

**实 验 报 告**

**课程设计名称：** 操作系统原理

**专 业 班 级 ：** 物联网工程1902班

**学 生 姓 名 ：** 王源

**学 号 ：** 201916070216

**指 导 教 师 ：** 廖庆荣

**课程实验时间：** 2021年5月6日

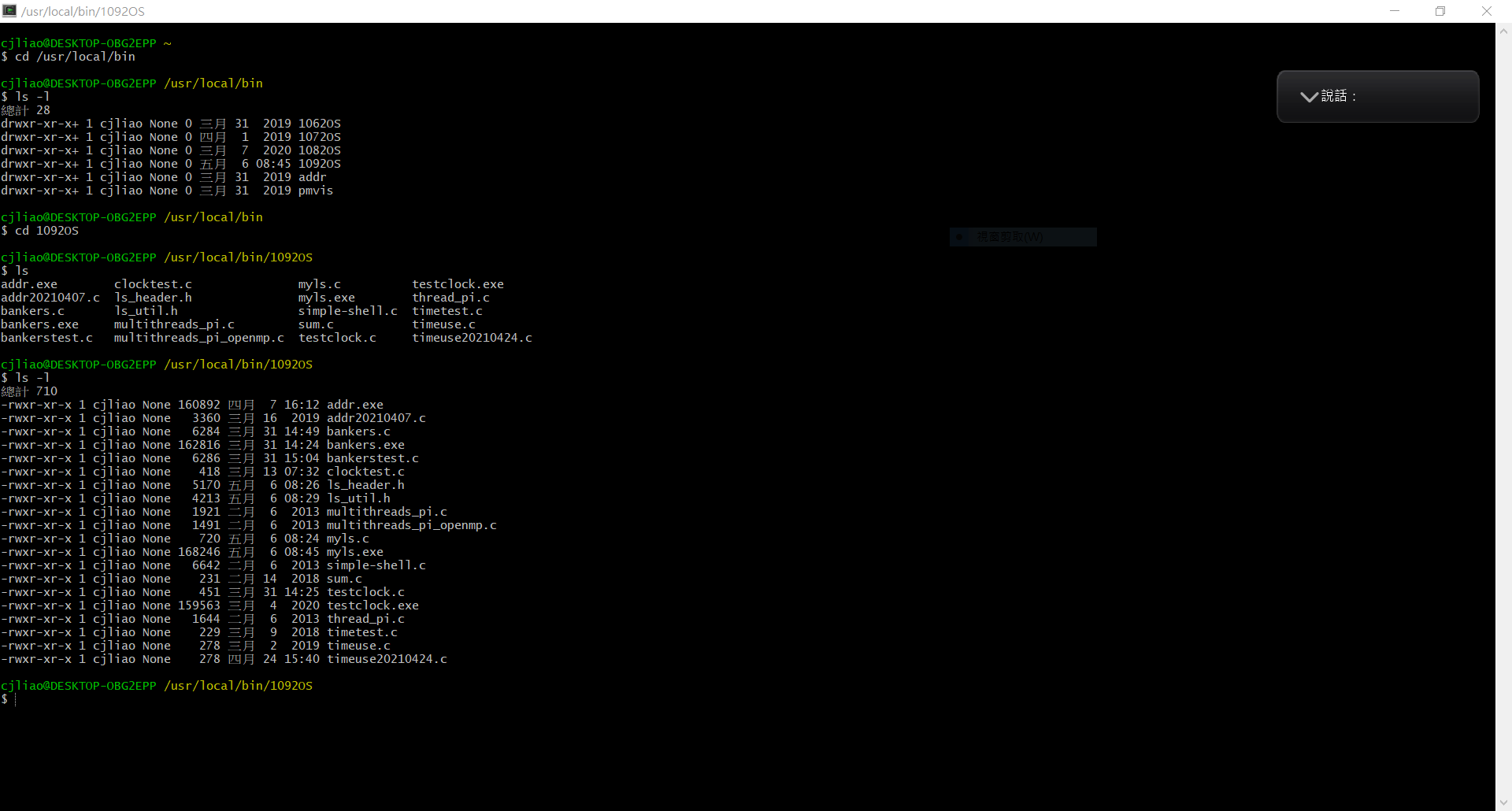
**实验四 文件管理的实现**

# 一、设计目的

1. 能了解操作系统的文件管理。

2. 能实现操作系统的文件管理功能。

# 二、设计要求与内容

1. 能实现操作系统文件管理列出所有文件ls的功能，如下图所示。

# 设计原理：包含程序想法和做法说明



说明：实验环境：ubuntu Linux操作系统、VMware\_ubuntu 20.04 LTS

vim\_g++、C/C++语言

Ls的基本概念：

1.ls命令是我们在进行linux操作中经常用到的一个命令，在设计文件服务器的时候，我们需要了解到ls 的底层实现。ls命令的功能是显示在当前目录下的所有文件，是最基本的命令之一。我们可以类似地通过使用dirent类和dirent结构体获取当前目录下的所有文件，将所有隐藏文件删去，将剩下文件排序输出即可。

Ls -l的基本概念：

2.ls -l的意义

显示当前目录下的所有文件的详细信息，以长格式显示目录下的内容列表。输出的信息从左到右依次包括文件名，文件类型、权限模式、硬连接数、所有者、组、文件大小和文件的最后修改时间等。

例：-rw -rw -r--   1  using using  3102  7月 22 17:06  test.c

　　　　drwxrwxr -x  2  using using  4096  7月 22 18:39  testdir

　　　　lrwxrwxrwx  1  using using      17   7月 22 18:43  shared -> /media/sf\_shared/

　　其中深蓝色为目录文件，天蓝色为软连接文件（具体颜色和vimrc配置有关）。

　　第一字段：首字母代表的是文件类型 ，其中"-"为普通文件、"d"为目录文件、"c"为字符设备文件、"b"为块设备文件、"p"为管道文件、"l"为链接文件、"s"为socket文件。“rwx”分别代表拥有读、写和执行权限，“-”代表无对应权限。三个“rwx”依次代表文件所有者、文件所有者所在用户组、其它用户对文件拥有的权限。

　　第二字段：文件硬连接数量

　　第三字段：文件拥有者

　　第四字段：文件拥有者所在组

　　第五字段：文件大小（以字节为单位）

　　第六字段：文件最后更改时间

第七字段：文件名（若为链接文件则追加显示其链接的原文件的路径）

做法说明：

在linux下使用C/C++语言，通过调用Linux系统的目录访问API来实现一个类似于ls命令功能的小程序，主要是可以练习程序对命令的解析和目录API函数的使用。

