

# 北 京 林 业 大 学

## 2017 学年—2018 学年第 二 学期 Linux 应用实验报告书

专 业： 计 算 机 科 学 与 技 术 ( 创 新 实 验 班 )

班 级： 计创 16

姓 名： 陈楠 学 号： 161002107

实验地点： 计算中心 N09 任课教师： 李群

实验题目： Linux 下 C 编程

实验环境： Linux 操作系统

实验目的、实现内容、实验结果及结论分析等：

### 一．实验目的：

1. 掌握 vi 编辑器的使用；
2. 掌握 GCC 的编译器的使用；
3. 掌握调试工具 GDB 的使用；
4. 掌握编写 makefile 文件的方法。

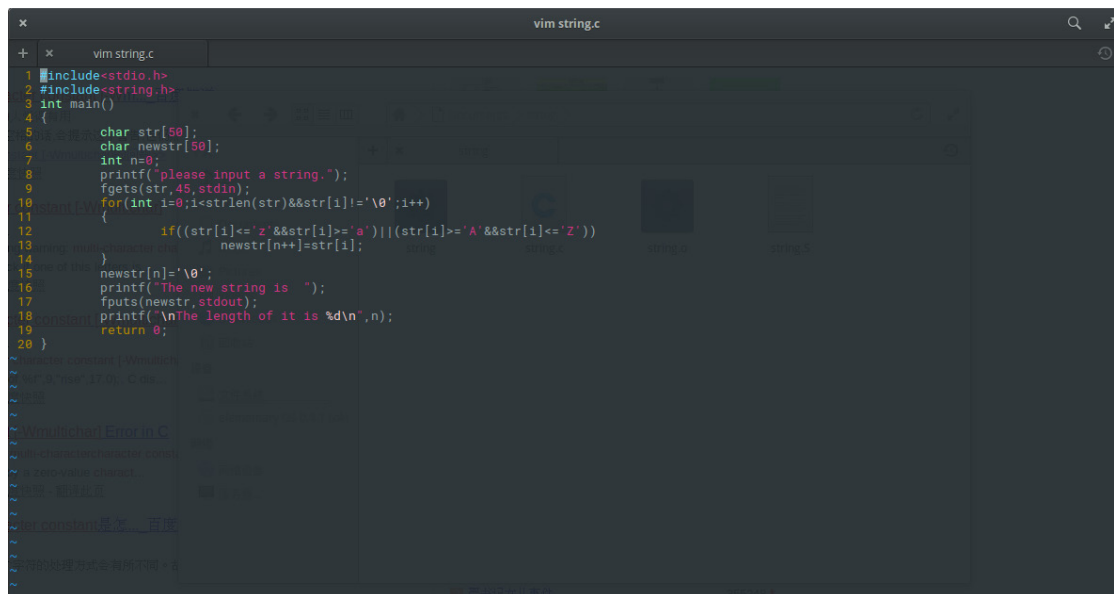
### 二．实验内容：

1. 用 vi 编辑器编辑一个应用程序，功能：输入一个字符串，过滤此串，只保留串中的字母字符，并统计新生成串中包含的字母个数。（例如：输入的字符串为 ab234\$df4，新生成的串为 abdf。）使用 gcc 进行编译，并分别使用 -E，-S，-c，-o，-static，-O2(用 time 命令)等选项，编译为可执行文件并执行。用抓图的形式保存整个程序的显示、编译、运行过程。（注：Linux 下输入字符串函数 fgets()、输出字符串函数 fputs()。）
2. 编写一个应用程序，功能：从键盘输入一个整数，判断其是否为素数，然后在 main 函数中输出相应的结论信息。（提示：素数也称质数。一个大于 1 的自然数，除了 1 和它自身外，不能被其他自然数整除的数叫做质数，否则称为合数。例如：7 是素数，8 不是素数。）使用 gcc 进行编译，并分别使用 -E，-S，-c，-o，-static，-O2(用 time 命令)等选项，编译为可执行文件并执行。用抓图的形式保存整个程序的显示、编译、运行过程。
3. 编写一个应用程序，功能是：实现 1~100 求和。使用 gdb 调试，调试中使用到本章节所介绍的 GDB 的几个主要功能来完成调试过程。用抓图的形式保存整个程序的调试、运行过程。
4. 实现一应用程序，该程序至少包含有两个 c 文件构成，使用 makefile 来完成对该程序的编译功能。

