**实验1：Linux基本操作**

**1. 实验目的**

1. 熟悉Ubuntu操作系统环境；
2. 会使用Ubuntu下的常用命令mkdir、cp、cd、ls、mv、rm等；
3. 熟悉vi、vim、gedit文本编辑器的使用，掌握在Linux环境下文本文件的编辑方法；
4. 掌握使用GCC编译、GDB调试代码的基本命令与操作。

**2. 实验内容**

1. 第一部分：Linux基本操作
2. 第二部分：VIM操作
3. 第三部分：GCC代码编译与运行
4. 第四部分：GDB代码调试

**3 实验准备**

可参考网址或资料：

* <http://www.runoob.com/linux/linux-tutorial.html>
* <http://www.yolinux.com/TUTORIALS/GDB-Commands.html>
* Linux Cookbook

更多linux信息请同学们动手上网搜索并实践。

**4 实验步骤与操作指导**

**第一部分：Linux基本操作**

请根据下表左列的“操作”说明，进行相关操作，并将对应命令填写在表格右列的“命令”一列。没有对应命令的不需写。

|  |  |  |
| --- | --- | --- |
| 步骤 | 操作 | 命令 |
| 1 | 登陆Ubuntu操作系统 |  |
| 2 | 在自己的某个目录（简称目录A）下新建一个子目录，并命名（假如我命令为B） | mkdir A  mkdir A/B |
| 3 | 回到目录A | cd .. |
| 4 | 进入自己刚建的子目录B | cd B |
| 5 | 在该子目录B下创建一个test.txt的文件 | touch test.txt |
| 6 | 打开该test.txt文件，并输入"Hello, operation system students!"，然后关闭该文件。 | vi test.txt  输入  : wq |
| 7 | 拷贝test.txt文件并重命名为temp.txt | cp ./test.txt ./temp.txt |
| 8 | 删除temp.txt文件 | rm temp.txt |
| 9 | 查看自己当前所在路径 | pwd [-P] |
| 10 | 清除屏幕所有显示 | clear |
| 11 | 查看当前目录所有文件的list | ls |
| 12 | 用uname相关命令显示你的操作系统的所有信息 | uname -a |
| 13 | 重命名test.txt文件为test1.txt | mv test.txt test1.txt |
| 14 | 将test1.txt复制到目录A下并重命名为test2.txt | cp test1.txt ../test2.txt |
| 15 | 使用帮助查看关于命令dir的说明 | man dir |
| 16 | 删除自己刚创建的目录A | cd ..  cd ..  rm -rf A |

**第二部分：VIM操作**

请根据以下步骤练习使用VIM（或者你喜欢的编辑器）。

（1）启动vim，新建一个mycode.c的文件。

（2）在Mycode.c中输入以下C程序

#include<stdio.h>

int main( )

{

printf(“Hello world!\n”);

return 0;

}

（3）保存文件并退出。

（4）用vim再次打开文件mycode.c

（5）修改程序为：

#include<stdio.h>

int main( )

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" Hello world!\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

return 0;

}

（5）再次保存上述程序。

**第三部分：****GCC代码编译与运行**

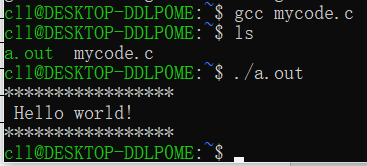
（1）使用GCC对第二部分的mycode.c代码进行编译并运行，写出命令及运行结果。

**//给出命令与结果**

gcc mycode.c

ls

./a.out

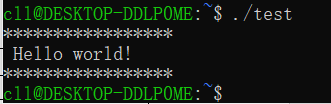


gcc mycode.c -o test

ls

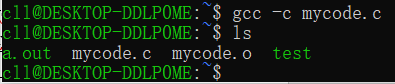
./test





gcc -c mycode.c

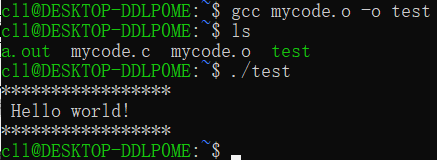
ls



gcc mycode.o -o test

ls

./test



（2）依次尝试以下命令，并根据你的理解在表格右列填写如何运行生成的文件，以及这些命令的作用。

|  |  |  |
| --- | --- | --- |
| 命令 | 运行生成文件 | 说明 |
| gcc mycode.c | ./a.out | 直接运行代码 |
| gcc mycode.c -o test | ./test | 将mycode.c预处理、汇编、编译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。 |
| gcc -c mycode.c | 无法运行 | 只激活预处理、编译和汇编,也就是只把程序做成obj文件,即将mycode.c编译输出为mycode.o文件。 |
| gcc mycode.o -o test | ./test | 通过gcc编译了obj文件，-o选项用来指定输出文件的文件名。 |