仿写ls -l的功能（编写myls程序） ,参数通过命令行传入：

1、获取当前工作目录路径并对该目录实现遍历。  
2、仿ls -l以列表形式出当前工作目录下的所有文件（包括子目录)  
需显示的文件属性有：  
文件类型 权限 硬链接数 所有者用户名 所有者所在组用户名 文件大小 最后修改时间。

用到的函数、结构体：  
getcwd strncmp opendir closedir stat

宏：

S\_ISDIR(st\_mode) //是否是一个目录  
S\_ISCHR(st\_mode) //是否是一个字符设备  
S\_ISBLK(st\_mode) //是否是一个块设备  
S\_IRUSR           //用户读权限  
S\_IWUSR //用户写权限  
S\_IRGRP  //用户组读权限  
S\_IWGRP //用户组写权限  
S\_IROTH //其他组都权限  
S\_IWOTH //其他组写权限

(1) 首先使用opendir()函数打开目录， 返回指向该目录的DIR结构体。

(2)接着，调用readdir()函数读取这个目录下所有文件，其中应该包括目录本

身，返回指向该目录下所有文件的dirent 结构体。

(3)最后，遍历dirent结构体，调用stat来获取每个文件的详细信息并存储在

stat结构体中。

主要结构体：

1.DIR结构体类似于FILE,是一个部结构，以下几个函数用这个部结构保存当

前正在被读取的目录的有关信息。函数DIR \*opendir(const char \*pathname)，即

打开文件目录，返回的就是指向DIR结构体的指针。

2.对于dirent结构体，首先我们要弄清楚目录文件( directory file)的概念。这种

文件包含了其他文件的名字以及指向与这些文件有关的信息的指针。从定义能够

看出，dirent 不仅仅指向目录，还指向目录中的具体文件，readdir 函数同样也读

取目录下的文件。

3.如果想获得类似ls -l那种效果的文件信息，必须要靠stat 函数了。通过readdir

函数读取到的文件名存储在结构体dirent的d\_name成员中，而函数

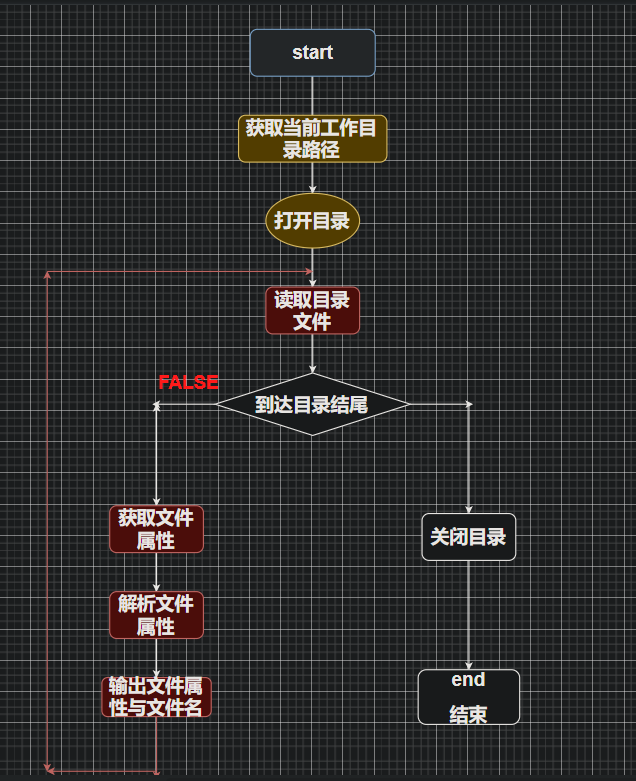
int stat(const char \*file\_ name, struct stat \*buf);的作用就是获取文件名为d\_ name的文件的详细信息，存储在stat 结构体中。

# 流程图

Ls流程图：



LS -l流程图：



# 源代码：程序一定要加上批注 /\* … \*/

Myls.cpp:

/\*操作系统实验课OS\_04

物联网1902班

学号：201916070216

姓名：王源

\*/

#include <iostream>

#include <ctime>

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <pwd.h>

#include <grp.h>

#include <dirent.h>

#include <sys/stat.h>

#include <string.h>

#include <unistd.h>

using namespace std;

char filename[100][255]; //存储文件名字符串

int filenum = 0;

// 将文件权限描述id转换为读写权限字符

void stmode\_to\_rwx(int mode, char str[]) {

strcpy(str, "----------");

if (S\_ISDIR(mode)) str[0] = 'd';

if (S\_ISCHR(mode)) str[0] = 'c';

if (S\_ISBLK(mode)) str[0] = 'b';

if ((mode & S\_IRUSR)) str[1] = 'r';

if ((mode & S\_IWUSR)) str[2] = 'w';

if ((mode & S\_IXUSR)) str[3] = 'x';

if ((mode & S\_IRGRP)) str[4] = 'r';

if ((mode & S\_IWGRP)) str[5] = 'w';

if ((mode & S\_IXGRP)) str[6] = 'x';

if ((mode & S\_IROTH)) str[7] = 'r';

if ((mode & S\_IWOTH)) str[8] = 'w';

if ((mode & S\_IXOTH)) str[9] = 'x';

}

int ls\_dir(char\* dirpath) {

DIR\* d; //DIR \*opendir(const char \*pathname)，目录的返回结构体指针

struct dirent\* dirfile; //用该结构体保存目录项

struct stat dirfileinfo; //目录项描述结构体

if (!(d = opendir(dirpath))) {

printf("error opendir %s\n", dirpath);

return -1;

}

while ((dirfile = readdir(d)) != NULL) {

// 过滤.xx目录项

if (strncmp(dirfile->d\_name, ".", 1) == 0)

continue;

stat(dirfile->d\_name, &dirfileinfo);

cout << dirfile->d\_name << " ";

}

cout << endl;

closedir(d);

return 1;

}

int ls\_l\_dir(char\* dirpath) {

DIR\* d; //DIR \*opendir(const char \*pathname)，目录的返回结构体指针

struct dirent\* dirfile; //用该结构体保存目录项

struct stat dirfileinfo; //目录项描述结构体

if (!(d = opendir(dirpath))) {

printf("error opendir %s\n", dirpath);

return -1;

}

struct passwd\* userinfo;

struct group\* groupinfo;

while ((dirfile = readdir(d)) != NULL) {

// 过滤.xx目录项

if (strncmp(dirfile->d\_name, ".", 1) == 0)

continue;

strcpy(filename[filenum++], dirfile->d\_name); //存储目录项名称

stat(dirfile->d\_name, &dirfileinfo); //得到指定文件名的描述信息

userinfo = getpwuid(dirfileinfo.st\_uid);

groupinfo = getgrgid(dirfileinfo.st\_gid);

char dirauth[11];

stmode\_to\_rwx(dirfileinfo.st\_mode, dirauth);

cout << dirauth << " " << dirfileinfo.st\_nlink << " " << userinfo->pw\_name << " " << groupinfo->gr\_name << " " << dirfileinfo.st\_size << " ";

printf(" %.12s", 4 + ctime(&dirfileinfo.st\_mtime));

cout << " " << dirfile->d\_name;

cout << endl;