### 三. 实验结果：

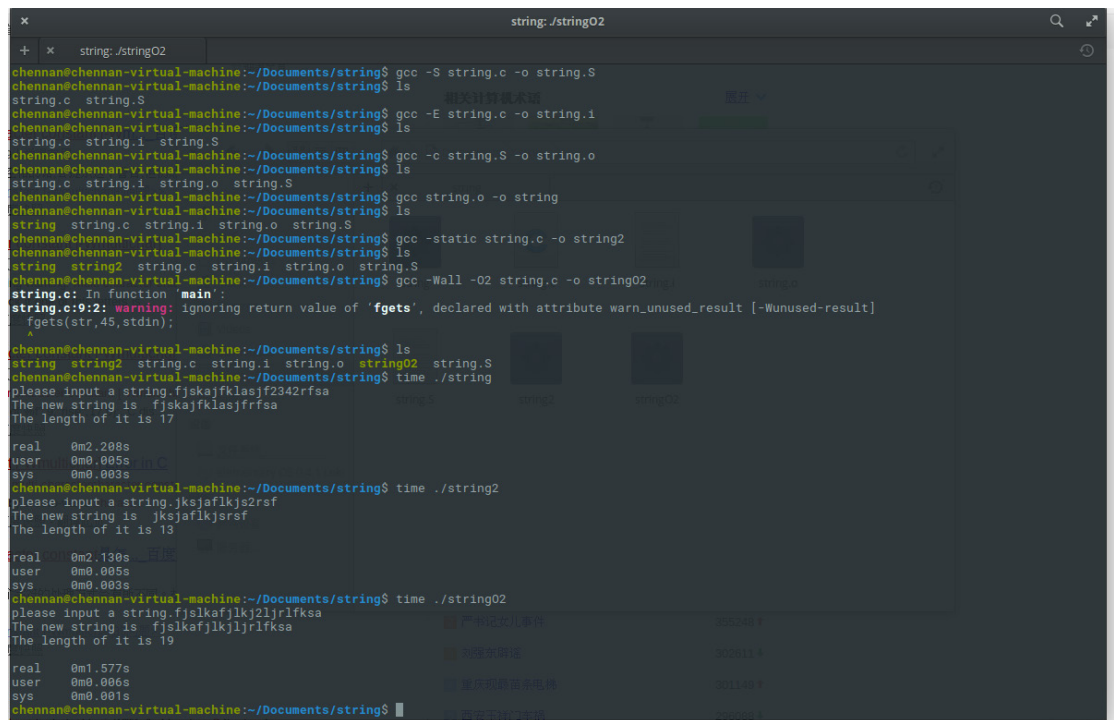
1. 用 vi 编辑器编辑一个应用程序，功能：输入一个字符串，过滤此串，只保留字符串中的字母字符，并统计新生成串中包含的字母个数。（例如：输入的字符串为 ab234\$df4，新生成的串为 abdf。）使用 gcc 进行编译，并分别使用 -E, -S, -c, -o, -static, -O2(用 time 命令)等选项，编译为可执行文件并执行。用抓图的形式保存整个程序的显示、编译、运行过程。（注：Linux 下输入字符串函数 fgets()、输出字符串函数 fputs()。

#### 建立 string.c 文件



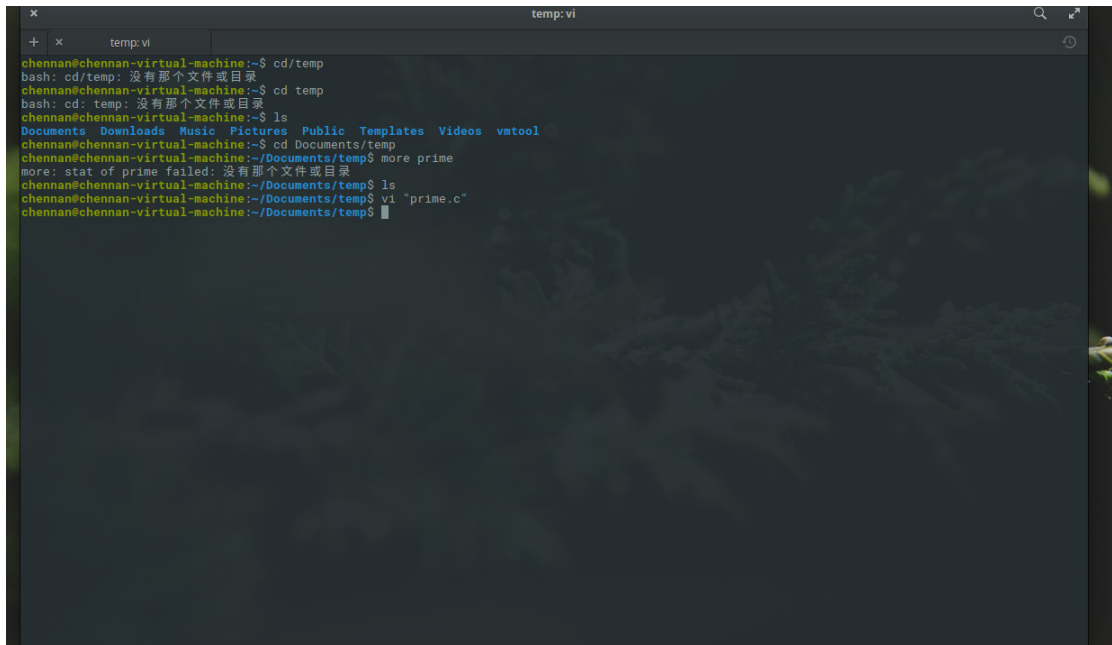
```
vim string.c
1 #include<stdio.h>
2 #include<string.h>
3 int main()
4 {
5     char str[50];
6     char newstr[50];
7     int n=0;
8     printf("please input a string.");
9     fgets(str,45,stdin);
10    for(int i=0;i<strlen(str)&&str[i]!='\0';i++)
11    {
12        if((str[i]<='z'&&str[i]>='a')||((str[i]>='A'&&str[i]<='Z'))
13        {
14            newstr[n++]=str[i];
15        }
16    }
17    newstr[n]='\0';
18    printf("The new string is ");
19    fputs(newstr,stdout);
20    printf("\nThe length of it is %d\n",n);
21    return 0;
22 }
```

