一、实验目标：

熟悉 Linux 上 C 程序的编译和调试工具，包括以下内容：
1. 了解 Linux 操作系统及其常用命令
2. 掌握编译工具 gcc 的基本用法
3. 掌握使用 gdb 进行程序调试

二、实验环境与工件

1.个人电脑
2. Fedora 13 Linux 操作系统
3. gcc
4. gdb

三、实验内容与步骤

1. 根据实验一：实验环境配置与使用.ppt 熟悉 Linux 基本操作（P.1 – P.28），然后根据以下过程创建用户：用户名为学生名称加学号，如**吴坤汉**，学号**2015170297**，则该用户名为 wukunhan_2015170297。按照 1.1~1.3 完成并截图，截图需要有运行的命令及其结果。另外：**后面的题目必须在该新建用户下完成。（30 分）**

   实验步骤：

   1.1. 首先切换为超级用户

   输入 su 进入超级用户：

```
dongyunhao2019284073@ubuntu:~$ su
Password:
su: Authentication failure
```

发现出现 Authentication failure，可能是密码忘记了，此时，可以使用`*sudo passwd root*`来给 root 重设密码：

```
dongyunhao2019284073@ubuntu:~$ su
Password:
su: Authentication failure
dongyunhao2019284073@ubuntu:~$ sudo passwd root
New password:
Retype new password:
passwd: password updated successfully
```

如上，则密码已经成功的重设了。

#

1.2. 参考以下命令创建新用户，设置新建用户的密码，注意：只有设置了密码才能激活用户，否则无法以该用户身份登录

①首先查询当前用户：

```
dongyunhao2019284073@ubuntu:~$ whoami
dongyunhao2019284073
```

②创建新用户 dongyunhao_2019284073

在终端中输入 *sudo adduser dongyunhao_2019284073*

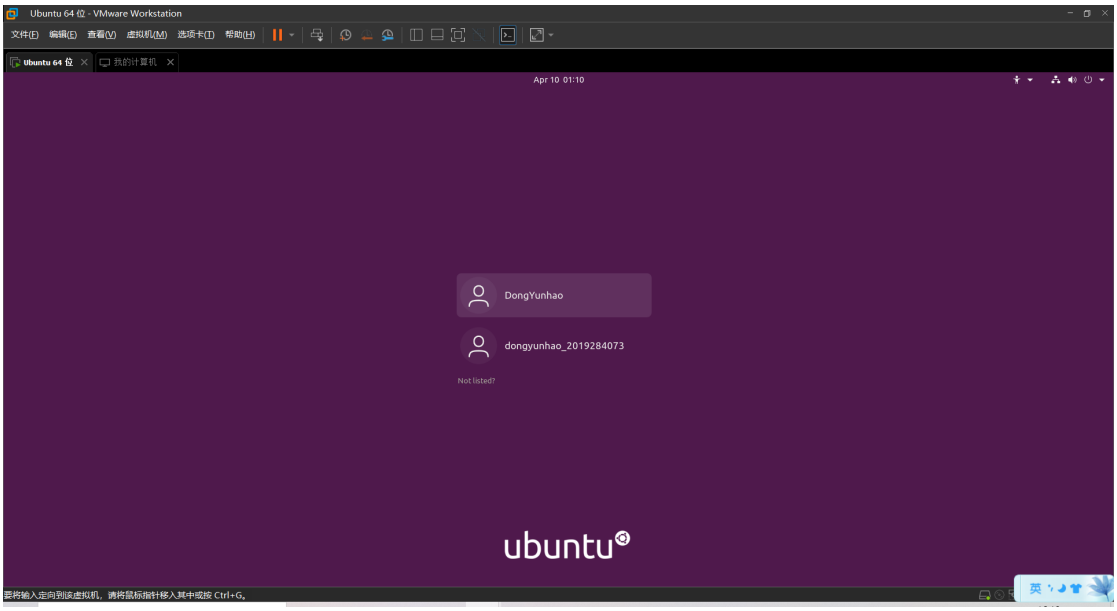
并输入超级用户密码与新创建账户密码后，再输入一些账户信息最后确认即

可完成创建。

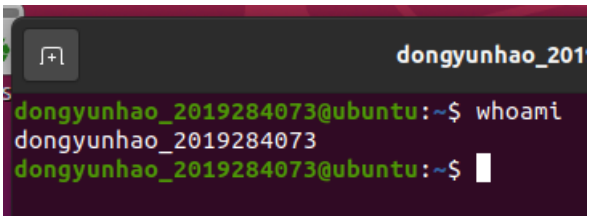
```
dongyunhao2019284073@ubuntu:~$ sudo adduser dongyunhao_2019284073
[sudo] password for dongyunhao2019284073:
Adding user `dongyunhao_2019284073' ...
Adding new group `dongyunhao_2019284073' (1001) ...
Adding new user `dongyunhao_2019284073' (1001) with group `dongyunhao_2019284073
' ...
Creating home directory `/home/dongyunhao_2019284073' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for dongyunhao_2019284073
Enter the new value, or press ENTER for the default
        Full Name []: dongyunhao_2019284073
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
```

