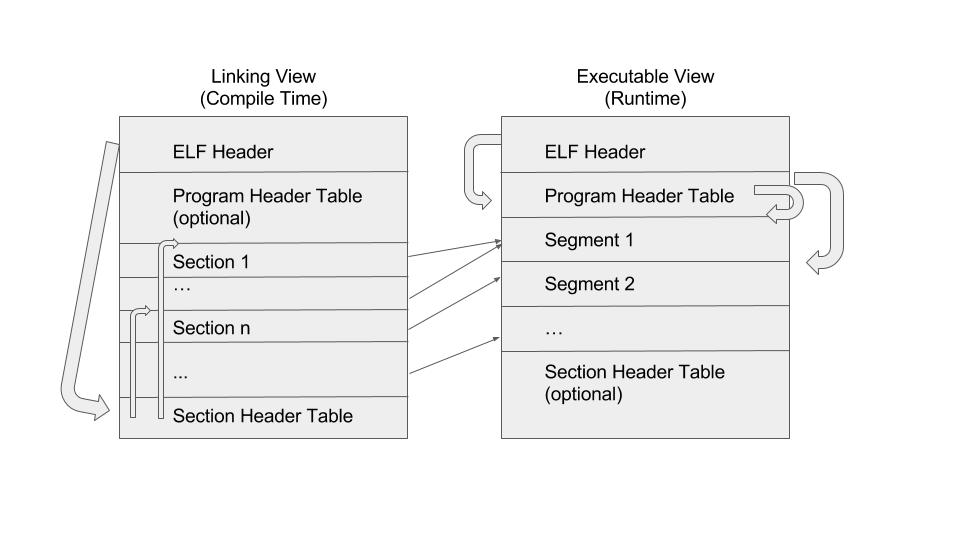
Group: JoJo on the Radio

Jonathan Bassi - Connor Francis - Crisrael Lucero

Homework 1: ELF Files

ELF stands for “Executable and Linking Format.” This format standardizes the format for executable software, object code files, and shared (read: imported) libraries. Before we can talk about the format, we should talk about the process that gets us to the format: compilation. In the most simplest terms, compiling code into the ELF files is the same as taking a program that someone wrote in high-level (human readable) code and *translating* it into something a computer can read (machine code). Once compilation or *translation* is finished and you have files in the ELF format, we can start dissecting the format and how the computer can read it. Code must be compiled before it can be ran by the computer, so we will distinguish the difference as compilation and runtime.

The format contains a single ELF Header and File Data. The File Data contains the Program Header Table, the Section Header Table, and Data (Sections and Segments) used throughout the program. Different portions of the ELF file show up during it’s two views: Linking View and Execution View. These are shown in the diagram below and each key word will be explained.



**ELF Header** - This can be likened to a road map of the object files because it defines the *offset* or starting locations of the Program Header Table and Section Header Table. It defines the sizes of the addresses, so that programs won’t unexpectedly run into addresses too small or large.

**Program Header Table** - This table is optional during the linking view. During the execution view, this table tells the system how to to create a *process image*. These are further described by the Segments.

**Section Header Table** - This table is optional during the execution view. This table contains information about the file’s sections and where to locate them; each section has an entry on this table, which will be further described by the Sections.

**Sections** - Information needed during compile time (when program is being *translated*). Sections contain important data for linking and relocation, instructions, and a symbol table. Symbol tables contain a formatted list of program entry points and variables used within the file.

**Segments** - Information needed during run time (when program is running). Segments contain zero or more Sections, depending on if the segment of code being used is from another library.

**Linking View -** The view being used during compilation or *translation* time.

**Execution View -** The view being used during runtime.

Through the explanation and diagram, it is easy to see that when compilation starts, we are put into the Linking View and the ELF Header points to the *offset* of the Section Header Table. The Section Header Table will then point to each section, which contains information such as where to find referenced structures. When the program is compiled and *read* or executed by the computer, we enter the Executable View, in which the ELF Header points to the *offset* of the Program Header Table. The Program Header Table points to each Segment which is called upon executing the program, and each Segment contains zero or more Sections to execute code off of.