#### 编译运行 string.c 文件



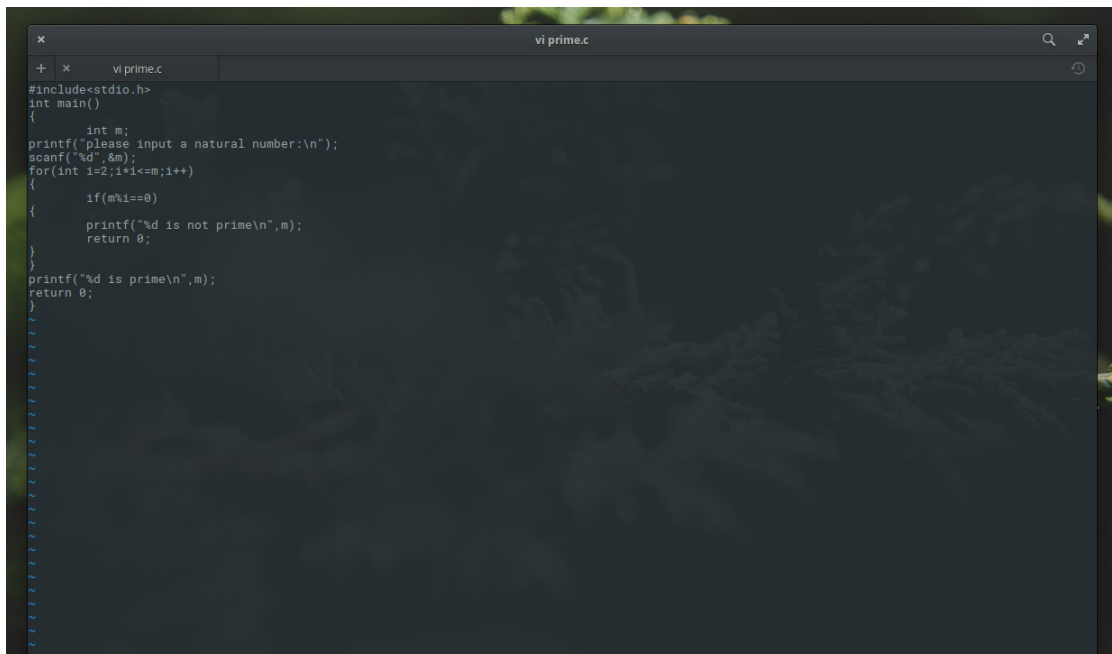
```
string: ./stringO2
chennan@chennan-virtual-machine:~/Documents/string$ gcc -S string.c -o string.S
chennan@chennan-virtual-machine:~/Documents/string$ ls
string.c string.S
chennan@chennan-virtual-machine:~/Documents/string$ gcc -E string.c -o string.i
chennan@chennan-virtual-machine:~/Documents/string$ ls
string.c string.i string.S
chennan@chennan-virtual-machine:~/Documents/string$ gcc -c string.S -o string.o
chennan@chennan-virtual-machine:~/Documents/string$ ls
string.c string.i string.o string.S
chennan@chennan-virtual-machine:~/Documents/string$ gcc string.o -o string
chennan@chennan-virtual-machine:~/Documents/string$ ls
string string.c string.i string.o string.S
chennan@chennan-virtual-machine:~/Documents/string$ gcc -static string.c -o string2
chennan@chennan-virtual-machine:~/Documents/string$ ls
string string2 string.c string.i string.o string.S
chennan@chennan-virtual-machine:~/Documents/string$ gcc -Wall -O2 string.c -o stringO2
string.c: In function 'main':
string.c:9:12: warning: ignoring return value of 'fgets', declared with attribute warn_unused_result [-Wunused-result]
   fgets(str,45,stdin);
   ^
chennan@chennan-virtual-machine:~/Documents/string$ ls
string string2 string.c string.i string.o stringO2 string.S
chennan@chennan-virtual-machine:~/Documents/string$ time ./string
please input a string.fjskajfklsjf2342rfsa
The new string is fjskajfklsjfrfsa
The length of it is 17
real    0m2.208s
user    0m0.005s
sys     0m0.000s
chennan@chennan-virtual-machine:~/Documents/string$ time ./string2
please input a string.jksjafkjs2rsf
The new string is jksjafkjsrsf
The length of it is 13
real    0m2.130s
user    0m0.005s
sys     0m0.000s
chennan@chennan-virtual-machine:~/Documents/string$ time ./stringO2
please input a string.fjslkafjlkj2lrlfksa
The new string is fjslkafjlkjlrlfksa
The length of it is 19
real    0m1.577s
user    0m0.005s
sys     0m0.001s
chennan@chennan-virtual-machine:~/Documents/string$
```

2. 编写一个应用程序，功能：从键盘输入一个整数，判断其是否为素数，然后在 main 函数中输出相应的结论信息。（提示：素数也称质数。一个大于 1 的自然数，除了 1 和它自身外，不能被其他自然数整除的数叫做质数，否则称为合数。例如：7 是素数，8 不是素数。）使用 gcc 进行编译，并分别使用 -E, -S, -c, -o, -static, -O2(用 time 命令)等选项，编译为可执行文件并执行。用抓图的形式保存整个程序的显示、编译、运行过程。



```
chennan@chennan-virtual-machine:~$ cd /temp
bash: cd /temp: 没有那个文件或目录
chennan@chennan-virtual-machine:~$ cd temp
bash: cd: temp: 没有那个文件或目录
chennan@chennan-virtual-machine:~$ ls
Documents Downloads Music Pictures Public Templates Videos vntool
chennan@chennan-virtual-machine:~$ cd Documents/temp
chennan@chennan-virtual-machine:~/Documents/temp$ more prime
more: stat of prime failed: 没有那个文件或目录
chennan@chennan-virtual-machine:~/Documents/temp$ ls
chennan@chennan-virtual-machine:~/Documents/temp$ vi "prime.c"
chennan@chennan-virtual-machine:~/Documents/temp$
```