**第四部分：GDB代码调试**

请根据以下调试举例写出对应命令，并给出命令与结果，其中结果可通过贴图的方式给出。通过该过程掌握VIM的常用命令。

（1）使用vim编写程序mycode.c，具体内容可以自由发挥。例如下面这段：

#include<stdio.h>

int main( )

{

int i=0;

int a=0;

for (i=0;i<5;i++)

{

a = a+2;

}

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" Hello world!\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

return 0;

}

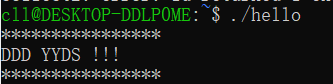
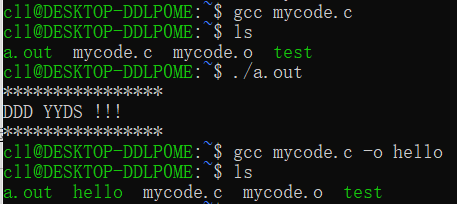
（2）在命令行里使用下列命令运行这段代码：

1）在命令行里输入gcc mycode.c，然后按回车。

2）在命令行里输入ls，会发现生成了绿色的a.out文件。

3）在命令行里输入./a.out，给出输出的结果。

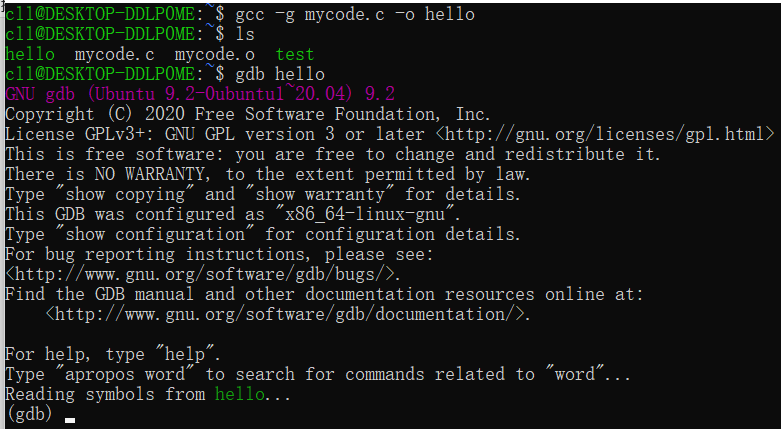
4）下面，请你重新编译这段代码，并生成名为hello的可执行文件，然后运行hello。



