实验2 Linux程序开发基础

1. 实验目的

学习利用GCC编译程序环境运行和编写程序；

1. 预备知识
2. Gcc 编译器能将C、C++语言源程序、汇程式化序和目标程序编译、连接成可执行文件，如果没有给出可执行文件的名字，gcc将生成一个名为a.out的文件。 在Linux系统中，可执行文件没有统一的后缀，系统从文件的属性来区分可执行文件和不可执行文件。而gcc则通过后缀来区别输入文件的类别。

.c为后缀的文件: C语言源代码文件；

.a为后缀的文件: 是由目标文件构成的档案库文件；

.C，.cc或.cxx 为后缀的文件: 是C++源代码文件；

.h为后缀的文件: 是程序所包含的头文件；

.i为后缀的文件: 是已经预处理过的C源代码文件；

.ii为后缀的文件: 是已经预处理过的C++源代码文件；

.m为后缀的文件: 是Objective-C源代码文件；

.o为后缀的文件: 是编译后的目标文件；

.s为后缀的文件: 是汇编语言源代码文件；

.S为后缀的文件: 是经过预编译的汇编语言源代码文件。

1. vim编辑器分为三种主要模式：

* 命令模式（编辑模式）：默认模式，移动光标，剪切/粘贴文本（界面表现：左下角显示文件名或为空）
* 插入模式（输入模式）：修改文本（界面表现：左下角显示—INSERT–）插入模式下，按ESC按键返回命令模式
* 末行模式（扩展模式）：保存、退出等（界面表现：左下角显示—VISUAL–）末行模式下连续按两次ESC按键返回末行模式

Esc键：退出当前模式

两次Esc键：总是返回到命令模式

1. 打开文件语法：vim [options] [file …]

例如:【vim abc.txt】打开abc.txt文件

例如：【vim +# abc.txt】打开abc.txt文件，光标定位在abc.txt文件的第#行

例如：【vim + abc.txt】打开abc.txt文件，光标定位在最后一行

例如：【vim +/PATTERN abc.txt】打开abc.txt文件，定位第一次被PATTERN(模式)匹配到的行的行首。

1. 模式之间的切换

注意：vim打开文件后，默认进入的模式为：命令模式：

命令模式下进入插入模式（输入模式）输入：【i】或者【o】或者【a】等

命令模式下进入末行模式（扩展模式）输入：【:】

1. 使用vim编辑多个文件

【vim FILE1 FILE2 FILE3】可以同时编辑FILE1 FILE2 FILE3这三个文件

【:next】切换至下一个文件

【:prev】切换至前一个文件

【:last】切换至最后一个文件

【:first】切换至第一个文件

1. vim的命令模式（编辑模式）常用快捷键
2. 模式转换

a) 【i】:在当前光标所在字符的前面，转为输入模式

b) 【a】:在当前光标所在字符的后面，转换为输入模式

c) 【o】:在当前光标所在行的下方，新建一行，并转换为输入模式：

d) 【I】:在当前光标所在行的行首，转换为输入模式

e) 【A】:在当前光标所在行的行尾，转换为输入模式

f) 【O】:在当前光标所在行的上方，新建一行，并转换为输入模式;