建立并编写 prime.c 文件

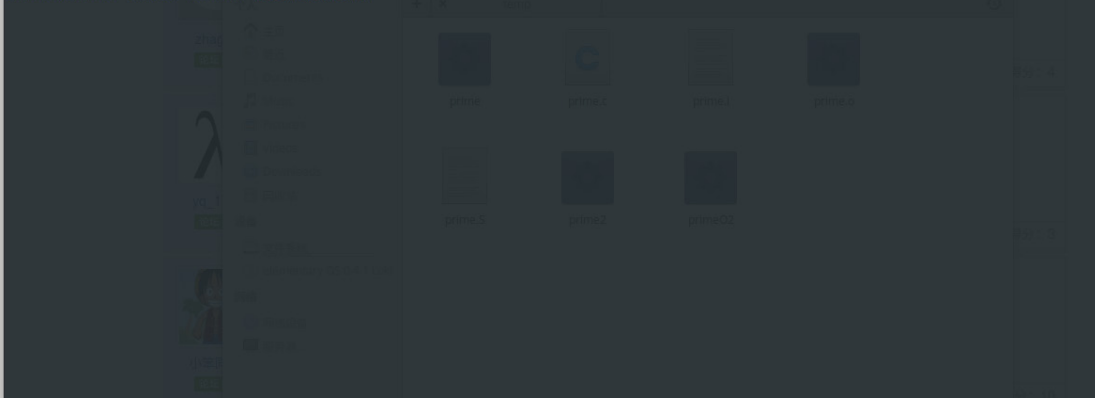


```
vi prime.c
#include<stdio.h>
int main()
{
    int m;
    printf("please input a natural number:\n");
    scanf("%d",&m);
    for(int i=2;i<=m;i++)
    {
        if(m%i==0)
        {
            printf("%d is not prime\n",m);
            return 0;
        }
    }
    printf("%d is prime\n",m);
    return 0;
}
```

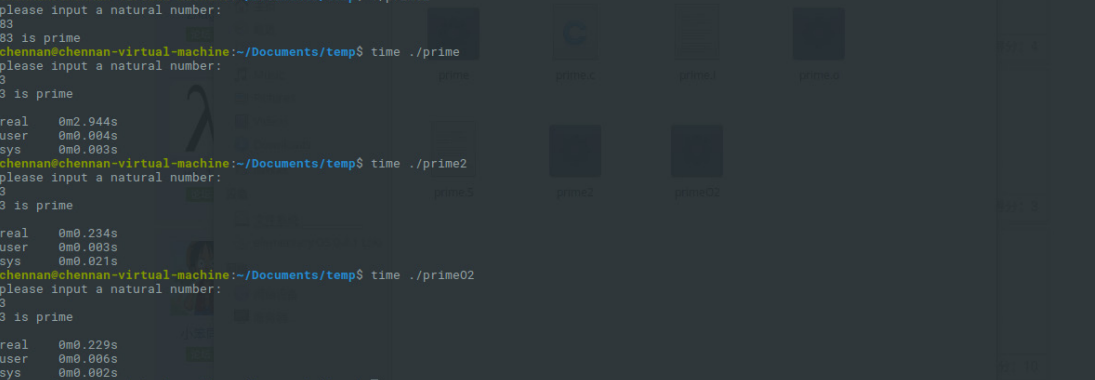
## 编译并运行 prime.c 文件

```
temp: gcc
chennan@chennan-virtual-machine:~/Documents/temp$ gcc -E prime.c -o prime.i
chennan@chennan-virtual-machine:~/Documents/temp$ ls
prime.c prime.i prime.S
chennan@chennan-virtual-machine:~/Documents/temp$ gcc -S prime.c -o prime.S
chennan@chennan-virtual-machine:~/Documents/temp$ ls
prime.c prime.i prime.S
chennan@chennan-virtual-machine:~/Documents/temp$ gcc -c prime.S -o prime.o
chennan@chennan-virtual-machine:~/Documents/temp$ ls
prime.c prime.i prime.o prime.S
chennan@chennan-virtual-machine:~/Documents/temp$ gcc prime.o -o prime
chennan@chennan-virtual-machine:~/Documents/temp$ ls
prime prime.c prime.i prime.o prime.S
chennan@chennan-virtual-machine:~/Documents/temp$ gcc -static prime.c -o prime2
chennan@chennan-virtual-machine:~/Documents/temp$ ls
prime prime2 prime.c prime.i prime.o prime.S
```

```
temp: ls
chennan@chennan-virtual-machine:~/Documents/temp$ gcc -Wall -O2 prime.c -o primeO2
chennan@chennan-virtual-machine:~/Documents/temp$ ls
prime prime2 prime.c prime.i prime.o primeO2 prime.S
chennan@chennan-virtual-machine:~/Documents/temp$
```



```
temp: ./primeO2
prime prime2 prime.c prime.i prime.o primeO2 prime.S
chennan@chennan-virtual-machine:~/Documents/temp$ ./prime
please input a natural number:
89
89 is prime
chennan@chennan-virtual-machine:~/Documents/temp$ ./prime2
please input a natural number:
83
83 is prime
chennan@chennan-virtual-machine:~/Documents/temp$ time ./prime
please input a natural number:
3
3 is prime
real    0m2.944s
user    0m0.004s
sys     0m0.003s
chennan@chennan-virtual-machine:~/Documents/temp$ time ./prime2
please input a natural number:
3
3 is prime
real    0m0.234s
user    0m0.003s
sys     0m0.021s
chennan@chennan-virtual-machine:~/Documents/temp$ time ./primeO2
please input a natural number:
3
3 is prime
real    0m0.229s
user    0m0.006s
sys     0m0.002s
chennan@chennan-virtual-machine:~/Documents/temp$
```



