_SYMTAB`. See Symbol Sort Sections.

DT_SUNW_TLSSORTSZ

The total size, in bytes, of the `DT_SUNW_TLSSORT` array.

DT_SUNW_CAPINFO

The address of the array of symbol table indices that provide the association of symbols to their capability requirements. See Capabilities Section.

DT_SUNW_STRPAD

The total size, in bytes, of the unused reserved space at the end of the dynamic string table. If `DT_SUNW_STRPAD` is not present in an object, no reserved space is available.

DT_SUNW_CAPCHAIN

The address of the array of capability family indices. Each family of indices is terminated with a `0` entry.

DT_SUNW_LDMACH

The machine architecture of the link-editor that produced the object. `DT_SUNW_LDMACH` uses the same `EM_` integer values used for the `e_machine` field of the ELF header. See ELF Header. `DT_SUNW_LDMACH` is used to identify the class, 32–bit or 64–bit, and the platform of the link-editor that built the object. This information is not used by the runtime linker, but exists purely for documentation.

DT_SUNW_CAPCHAINENT

The size, in bytes, of the `DT_SUNW_CAPCHAIN` entries.

DT_SUNW_CAPCHAINSZ

The total size, in bytes, or the `DT_SUNW_CAPCHAIN` chain.

DT_SYMINFO

The address of the symbol information table. This element requires that the `DT_SYMINENT` and `DT_SYMINSZ` elements also be present. See Syminfo Table Section.

DT_SYMINENT

The size, in bytes, of the DT_SYMINFO information entry.

DT_SYMINSZ

The total size, in bytes, of the DT_SYMINFO table.

DT_VERDEF

The address of the version definition table. Elements within this table contain indexes into the string table DT_STRTAB. This element requires that the DT_VERDEFNUM element also be present. See Version Definition Section.

DT_VERDEFNUM

The number of entries in the DT_VERDEF table.

DT_VERNEED

The address of the version dependency table. Elements within this table contain indexes into the string table DT_STRTAB. This element requires that the DT_VERNEEDNUM element also be present. See Version Dependency Section.

DT_VERNEEDNUM

The number of entries in the DT_VERNEEDNUM table.

DT_RELACOUNT

Indicates the RELATIVE relocation count, which is produced from the concatenation of all Elf32_Rela, or Elf64_Rela relocations. See Combined Relocation Sections.

DT_RELCOUNT

Indicates the RELATIVE relocation count, which is produced from the concatenation of all Elf32_Rel relocations. See Combined Relocation Sections.

DT_AUXILIARY

The DT_STRTAB string table offset of a null-terminated string that names one or more auxiliary filtees. See Generating Auxiliary Filters.

DT_FILTER

The DT_STRTAB string table offset of a null-terminated string that names one or more standard filtees. See Generating Standard Filters.

DT_CHECKSUM

A simple checksum of selected sections of the object. See gelf_checksum(3ELF).

DT_MOVEENT

The size, in bytes, of the DT_MOVETAB move entries.

DT_MOVESZ

The total size, in bytes, of the DT_MOVETAB table.

DT_MOVETAB

The address of a move table. This element requires that the DT_MOVEENT and DT_MOVESZ elements also be present. See Move Section.

DT_CONFIG

The DT_STRTAB string table offset of a null-terminated string defining a configuration file. The configuration file is only meaningful in an executable, and is typically unique to this object. See Configuring the Default Search Paths.

DT_DEPAUDIT

The DT_STRTAB string table offset of a null-terminated string defining one or more audit libraries. See Runtime Linker Auditing Interface.

DT_AUDIT

The DT_STRTAB string table offset of a null-terminated string defining one or more audit libraries. See Runtime Linker Auditing Interface.

DT_FLAGS_1

Flag values specific to this object. See Table 13-10.

DT_VALRNGLO - DT_VALRNGHI

Values in this inclusive range use the d_un.d_val field of the dynamic structure.

DT_ADDRRNGLO - DT_ADDRRNGHI

Values in this inclusive range use the d_un.d_ptr field of the dynamic structure. If any adjustment is made to the ELF object after the object has been built, these entries must be updated accordingly.

DT_SPARC_REGISTER

The index of an STT_SPARC_REGISTER symbol within the DT_SYMTAB symbol table. One dynamic entry exists for every STT_SPARC_REGISTER symbol in the symbol table. See Register Symbols.

DT_LOPROC - DT_HIPROC

Values in this inclusive range are reserved for processor-specific semantics.