Link 3. Shared Library

Young W. Lim

2022-09-24 Sat

Outline

Based on

"Study of ELF loading and relocs", 1999
http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.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

Creating shared libary

- *create* a file `library.c`
- *compile* the `library.c` file

```
gcc -shared -fPIC -o liblibrary.so library.c
```

  - `-shared` instructs the compiler
    that we are building a shared library
  - `-fPIC` is to generate position-independent code
  - generates a shared library `liblibrary.so`
    in the current working directory.
  - We have our shared object file
    (shared library name in Linux)
    ready to use.

`https://www.geeksforgeeks.org/working-with-shared-libraries-set-2/`

Using shared libary (1)

- *create* a file `application.c`
- *compile* the `application.c` file

```
gcc application.c -L /home/coding/ -library -o sample
```

  - `-library` instructs the compiler
    to look for symbol definitions
    that are not available in the current code
  - the option `-L` is a hint to the compiler
    to look in the directory followed by the option
    for any shared libraries (during link-time only).
  - generates an executable named `sample`

`https://www.geeksforgeeks.org/working-with-shared-libraries-set-2/`

Using shared libary (2)

- By default, it will not look into the current working directory
- You have to explicitly instruct the tool chain to provide proper paths
- otherwise, when you invoke the executable, the dynamic linker will not be able to find the required shared library

https://www.geeksforgeeks.org/working-with-shared-libraries-set-2/

Using shared libary (3)

- The dynamic linker searches standard paths
  available in the LD_LIBRARY_PATH
  and also searches in the system cache
- We have to add our *working directory*
  to the LD_LIBRARY_PATH environment variable

  ```
  export LD_LIBRARY_PATH=/home/work/:$LD_LIBRARY_PATH
  ```
- You can now invoke our executable as shown.
  ```
  ./sample
  ```

https://www.geeksforgeeks.org/working-with-shared-libraries-set-2/

gcc -llibrary (1)

- `-llibrary`
  `-l library`
  - Search the library named `library` when linking
    - the second alternative with the library `-l library`
      as a separate argument is only for POSIX compliance
      and is not recommended
  - The `-l` option is passed directly to the linker by GCC.
  - the linker searches
    a standard list of directories for the library.
    - the directories searched include
      several standard system directories
      plus any that you specify with `-L`

`man gcc`

gcc -llibrary (2)

- -llibrary
  -l library
  - static libraries are archives of object files,
    and have file names like liblibrary.a
    some targets also support shared libraries,
    which typically have names like liblibrary.so
    if both static and shared libraries are found,
    the linker gives preference to linking
    with the shared library
    unless the -static option is used.

man gcc

gcc -llibrary (3)

- -llibrary
  -l library
  - it makes a difference where in the command you write this option;
    the linker searches and processes libraries and object files
    in the order they are specified.
    - Thus, foo.o -lz bar.o searches library z
      after file foo.o but before bar.o
    - if bar.o refers to functions in z,
      those functions may not be loaded.

man gcc

gcc -fpic (1)

- -fpic
  - generate position-independent code (PIC) suitable for use in a shared library, if supported for the target machine.
  - Such code accesses all constant addresses through a global offset table (GOT).
  - The dynamic loader resolves the GOT entries when the program starts
    - the dynamic loader is not part of GCC; it is part of the operating system

```
man gcc
```

gcc -fpic (2)

- -fpic
  - If the GOT size for the linked executable
    exceeds a machine-specific maximum size,
    you get an error message from the linker
    indicating that -fpic does not work;
    in that case, recompile with -fPIC instead.
    - These maximums are 8k on the SPARC,
      28k on AArch64 and 32k on the m68k and RS/6000.
    - The x86 has no such limit

```
man gcc
```

gcc -fPIC

- -fPIC
  - if supported for the target machine,
    emit position-independent code,
    suitable for dynamic linking and
    avoiding any limit on the size of the global offset table
    - This option makes a difference on
      AArch64, m68k, PowerPC and SPARC (not on x86)
  - Position-independent code requires special support,
    and therefore works only on certain machines.
  - When this flag is set, the macros __pic__ and __PIC__
    are defined to 2.

```
man gcc
```

gcc -shared (1)

- `-shared`
  - produce a shared object
    which can then be *linked* with other objects
    to form an executable
  - not all systems support this option.
  - For predictable results, you must also specify
    the same set of options used for *compilation*
    (`-fpic`, `-fPIC`, or model suboptions)
    when you specify this linker option

```
man gcc
```

gcc -shared (2)

- Shared libraries and executables use the same format:
  they are both loadable images. However,
  - shared libraries are usually position-independent,
    executables are often not
    - This affects code generation: for position-independent
      you have to load globals or jump to functions
      using relative addresses
  - executables have an entry point
    which is where execution starts.
    - this is usually not `main()`,
      because `main()` is a function, and functions return,
      but execution should never return from the entry point

`https://stackoverflow.com/questions/25084855/how-does-gcc-shared-option-affect-the`

gcc -shared (3)

- parameters for collect2 without -shared:

```
-dynamic-linker
/lib64/ld-linux-x86-64.so.2
/usr/lib/gcc/x86_64-linux-gnu/4.7/../../../x86_64-linux-gnu/crt1.o
/usr/lib/gcc/x86_64-linux-gnu/4.7/crtbegin.o
/usr/lib/gcc/x86_64-linux-gnu/4.7/crtend.o
```

