Hunter Damron

Graham McDonald

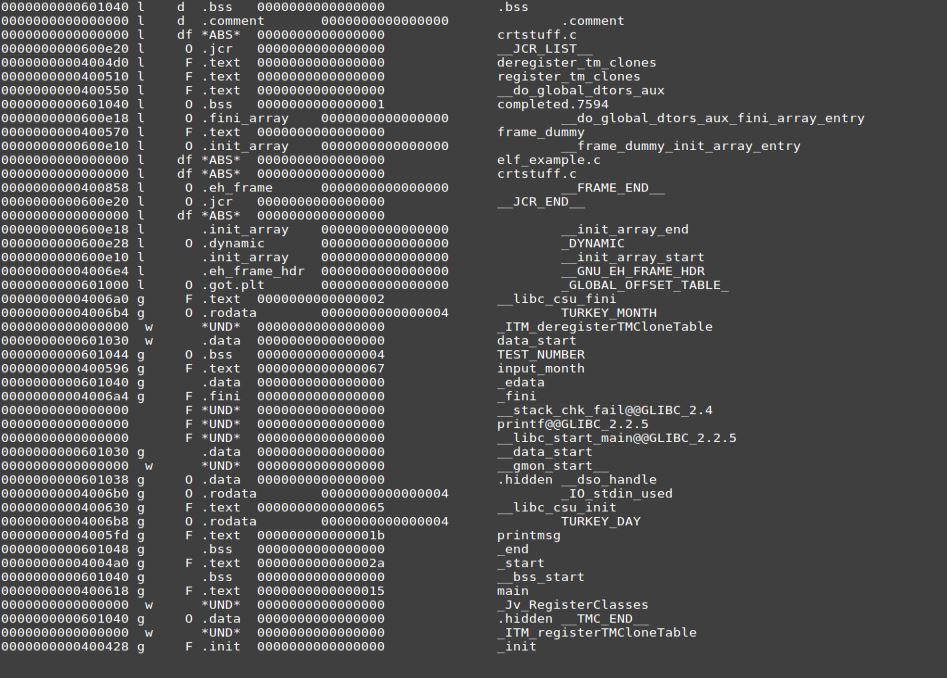
Reed Segars

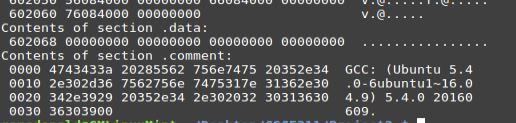
ELF Analysis

The Executable and Linkable Format (ELF) is the generic file format for executables in Linux based systems. It defines the structure for binaries, libraries, and core files. The three major components include the ELF header, sections, and segments. In our analysis, we created a simple program (elf\_example.cc) with the binary executable ./elf. This program simply prompts the user for the correct month (11) and day (22) of Thanksgiving and does some error checking. The code was broken up into several functions and made use of constants in order to better illustrate the ELF file’s sections and segments.

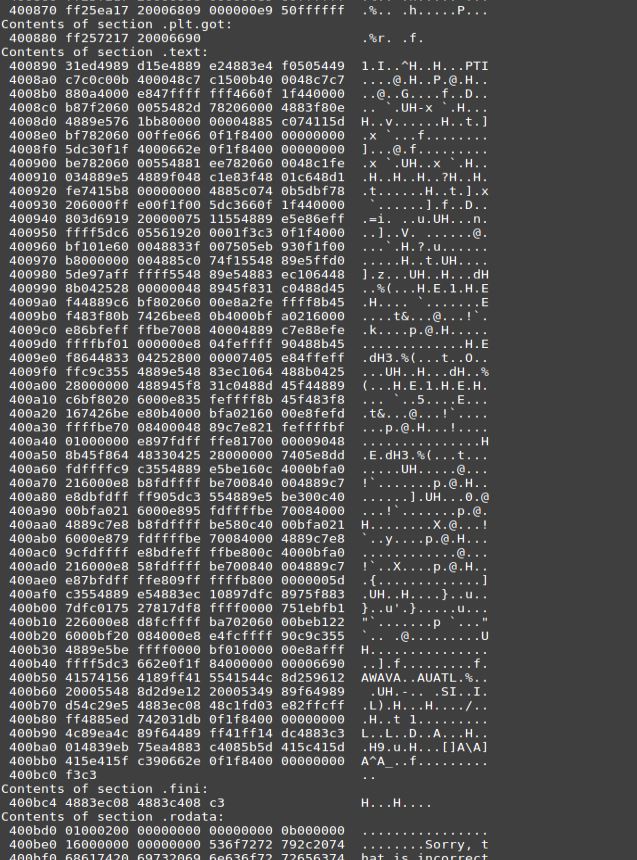
Sections contain the information needed for linking an object file to form an executable.