g) 【cc】删除当前行并输入新内容，相当于S。扩展【#cc】

h) 【C】删除当前光标到行尾，并切换成插入模式

1. 删除命令（剪切命令）

d命令删除的字符会默认复制到剪贴板

a) 【d】删除命令，可以结合光标跳转字符，实现范围删除

b) 【d$】删除到行尾

c) 【d^】删除到非空行首

d) 【d0】删除到行首

e) 【dd】删除光标所在行

f) 【#dd】多行删除

g) 【D】从当前光标位置一直删除到行尾，留空行，等同于【d$】

h) 【de】

1. 复制命令（y,yank）

a) 【y】复制，可以结合光标跳转字符，实现范围复制（行为与d有类似之处）

b) 【y$】复制到行尾

c) 【y^】复制到非空行首

d) 【y0】复制到行首

e) 【yy】复制当前行

f) 【#yy】复制多行

g) 【Y】整行复制

h) 【ye】

i) 【yw】

j) 【yb】

k) 【#COMMAND】

1. 粘贴命令（p,paste）

a) 【p】缓冲区存的如果为整行，则粘贴当前光标所在行的下方，否则粘贴至当前光标所在处的后面

b) 【P】缓冲区存的如果为整行，则粘贴当前光标所在行的上方，否则粘贴至当前光标所在处的前面

1. vim的末行模式（扩展模式）
2. 退出命令

【:q】退出

【:x】保存退出

【:wq】保存并退出

【:q!】强制退出并或略所有更改

【:e!】放弃所有修改，并打开源文件

注意：在默认模式的退出方法

【ZZ】保存并退出

【ZQ】不保存退出

1. 实验内容
2. GCC的使用格式如下：

gcc [options][filenames]，其中options就是编译器所需要的参数，filenames给出相关的文件名称。下面是一个简单的源程序示例：

#include<stdio.h>

int main()

{

printf(“Hello world,Linux programming!\n”);

return 0;

}

运行时，输入：#gcc hello.c –o hello //这样会编译出一个名为hello.out的输出文件

#./hello

输出：Hello world,Linux programming!

1. Gcc的主要参数：

-o 定义输出文件

-E 完成预处理，预编译停止

-c 只编译生成.o，不连接

-x 指定编译步骤

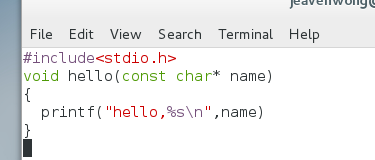
-g gdb调用，在可执行程序中包含标准调试信息

-O,O1,O2,O3,O4,O5 优化级别

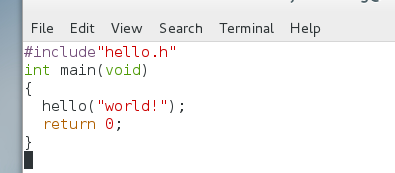
-w 关闭所有警告

-Wall 允许所有有用的警告

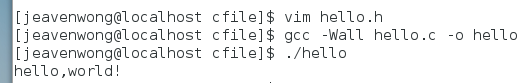
1. 以头文件的形式编写程序
2. 首先使用vim命令进入编辑模式，编写一个一个头文件hello.h

[](https://images2015.cnblogs.com/blog/1154141/201706/1154141-20170604131231789-275301236.png)

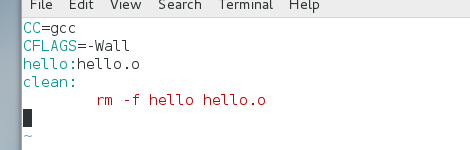
1. 然后使用vim命令进入编辑模式，编写一个主程序hello.c

[](https://images2015.cnblogs.com/blog/1154141/201706/1154141-20170604131245493-1302558506.png)

直接编译：

[](https://images2015.cnblogs.com/blog/1154141/201706/1154141-20170604131303196-2122198604.png)

1. 了解makefile使用。写一个简单的makefile，来完成以上功能：

[](https://images2015.cnblogs.com/blog/1154141/201706/1154141-20170604131316399-700816257.png)

注意make是自动执行当前目录下的Makefile文件里面的指令的，Makefile里面的指令行得用tab键开始，否则无法执行

[https://images2015.cnblogs.com/blog/1154141/201706/1154141-20170604131329008-1441408398.png](https://images2015.cnblogs.com/blog/1154141/201706/1154141-20170604131329008-1441408398.png)

可以看到生成了hello绿色的可执行文件

[https://images2015.cnblogs.com/blog/1154141/201706/1154141-20170604131340414-797619228.png](https://images2015.cnblogs.com/blog/1154141/201706/1154141-20170604131340414-797619228.png)

输出正确的结果：

[https://images2015.cnblogs.com/blog/1154141/201706/1154141-20170604131352711-759489865.png](https://images2015.cnblogs.com/blog/1154141/201706/1154141-20170604131352711-759489865.png)

1. 实验题目
2. 使用vim编辑器编写冒泡排序或者选择排序，使用gcc编译运行并截图保存结果。
3. 使用vim编辑器编写折半查找，使用gcc编译运行并截图保存结果。
4. 实验过程和输出结果可以用文字说明和截图放入实验报告中，将调试运行成功的程序和写好的实验报告一起压缩打包，以实验X-学号-姓名.rar这样的形式命名，并上传提交。