Lab02: GCC & GDB & Makefile

1. 实验目的

通过使用 GCC & GDB & Makefile 熟悉 Linux 中三种常用工具的基本操作，深入了解 Linux 编程环境。

提示：

- 开始实验前请仔细阅读书中第二章内容。

2. 实验内容

- GCC/GDB 基本使用方法
- GDB 调试功能
- 静态库和动态库的创建和使用
- 熟悉 make 命令的使用和 Makefile 中的规则编写

3. 实验指南

3.1. GCC

- 无选项编译链接
    - 用法：gcc test.c

    作用：将 test.c 预处理、编译、汇编并链接形成可执行文件。这里未指定输出文件，默认输出为 a.out。
- 选项 -o
    - 用法：gcc test.c -o test

    作用：将 test.c 预处理、编译、汇编并链接形成可执行文件 test。-o 选项用来指定输出文件的文件名。
- 选项 -E

- 用法：gcc -E test.c -o test.i

  作用：将 test.c 预处理输出 test.i 文件。

- 选项 -S
  - 用法：gcc -S test.i

  作用：将预处理输出文件 test.i 编译成 test.s 文件。

- 选项 -c
  - 用法：gcc -c test.s

  作用：将汇编语言文件 test.s 汇编成目标代码 test.o 文件。

- 无选项链接
  - 用法：gcc test.o -o test

  作用：将目标代码文件 test.o 链接成最终可执行文件 test。

- 选项 -O
  - 用法：gcc -O1 test.c -o test

  作用：使用编译优化级别 1 编译程序。级别为 1~3，级别越大优化效果越好，但编译时间越长。

官方文档：GCC, the GNU Compiler Collection

3.2. GDB 调试器的使用

- 在使用 gcc 对程序编译时，需要**加上-g 参数**（产生调试信息）才能使 GDB 进行调试。
- 输入 help 命令获得帮助
- 输入 quit 或者按 Ctrl+D 组合键退出 GDB。
- 启动程序准备调试方法
  - 方法一：在执行 GDB 命令时加上要调试的可执行程序名称，如“GDB yourprogram”；
  - 方法二：先输入 GDB，在 GDB 中输入 file yourprogram 加载需要调试的程序。最后使用 run 或者 r 命令开始执行，也可以使用 run parameter 方式传递参数

命令	命令缩写	命令说明
list	l	显示多行源代码
break	b	设置断点，程序运行到断点的位置会停下来
info	i	描述程序运行的状态
run	r	开始运行程序
display	disp	跟踪查看某个变量，每次停下来都显示它的值
step	s	执行下一条语句，若该语句为函数调用，则进入函数执行其第一条语句
next	n	执行下一条语句，若该语句为函数调用，不会进入函数内部执行（即不会一步一步地调试函数内部语句）
print	p	打印内部变量
continue	c	继续程序的执行直到遇到下一个断点

命令	命令缩写	命令说明
set var		设置变量的值
start		开始执行程序，在 main 函数第一条语句前面停下
file		装入需要调试的文件
kill	k	终止正在调试的程序
watch		监视变量值的变化
backtrace	bt	查看函数调用的信息
frame	f	查看栈帧
quit	q	退出 GDB 环境

- 扩展：GDB Text User Interface

  > The GDB Text User Interface (TUI) is a terminal interface which uses the `curses` library to show the source file, the assembly output, the program registers and GDB commands in separate text windows. The TUI mode is supported only on platforms where a suitable version of the `curses` library is available.
  >
  > The TUI mode is enabled by default when you invoke GDB as ‘gdb -tui’. You can also switch in and out of TUI mode while GDB runs by using various TUI commands and key bindings, such as `tui enable` or C-x C-a. See TUI Commands, and TUI Key Bindings.

