

Center of Information Technology and Scientific Computing

Department of Software Engineering

System Programming

Assignment l

Prepared by:

Hanan Mohsin ATR/1489/09

Section-2

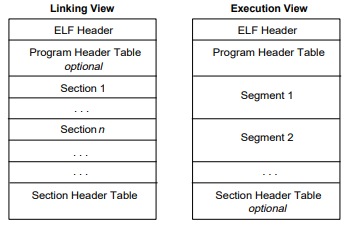
Date:- 12/26/2019

**Executable and Linkable Format**

**Overview**

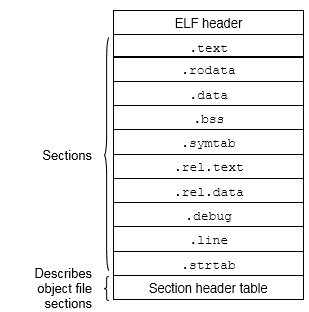
There are three main types of object files:-

* A re-locatable file holds code and data suitable for linking with other object files to create an executable or a shared object file.
* An executable file holds a program suitable for execution.
* A shared object file holds code and data suitable for linking in two contexts.

Object files participate in program linking (building a program) and program execution (running a program). For convenience and efficiency, the object file format provides parallel views of a file’s contents, reflecting the differing needs of these activities.

ELF is a standard file format for executable, object code, shared libraries, and core dumbs.

**Structures of ELF file**

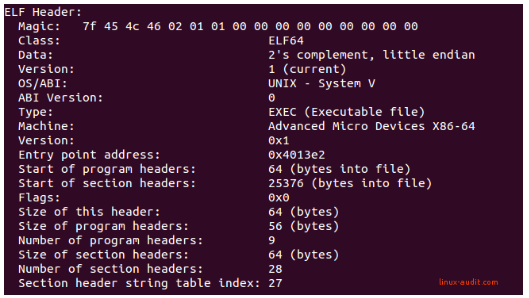
****

Due to the extensible design of ELF files, the structures differs per file. An ELF file consists of:

1. ELF header
2. File data

**ELF header**

Command to see the structure of a file:- **readelf**



ELF header starts with some magic. The ELF header magic provides information about the file. The first 4 hexadecimal parts define that this is an ELF file (45=**E**, 4c=**L**,46=**F**), prefixed with the 7f value.

This ELF header is mandatory. It ensures that data is correctly interpreted during linking or execution.

**File data**

Besides the ELF header, ELF files consist of three parts.

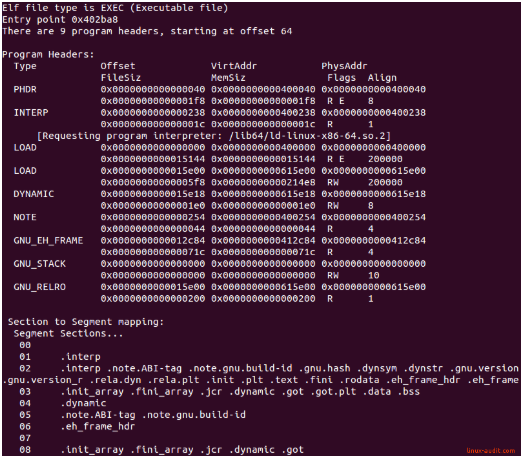
* Program Headers or Segments
* Section Headers or Sections
* Data

**Program headers**

An ELF file consists of zero or more segments, and describe how to create a process/memory image for runtime execution. When the kernel sees these segments, it uses them to map them into virtual address space, using the mmpa(2) system call. In other words, it converts predefined instructions into memory image. If your ELF file is normal binary, it requires these program headers. Otherwise, it simply won’t run. It uses these headers, with the underlying data structure, to form a process.

Command to see the program headers:

* **dumpelf(pax-utilis)**
* **elfls –S/bin/ps**
* **eu-readelf –program-headers/bin/ps**



**Section headers**

The section headers define all the sections in the file. As said, the “view” is used for linking and relocation. Sections can be found in an ELF binary after GNU C compiler transformed C code into assembly, followed by GNU assembler, which creates objects of it. For executable files there are four main sections: .text, .data, .rodata, and .bss. Each of these sections is loaded with different access right.

.text:- Contains executable code. It will be packed into segments with read and execute access rights. It is only loaded once, as the content change.

.data:- Initialized data, with read/write access rights.

.rodata:- Initialized data, with read access rights only.

.bss:- Uninitialized data, with read/write access rights.

Command to see the program headers:

* **dumpelf**
* **elfls –p/bin/ps**
* **eu-readelf -section-headers/bin/ps**
* **readelf –S/bin/ps**
* **objdump –h/bin/ps**

**Dynamic Linking and Loading**

**Linking**

Linker collects procedures and links them together into one executable program .

There are two approaches of linking:-

* Static linking
* Dynamic linking

**Dynamic Linking**

In dynamic linking complete linking is postponed until execution time. Stub is used to locate the appropriate memory-resident library routine. Stub replaces itself with the address of the routine, and executes the routine. Operating system needs to check if routine is in processes’ memory address space.

Advantage of Dynamic Linking

* The executable is smaller (does not include the library information explicitly).
* When the library is changed, the code that references it does not usually need to be recompiled.
* The executable accesses the .DLL at runtime; therefore, multiple codes can access the same .DLL at the same time therefore it saves memory.

Disadvantage of Dynamic Linking

* Need to load shared objects (once) and need to resolve addresses (once or every time).
* Could have the library, but wrong version.

**Loading**

It loads a program file into the memory for execution

There are two approaches of loading:-

* Static loading
* Dynamic loading

**Dynamic Loading**

Program file can either be program-controlled or linker-assisted dynamic loading.

In program-controlled dynamic loading a load system is called to invoke loader. It has ability to leave symbols unresolved and resolve at run time.

In linker-assisted dynamic loading, programmer marks modules as “dynamic” to linker

**Static libraries**

Static libraries are reusable in multiple programs, are locked into a program at compile time. Using static libraries means every file in your program must have it’s own copy of the library’s file at compile time.

The upside of using static libraries is that it is not susceptible to breaking. Another benefit of using static libraries is execution speed at runtime. Because it’s object code is already included in the executable file, multiple calls to function can be handled much more quickly than a dynamic library’s code.

The downside of using static library is that its code is locked into final executable file and cannot be modified without a re-compile.

**Dynamic libraries**

Dynamic or shared libraries exist as separate files outside of the executable file. Because dynamic libraries live outside of executable file, the program needs only to make only one copy of the library’s system at compile time.

The upside of using dynamic library is that it can be modified without the need to re-compile. Another benefit of using dynamic library is that multiple running application use the same library without the need for each to have its own copy.

The downside of using dynamic library is that a program is much more susceptible to breaking, If a dynamic library for example becomes corrupt, the executable file may no longer work.