1.3. 注销当前用户，并以新建的用户身份登录，登录后运行 $ *whoami*，并进行截图；

①点击左上角 log out->switch user 并选择新创建的用户，输入密码后即可登录



②查看当前用户，在终端中输入 *whoami* 即可查看当前用户名



2. 新建用户主目录下创建子目录：**gdbdebug**，并进入 gdbdebug 子目录。将过程和结果截图。（10 分）

①输入命令：*sudo mkdir gdbdebug* 并运行

```
dongyunhao_2019284073@ubuntu:~$ sudo mkdir gdbdebug
[sudo] password for dongyunhao_2019284073:
```

可以发现报错，提示“not in the sudoers file”这说明当前用户（新创建的用户）不具有 sudo 的权限，因此我们需要给当前用户权限。此时需要进入 root 进行操作

②进入 root：

输入 *su* 后再输入 visudo 进入权限管理操作文件页面。并给当前用户 dongyunhao_2019284073 添加权限后退出即可。

```
root@ubuntu: /home/dongyunhao_2019284073
  GNU nano 4.8                    /etc/sudoers.tmp                    Modified

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
dongyunhao_2019284073 ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
sudo    ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d


                          [ Unbindable key: M-^A ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

③重新创建文件夹：

此时再使用 *sudo mkdir gdbdebug* 命令，即可成功创建

```
dongyunhao_2019284073@ubuntu:~$ sudo mkdir gdbdebug
[sudo] password for dongyunhao_2019284073:
dongyunhao_2019284073@ubuntu:~$
```

④查看创建的文件夹：

输入 ls 命令查看文件夹

```
dongyunhao_2019284073@ubuntu:~$ ls
Desktop    Downloads  Music     Public     Videos
Documents  gdbdebug   Pictures  Templates
```

⑤进入文件夹：

输入 *cd gdbdebug* 即可进入文件夹