3. 编写一个应用程序，功能是：实现 1~100 求和。使用 gdb 调试，调试中使用到本章节所介绍的 GDB 的几个主要功能来完成调试过程。用抓图的形式保存整个程序的调试、运行过程。

## 进入 gdb

```
gdb sum

chennan@chennan-virtual-machine:~/Documents/temp2$ vim "sum.c"
chennan@chennan-virtual-machine:~/Documents/temp2$ gcc -g sum.c -o sum
chennan@chennan-virtual-machine:~/Documents/temp2$ gdb sum
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from sum...done.
(gdb)
```

## 显示程序

```
gdb sum

chennan@chennan-virtual-machine:~/Documents/temp2$ vim "sum.c"
chennan@chennan-virtual-machine:~/Documents/temp2$ gcc -g sum.c -o sum
chennan@chennan-virtual-machine:~/Documents/temp2$ gdb sum
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from sum...done.
(gdb) list
1      #include<stdio.h>
2      int main()
3      {
4          int n=100,sum=0;
5          for(int i=1;i<=n;i++)
6          {
7              sum+=i;
8          }
9          printf("The sum of 1 - 100 is %d\n",sum);
10         return 0;
(gdb)
```

## 设置断点

```
gdb sum

chennan@chennan-virtual-machine:~/Documents/temp2$ vim "sum.c"
chennan@chennan-virtual-machine:~/Documents/temp2$ gcc -g sum.c -o sum
chennan@chennan-virtual-machine:~/Documents/temp2$ gdb sum
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from sum...done.
(gdb) list
1      #include<stdio.h>
2      int main()
3      {
4          int n=100,sum=0;
5          for(int i=1;i<=n;i++)
6          {
7              sum+=i;
8          }
9          printf("The sum of 1 - 100 is %d\n",sum);
10         return 0;
(gdb) break 7
Breakpoint 1 at 0x400545: file sum.c, line 7.
(gdb)
```

显示断点信息

```
gdb sum
chennan@chennan-virtual-machine:~/Documents/temp2$ gcc -g sum.c -o sum
chennan@chennan-virtual-machine:~/Documents/temp2$ gdb sum
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from sum...done.
(gdb) list
1      #include<stdio.h>
2      int main()
3      {
4          int n=100,sum=0;
5          for(int i=1;i<=n;i++)
6          {
7              sum+=i;
8          }
9          printf("The sum of 1 - 100 is %d\n",sum);
10         return 0;
(gdb) break 7
Breakpoint 1 at 0x400545: file sum.c, line 7.
(gdb) info b
Num      Type      Disp Enb Address            What
1        breakpoint keep y   0x000000000400545 in main at sum.c:7
(gdb)
```

执行程序

```
(gdb) list
1      #include<stdio.h>
2      int main()
3      {
4          int n=100,sum=0;
5          for(int i=1;i<=n;i++)
6          {
7              sum+=i;
8          }
9          printf("The sum of 1 - 100 is %d\n",sum);
10         return 0;
(gdb) break 7
Breakpoint 1 at 0x400545: file sum.c, line 7.
(gdb) info b
Num      Type      Disp Enb Address            What
1        breakpoint keep y   0x000000000400545 in main at sum.c:7
(gdb) run
Starting program: /home/chennan/Documents/temp2/sum
Breakpoint 1, main () at sum.c:7
7          sum+=i;
(gdb)
```

打印变量值

```
1      #include<stdio.h>
2      int main()
3      {
4          int n=100,sum=0;
5          for(int i=1;i<=n;i++)
6          {
7              sum+=i;
8          }
9          printf("The sum of 1 - 100 is %d\n",sum);
10         return 0;
(gdb) break 7
Breakpoint 1 at 0x400545: file sum.c, line 7.
(gdb) info b
Num      Type      Disp Enb Address            What
1        breakpoint keep y   0x000000000400545 in main at sum.c:7
(gdb) run
Starting program: /home/chennan/Documents/temp2/sum
Breakpoint 1, main () at sum.c:7
7          sum+=i;
(gdb) print sum
$1 = 0
(gdb) print n
$2 = 100
(gdb) print i
$3 = 1
(gdb)
```



```
1 int n=100,sum=0;
2 for(int i=1;i<=n;i++)
3 {
4     sum+=i;
5 }
6 printf("The sum of 1 - 100 is %d\n",sum);
7 return 0;
(gdb) break 7
Breakpoint 1 at 0x400545: file sum.c, line 7.
(gdb) info b
Num      Type      Disp Enb Address             What
1 breakpoint keep y  0x0000000000400545 in main at sum.c:7
(gdb) run
Starting program: /home/chennan/Documents/temp2/sum
Breakpoint 1, main () at sum.c:7
7     sum+=i;
(gdb) print sum
$1 = 0
(gdb) print n
$2 = 100
(gdb) print i
$3 = 1
(gdb) next
7     sum+=i;
(gdb) step
7     sum+=i;
Breakpoint 1, main () at sum.c:7
7     sum+=i;
(gdb) steo
Undefined command: "steo". Try "help".
(gdb) step
7     sum+=i;
(gdb) step
7     sum+=i;
(gdb) 
```

