Figure 1-13: `sh_link` **and** `sh_info` **Interpretation**

sh_type	sh_link	sh_info
SHT_DYNAMIC	The section header index of the string table used by entries in the section.	0
SHT_HASH	The section header index of the symbol table to which the hash table applies.	0
SHT_REL SHT_RELA	The section header index of the associated symbol table.	The section header index of the section to which the relocation applies.
SHT_SYMTAB SHT_DYNSYM	The section header index of the associated string table.	One greater than the symbol table index of the last local symbol (binding STB_LOCAL).
other	SHN_UNDEF	0

Special Sections

Various sections hold program and control information. Sections in the list below are used by the system and have the indicated types and attributes.

Figure 1-14: Special Sections

Name	Type	Attributes
.bss	SHT_NOBITS	SHF_ALLOC + SHF_WRITE
.comment	SHT_PROGBITS	none
.data	SHT_PROGBITS	SHF_ALLOC + SHF_WRITE
.data1	SHT_PROGBITS	SHF_ALLOC + SHF_WRITE
.debug	SHT_PROGBITS	none
.dynamic	SHT_DYNAMIC	see below
.dynstr	SHT_STRTAB	SHF_ALLOC
.dynsym	SHT_DYNSYM	SHF_ALLOC
.fini	SHT_PROGBITS	SHF_ALLOC + SHF_EXECINSTR
.got	SHT_PROGBITS	see below
.hash	SHT_HASH	SHF_ALLOC
.init	SHT_PROGBITS	SHF_ALLOC + SHF_EXECINSTR
.interp	SHT_PROGBITS	see below
.line	SHT_PROGBITS	none
.note	SHT_NOTE	none
.plt	SHT_PROGBITS	see below
.rel*name*	SHT_REL	see below

Figure 1-14: Special Sections (continued)

`.rela`*name*	`SHT_RELA`	see below
`.rodata`	`SHT_PROGBITS`	`SHF_ALLOC`
`.rodata1`	`SHT_PROGBITS`	`SHF_ALLOC`
`.shstrtab`	`SHT_STRTAB`	none
`.strtab`	`SHT_STRTAB`	see below
`.symtab`	`SHT_SYMTAB`	see below
`.text`	`SHT_PROGBITS`	`SHF_ALLOC + SHF_EXECINSTR`

`.bss` This section holds uninitialized data that contribute to the program's memory image. By definition, the system initializes the data with zeros when the program begins to run. The section occupies no file space, as indicated by the section type, `SHT_NOBITS`.

`.comment` This section holds version control information.

`.data` and `.data1`
These sections hold initialized data that contribute to the program's memory image.

`.debug` This section holds information for symbolic debugging. The contents are unspecified.

`.dynamic` This section holds dynamic linking information. The section's attributes will include the `SHF_ALLOC` bit. Whether the `SHF_WRITE` bit is set is processor specific. See Part 2 for more information.

`.dynstr` This section holds strings needed for dynamic linking, most commonly the strings that represent the names associated with symbol table entries. See Part 2 for more information.

`.dynsym` This section holds the dynamic linking symbol table, as ''Symbol Table'' describes. See Part 2 for more information.

`.fini` This section holds executable instructions that contribute to the process termination code. That is, when a program exits normally, the system arranges to execute the code in this section.

`.got` This section holds the global offset table. See ''Special Sections'' in Part 1 and ''Global Offset Table'' in Part 2 for more information.

`.hash` This section holds a symbol hash table. See ''Hash Table'' in Part 2 for more information.

`.init` This section holds executable instructions that contribute to the process initialization code. That is, when a program starts to run, the system arranges to execute the code in this section before calling the main program entry point (called `main` for C programs).

`.interp` This section holds the path name of a program interpreter. If the file has a loadable segment that includes the section, the section's attributes will include the `SHF_ALLOC` bit; otherwise, that bit will be off. See Part 2 for more information.

`.line` This section holds line number information for symbolic debugging, which describes the correspondence between the source program and the machine code. The contents are unspecified.

`.note`	This section holds information in the format that ''Note Section'' in Part 2 describes.
`.plt`	This section holds the procedure linkage table. See ''Special Sections'' in Part 1 and ''Procedure Linkage Table'' in Part 2 for more information.
`.rel`*name* and `.rela`*name*	These sections hold relocation information, as ''Relocation'' below describes. If the file has a loadable segment that includes relocation, the sections' attributes will include the `SHF_ALLOC` bit; otherwise, that bit will be off. Conventionally, *name* is supplied by the section to which the relocations apply. Thus a relocation section for `.text` normally would have the name `.rel.text` or `.rela.text`.
`.rodata` and `.rodata1`	These sections hold read-only data that typically contribute to a non-writable segment in the process image. See ''Program Header'' in Part 2 for more information.
`.shstrtab`	This section holds section names.
`.strtab`	This section holds strings, most commonly the strings that represent the names associated with symbol table entries. If the file has a loadable segment that includes the symbol string table, the section's attributes will include the `SHF_ALLOC` bit; otherwise, that bit will be off.
`.symtab`	This section holds a symbol table, as ''Symbol Table'' in this section describes. If the file has a loadable segment that includes the symbol table, the section's attributes will include the `SHF_ALLOC` bit; otherwise, that bit will be off.
`.text`	This section holds the ''text,'' or executable instructions, of a program.

Section names with a dot (`.`) prefix are reserved for the system, although applications may use these sections if their existing meanings are satisfactory. Applications may use names without the prefix to avoid conflicts with system sections. The object file format lets one define sections not in the list above. An object file may have more than one section with the same name.

Section names reserved for a processor architecture are formed by placing an abbreviation of the architecture name ahead of the section name. The name should be taken from the architecture names used for `e_machine`. For instance .FOO.psect is the psect section defined by the FOO architecture. Existing extensions are called by their historical names.

Pre-existing Extensions	
`.sdata`	`.tdesc`
`.sbss`	`.lit4`
`.lit8`	`.reginfo`
`.gptab`	`.liblist`
`.conflict`	