```
dongyunhao_2019284073@ubuntu:~$ cd gdbdebug
dongyunhao_2019284073@ubuntu:~/gdbdebug$
```

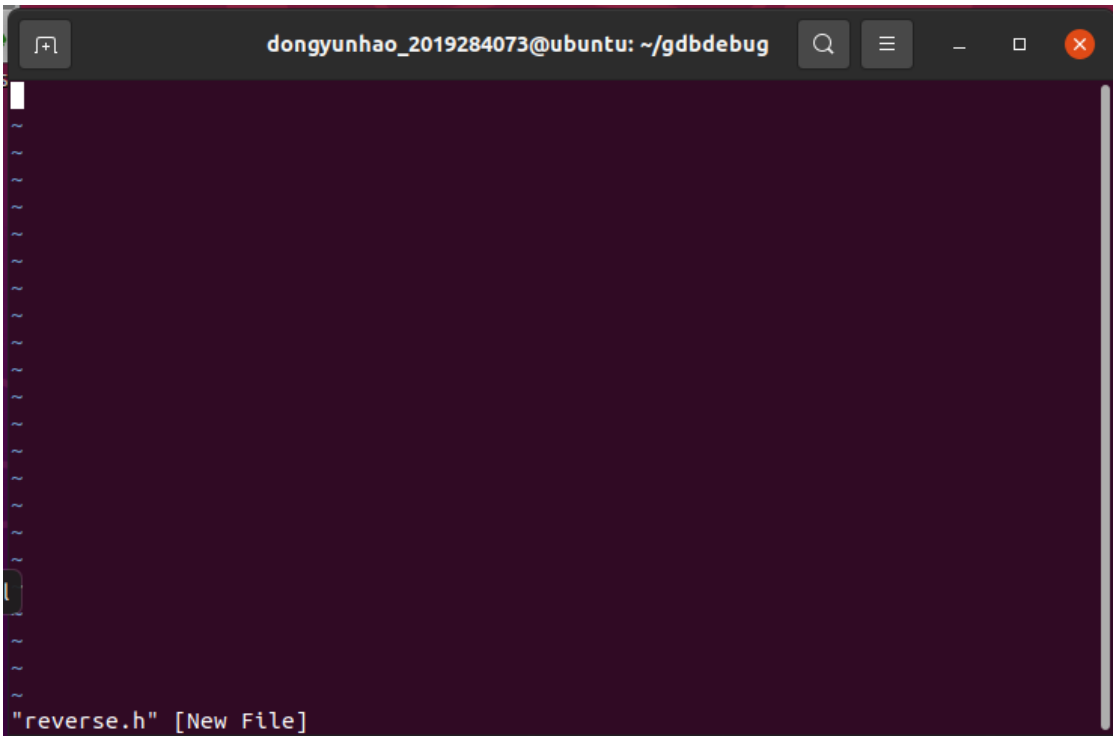
3. 使用 vi 编辑以下两个文件并编译和运行，截图（30 分）

3.1. 编辑 reverse.h

①创建 reverse.h 文件：

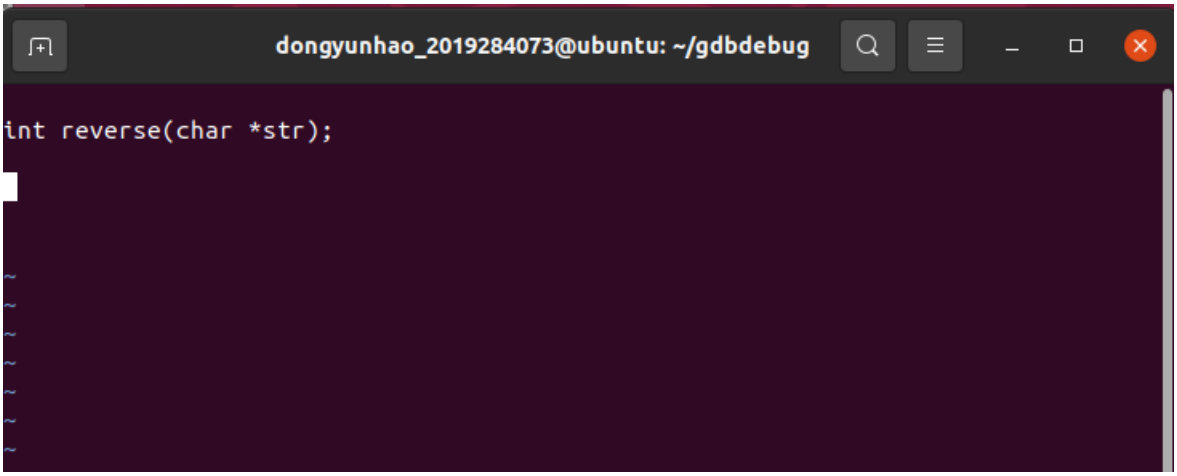
输入 vi reverse.h 并运行