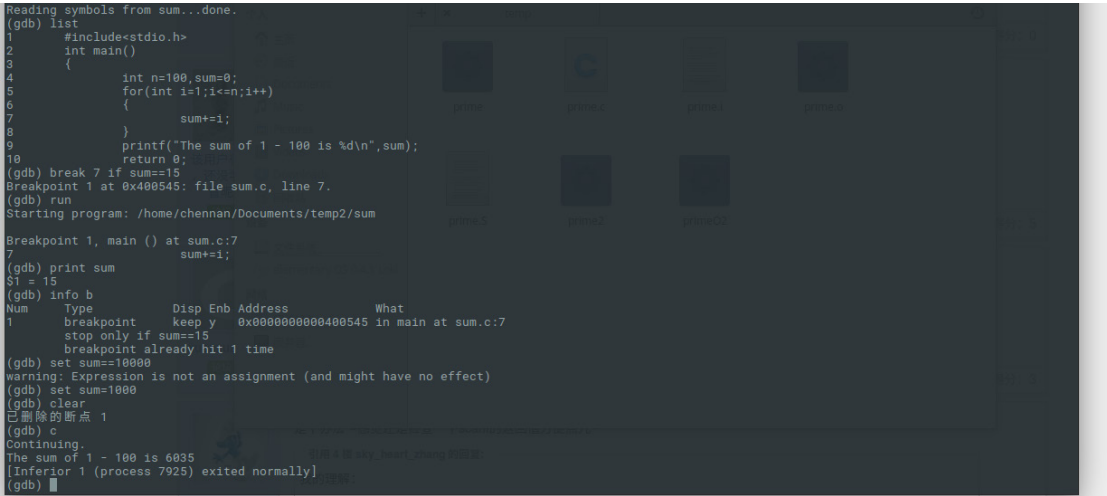
## 按步执行

```
x gdb sum
+ x gdb sum
(gdb) print n
$2 = 100
(gdb) print i
$3 = 1
(gdb) next
5     for(int i=1;i<=n;i++)
(gdb) step
Breakpoint 1, main () at sum.c:7
7     sum+=i;
(gdb) steo
Undefined command: "steo". Try "help".
(gdb) step
5     for(int i=1;i<=n;i++)
(gdb) ^CQuit
(gdb) next
Breakpoint 1, main () at sum.c:7
7     sum+=i;
(gdb) step
5     for(int i=1;i<=n;i++)
(gdb) step
Breakpoint 1, main () at sum.c:7
7     sum+=i;
(gdb) step
5     for(int i=1;i<=n;i++)
(gdb) next
Breakpoint 1, main () at sum.c:7
7     sum+=i;
(gdb) next
5     for(int i=1;i<=n;i++)
(gdb) next
Breakpoint 1, main () at sum.c:7
7     sum+=i;
(gdb) c
Continuing.
Breakpoint 1, main () at sum.c:7
7     sum+=i;
(gdb) 
```

## 删除断点，运行到程序结束

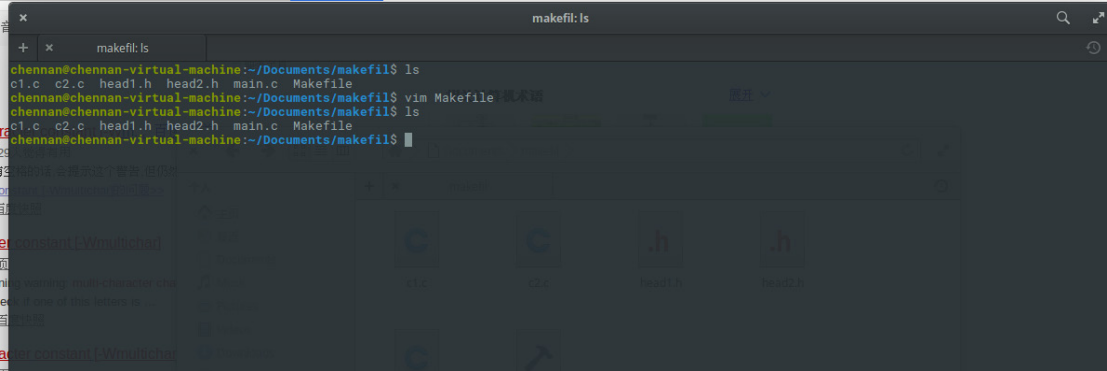
```
(gdb) c 20
Will ignore next 19 crossings of breakpoint 1. Continuing.
Breakpoint 1, main () at sum.c:7
7     sum+=i;
(gdb) c
Continuing.
Breakpoint 1, main () at sum.c:7
7     sum+=i;
(gdb) clear
已删除的断点 1
(gdb) c
Continuing.
The sum of 1 - 100 is 5050
[Inferior 1 (process 7549) exited normally]
(gdb) 
```

