桂林电子科技大学

**实验五 C语言开发工具**  实验报告

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | **实验5 C语言开发工具** | | | | | | | |  | 辅导员意见：  成绩 辅导员  签 名 |
| 院 系 | 计算机与信息安全学院 | | | 专业 | | 计算机科学与技术 | | |
| 学 号 | 1600300128 | | | 姓名 | | 王涛 | | |
| 实验日期 | 2018 | 年 | 12 | | 月 | | 15 | 日 |
|  |  | | | | | | | |

## 实验目的

1、 掌握gcc的编译流程、编译方法和使用；

2、 掌握Linux调试器工具GDB的使用；

3、 理解make工具的功能，掌握编写makefile文件进行工程项目管理。

## 实验要求

1、 理解gcc的编译选项，并通过设置选项进行高级操作；

2、 理解掌握gdb的各种调试命令，并对具体实例进行调试；

3、 理解makefile变量和各种语法规则，并对具体工程编写makefile文件。

## 实验步骤

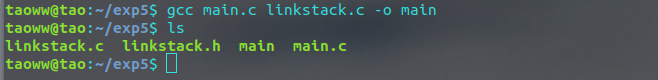
### 一、编译器gcc的使用

1. 在home目录下新建文件夹exp5，将linkstack.h linkstack.c 和main.c 复制到exp5目录

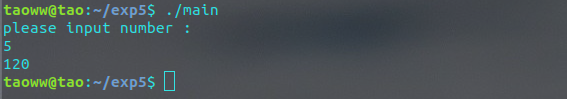


1. 编译linkstack.c 和main.c文件，使用链接选项 -o直接生成可执行文件，并观察exp5目录下内容的变化。

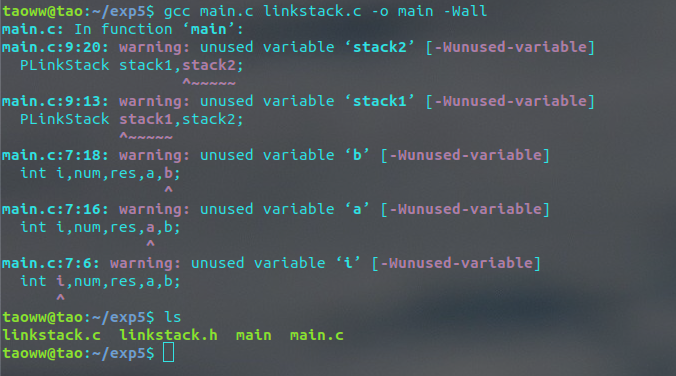
**gcc main.c linkstack.c –o main**



1. 运行main，输入5，记录下运行结果。

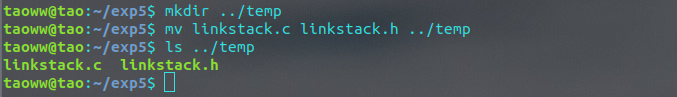


1. 添加编译选项-Wall，重新编译并运行。观察编译提示，说明添加-Wall选项的作用，并说明警告是否影响了输出结果。

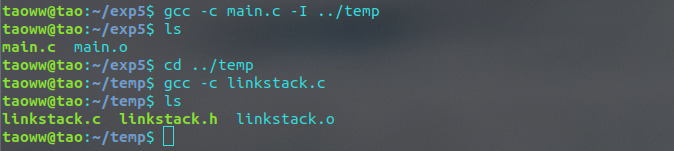


-Wall选项编译后会输出警告信息，不会影响编译输出结果。

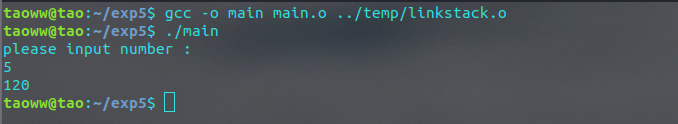
1. 在home目录下新建文件夹temp，将exp5目录下linkstack.h和linkstack.c复制到temp目录，并且删除exp5目录下linkstack.h和linkstack.c



1. 分别编译linkstack.c 和main.c文件，注意两个文件时需要编译选项-c生成.o目标文件，同时编译main.c时需要通过编译选项 -I 指出头文件linkstack.h的路径，注意比较没有指出头文件路径的错误提示。