```
dongyunhao_2019284073@ubuntu:~/gdbdebug$ vi reverse.h
```

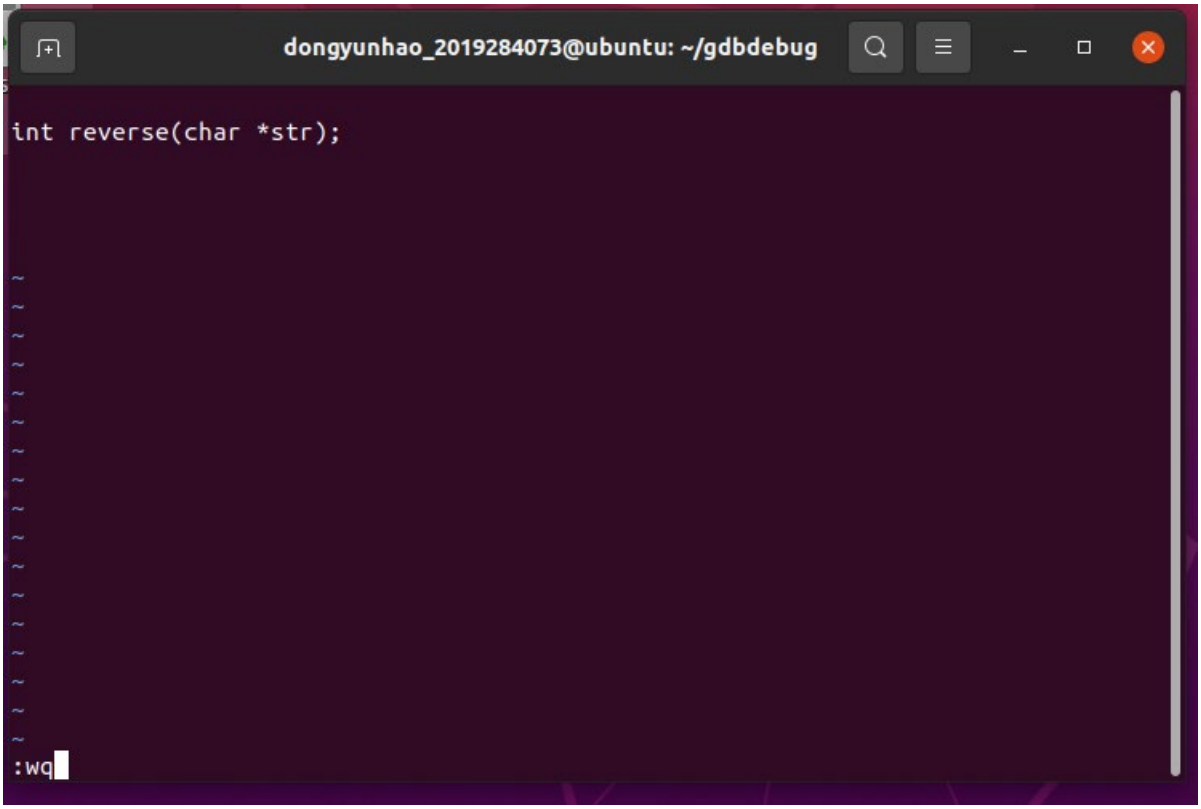


②编辑 reverse.h

进入编辑页面后先输入 i 以切换到文字输入模式，然后输入如下代码



输入完成后按“esc”结束输入，并键入“：wq”对代码进行保存

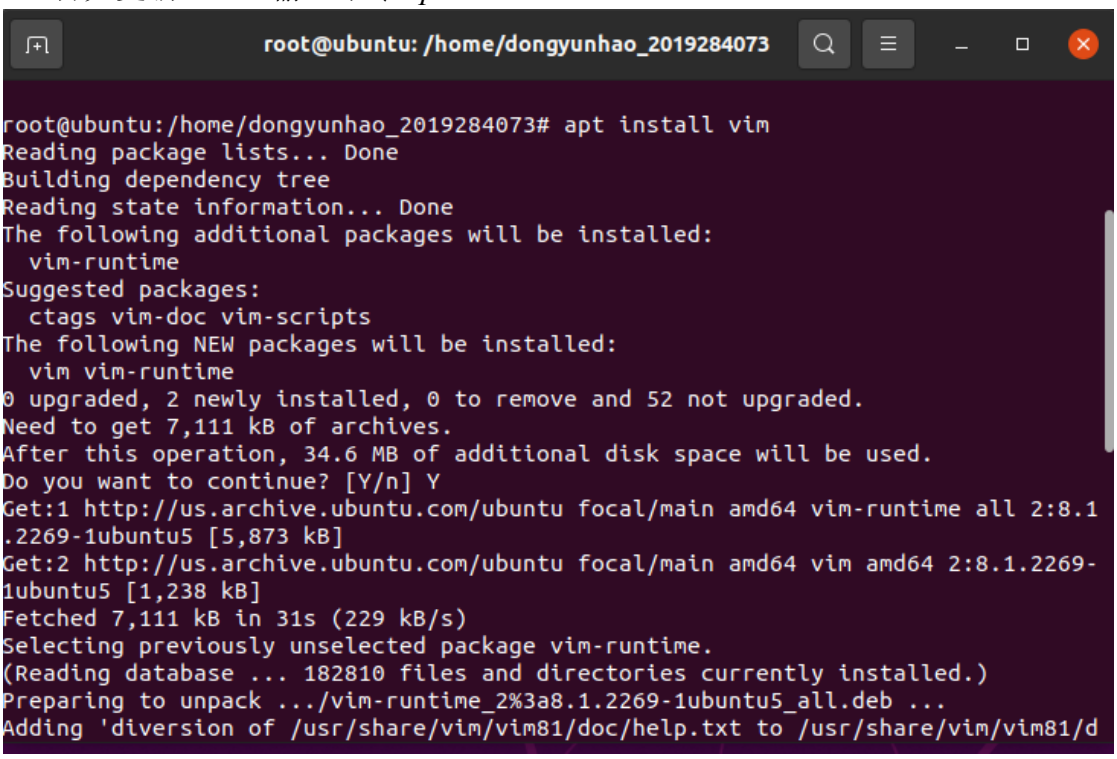


3.2. 编辑 reverse.c

①显示代码高亮与行号：

在上一步骤的代码编写中，发现在 vim 进行编辑时代码没有代码高亮也没有行号。因此可以打开对应的设置。

首先更新 vim，输入命令 *apt install vim*



将自动更新下载 vim，完成下载后，进入 vim 的设置文件。输入 *vim ~/.vimrc*，并输入如下命令：