设置断点中变量到具体值时停止。  
并给变量重新赋值。

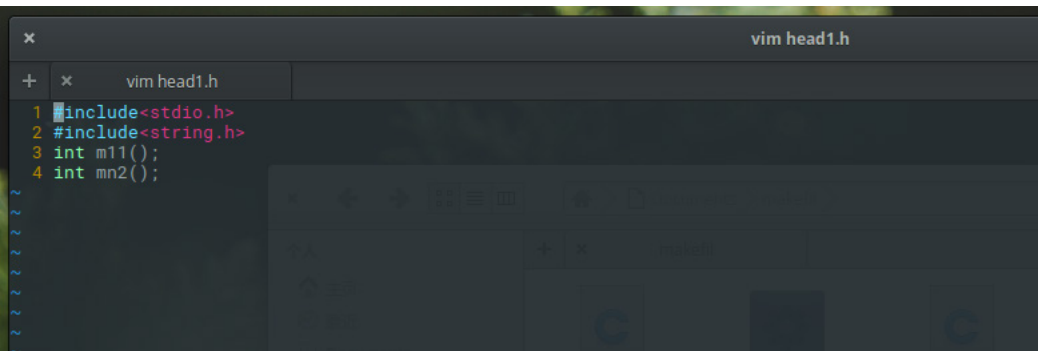


4. 实现一应用程序，该程序至少包含有两个 c 文件构成，使用 makefile 来完成对该程序的编译功能。

建立 head1.h head2.h c1.c c2.c main.c 文件。

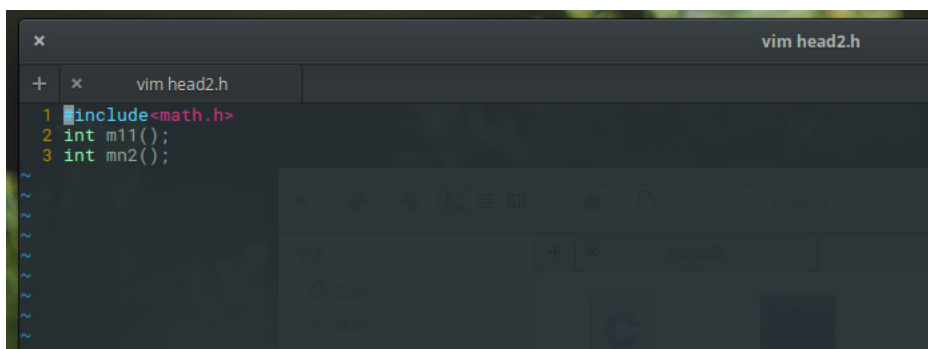


head1.h 文件



head2.h 文件

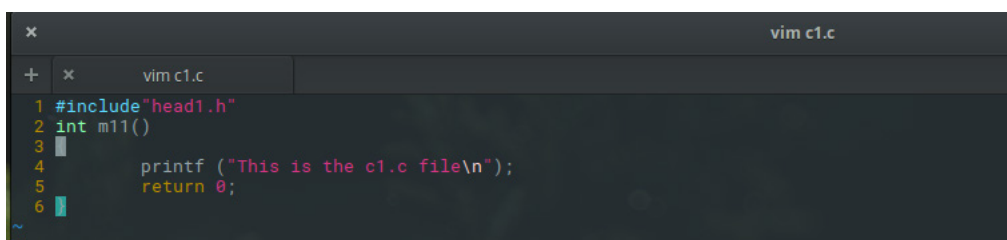


A screenshot of a vim editor window titled 'vim head2.h'. The editor shows a C header file with the following content:

```
1 #include<math.h>
2 int m11();
3 int mn2();
```

The editor interface includes a tab bar at the top with a '+' icon and a close 'x' icon. A vertical line of tilde '~' characters is visible on the left margin.

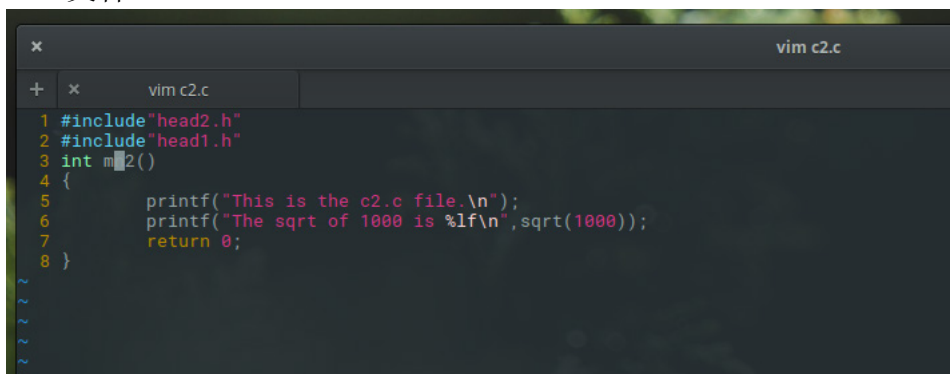
c1.c 文件

A screenshot of a vim editor window titled 'vim c1.c'. The editor shows a C source file with the following content:

```
1 #include"head1.h"
2 int m11()
3 {
4     printf ("This is the c1.c file\n");
5     return 0;
6 }
```

The editor interface includes a tab bar at the top with a '+' icon and a close 'x' icon. A vertical line of tilde '~' characters is visible on the left margin.

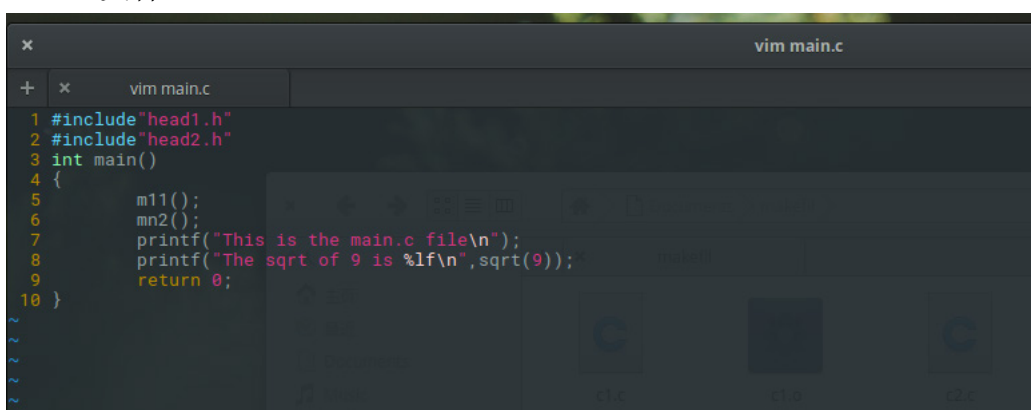
c2.c 文件

A screenshot of a vim editor window titled 'vim c2.c'. The editor shows a C source file with the following content:

```
1 #include"head2.h"
2 #include"head1.h"
3 int m2()
4 {
5     printf("This is the c2.c file.\n");
6     printf("The sqrt of 1000 is %lf\n",sqrt(1000));
7     return 0;
8 }
```

The editor interface includes a tab bar at the top with a '+' icon and a close 'x' icon. A vertical line of tilde '~' characters is visible on the left margin.