- parameters for collect2 with -shared:

```
-shared
/usr/lib/gcc/x86_64-linux-gnu/4.7/crtbeginS.o
/usr/lib/gcc/x86_64-linux-gnu/4.7/crtendS.o
```

https://stackoverflow.com/questions/25084855/how-does-gcc-shared-option-affect-the

gcc -shared (4)

- you still have to use -fpic or -fPIC, when -shared is used
- it looks like code generation is not affected:
- crt1.o (the C runtime) is only included
  when linking the executable,
  and thus when -shared is not used
  or when Wl,-shared is used

https://stackoverflow.com/questions/25084855/how-does-gcc-shared-option-affect-the

gcc -shared (5)

- Using nm crt1.o

```
$ nm /usr/lib/x86_64-linux-gnu/crt1.o
0000000000000000 R _IO_stdin_used
0000000000000000 D __data_start
                 U __libc_csu_fini
                 U __libc_csu_init
                 U __libc_start_main
0000000000000000 T _start
0000000000000000 W data_start
                 U main
```

- seems to define something to do with stdin, as well as _start (which is the entry point),
- has an undefined reference to main

https://stackoverflow.com/questions/25084855/how-does-gcc-shared-option-affect-the

gcc -shared (6)

- passing -shared to gcc (gcc -shared)
  - gcc -shared -Wl,-soname,libtest.so -o libtest.so *.o
  - may enable or disable other flags at link time.
    different *~crt*~* files might be involved.
  - To get more information, grep for -shared
    in GCC's gcc/config/ directory and subdirectories.
- passing -shared to ld (gcc -Wl,-shared).
  - gcc -Wl,-shared -Wl,-soname,libtest.so -o libtest.so *.o
  - on i386 FreeBSD, gcc -shared will link in object file crtendS.o,
    while without -shared, it will link in crtend.o instead.

https://stackoverflow.com/questions/4623915/difference-between-shared-and-wl-share

gcc -Ldir (1)

- -Ldir
  - Add directory dir to the list of directories to be searched for -l.
- -Lsearchdir
  --library-path=searchdir
- Add path searchdir to the list of paths
  that ld will search for archive libraries and ld control scripts.
- You may use this option any number of times.
- The directories are searched in the order
  in which they are specified on the command line.
- Directories specified on the command line are searched
  before the default directories.
- All -L options apply to all -l options,
  regardless of the order in which the options appear.

http://ftp.gnu.org/old-gnu/Manuals/ld-2.9.1/html_node/ld_3.html

gcc -Ldir (2)

- -Ldir
  - Add directory dir to the list of directories to be searched for -l.
- -Lsearchdir
  --library-path=searchdir
- The default set of paths searched (without using -L)
  depends on which emulation mode ld is using,
  and in some cases also on how it was configured.
- See section Environment Variables.
- The paths can also be specified in a link script
  with the SEARCH_DIR command.
- Directories specified this way are searched at the point
  in which the linker script appears in the command line.

http://ftp.gnu.org/old-gnu/Manuals/ld-2.9.1/html_node/ld_3.html

gcc -Wl,option

- -Wl,/option/
  - pass option as an *option* to the linker.
  - if option contains *commas*,
    it is split into multiple options at the *commas*
  - use the *comma* (,) syntax to *pass* an argument to the option.
    `-Wl,-Map,output.map`
    passes `-Map output.map` to the linker.
  - When using the GNU linker, you can also
    get the same effect with =
    `-Wl,-Map=output.map`

`man gcc`

ld -R filename

- -R filename
  --just-symbols=filename
  - read *symbol names* and their *addresses* from filename
  - but do not *relocate* it or *include* it in the output.
    - this allows your output file to refer symbolically
      to absolute locations of memory defined in other programs.
  - may use this option more than once
  - for compatibility with other ELF linkers,
    if the -R option is followed by a directory name,
    rather than a file name, it is treated as the -rpath option.

```
man ld
```

ld -rpath=dir (1)

- `-rpath=dir`
  - add a directory to the runtime library search path
  - used when linking an ELF executable with shared objects
  - all `-rpath` arguments are *concatenated* and *passed* to the runtime linker, which uses them to *locate* shared objects at runtime

```
man ld
```

ld -rpath=dir (2)

- -rpath=dir
  - also used when *locating* shared objects which are needed by shared objects explicitly included in the link;
    - see the description of the -rpath-link option.
    - Searching -rpath in this way is only supported by native linkers and cross linkers which have been configured with the --with-sysroot option.

```
man ld
```

ld -rpath=dir (3)

- -rpath=dir
  - if -rpath is not used when *linking* an ELF executable, the contents of the environment variable LD_RUN_PATH will be *used* if it is defined.
  - If a -rpath option is used, the runtime search path will be formed *exclusively* using the -rpath options, *ignoring* the -L options.
  - This can be useful when using gcc, which adds many -L options which may be on NFS mounted file systems.

```
man ld
```

ld -rpath=dir (4)

- -rpath=dir
  - for compatibility with other ELF linkers, if the -R option is followed by a directory name, *rather than* a file name, it is treated as the -rpath option.
  - the -rpath option may also be used on SunOS.
  - By default, on SunOS, the linker will form a runtime search path out of all the -L options it is given.

```
man ld
```