  GDB 的更多使用方法可以参阅**GDB User Manual** (PDF)

3.3. Makefile 的使用

> 代码变成可执行文件，叫做编译（compile）；先编译这个，还是先编译那个（即编译的安排），叫做构建（build）。Make是最常用的构建工具，诞生于 1977 年，主要用于 C 语言的项目。但是实际上，任何只要某个文件有变化，就要重新构建的项目，都可以用 Make 构建。

- 当使用 make 构建项目时，具体的构建规则被写在一个叫做 Makefile 的文件中。本次课程的内容有了解 make 命令、了解 Makefile 文件的概念并学习 Makefile 文件的基本语法。还可以参考阮一峰的 make 命令教程。遇到不懂的地方可以查阅官方文档GNU make。

注：教材 P49“vpath %.xyz” 中间不应该有'.',应为'vpath % xyz'

4. 实验习题

- GCC 理解（请用命令行操作并给出截图）
  - 在 Linux 下创建并写入一个 C 程序文件，可以输出 `hello,world!` ，命名为 `test.c` 。
  - GCC 将一个源程序转换为可执行文件经历了哪些主要步骤?
  - 请利用 `test.c` 用 GCC 命令将其将其转换为可执行程序的主要过程表示出来。
- 静态库和动态库的操作

- 解压 lab02.zip
- 静态库
  - 使用 gcc 命令分别将 `mytool1.c` 和 `mytool2.c` 编译成 .o 目标文件
  - 执行下面两个命令

    ```
    ar cr libmylib.a mytool2.o mytool1.o
    ```

    ```
    gcc -o main main.c -L. -lmylib
    ```

    说明上述两个命令完成了什么事?
  - 查看 `main` 文件大小，并记录
  - 执行 `./main`
  - 删除之前生成的静态库文件，重新执行 `./main` 命令，对比上一步骤得到的结果，你有什么发现？并解释原因。
- 动态库
  - 执行下面两个命令

    ```
    gcc -c -fPIC mytool2.c -o mytool2.o
    gcc -c -fPIC mytool1.c -o mytool1.o
    gcc -shared -o libmylib.so mytool2.o mytool1.o
    ```

    ```
    gcc -o main main.c -L. -lmylib
    ```
  - 查看 `main` 文件大小，并和之前的作比较，解释原因。
  - 执行命令将当前目录添加到库搜索路径中

    ```
    export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:;
    ```
  - 执行 `./main`
  - 删除之前生成的动态库文件，重新执行 `./main` 命令，对比上一步骤得到的结果，你有什么发现?
- 综合实验，你觉得静态库和动态库的区别和相同点是什么？谈谈他们的优缺点。

- GDB 命令操作

  按顺序执行如下操作

  （1）输入如下程序保存为 test.c

```c
#include <stdio.h>
int main() {
    int num;
    do
    {
        printf("Enter a positive integer: ");
        scanf("%d", &num);
    }
    while(num < 0);

    int factorial;
    for(int i = 1; i<=num; i++)
        factorial = factorial*i;

    printf("%d! = %d\n", num, factorial);
    return 0;
```

```
}
```

（2）使用 gcc -g test.c 命令编译生成可执行文件 a.out

（3）执行 gdb a.out 命令

（5）在 main 函数处设置断点

（6）输入 run 命令开始程序

（7）多次输入 next 命令使程序运行到第 13 行,使用 print 命令打印 num 的值

（8）继续调试至程序第 16 行,使用 print 命令打印 factorial 的值

（9）使用 run 命令再次调试程序

（10）在程序第 10 行加入断点

（11）使用 continue 命令使程序运行到断点处

（12）使用 next 命令

（13）使用 print 命令打印 i 和 factorial 的值

（14）使用 p factorial=1 命令改变 factorial 的值

（15）使用 info locals 查看所有局部变量值

（16）继续调试至程序结束

（17）说明源程序中存在的错误

- 综合实验，跟用 printf 函数打印输出相比，采用 gdb 调试的优点有哪些?

- Makefile 操作

  解压 `make_practice.zip` ，补全 Makefile 文件，要求对主模块进行编译，包含 clean 模块可删除目标文件和中间生成文件.

  make_practice 文件夹的目录结构如下

```
.
├── include
│   ├── dylib.h
│   ├── fun1.h
│   └── fun2.h
├── lib
│   └── libdylib.so
├── Makefile
└── src
    ├── fun1.c
    ├── fun2.c
    └── main.c
```

- 综合实验，Make 工具是如何知道哪些文件需要重新生成，哪些不需要的?