ELF2 4A Dynamic Linking

Young W. Lim

2019-10-28 Mon

Outline

1. Based on

2. Shared Object File

3. Run time linking

Based on

Oracle Document

https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter4-1/index.html

Compling 32-bit program on 64-bit gcc

- `gcc -v`
- `gcc -m32 t.c`
- `sudo apt-get install gcc-multilib`
- `sudo apt-get install g++-multilib`
- `gcc-multilib`
- `g++-multilib`
- `gcc -m32`
- `objdump -m i386`

PIC shared object

- When a shared object is built from PIC, relocatable references are generated as indirections through data in the shared object's data segment.
- The code within the text segment requires no modification.
- All relocation updates are applied to corresponding entries within the data segment.

https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter4-1/index.html

GOT (1)

- Position-independent code cannot, in general, contain absolute virtual addresses.
- Global offset tables hold absolute addresses in private data
- Addresses are therefore available without compromising the position-independence and shareability of a program's text.
- A program references its GOT using position-independent addressing and extracts absolute values.

- This technique redirects position-independent references to absolute locations.

https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter4-1/index.html

GOT (2)

- Initially, the GOT holds information
  as required by its relocation entries.
- After the system creates memory segments for a loadable object file,
  the runtime linker processes the relocation entries.
- Some relocations can be of type R_xxxx_GLOB_DAT,
  referring to the GOT.

https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter4-1/index.html

GOT (3)

- The runtime linker determines the associated symbol values,
  calculates their absolute addresses,
  and sets the appropriate memory table entries to the proper values.
- Although the absolute addresses are unknown
  when the link-editor creates an object file,
  the runtime linker knows the addresses of all memory segments and can thus calculate the absolute addresses of the symbols contained therein.

`https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter4-1/index.html`

GOT (4)

- If a program requires direct access to the absolute address of a symbol, that symbol will have a GOT entry.
- Because the executable file and shared objects have separate a GOT, a symbol's address can appear in several tables.
- The runtime linker processes all the GOT relocations before giving control to any code in the process image.
- This processing ensures that absolute addresses are available during execution.

`https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter4-1/index.html`

GOT (5)

- The system can choose different memory segment addresses for the same shared object in different programs.
- The system can even choose different library addresses for different executions of the same program.
- Nonetheless, memory segments do not change addresses once the process image is established.
- As long as a process exists, its memory segments reside at fixed virtual addresses.

- A GOT format and interpretation are processor-specific.
- The symbol GLOBAL_OFFSET_TABLE can be used to access the table.
- This symbol can reside in the middle of the .got section,

https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter4-1/index.html

run time linking (1)

- runtime linking involves the binding of objects, usually generated from one or more previous link-edits, to generate a runnable process.
- During the generation of these objects by the link-editor, appropriate bookkeeping information is produced to represent the verified binding requirements.
- This information enables the runtime linker to load, relocate, and complete the binding process.

`https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter1-3/index.html`

run time linking (2)

- During process execution, the facilities of the runtime linker are made available.
- These facilities can be used to extend the process' address space by adding additional shared objects on demand.
- The two most common components involved in runtime linking are dynamic executables and shared objects.

https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter1-3/index.html

run time linking (3)

- Dynamic executables are applications that are executed under the control of a runtime linker.
- These applications usually have dependencies in the form of shared objects, which are located, and bound by the runtime linker to create a runnable process.
- Dynamic executables are the default output file generated by the link-editor.

`https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter1-3/index.html`

run time linking (4)

- Shared objects provide the key building-block to a dynamically linked system.
- A shared object is similar to a dynamic executable, however, shared objects have not yet been assigned a virtual address.

https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter1-3/index.html

run time linking (5)

- Dynamic executables usually have dependencies on one or more shared objects.
- Typically, one or more shared objects must be bound to the dynamic executable to produce a runnable process.
- Because shared objects can be used by many applications, aspects of their construction directly affect shareability, versioning, and performance.

https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter1-3/index.html

run time linking (6)

- Shared object processing by the link-editor or the runtime linker can be distinguished by the environment
  in which the shared object is used.
  - compilation environment
  - runtime environment

https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter1-3/index.html

run time linking (7)

- compilation environment
  - Shared objects are processed by the link-editor to generate dynamic executables or other shared objects. The shared objects become dependencies of the output file being generated.

`https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter1-3/index.html`

run time linking (8)

- runtime environment
  - Shared objects are processed by the runtime linker,
    together with a dynamic executable,
    to produce a runnable process.

`https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter1-3/index.html`

run time linking ()

https://docs.oracle.com/cd/E19120-01/open.solaris/819-0690/chapter1-3/index.html