}

closedir(d);

return 1;

}

int main(int args, char\* argv[]) {

char pwddir[255]; //存储当前工作环境下绝对路径

getcwd(pwddir, 255); //得到当前工作环境绝对路径

if (args > 1) {

//printf("your command : ./ %s,%s",argv[0],argv[1]);

int i = 2;

if (memcmp(argv[1], "-l", i) == 0) {

ls\_l\_dir(pwddir); //遍历当前工作环境路径下

}

else {

printf("Usage:%s -l \n", argv[0]);

}

}

else {

ls\_dir(pwddir);

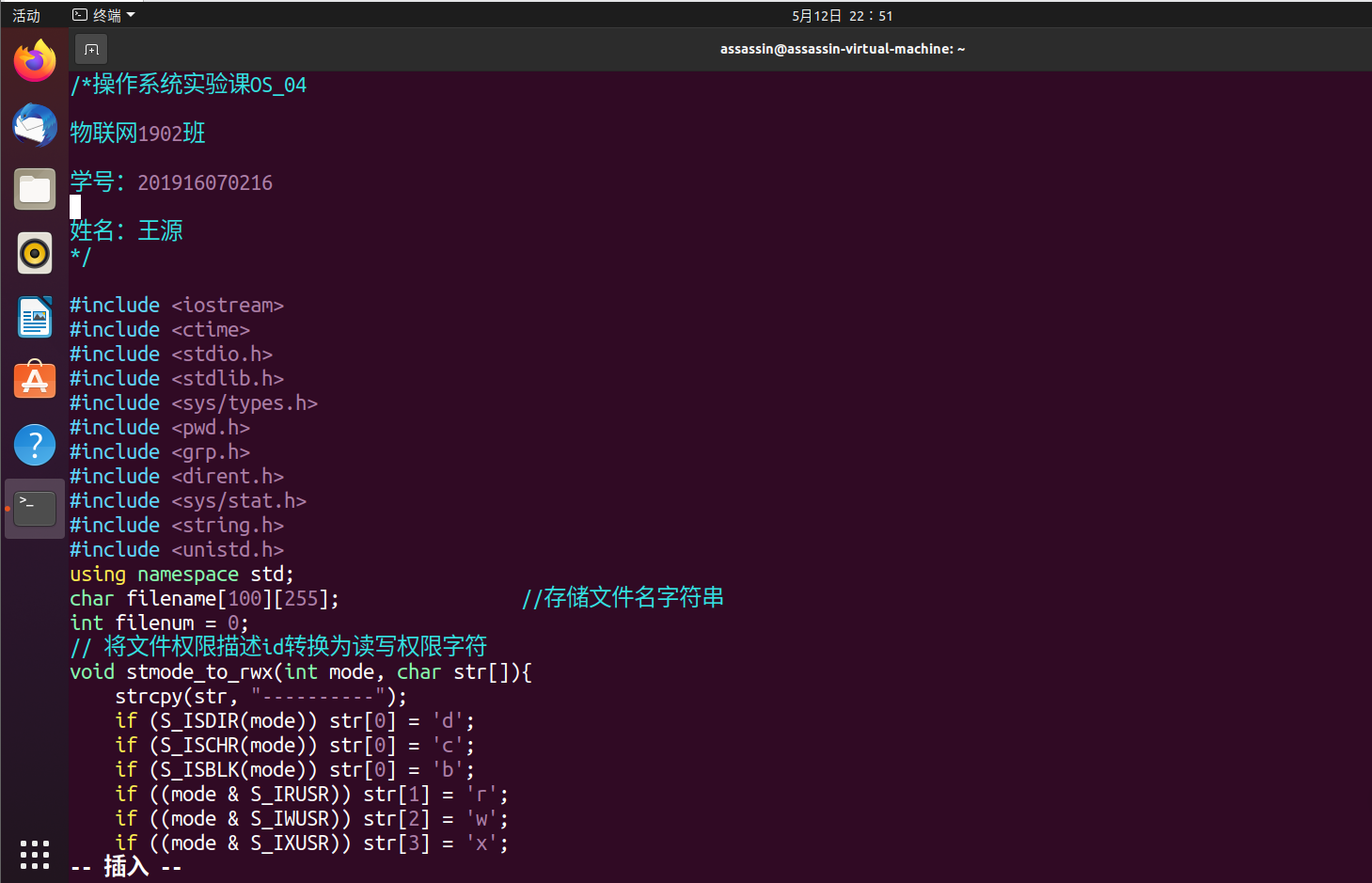
}

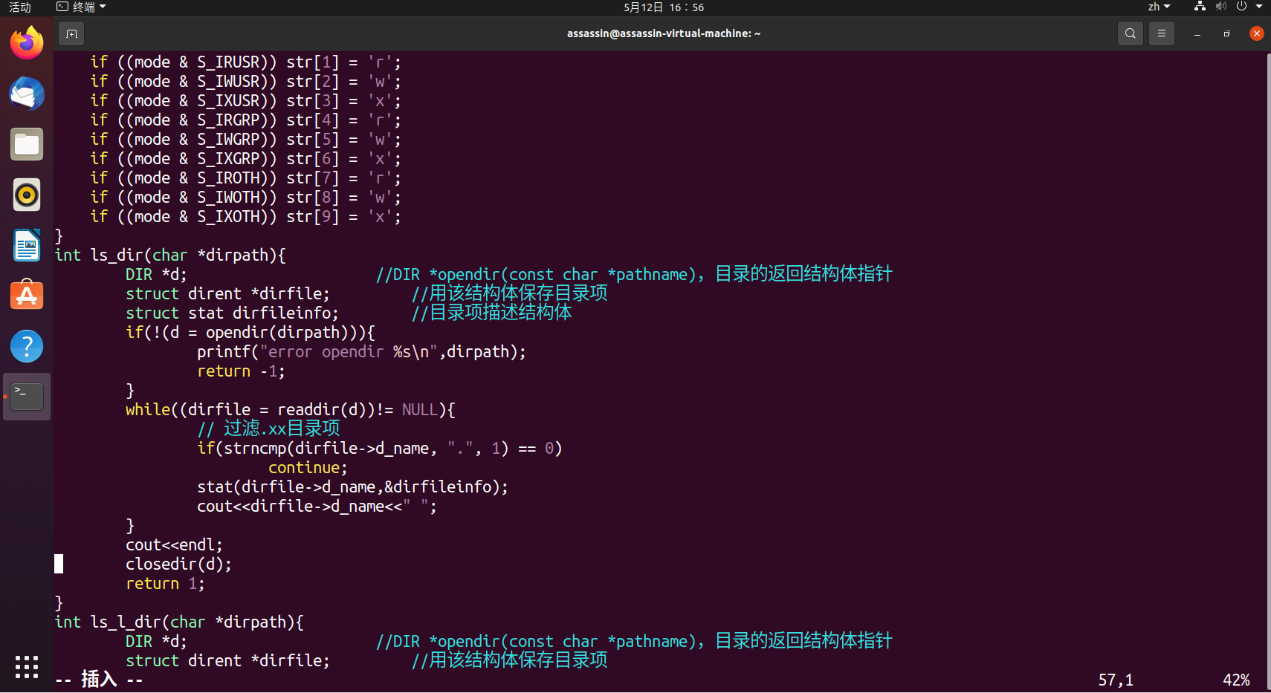
return 0;