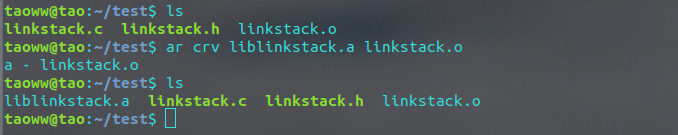
1. 链接main.o和linkstack.o生成可执行文件main，并运行main。



1. 删除main.o、linkstack.o和main

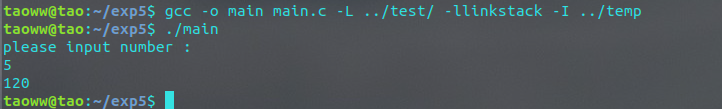


1. 在test目录下编译linkstack.c，并打包成一个静态库liblinkstack.a

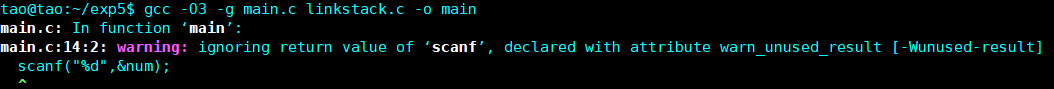


1. 编译main.c生成可执行文件。注意需要设置编译选项-L告诉编译器到哪里找需要的库文件，设置编译选项-I告诉编译器到哪里找需要的头文件。注意即使库文件在当前目录，也要告诉编译器L.

gcc main.c –L ../test/ -llinkstack –o main



1. 删除编译后的文件，只保留原始的.c和.h文件，设置优化等级选项-On，要求优化等级为3级，并设置调试选项-g生成调试信息，重新编译，观察提示，并说明是不是优化等级越高越好。



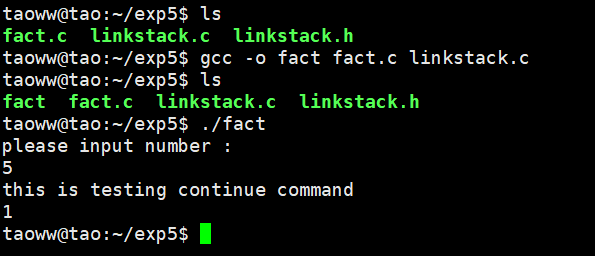
gcc - O1 首先o1上面还有一个O0,那个是不提供任何优化，项目中几乎不会使用，而o1使用就非常广泛了，o1是最基本的优化，主要对代码的分支，表达式，常量来进行优化，编译器会在较短的时间下将代码变得更加短小，这样体积就会变得更小，会减少内存的占用率，在操作系统进行内存调度时就会更快。但是事情没有绝对的优点，当一个庞大的程序被拆碎细分的话，内存占用会大大增加，由于当今系统大多数都是多线程，就会出现卡顿和反应延迟。

gcc - O2 这个优化级别是O1的进阶，在上一级的基础上会进行更严格的细分，最重要的是加入了寄存器的使用。寄存器是cpu中重要的组成部分，此外还有运算器和控制器，计算机顾名思义，要进行各种庞杂的计算，由于CPU速度较快，所以计算的中间结果都会保存在寄存器中，这样可以大大提高系统的效率，但是寄存器造价高昂，数量有限，所以一般来说程序不会放在寄存器中，另一种将代码放在寄存器的方式是使用register修饰变量，适用于频繁调用的变量。

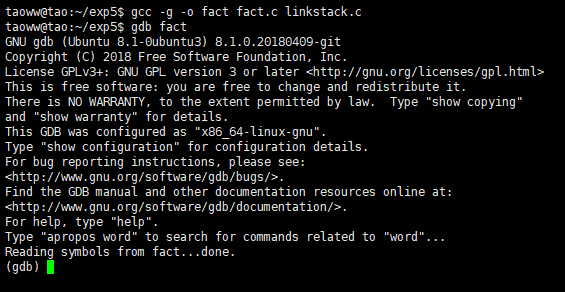
gcc - O3 这个优化属于非常强大的优化，因为编译器会进行预测，对循环每一层的预测，以便于将循环拆分，可以提高执行效率。编译器还会试图用已有的值来代替未知的值，并且还会用加代替乘，因为运算器的特性，乘法十分复杂耗时。当然O3的缺点最明显，那就是O3因为试图预测程序的走向，可能会出现误差，导致错误和程序不可逆转的走向。所以一般O3不建议使用。