（3）开始使用GDB调试

**//给出命令与结果**

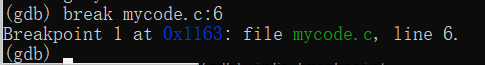
gdb hello



（4）在GDB下进行分步调试，依次学习如何使用命令break、info、next

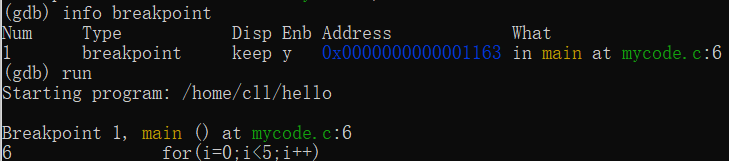
**//给出命令与结果**

break hello:6

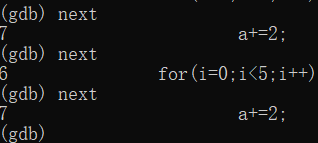


info breakpoint

run



next



（5）自己写一段代码，里面定义一个变量i，在GDB下进行分步调试，完成以下操作。

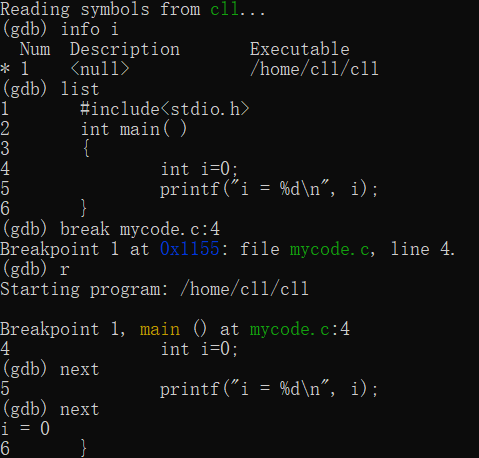
a. 在gdb里打印变量i的值

**//给出命令与结果**

break mycode.c:4

r

next



b. 在gdb里打印变量i的地址

**//给出命令与结果**

print &i



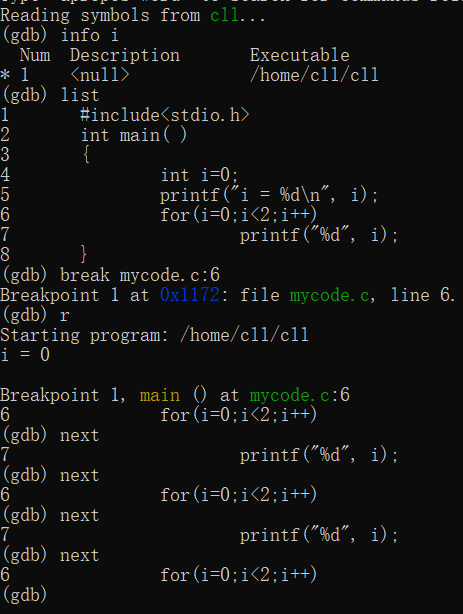
c. 给代码加点功能，然后对修改的代码使用gdb分步调试，例如加一个循环，每次i增加1就打印一次，循环100次。

**//给出命令与结果，此处不需要严格的调试100次，只要举例说明你会根据设置的条件进行调试即可。**

break mycode.c:6

r

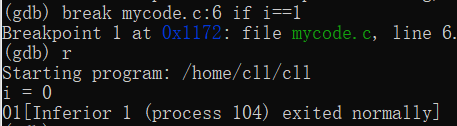
next



d. 在某些条件下设置断点，例如当i=20时在第5行设置断点

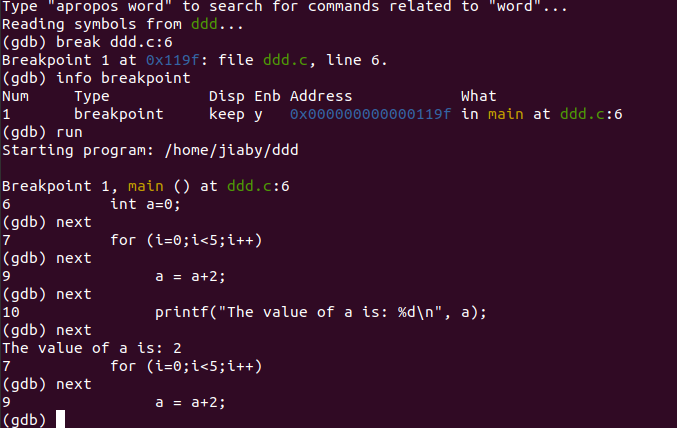
**//给出命令与结果**

break mycode.c:6 if i==1



**下面是在gdb下进行调试的基本操作举例。**





**5 实验总结**

请根据自己的实验情况对本次实验进行总结，例如，是否了解Linux的基本操作命令，在实验中遇到了什么困难以及如何解决的。

本次实验通过命令行实现一些平时经常用到的操作，我对基于Linux系统的基本操作命令有了初步认识。印象最深的还是GCC代码编译与运行的部分，通过四个不同的命令，我基本了解了几种关于编译命令的结果以及处理方法，比如生成了obj文件后要再次编译才能算是把代码执行了。

困难阻碍比较大的部分还是最后的gdb调试部分，经常性的报错加上对基于Linux下命令的含义认识不到位，这让我对错误不断，几经调试才终于成功。