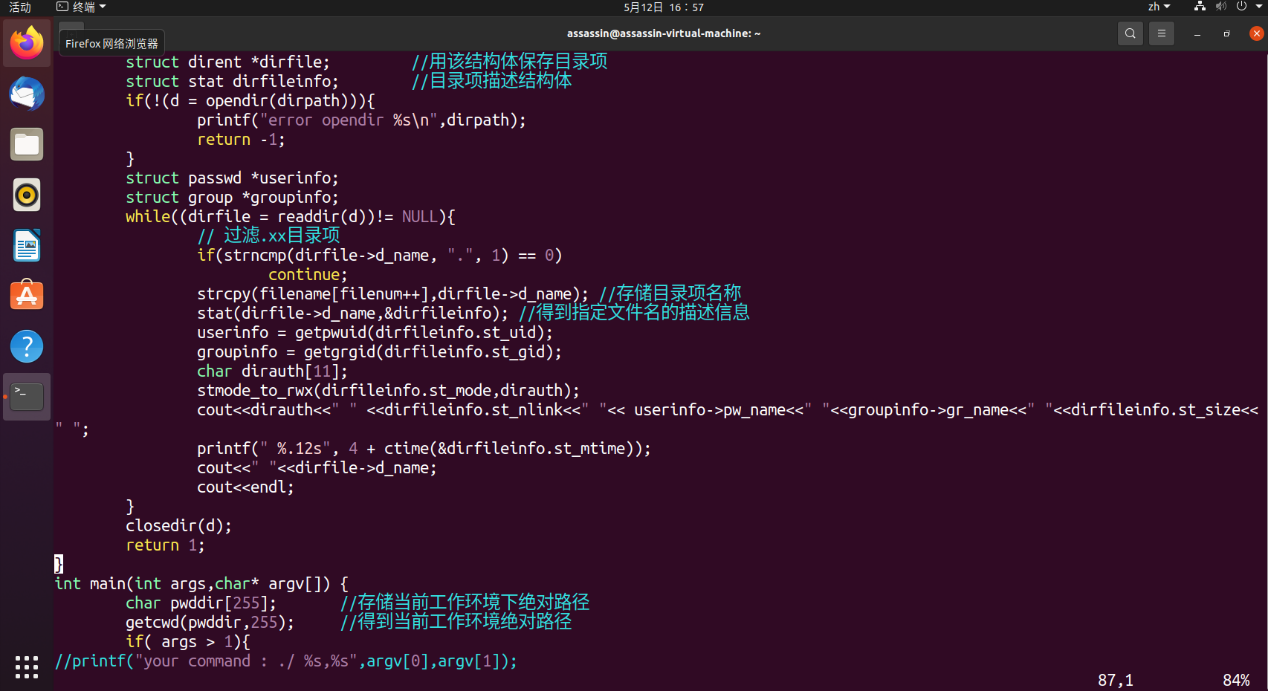
}

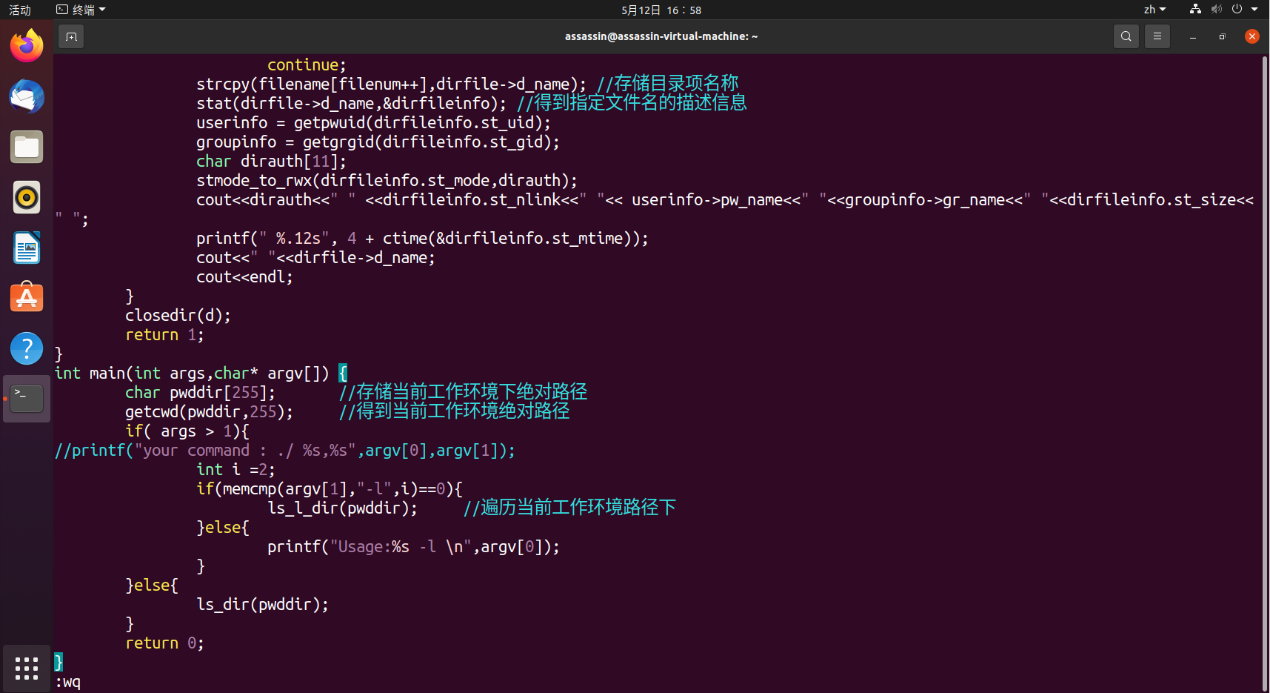
# 实验结果与说明：包含程序运行结果截图，结果要先显示出你自己的班级、学号、姓名。

