Module02-02
Linux 开发环境：使用 gcc/g++

Linux 开发环境 - 使用 gcc/g++

- vim
- ➔ 使用 gcc/g++
- make 和 makefile
- 使用 gdb
- CVS
- Eclipse CDT

使用 gcc/g++

这节的课程中，我们将熟悉下列内容：

- 关于 GCC
- 生成可执行文件
- 生成共享库
- 生成静态库
- 调用共享库
- 调用静态库
- 共享库路径问题
- gcc/g++ 常用选项

使用 gcc/g++ - 关于 GCC

- 关于 GCC：
  - GCC - GNU Compiler Collection， GNU 编译器套件
  - 1985 年有自由软件基金会的创立者 Richard M Stallman 开始开发
  - 现已包含支持多种语言的编译器：
    - C (gcc)、 C++ (g++)
    - Objective-C、 Objective-C++
    - Fortran (GNat)
    - Ada (GFortran)
    - Java (GCJ 编译器、 GIJ 解释器)
    - 其他语言如： Pascal、 PL/I、 D 等

使用 gcc/g++ - 关于 GCC

- 支持的系统平台：
  - 支持几乎所有 UNIX 和类 UNIX 平台
    - UNIX / BSD / Mac OSX / Linux
  - 通过多种途径支持 Windows
    - MinGW： Minimalist GNU for Windows
    - Cygwin
- 支持的硬件平台：
  - 是目前支持硬件平台最为广泛的编译器套件，支持众多的硬件平台：从嵌入式设备到大型机平台，都有 GCC 的移植版本

使用 gcc/g++ - 生成可执行文件

- 生成可执行文件（以 g++ 为例）：
  - 1，不加任何选项，生成名为 a.out 的可执行文件

```
$ g++ hello.cpp
$ ls
a.out  hello.cpp
```

  - 2，先生成目标文件 (.o 文件)，再生成（指定名称的）可执行文件

```
$ g++ -c hello.cpp
$ ls
a.out  hello.cpp  hello.o
$ g++ -o"hello" hello.o
$ ls
a.out  hello  hello.cpp  hello.o
```

使用 gcc/g++ - 生成共享库

- 生成共享库（以 g++ 为例）：
  - 本次项目的目录结构

```
./speak
|--- ./test_drv/
|      |--- test_drv.cpp
|      +--- test_drv          # 可执行文件，用于测试 speak 库功能
|--- ./inc/
|      |--- say_goodbye.h
|      +--- say_hello.h
|--- ./src/
|      |--- say_goodbye.cpp
|      |--- say_goodbye.o     # 目标文件（object 文件）
|      |--- say_hello.cpp
|      +--- say_hello.o       # 目标文件（object 文件）
|--- ./lib/
|      |--- libspeak_a.a      # 静态库
|      +--- libspeak.so       # 共享库
|
```

使用 gcc/g++ - 生成共享库

- 生成共享库（以 g++ 为例）（续）：
  - 共享库的命名

    lib(前缀)+ 库名 +.so(后缀)+[. 版本]，如： libspeak.so.1.0.9
  - 1、生成目标文件 (.o 文件)

```
$ cd src
$ g++ -c say_goodby.cpp -I"../inc"   # 注意 -I  选项！
$ g++ -c say_hello.cpp -I"../inc"
$ ls *.o
say_goodby.o  say_hello.o
```

  - 2、生成共享库的 2 种常用方式：

```
#1，使用 -shared 选项， GCC 风格
$ g++ -shared -o ../lib/libspeak.so say_goodby.o say_hello.o
$
#2，调用 ld 命令
$ ld -G -o ../lib/libspeak.so say_goodby.o say_hello.o
```

使用 gcc/g++ - 生成静态库

- 生成静态库（以 g++ 为例）：
  - 静态库的命名

    lib(前缀)+ 库名 +.a ， 如： libspeak.a
  - 1 、生成目标文件 (.o 文件)

```
$ cd src
$ g++ -c say_goodbye.cpp -I"../inc"   # 注意 -I  选项！
$ g++ -c say_hello.cpp -I"../inc"
$ ls *.o
say_goodby.o  say_hello.o
```

  - 2 、生成静态库

```
# 使用 ar 命令
$ ar -r ../lib/libspeak_a.a say_goodbye.o say_hello.o
```

使用 gcc/g++ - 调用共享库

- 调用共享库
  - 编译及链接共享库

```
$ cd ../test_drv
$ ls
test_drv.cpp
$ g++ -o test_drv test_drv.cpp -I"../inc" -L"../lib"
-lspeak     # 注意 -L 和 -l 选项
$ ls
test_drv  test_drv.cpp
```

  - 运行

```
# 一定要注意: 设置 LD_LIBRARY_PATH 环境变量
$ export LD_LIBRARY_PATH="../lib:$LD_LIBRARY_PATH"
$ ./test_drv
......
```

使用 gcc/g++ - 调用静态库

- 调用静态库
  - 编译及链接静态库

```
$ cd ../test_drv
$ ls
test_drv.cpp
$ g++ -o test_drv test_drv.cpp -I"../inc" -L"../lib"
-lspeak_a     # 注意 -L 和 -l 选项
$ ls
test_drv  test_drv.cpp
```

  - 运行

```
$ ./test_drv
......
```

使用 gcc/g++ - 共享库路径问题

- 解决共享库路径问题
  - gcc/g++ 编译期：
    - 使用 -L 选项指定所需链接的共享库所在目录（如果共享库或其连接文件位于 /usr/lib 、 /usr/local/lib 或由 LD_LIBRARY_PATH 指定的目录下，则不需指定该选项）
    - 使用 -l 选项指定所需链接的共享库名称（无前缀、后缀）
  - 程序运行期：
    - 使用 LD_LIBRARY_PATH 环境变量来指定所需引用的共享库所在目录 (临时指定或永久性写入相关文件，普遍适用于 UNIX 世界)
    - 使用 ldconfig 机制（需 root 权限）：
      - 首先，在 /etc/ld.so.conf.d/ 下创建一个 .conf 文件，如 libspeak.conf ，内容为共享库所在目录的绝对路径
      - 然后，运行 ldconfig

使用 gcc/g++ - 常用选项

- gcc/g++ 常用选项

选项	说明
-c	生成目标文件（.o 文件，object file）
-o	指定输出文件的名称，如 -o"hello.o"、-o libspeak.so
-I	指定头文件搜索路径
-L	指定需链接的库文件路径
-l	指定需链接的库名（不包括前缀、后缀）
-g	在目标文件中包含 debug 信息
-O	指定优化级别，O0 不优化，O1~O3 不同级别的优化
-Wall	打开所有警告信息
-w	禁止警告信息

使用 gcc/g++ - 小结

- 需要特别注意的是共享库的路径问题
- gcc/g++ 的功能远远不止于我们所列举的这些内容，但常规工作都可以通过这小节列举的操作来完成，更详细的介绍，请 man gcc
- 编译与链接是非常深奥的课题，如果想比较透彻的了解这个过程，需研读编译原理方面的书籍