```
root@ubuntu: /home/dongyunhao_2019284073
1 syntax on
2 set number
3
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"~/.vimrc" 3L, 22C                                    1,1           All
```

保存并退出后即可看到代码已经有了代码高亮和行号。

②创建并编写 reverse.c

与上一部分中创建的代码相同，输入 vi reverse.c 并运行。并在其中写入代码。

```
dongyunhao_2019284073@ubuntu: ~/gdbdebug
#include<stdio.h>
#include"reverse.h"
/************************/
int reverse(str)
        char *str;
{
        int i;
        int len;
        char c;
        len = strlen(str);
        for(i = 0;i < len/2; i++)
        {
                c = *str +i;
                *(str + i) = *str + len - i - 1;
                *(str + len -i -1) = c;
        }
}

int main(void)
{
        char str[1024];
        printf("Give me a word to reverse:\n");
        scanf("%s",&str);
-- INSERT --                                          1,1           Top
```

代码编写完成后输入“wq：”进行保存并退出。

3.3. 按以下步骤编译，如有警告信息，请修改代码至无警告信息

$gcc －Wall reverse.c －o reverse

①输入命令并执行

```
dongyunhao_2019284073@ubuntu:~/gdbdebug$ gcc -Wall reverse.c -o reverse

Command 'gcc' not found, but can be installed with:

apt install gcc
Please ask your administrator.
```

发现未安装 gcc，则手动进行安装

②安装 gcc

首先切换到 root 并输入代码进行安装

```
dongyunhao_2019284073@ubuntu:~/gdbdebug$ su
Password:
root@ubuntu:/home/dongyunhao_2019284073/gdbdebug# apt install gcc
```

弹出的提示信息中选择 Yes

```
root@ubuntu:/home/dongyunhao_2019284073/gdbdebug# apt install gcc
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu gcc-9 libasan5 libatomic1
  libbinutils libc-dev-bin libc6-dev libcrypt-dev libctf-nobfd0 libctf0
  libgcc-9-dev libitm1 liblsan0 libquadmath0 libtsan0 libubsan1 linux-libc-dev
  manpages-dev
Suggested packages:
  binutils-doc gcc-multilib make autoconf automake libtool flex bison gcc-doc
  gcc-9-multilib gcc-9-doc gcc-9-locales glibc-doc
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu gcc gcc-9 libasan5
  libatomic1 libbinutils libc-dev-bin libc6-dev libcrypt-dev libctf-nobfd0
  libctf0 libgcc-9-dev libitm1 liblsan0 libquadmath0 libtsan0 libubsan1
  linux-libc-dev manpages-dev
0 upgraded, 21 newly installed, 0 to remove and 52 not upgraded.
Need to get 20.3 MB of archives.
After this operation, 93.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

```
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 binutils-comm
on amd64 2.34-6ubuntu1.1 [207 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libbinutils a
md64 2.34-6ubuntu1.1 [475 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libctf-nobfd0
 amd64 2.34-6ubuntu1.1 [47.1 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libctf0 amd64
 2.34-6ubuntu1.1 [46.6 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 binutils-x86-
64-linux-gnu amd64 2.34-6ubuntu1.1 [1,613 kB]
9% [5 binutils-x86-64-linux-gnu 423 kB/1,613 kB 26%]
```

等待安装完毕即可

```
root@ubuntu: /home/dongyunhao_2019284073/gdbdebug
Setting up linux-libc-dev:amd64 (5.4.0-70.78) ...
Setting up libctf-nobfd0:amd64 (2.34-6ubuntu1.1) ...
Setting up libasan5:amd64 (9.3.0-17ubuntu1~20.04) ...
Setting up libquadmath0:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up libatomic1:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up libubsan1:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up libcrypt-dev:amd64 (1:4.4.10-10ubuntu4) ...
Setting up libbinutils:amd64 (2.34-6ubuntu1.1) ...
Setting up libc-dev-bin (2.31-0ubuntu9.2) ...
Setting up liblsan0:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up libitm1:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up libtsan0:amd64 (10.2.0-5ubuntu1~20.04) ...
Setting up libctf0:amd64 (2.34-6ubuntu1.1) ...
Setting up libgcc-9-dev:amd64 (9.3.0-17ubuntu1~20.04) ...
Setting up libc6-dev:amd64 (2.31-0ubuntu9.2) ...
Setting up binutils-x86-64-linux-gnu (2.34-6ubuntu1.1) ...
Setting up binutils (2.34-6ubuntu1.1) ...
Setting up gcc-9 (9.3.0-17ubuntu1~20.04) ...
Setting up gcc (4:9.3.0-1ubuntu2) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
root@ubuntu:/home/dongyunhao_2019284073/gdbdebug#
```