**Linux g++下用vim编辑器实现代码：**

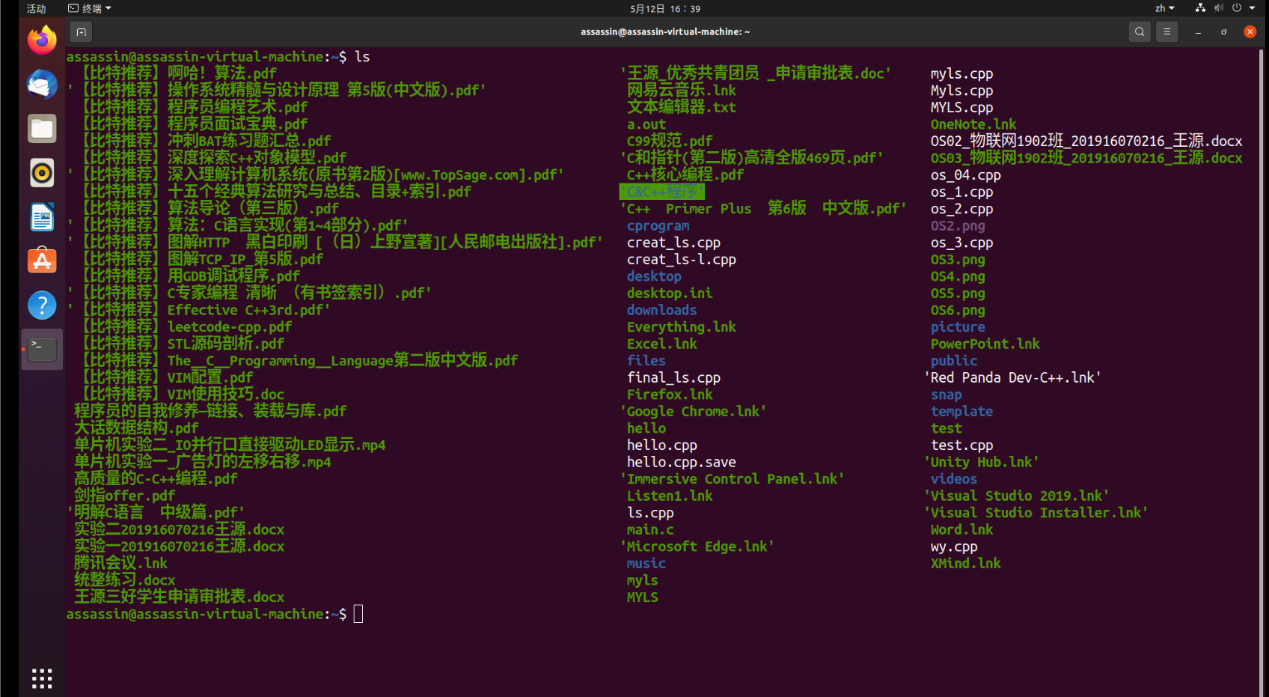




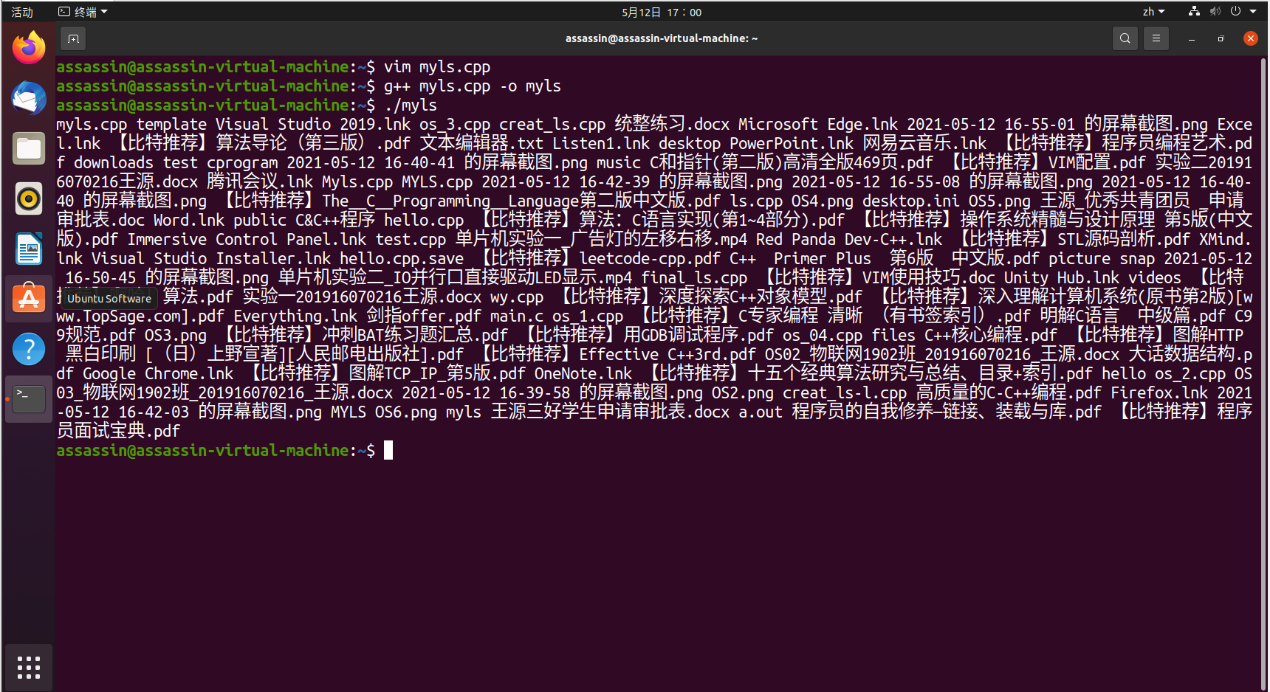




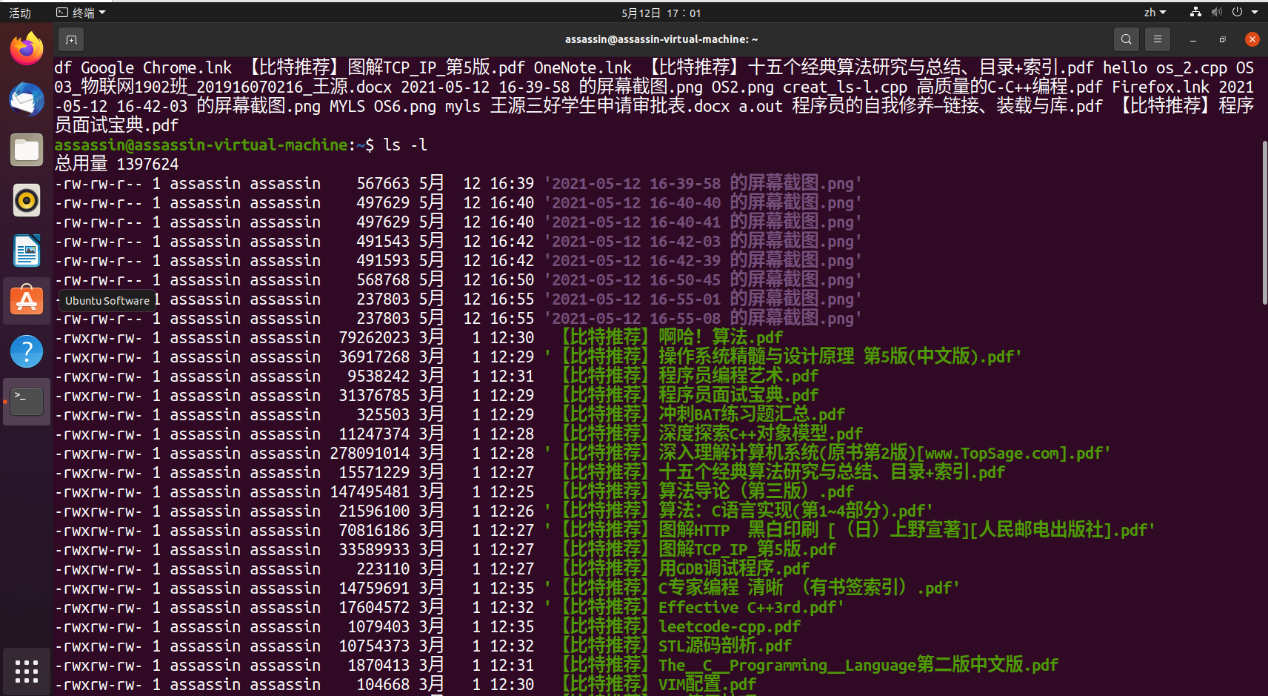
**Linux shell自带的ls：**

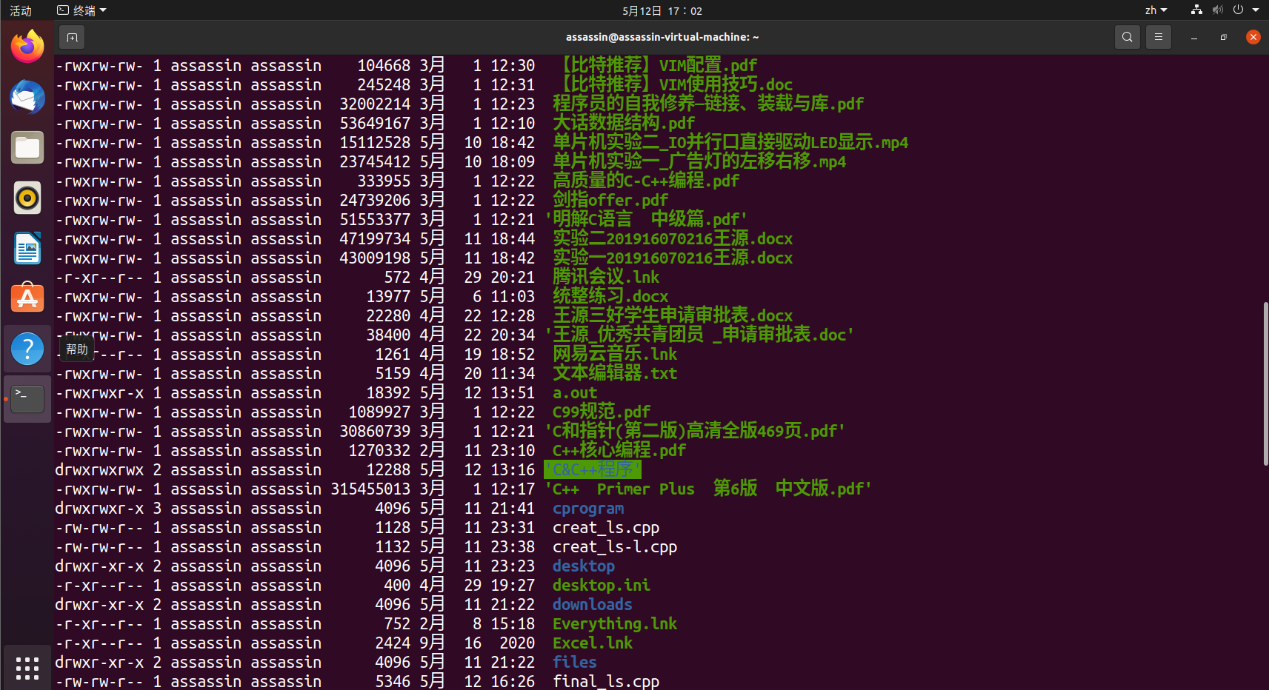


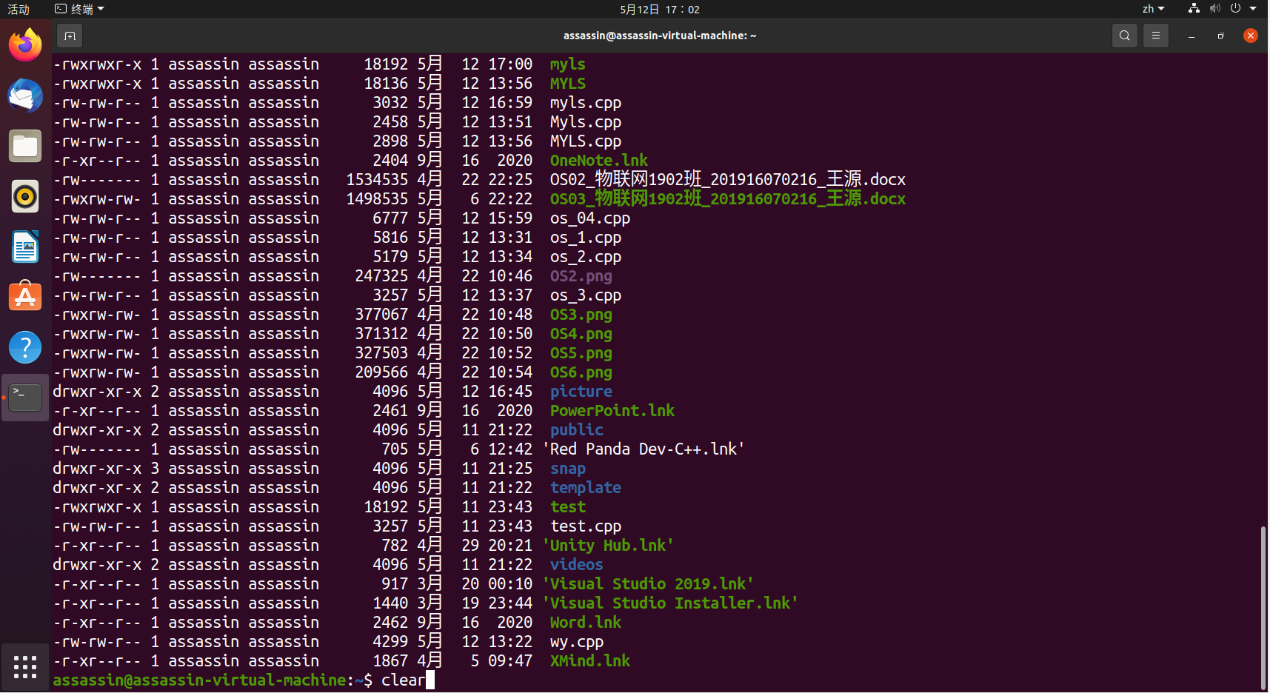
**Linux下自己实现的myls：**

****

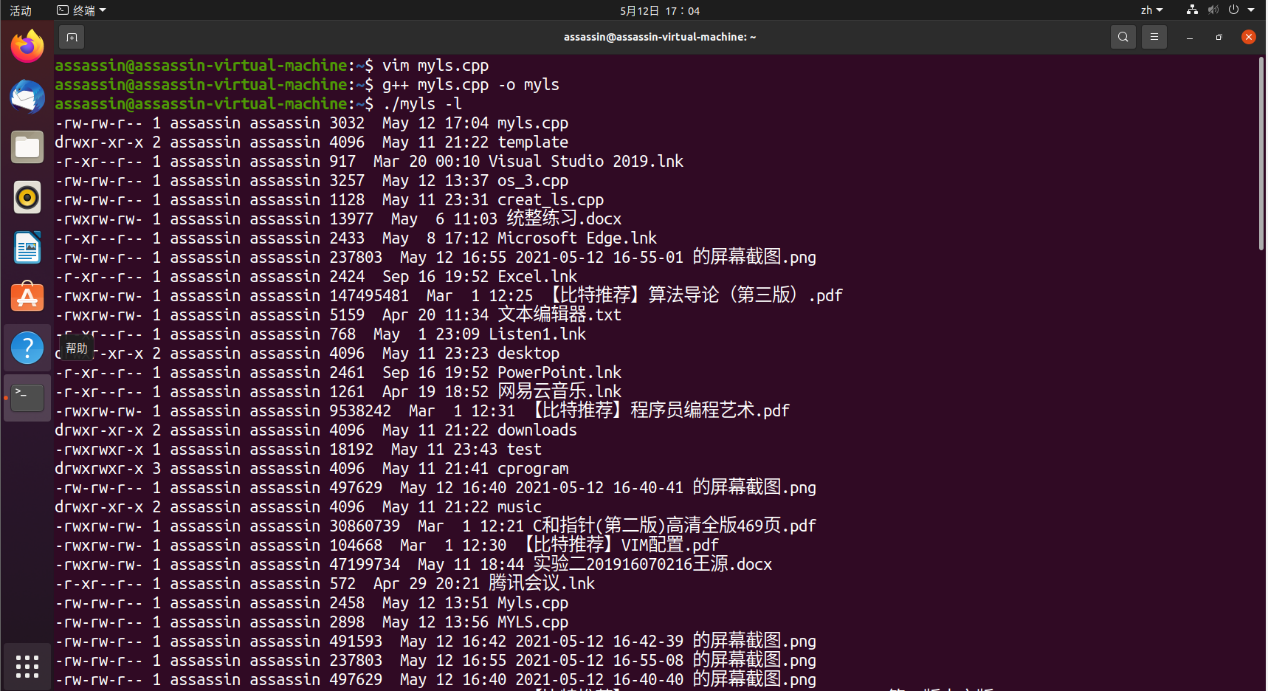
**Linux shell自带的ls -l：**

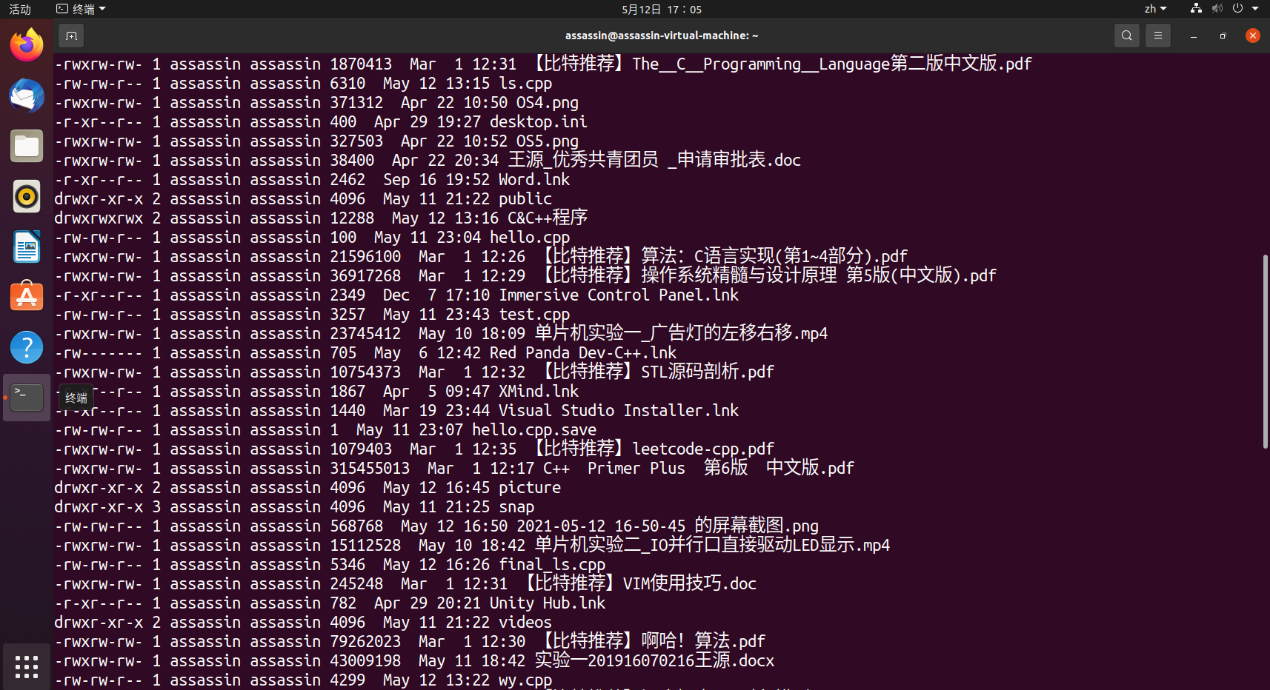
****

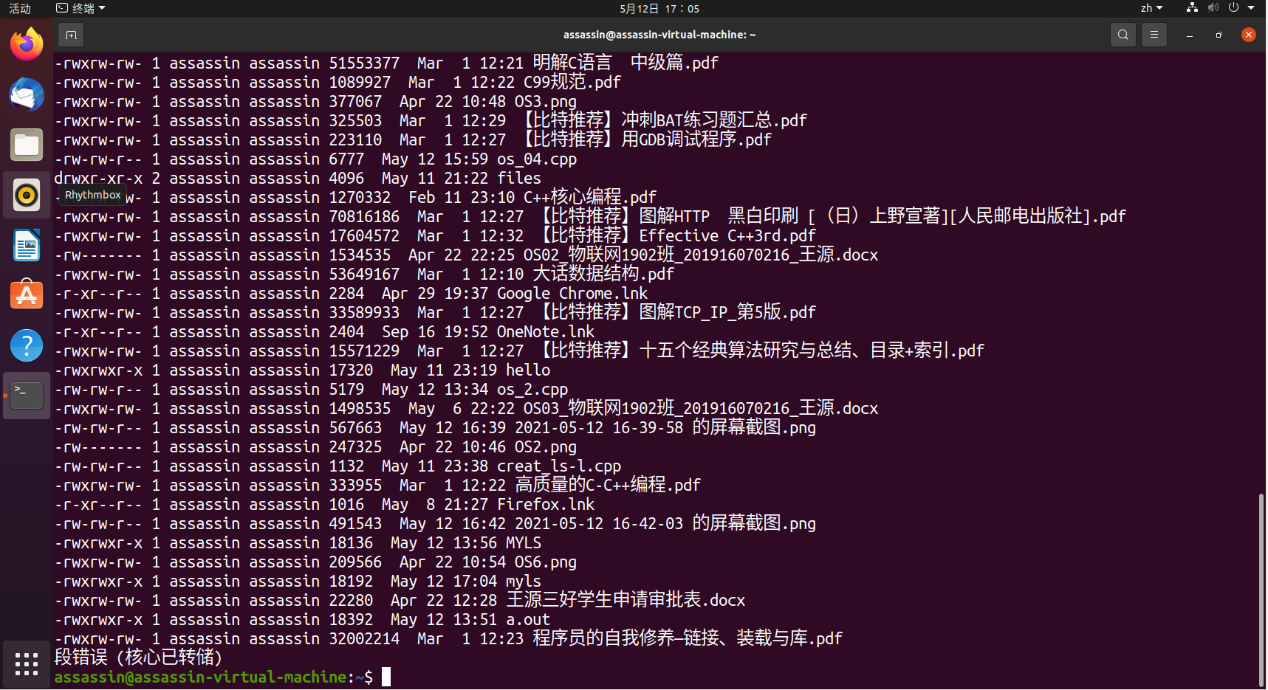




