1) An Executable and Linkable Format (ELF) file is a common file format used for object files that will be executed in a Linux system. These types of files are designed to easily load the necessary data into the memory dynamically at run-time. There are different types of ELF files, but they all share the same concept, which is they are comprised of sections. Each file first begins with an ELF header, which declares whether the program requires 32-bit or 64-bit addressing. Along with the file header, there could also be a Program header or a Section header. The Program header gives the address of the executable code, while the Section header which points to the Data section.

As shown in the diagram, there are several different sections in each ELF file, but from a general sense there are 3 that are the main focus. These sections are the .text, .data, and .bss sections. The executable code is always located in the .text section. This section is always loaded into the appropriate amount of memory needed, then it is marked as ‘read-only’ meaning it cannot be changed. This is basically our program. Next, the .data section is basically our data, for a lack of better word. All of our variables that we set will be placed in this section until they are needed by the program. Now, if we have any variables that we didn’t set with a value those are placed in the .bss section.

When an ELF file is first executed, the kernel (the manager of the O.S.) goes to the header to find out where everything is located. It first goes to the .text section then properly loads the code inside into the proper memory space. Next, it jumps to the .data section, and loads the set variables into the address space. Finally, the kernel looks at the .bss section an opens up enough memory to hold the variables when they are set. Although they have no value, the kernel knows what type of data the variable will hold thus allowing it to open up the maximum potential space that would be needed once it is declared. By breaking up the file into separate sections, the kernel is able to quickly and efficiently load all the necessary data into the memory, which in the end makes for a quicker overall process.