### 二、用GDB调试程序BUG

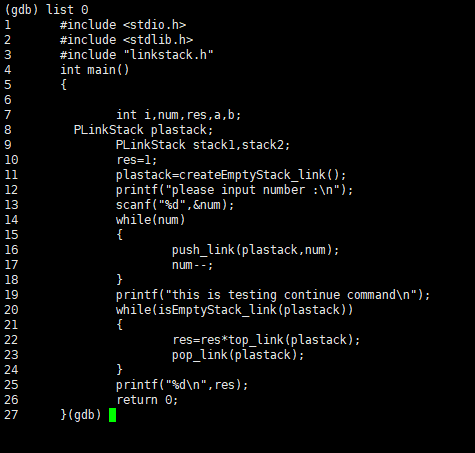
1. fact.c实现阶乘功能，编译运行程序fact.c，输入5，结果显示1，观察结果不正确；



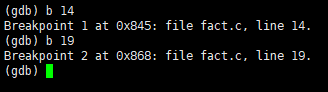
1. 重新编译gcc –g的选项编译这段代码；
2. 启动gdb调试程序



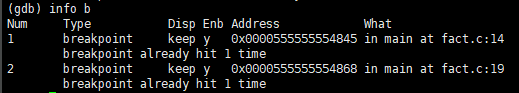
1. 列出源代码



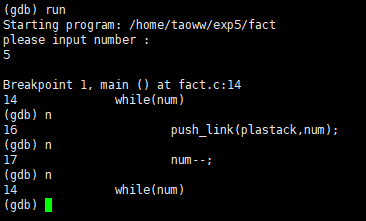
1. 在14行和19行设置断点



1. 执行程序
2. 查看断点



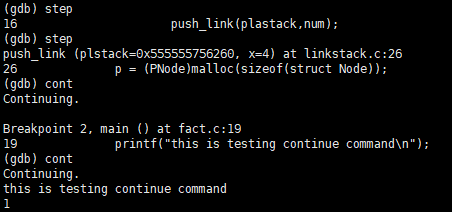
1. 执行单步跟踪



1. 打印num变量以观察代码执行情况



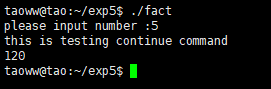
1. 分别执行step，执行cont，察看二者的区别



1. 退出调试



1. 纠正错误，更改源程序并得到正确的结果。

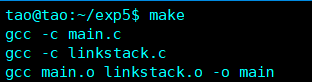


### 三、编写包含多个文件的makefile

* 1. 已知工程proj包含3个源程序linkstack.h linkstack.c 和main.c，为其编写makefile文件。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | main:main.o linkstack.o  gcc main.o linkstack.o -o main  main.o:main.c linkstack.h  gcc -c main.c  linkstack.o:linkstack.c linkstack.h  gcc -c linkstack.c |

* 1. 键入 **make** ，查看结果，对实验进行解析，并说明其功能。





解析：

该makefile的功能是编译连接生成目标文件main，main依赖于main.o和linkstack.o，main.o依赖main.c和linkstack.h编译生成main.o，linkstack.o依赖linkstack.c和linkstack.h编译生成linkstack.o.最终连接main.o和linkstack.o生成main。

* 1. 重新编写makefile

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | main:main.o linkstack.o  gcc $^ -o $@  main.o:main.c linkstack.h  linkstack.o:linkstack.c linkstack.h  clean:  @echo "cleanning the project"  rm \*.o |

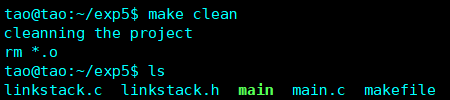
* 1. 键入 **make**，查看结果，对实验进行解析，并说明其功能。



解析：

$^表示当前依赖文件的集合，$@表示当前目标文件而main.o又依赖于main.c和linkstack.h,linkstack.o依赖于linkstack.c linkstack.h。所以gcc $^ -o $@相当于gcc main.c linkstack.c linkstack.h -o main。同样是编译连接生成目标文件main。