③编译并运行

再次输入命令进行编译。可以观察到如下几个信息：

```
dongyunhao_2019284073@ubuntu: ~/gdbdebug
reverse.c  reverse.h
dongyunhao_2019284073@ubuntu:~/gdbdebug$ gcc -Wall reverse.c -o reverse
reverse.c: In function ‘reverse’:
reverse.c:10:8: warning: implicit declaration of function ‘strlen’ [-Wimplicit-f
unction-declaration]
   10 |  len = strlen(str);
      |        ^~~~~~
reverse.c:10:8: warning: incompatible implicit declaration of built-in function
‘strlen’
reverse.c:3:1: note: include ‘<string.h>’ or provide a declaration of ‘strlen’
    2 | #include"reverse.h"
  +++ |+#include <string.h>
    3 | /************************/
reverse.c: In function ‘main’:
reverse.c:23:10: warning: format ‘%s’ expects argument of type ‘char *’, but arg
ument 2 has type ‘char (*)[1024]’ [-Wformat=]
   23 |  scanf("%s",&str);
      |         ~^  ~~~~
      |          |  |
      |          |  char (*)[1024]
      |          char *
reverse.c: In function ‘reverse’:
reverse.c:17:1: warning: control reaches end of non-void function [-Wreturn-type
]
```

a. strlen 函数未声明。又注意到提示说<string.h>中含有对 strlen 函数的定义，故可以在代码中添加此头文件

b.在 main 函数中%s 的输入格式不应存在 char *格式，但第二个参数中存在 char *格式，因此需要去掉取地址符

c.int 型返回值函数没有返回值。在函数末尾补一个 return 1 即可

进入代码进行修改，修改完成后保存即可。

④重新进行编译

```
dongyunhao_2019284073@ubuntu:~/gdbdebug$ vi reverse.c
dongyunhao_2019284073@ubuntu:~/gdbdebug$ gcc -Wall reverse.c -o reverse
dongyunhao_2019284073@ubuntu:~/gdbdebug$
```

可以看到重新编译后警告信息消失了

3.4. 运行程序

输入代码 ./reverse 进行运行

```
dongyunhao_2019284073@ubuntu:~/gdbdebug$ ./reverse
Give me a word to reverse:
MattDong
REVERSED:TZYXWVUM
```

虽然程序执行了，但输出是错误的，因此我们需要进行 debug 并对代码进行修改调试。

4. 按照以下过程调试并修正 reverse.c,请参考过程截图。(30 分)

4.1. 编译时加入调试信息

$gcc -g reverse.c -o reverse1

```
dongyunhao_2019284073@ubuntu:~/gdbdebug$ gcc -g reverse.c -o reverse1
```

4.2. 启用 GDB 调试

```
dongyunhao_2019284073@ubuntu:~/gdbdebug$ gdb ./reverse1
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 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 ./reverse1...
(gdb)
```

4.3. 键入 list,查看源代码并根据行号/函数名设置断点（断点位置仅供参考，不需雷同）

①输入 list 对代码进行查看

```
(gdb) list
10              int i;
11              int len;
12              char c;
13              len = strlen(str);
14              for(i = 0;i < len/2; i++)
15              {
16                      c = *str +i;
17                      *(str + i) = *str + len - i - 1;
18                      *(str + len -i -1) = c;
19              }
(gdb) break main
```

②添加并查看断点：

由于函数中第 16,17,18 行发生的操作比较容易发生问题，故在此添加断点。

```
(gdb) break main
Breakpoint 1 at 0x1267: file reverse.c, line 24.
(gdb) break 16
Breakpoint 2 at 0x11f1: file reverse.c, line 16.
(gdb) break 17
Breakpoint 3 at 0x1202: file reverse.c, line 17.
(gdb) break 18
Breakpoint 4 at 0x1229: file reverse.c, line 18.
```

完成断点添加后，可以输入 info break 对断点信息进行查看

```
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000001267 in main at reverse.c:24
2       breakpoint     keep y   0x00000000000011f1 in reverse at reverse.c:16
3       breakpoint     keep y   0x0000000000001202 in reverse at reverse.c:17
4       breakpoint     keep y   0x0000000000001229 in reverse at reverse.c:18
```

