Linux编程环境

**一、 实验目的**

1.理解vi 的三种运行模式及其切换方法

2.学会使用vi 的各种操作命令进行文本文件的编辑

3.用 vi 编写 Linux 下的 C 语言源程序，会使用 GCC 编译器进行编译

4.编写运行简单程序

**二、实 验 内容**

**（一 ）vi 编辑器 的使用**

**1. vi 编辑器 三种模式 （命令行模式、插入模式、底行模式） 的 切换及应用**

**（1）使用 root 账户登录 Linux 系统，切换到目录/root/test**

**（2）执行 vi，不保存直接退出，操作如下：**

#vi （进入 vi 编辑界面）

:q （从命令行模式切换为底行模式，不保存直接退出）

**（3）建立 hello.c 文件**

#vi hello.c

在 vi 界面下输入 i，左下角出现“INSERT”,提示当前为插入模式。

输入一段文字：

Hello world!

Do it myself!

按 esc 键，进入命令行模式。这时“INSERT”消失。

输入:（冒号）进入底行模式，输入wq，按回车键，保存文件并退出 vi。

查看 hello.c 文件内容：

#cat hello.c

**（4）编辑 hello.c 文件（增加、删除、修改）**

#vi hello.c

进入 vi 界面，输入 i，进入插入模式

在原文下面再增加一行文字：Operating system is very easy!

用方向键将光标移动到“Do it myself!”所在行任意位置

按 esc 键，进入命令行模式输入命令 dd，删除光标所在一行（即删除 Do it myself!一行）（注意：若仅输入d 后再回车是将光标所在行及以下所有行全部删除）

将光标移动到 system 的第一个“s”处，输入 i，进入插入模式，对文本进行修改。

**（5）对 hello.c 文件内容进行复制、粘贴**

按 esc 进入命令行模式，将光标定位到要复制行，输入命令 yy，复制该行。

将光标移动到待粘贴行上一行开头处，输入 p，完成行粘贴。

**（6）在 hello.c 文件中搜索、替换字符**

在命令行模式下输入:（冒号）进入底行模式，输入/is，搜索字符 is，可以看到

光标定位到该字符所在行行首处。

**（7）vi 的其他功能请参考相关参考资料自行练习。**

**2. 使用 i vi 编辑器 编写 C语言 源 程序 sum.c并编译运行**

进入 vi 编辑界面

输入如下程序代码：

#include <stdio.h>

int sum(int n);

main()

{ int s=0;

int i,n;

for(i=0;i<=50;i++)

{

s=i+s;

}

s=s+sum(20);

printf("the result is %d\n",s);

}

int sum(int n)

{

int total=0;

int i;

for(i=0;i<=n;i++)

total=total+i;

return (total);

}

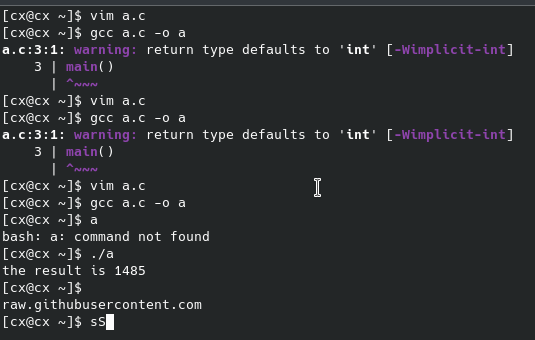
:wq （退出 vi 编辑器）

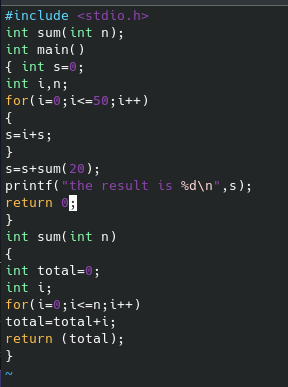
输入编译命令，产生二进制可执行程序sum：

#gcc -o sum sum.c

执行程序：

#./sum



****

**（ 二 ） GCC 编译器 的使用**

**1. gcc 编译器的 直接编译**

**（1）使用 vi 编辑器，编辑 C 语言源程序hello.c，代码如下：**

#include <stdio.h>

void main()