* 1. 键入 **make** **clean，**查看结果，对实验进行解析，并说明其功能。



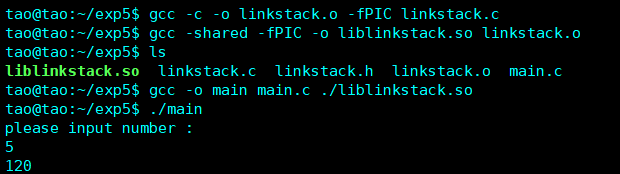
解析：

键入make clean 就会执行clean:里面的命令,功能是输出提示信息"cleanning the project"之后删除所有以.o结尾的文件。

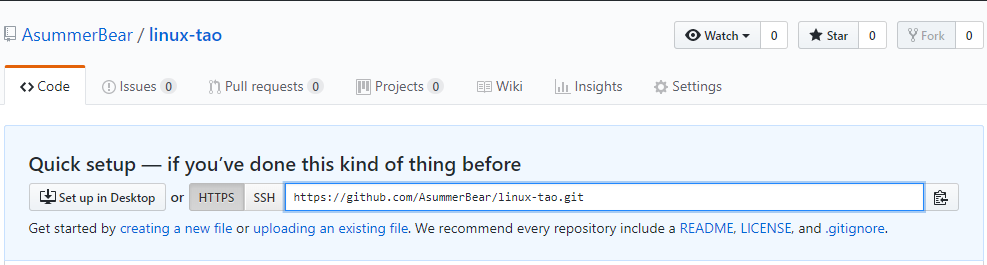
### 四、实验扩展：动态库和github

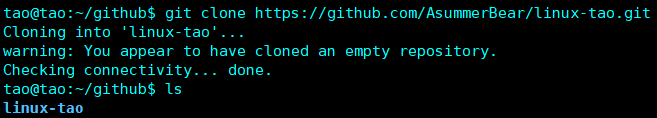
* 1. 动态库的编译、链接和运行

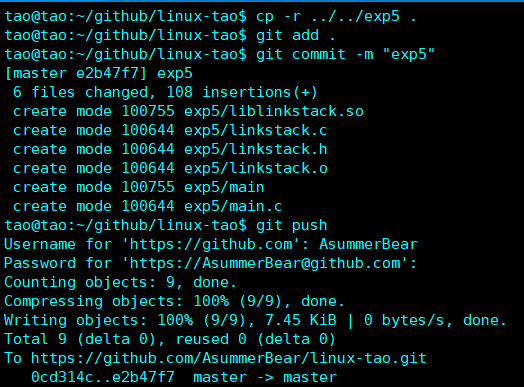
将linkstack.c打包成动态库，并在main程序中链接调用执行。

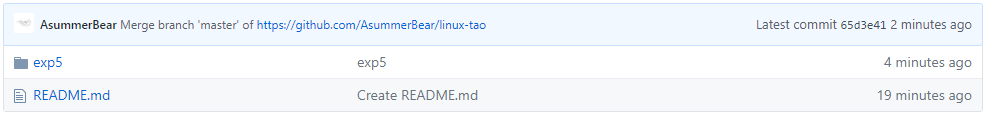


2、项目版本github学习和使用，将自己的代码在github上进行托管。









## 问题记录和实验总结

问题记录：

1.指定静态库路径无法找到静态库文件

解决：静态库编译时一定要命名为libxxx.a的格式，这样通过-L指定的路径中，通过-lxxx默认访问的是libxxx.a，若命名不按指定格式，就会找不到静态库文件。

2.编写makefile报错问题

解决：makefile出现错误往往是因为格式不对，makefile语法要求比较严格，缺少一个tab都会报错，如果使用vim自定义语法高亮，可以通过颜色提示保证编写格式无误。

通过本次实验，学会了基本的gcc编译c的命令，更加深刻理解了gcc的编译流程，大体明白了预处理、编译、链接等步骤。同时也学会了gdb调试工具的基本使用，利用gcc+gdb+vim在linux终端下，可以类比之前学过的c的IDE图形化操作模式完成程序的编译调试。另外还学习了makefile，这个以前没有接触过比较生疏，通过实验的样例和老师的讲解理解了基本的使用方法，明白了makefile在工程管理中的作用。在拓展部分的github的使用中，发现了这个仓库的好用之处，可以快捷方便的将项目代码托管到github上管理自己的文件。