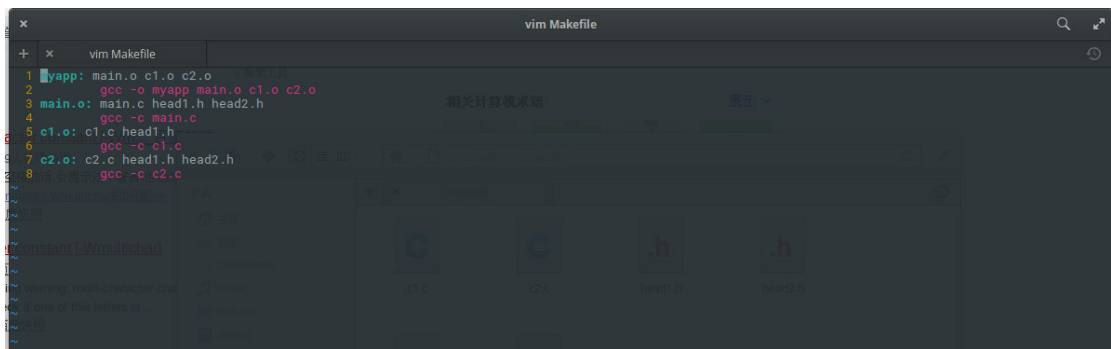
main.c 文件

A screenshot of a vim editor window titled 'vim main.c'. The editor shows a C source file with the following content:

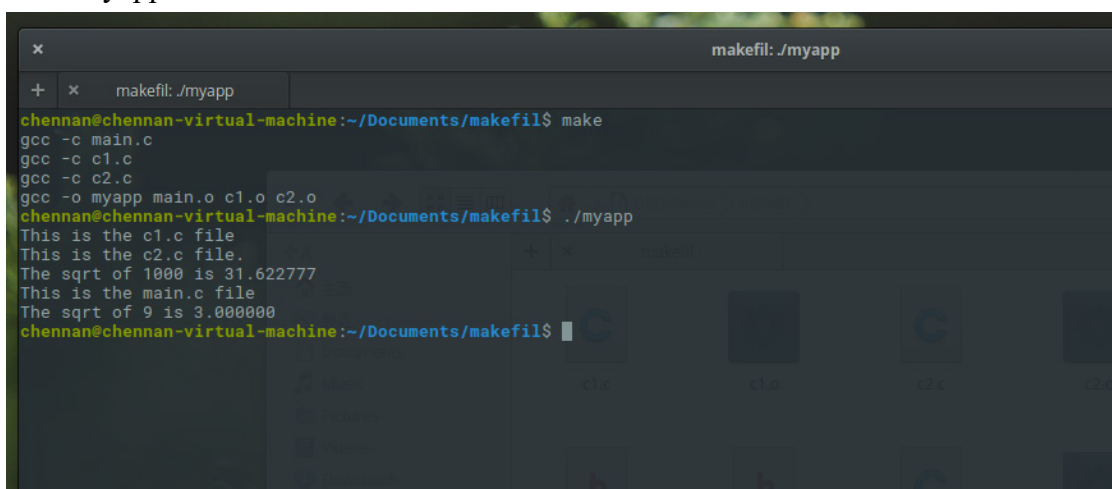
```
1 #include"head1.h"
2 #include"head2.h"
3 int main()
4 {
5     m11();
6     mn2();
7     printf("This is the main.c file\n");
8     printf("The sqrt of 9 is %lf\n",sqrt(9));
9     return 0;
10 }
```

The editor interface includes a tab bar at the top with a '+' icon and a close 'x' icon. A vertical line of tilde '~' characters is visible on the left margin. At the bottom of the editor, there are three icons labeled 'c1.c', 'c1.o', and 'c2.c'.

依赖关系如下 makefile 文件所示:



执行 makefile 文件，编译 myapp 程序。  
执行 myapp 程序，运行正常，输出正确结果。



## 四. 结论分析:

通过本次 Linux 下 C 编程实验,我对 Linux 下的 C 编程有了更加深入地理解。在此次实验中,我首先使用了 vi 编辑器,后来换成了功能更加强大的 vim 编辑器,通过在其中编写程序,我对 vi/vim 编辑器的操作有了一定的了解。写完程序后由自己而不是 IDE 来编译代码,更令我对 C 程序的编译过程有了深入的理解,编译中不同的命令会产生不同的结果,而且他们之间还有严格的先后顺序,这要求我们必须真正理解 C 程序的编译原理,才能不出错的完成这一步。写代码出 bug 是再正常不过的一件事,通过 GDB 强大的调试功能,我们可以逐步,仔细的查找程序中的问题,改正错误。越大的工程代码量越大,当文件变多后,手动编译程序变得越来越麻烦,这时候善于利用 makefile,可以节约很多工作。而且编写简单的 makefile 文件,也让我对 makefile 的原理与使用方法有了更深入的理解。