{

char msg[80]=“Hello,world!”;

printf(“%s\n”,msg);

}

**（2）编译程序，产生二进制可执行程序 a.out**

#gcc hello.c

**（3）执行程序**

#./a.out

**2. gcc 编译器的分步编译**

**（1）用 vi 编辑器编写C 语言源程序 test.c，代码如下：**

#define number (1+2\*3)

int main()

{

int n;

n=number+3;

return 0;

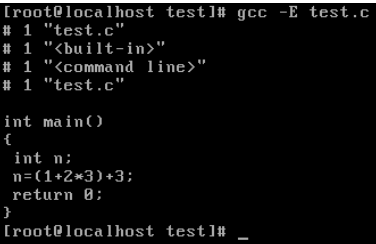
}

**（2）对 test.c 进行预编译（preprocessing）**

通过在 GCC 后加-E 选项完成对代码的预编译：

#gcc –E test.c

预编译结果如下：



使用如下预编译命令，将预编译结果保存在 test.i 文件中：

#gcc –E test.c -o test.i

查看 test.i 内容：

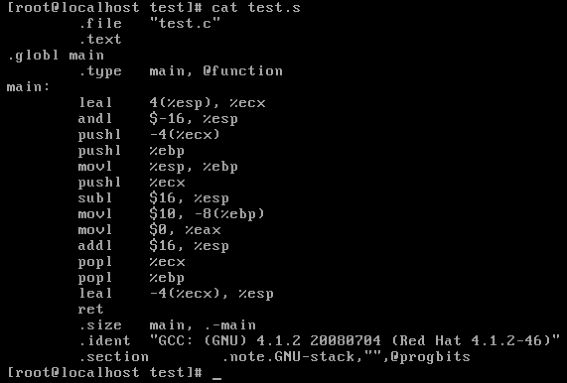
#cat test.i

**（3）编译（compilation）**

将预编译后的文件转换成汇编语言，并自动生成后缀为.s 的文件：

#gcc -S test.c

编译结果如下：



**（4）汇编（assembly）**

对 test.c 进行汇编，生成后缀名为.o 的目标程序：

#gcc –c test.c

**（5）连接（linking）**

连接目标代码，生成二进制可执行程序test：

#gcc -o test test.o

或者执行如下指令：

#gcc -o test test.c

**（6）执行编译后的可执行程序**

#./test

**（7）在 test.c 的基础上编写 test1.c 程序，再按上述分步编译过程重新进行编译，并对两程序编译过程信息输出及结果进行分析和对比。test1.c 程序代码如下：**

#include <stdio.h>

#define number (1+2\*3)

int main()

{

int n;

n=number+3;

printf(“n=number+3=(1+2\*3)+3=%d\n”,n);

}

1. **用vi建立C++/C程序后，学习使用gcc编译器编译执行，查看结果。  
   任务1. helloworld程序。  
   任务2. （二选一）判断一个年份是平年还是闰年。  
    打印九九乘法表。**

任务1：

使用vi编写helloworld.cpp程序（安装了vim的同学，可以输入vim helloworld.cpp），保存。程序代码如下：

using namespace std;

#include<iostream>

int main()

{

cout<<”hello,world!\n”;

return 0;

}

或编写helloworld.c程序，代码如下：

#include<stdio.h>

int main()

{

printf(“hello,world!\n”);

return 0;

}

然后，利用g++ helloworld.cpp或gcc helloworld.c指令编译执行。（如果没有任何输出提示，恭喜你，编译执行成功。否则，会提示出错信息，请检查语法并修改。）最后，在命令行输入 ./a.out 查看结果。

任务2：

[题1]设计一个程序，由用户输入一个年份，程序判断是平年还是闰年，输出结果。将你的代码以及做为测试数据的输入输出写出来。练习使用make工具及gdb调试器。

（一个年份是闰年的条件为:

如果该年份不能被100整除，但能被4整除，则是闰年。

或者该年份能被400整除，则也是闰年。

例如：1992年是闰年，2000年是闰年，1990年是平年，1900年是平年。）

[题2]设计一个程序，打印99乘法表。要求打印出如下结果：

1\*1=1 1\*2=2 1\*3=3 1\*4=4 1\*5=5 1\*6=6 1\*7=7 1\*8=8 1\*9=9

2\*1=2 2\*2=4 2\*3=6 2\*4=8 2\*5=10 2\*6=12 2\*7=14 2\*8=16 2\*9=18

……（略）

9\*1=9 9\*2=18 9\*3=27 9\*4=36 9\*5=45 9\*6=54 9\*7=63 9\*8=72 9\*9=81

将你的代码写出来。

**提高篇：用Gdb调试有问题的程序（选做）**

**通过调试一个有问题的程序，使读者进一步熟练使用Vi 操作，而且熟练掌握Gcc 编译命令及Gdb 的调试命令。**

**要求：将原来有错的程序经过Gdb调试，找出问题所在，并修改源代码，输出正确的倒序显示字符串的结果。**

**1、操作内容：**

（1）使用Vi 编辑器，将以下代码输入到名为greet.c 的文件中。此代码的原意为输出倒序main函数中定义的字符串，但结果显示没有输出。代码如下所示：

#include <stdio.h>

int display1(char \*string);

int display2(char \*string);

int main ()

{

char string[] = "Embedded Linux";

display1 (string);

display2 (string);

}

int display1 (char \*string)

{

printf ("The original string is %s \n", string);

}

int display2 (char \*string1)

{

char \*string2 = NULL;

int size = 0, i = 0;

size = strlen (string1);

string2 = (char \*) malloc (size + 1);

for (i = 0; i < size; i++)

string2[size - i] = string1[i];

string2[size+1] = ' ';

printf("The string afterward is %s\n",string2);

}

（2）使用Gcc编译这段代码，注意要加上“-g”选项以方便之后的调试。

（3）运行生成的可执行文件，观察运行结果。

（4）使用Gdb调试程序，通过设置断点、单步跟踪，一步步找出错误所在。

（5）纠正错误，更改源程序并得到正确的结果。

**2、操作指导：**

（1）在工作目录上新建文件greet.c，并用Vi 启动：vi greet.c。

（2）在Vi 中输入以上代码。

（3）在Vi 中保存并退出：wq。

（4）用Gcc编译：gcc -g greet.c -o greet。

（5）运行greet：./greet，输出为：

The original string is Embedded Linux

The string afterward is

可见，该程序没有能够倒序输出。

（6）启动Gdb调试：gdb greet。

（7）查看源代码，使用命令“l”。

（8）在 display2函数 for 循环处（line = X）设置断点，使用命令“b X”。

（9）在行 display2函数 printf函数处（line = X）设置断点，使用命令“b X”。

（10）查看断点设置情况，使用命令“info b”。

（11）运行代码，使用命令“r”。

（12）单步运行代码，使用命令“n”。

（13）查看暂停点变量值，使用命令“p string2[size - i]”。

（14）继续单步运行代码数次，并使用命令查看，发现string2[size-1]的值正确。

（15）继续程序的运行，使用命令“c”。

（16）程序在printf前停止运行，此时依次查看string2[0]、string2[1]…，发现string[0]没有被正确赋值，而后面的复制都是正确的，这时，定位程序第31行，发现程序运行结果错误的原因在于“size-1”。由于i 只能增到“size-1”，这样string2[0]就永远不能被赋值而保持NULL，故输不出任何结果。

（17）退出Gdb，使用命令q。

（18）重新编辑greet.c，把其中的“string2[size - i] = string1[i]”改为“string2[size – i - 1] =string1[i];”即可。

（19）使用Gcc重新编译：gcc -g greet.c -o greet。

（20）查看运行结果：./greet

The original string is Embedded Linux

The string afterward is xuniL deddedbmE

这时，输入结果正确。