③开始调试

输入 run 进行调试

4.4.　　观察变量值，并作分析，推测错误（过程仅供参考，不需雷同）

开始运行，在程序运行中可以输入 c 进行跨越一个断点的运行

```
Breakpoint 2, reverse (str=0x7fffffffdb70 "MattDong") at reverse.c:16
16                      c = *str +i;
(gdb) c
Continuing.

Breakpoint 3, reverse (str=0x7fffffffdb70 "MattDong") at reverse.c:17
17                      *(str + i) = *str + len - i - 1;
(gdb) c
Continuing.

Breakpoint 4, reverse (str=0x7fffffffdb70 "TattDong") at reverse.c:18
18                      *(str + len -i -1) = c;
(gdb) c
c LibreOffice Writer

Breakpoint 2, reverse (str=0x7fffffffdb70 "TattDonM") at reverse.c:16
16                      c = *str +i;
(gdb) c
Continuing.
```

```
Breakpoint 2, reverse (str=0x7fffffffdb70 "TZttDoUM") at reverse.c:16
16                      c = *str +i;
(gdb) c
Continuing.

Breakpoint 3, reverse (str=0x7fffffffdb70 "TZttDoUM") at reverse.c:17
17                      *(str + i) = *str + len - i - 1;
(gdb) c
Continuing.

Breakpoint 4, reverse (str=0x7fffffffdb70 "TZYtDoUM") at reverse.c:18
18                      *(str + len -i -1) = c;
(gdb) c
Continuing.
Terminal
Breakpoint 2, reverse (str=0x7fffffffdb70 "TZYtDVUM") at reverse.c:16
16                      c = *str +i;
(gdb) c
Continuing.

Breakpoint 3, reverse (str=0x7fffffffdb70 "TZYtDVUM") at reverse.c:17
17                      *(str + i) = *str + len - i - 1;
(gdb) c
Continuing.

Breakpoint 4, reverse (str=0x7fffffffdb70 "TZYXDVUM") at reverse.c:18
18                      *(str + len -i -1) = c;
(gdb) c
Continuing.
REVERSED:TZYXWVUM
```

从程序运行间可以看到，当第一次对字符数组操作时，第一个“M”本应被替换成“G”缺被替换成了“T”通过分析代码可知，代码的本意应该是通过指针交换第一个与最后一个字符，而实际上却是发生了偏移。这是因为，每次的指针应该是*(str+i)而不是*str+i，在 C 语言运行时，将先执行指针符号，再执行加法，因此需将两处都加上括号。

4.5. 修正程序并运行

通过上面的分析，对代码进行修改如下：

```
dongyunhao_2019284073@ubuntu: ~/gdbdebug
#include<stdio.h>
#include"reverse.h"
#include<string.h>


/************************/
int reverse(str)
        char *str;
{
        int i;
        int len;
        char c;
        len = strlen(str);
        for(i = 0;i < len/2; i++)
        {
                c = *(str +i);
                *(str + i) = *(str + len - i - 1);
                *(str + len -i -1) = c;
        }
        return 1;
}

int main(void)
{
        char str[1024];
        printf("Give me a word to reverse:\n");
        scanf("%s",str);
        reverse(str);
        printf("REVERSED:%s\n", str);
}
```

修改完成后重新进行编译并运行

```
dongyunhao_2019284073@ubuntu:~/gdbdebug$ vi reverse.c
dongyunhao_2019284073@ubuntu:~/gdbdebug$ gcc -Wall reverse.c -o reverse
dongyunhao_2019284073@ubuntu:~/gdbdebug$ ./reverse
Give me a word to reverse:
MattDong
REVERSED:gnoDttaM
```

