The ELF File Format

Figure: Linking and Execution Views: This figure illustrates the format of an ELF object file.

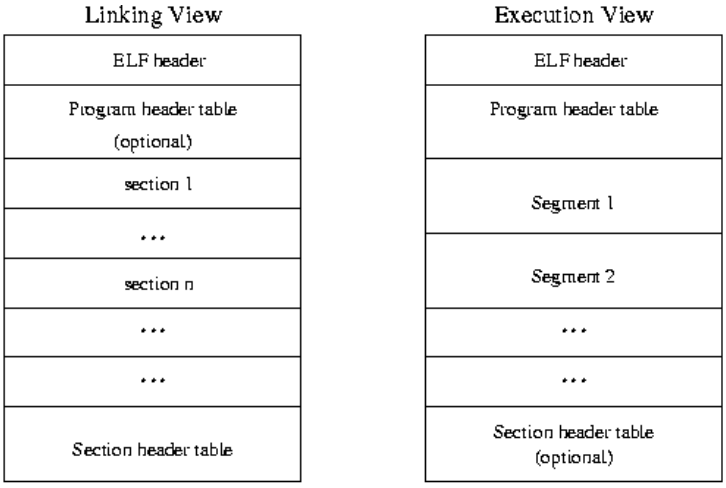

Figure: The ELF Header

```
#define EI_NIDENT    16

typedef struct {
    unsigned char    e_ident[EI_NIDENT];    //  file ID, interpretation
    Elf32_Half       e_type;                //  object file type
    Elf32_Half       e_machine;             //  target architecture
    Elf32_Word       e_version;             //  ELF version
    Elf32_Addr       e_entry;               //  starting virtual address
    Elf32_Off        e_phoff;               //  file offset to program hdr
    Elf32_Off        e_shoff;               //  file offset to section hdr
    Elf32_Word       e_flags;               //  processor-specific flags
    Elf32_Half       e_ehsize;              //  the ELF header's size
    Elf32_Half       e_phentsize;           //  program hdr entry size
    Elf32_Half       e_phnum;               //  program hdr entry number
    Elf32_Half       e_shentsize;           //  section hdr entry size
    Elf32_Half       e_shnum;               //  section hdr entry number
    Elf32_Half       e_shstrndx;            //  section hdr index for strings
} Elf32_Ehdr;
```

There are two views for each of the three file types described in the previous section. These views support both the linking and execution of a program. The two views are summarized in Figure 2.5 where the view on the left of the figure is the link view and the view on the right of the figure is the execution view. The link view of the ELF object file is partitioned by sections and the execution view of the ELF object file is partitioned by segments. Thus, the programmer interested in obtaining section information about the program items such as symbol tables, relocation, specific executable code or dynamic linking information will use the link view; the programmer interested in obtaining segment information such as the location of the `text` segment or `data` segment will use the execution view. The ELF access library, libelf, provides a programmer with tools to extract and manipulate ELF object file contents for either view. The ELF header describes the layout of the rest of the object file. It provides information on where and how to access the other sections. The Section Header Table gives the location and description of the sections and is mostly used in linking. The Program Header Table provides the location and description of segments and is mostly used in creating a programs' process image. Both sections and segments hold the majority of data in an object file including: instructions, data, symbol table, relocation information, and dynamic linking information.