**Linux下自己实现的myls -l：**







# 我所遭遇的问题

1. 首先一个很重要的问题就是，linux下C/C++编译环境的搭建。我对linux操作系统不是很熟悉，平常编写代码都是在windows上的集成编译环境下（如VS2019）完成，在ubuntu下当然就要在gcc/g++/gdb上编译。由于linux上的操作基本都是通过命令行来完成的，所以搭建环境花了我很大的功夫。
2. Vim编辑器的使用也很令人头痛，vim有三种模式，一般指令模式，编辑模式和命令行模式。首先默认进入的是一般指令模式，不能对代码进行编辑，一度令我很苦恼，后来查阅资料才知道可以enter ‘i’进入编辑模式。若想退出编辑模式enter “esc”,通过shift+‘：’保存并退出vim。
3. 实现ls-l时需要用到许多linux下的头文件和结构体，而之前对此毫无接触，所以刚开始上手觉得手足无措，在查阅了相关资料后慢慢摸索出其中的功能所在。
4. 自己实现时对文件系统不是很熟悉，对linux文件目录权限也是一知半解，总的来说，对linux文件管理不熟练导致linux用的不是很顺畅，运用命令行下载软件跟创建文件删除文件也是很难受。

# 收获、体会和建议：包含撰写程序的心得

我不是经常使用linux操作系统，平时编写代码的环境都是windows，由于形式需要转而在ubuntu上完成实验。首先碰到的难题就是linux命令掌握的不是很熟悉，所以导致在编译代码时环境搭建地很困难，听说公司中程序员大多使用linux操作系统，所以早些熟悉为好。

通过编写在linux环境下实现ls与ls-l,让我再一次明白在写程序之前把思路理清,需求列好有多重要。程序框架一定要一次性建好,并且要完整地分析整个程序的运行过程,不能值分析一半或只分析一部分。  
 需求也要一次性列好,不能编写边加,否则新的需求可能会冲击之前建好的程序框架,必要时就需要重构。  
 最后,感受最深的可能就是,代码一定要git，rm时一定要保持清醒.但是,毕竟每个程序员都会遇到这样的情况,千金难买经验和教训,以后做事情就要更加谨慎,也要养成代码git的好习惯。

从以前仅仅是单纯地使用系统命令，到现在可以自已通过编程来实现一些命令，能够感觉到些许的成就感。但同时，也发现了使用系统提供的命令和调用自己编写的命令感觉是相差很大的。在这编程和使用的过程中，尽管出现了很多困难，软件不熟悉，代码不断报错出现bug,甚至是开发环境不知道如何配置，但是通过查阅资料都顺利解决了这些问题。

**九．附件**

****

# 主要参考文献

1. [csdn社区 [(6条消息) Linux 简单实现 ls -l 命令\_如故的博客-CSDN博客](https://blog.csdn.net/sinat_36629696/article/details/79456589)](https://blog.csdn.net/r8l8q8/article/details/78374024)
2. 知乎 [自己实现一个简单的 ls 命令 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/64608414)

3.汤小丹，梁红兵等 《计算机操作系统》(文件管理)

**截止日期：2021.05.12 Thr. 23:00**

**档案命名方式：OS04\_班级学号姓名**