结果正确，程序运行无误

四、实验结果

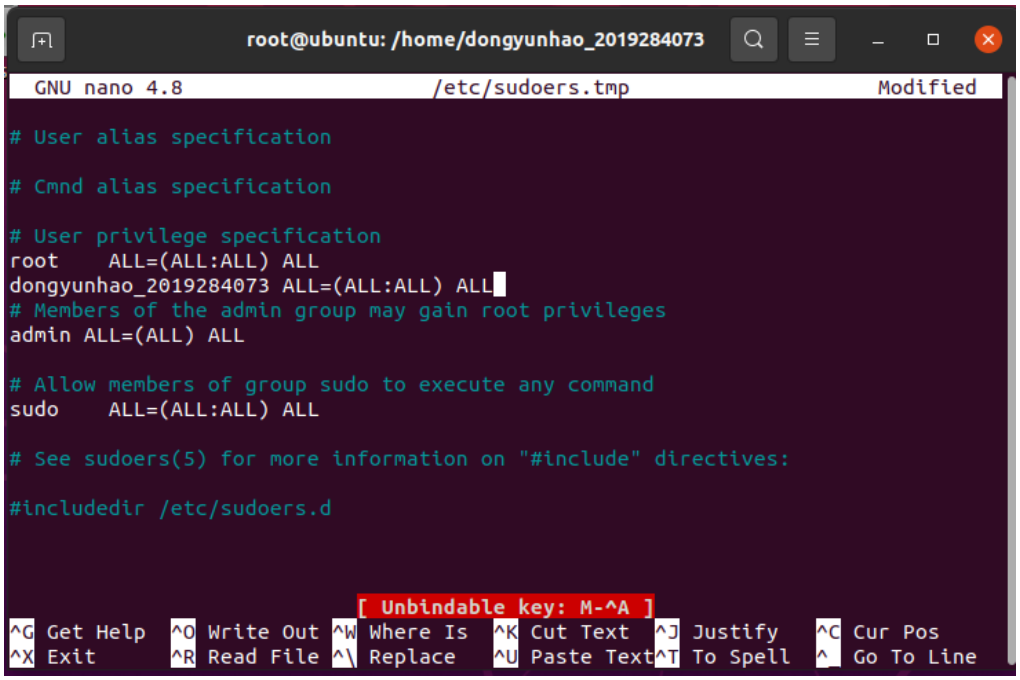
```
dongyunhao_2019284073@ubuntu:~/gdbdebug$ vi reverse.c
dongyunhao_2019284073@ubuntu:~/gdbdebug$ gcc -Wall reverse.c -o reverse
dongyunhao_2019284073@ubuntu:~/gdbdebug$ ./reverse
Give me a word to reverse:
MattDong
REVERSED:gnoDttaM
```

如上图，程序运行无误，最后也输出了正确的结果。

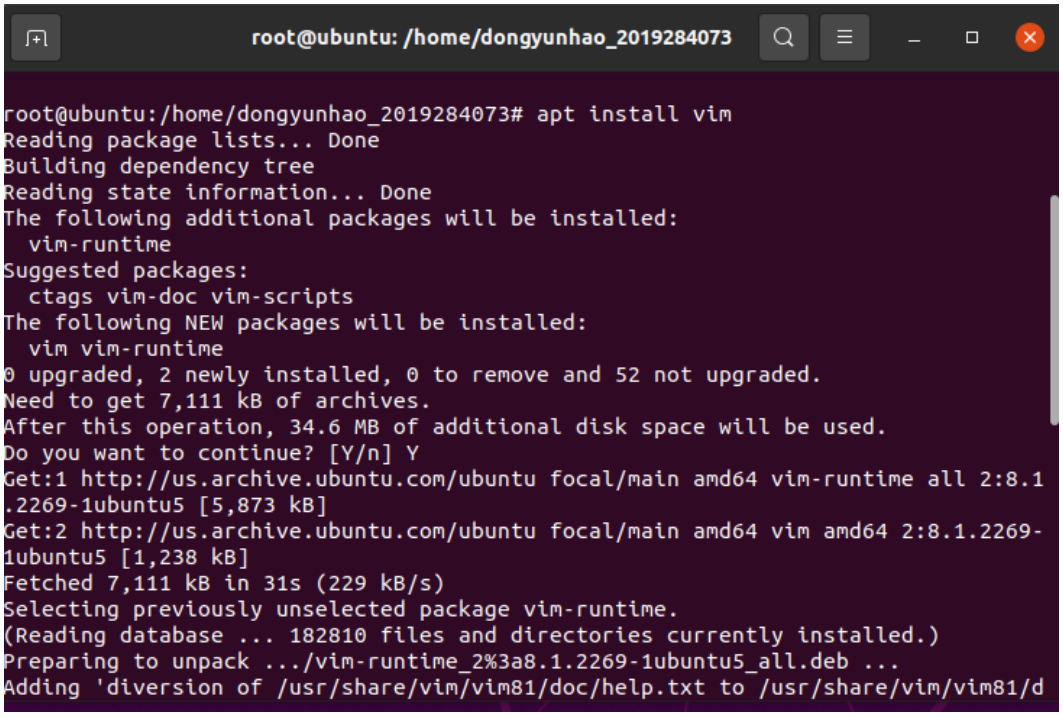
五、实验总结与体会

本次实验进行的比较顺利，但也遇到了一些问题：

1、Ubuntu 上 Linux 系统与 Windows 系统有一样的地方也有区别，例如对文件读写时可能会因为权限问题造成不能读写。此时需要进入 root 对相关权限文件进行修改。在权限文件中添加用户之后便可以顺利对文件进行读写操作了。



2、在编写代码过程中没有代码高亮与行号。我自己下载了新的 vim，并通过修改 vim 的配置文件最终得到解决。



3、编写 C 程序代码时要注意程序运行的优先级。本次实验中就是因为忽略了指针运算符（*）会在加法运算符（+）前执行导致程序错误。不过最后通过 debug 问题得到解决