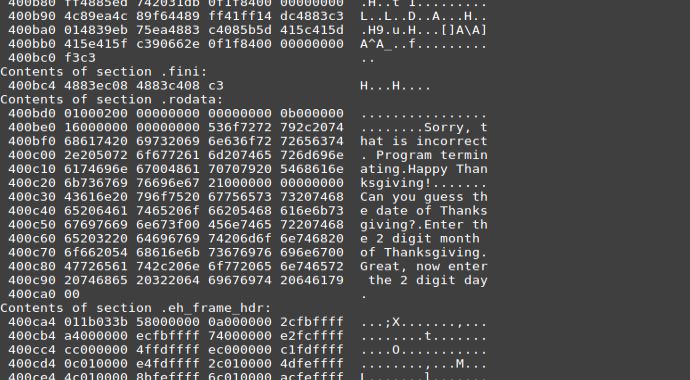
TEST\_NUMBER from the elf\_example.c program is stored in the .bss section. The variables input\_month, TURKEY\_MONTH, and TURKEY\_DAY from the elf\_example.cc program are stored in the .text section. The strings inside of the elf\_example.cc program are stored in the .rodata section. The compiler and operating system information are stored in the .comment section. The .data section is empty. During runtime, global variables will be retrieved from the .text section into .data section. The .interp section stores the dynamic linker file path.





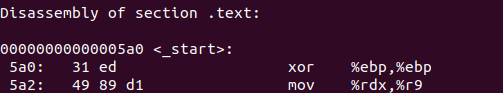


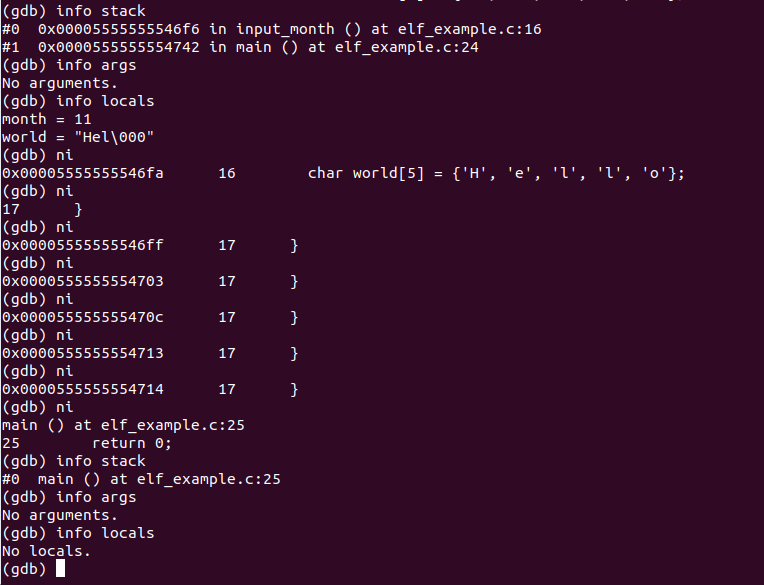


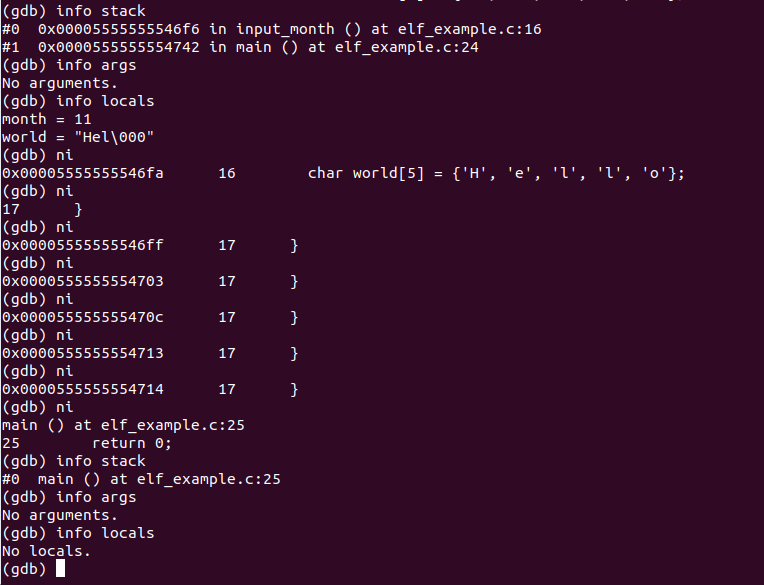


Segments break down the structure of an ELF executable into small chunks to prepare it to be loaded into memory. In terms of the memory address space, the common segments include code (text), data, stack, and heap. In our example, the .text section of the ELF file is stored in the code segment of the memory address space. This can be seen by examining the disassembly of the .text section which shows the code text being translated to assembly language instructions.





Another segment is the call stack segment. The call stack stores function calls and local variables of those function calls for temporary use during program execution. Using gdb, we can see that the functions and local variables are pushed and popped off the stack during execution.



The last segment we examined was the data segment. The data segment contains the static (global) variables that exist throughout program execution. In our code, we had two initialized global variables, TURKEY\_MONTH and TURKEY\_DAY. In the object dump, we can see that this data is present through the assembly language. There is an lea operation which calculates the effective address of the constant as well as a compare operation which compares the value stored in %eax (0xb or 11) with that of TURKEY\_MONTH which is stored in a stack frame register.

