ELF Symbol Meta-Information Implementation Details

August 2020

This document describes the precise changes to be made to the ELF gABI to implement Symbol Meta-Information.

4 Object Files

Sections

Table 1: Section types, `sh_type`

Name	Value
SHT_SYMTAB_META	19

SHT_SYMTAB_META This section contains the symbol meta-information entries for the file. The section might begin with a header, which contains some supplemental information.

Figure 1: .symtab_meta Header

```
typedef struct {
  unsigned char symtab_hash[20];
} Elf32_SMhdr;

typedef struct {
  unsigned char symtab_hash[20];
} Elf64_SMhdr;
```

symtab_hash For `.symtab_meta` format version >= 2, a 20-byte SHA-1 hash of the entire contents of `.symtab`.

Figure 2: `sh_link` and `sh_info` interpretation

Name	sh_link	sh_info
`SHT_SYMTAB_META`	The section header index of the associated symbol table.	The format version number of the symbol meta-information table (`ELFxx_SMH_VER`), and the section header index of the `.strtab_meta` string table used by entries in this section (`ELFxx_SMH_STR`).

(a) Accessors for the `sh_info` field

```
#define ELF32_SMH_STR(i)     ((i)>>8)
#define ELF32_SMH_VER(i)     ((unsigned char)(i))
#define ELF32_SMH_INFO(s,v) (((s)<<8)+(unsigned char)(v))

#define ELF64_SMH_STR(i)     ((i)>>32)
#define ELF64_SMH_VER(i)     ((i)&0xffffffffL)
#define ELF64_SMH_INFO(s,v) (((s)<<32)+((v)&0xffffffffL))
```

Figure 3: `.symtab_meta` Versions

Value	Meaning
0	Invalid Version
1	There is no header at the beginning of `.symtab_meta`.
2	A header containing the hash of `.symtab` is at the beginning of `.symtab_meta`.

Special Sections

Figure 4: Special Sections

Name	Type	Attributes
`.symtab_meta`	`SHT_SYMTAB_META`	`None`
`.strtab_meta`	`SHT_STRTAB`	`None`

.symtab_meta This section holds additional "meta-information" about symbols in `.symtab`. The different types of meta-information are described in "Symbol Meta-Information".

.strtab_meta If required, this section holds strings used as a value to certain types of symbol meta-information. It can be omitted if no symbol meta-information types require it.

Symbol Meta-Information

Note: This is a new subsection, intended to be placed at the end of the "Symbol Table" section, after the "Symbol Values" subsection.

ELF relocatable and executable files may contain a new section named `.symtab_meta`. This section describes additional information about symbols in `.symtab`. The section can be omitted from ELF files if there is no meta-information for any symbols, but if present, there can only be one section with this name and type.

Symbol Meta-Information Table Entries

Following the initial header of `.symtab_meta`, there is an array of symbol meta-information entries.

```
typedef struct {
  Elf32_Addr smi_info;
  Elf32_Word smi_value;
} Elf32_SymMetaInfo;

typedef struct {
  Elf64_Addr  smi_info;
  Elf64_Xword smi_value;
} Elf64_SymMetaInfo;
```

smi_info This field describes both the symbol table index of the ELF symbol this symbol meta-information this applies to, and the type of meta-information entry this is. A number of generic types are pre-defined. There are also reserved ranges for processor-specific and application-specific (i.e. vendor-specific) types.

```
#define ELF32_SMI_SYM(i)    ((i)>>8)
#define ELF32_SMI_TYPE(i)   ((unsigned char)(i))
#define ELF32_SMI_INFO(s,t) (((s)<<8)+(unsigned char)(t))

#define ELF64_SMI_SYM(i)    ((i)>>32)
#define ELF64_SMI_TYPE(i)   ((i)&0xffffffffL)
#define ELF64_SMI_INFO(s,t) (((s)<<32)+((t)&0xffffffffL))
```

smi_value The interpretation depends on the associated type. The value could be interpreted as a boolean, symbol table index, address, string table index etc.

Figure 5: Symbol Meta-Information Types

Value	Type	Format of Value
0	SMT_NONE	None
1	SMT_RETAIN	Boolean
2	SMT_LOCATION	Address
3	SMT_NOINIT	Boolean
4	SMT_PRINTF_FMT	Integer
0xC0	SMT_LOPROC	Processor-specific
0xDF	SMT_HIPROC	
0xE0	SMT_LOUSER	Vendor-specific
0xFF	SMT_HIUSER	

SMT_NONE This indicates an invalid or incomplete entry.

SMT_RETAIN A value of 1 indicates the associated symbol should be retained in the output executable file, even it appears unused and so the linker would normally garbage collect it. Other values result in the type being ignored.

SMT_LOCATION The VMA of the associated symbol in the output executable file should be set to the specified the value.

SMT_NOINIT A value of 1 indicates the associated data symbol should not be initialized by the runtime support code at program startup. Other values result in the type being ignored.

SMT_PRINTF_FMT The value indicates a byte offset into the `.strtab_meta` section. The section header table index of `.strtab_meta` is extracted from the `sh_info` value of `.symtab_meta`, using the `ELFxx_SMH_STR` accessor. The null-terminated string extracted from the string table is a de-duplicated list of format specifiers used by calls to printf-like functions, in the function whose symbol is pointed to by this entry.
The following C code:
`printf ("%d / %d = %f\n", ...);`
would generate the following string in `.strtab_meta`:
"`%d%f`".

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

SMT_LOUSER..SMT_HIUSER Values in this range are reserved for vendor-specific semantics.

Restrictions on applying symbol meta-information types to symbols
Symbol meta-information entries are always tied to a symbol in the symbol table, so there are no special rules regarding different symbols with the same name; the standard symbol binding rules apply.

No two entries in `.symtab_meta` can have the same `smi_info` value - each symbol must only have one value for a given meta-information type.

Figure 6: Symbol bindings and types permitted for metasyms

Symbol Meta-Information Type	Permitted Symbol Binding	Permitted Symbol Type
SMT_RETAIN	Any < STB_LOOS	STT_FUNC or STT_OBJECT or STT_COMMON
SMT_LOCATION	Any < STB_LOOS	STT_FUNC or STT_OBJECT or STT_COMMON
SMT_NOINIT	Any < STB_LOOS	STT_OBJECT or STT_COMMON
SMT_PRINTF_FMT	Any < STB_LOOS	STT_FUNC