

# Linux 操作系统综合实验

## 实验报告

---



---

日期 2025-5-5

---

# 目录

一.	实验目的 .....	3
二.	实验要求 .....	3
三.	题目分析及基本设计过程分析 .....	3
1.	题目分析 .....	3
2.	设计决策 .....	4
3.	实现过程 .....	4
四.	配置文件关键修改处的说明及运行情况 .....	4
1.	Samba 安装和配置 .....	5
2.	文件上传功能 .....	6
3.	测试功能 .....	7
4.	自动生成文档 .....	9
5.	权限控制 .....	11
五.	脚本源程序清单 .....	12
1.	自动生成代码记录脚本 auto-code-test.sh .....	12
2.	自动从目录删除文件的脚本 auto-delete.sh .....	14
3.	自动生成测试记录文档脚本 auto-test-record.sh .....	16
六.	实验过程中出现的问题及解决方法 .....	18
七.	实验体会 .....	18
八.	分工情况 .....	18

## 一. 实验目的

熟练掌握 Linux 操作系统的使用，掌握 Linux 的各项系统管理功能，掌握 Linux 下各类网络服务的安装、配置以及使用，并能用 shell 脚本实现简单的管理任务。

## 二. 实验要求

假设在一个小型软件公司中，有一个 leader、若干开发人员和测试人员。根据所学内容，使用一种或多种服务（如 ftp、samba、Http 等）搭建服务器，完成下述功能：

1. 开发人员能实现所编代码的上传，以“开发人员姓名+功能”命名代码。开发人员能看到所有代码列表，但是不能下载其他用户的代码。
2. 在指定的代码提交时间到时，运行脚本自动统计上传代码的开发人员名字、人数和提交时间，生成文档，供测试人员和 leader 查看。
3. 测试人员能够查看提交的代码，但无权修改代码，可选择一个代码进行测试。被选中测试的代码从当前目录清除，放入一个指定的新目录中。测试后生成相应的测试报告，以“开发人员姓名+功能+测试人员姓名”命名，并发布在某个指定的目录中，供开发人员查看。
4. 根据测试报告，生成测试记录文件，记录哪个测试人员测试了哪个开发人员的哪个代码以及处理时间。

## 三. 题目分析及基本设计过程分析

### 1. 题目分析

通过对实验要求的分析理解，本组认为本次实验主要任务包括文件共享、文件管理、脚本自动生成、权限管理，具体功能需求如下

**代码上传：**开发人员可以以固定的命名上传代码

**代码工作统计：**系统可自动定时生成关于代码上传情况的文档

**代码测试：**被测代码需要转移到新的目录，测试完成后测试人员可以固定命名上传测试报告

**测试记录生成：**测试完成后系统可自动生成测试记录文件

**权限控制：**系统为不同角色赋予不同权限。其中开发人员可以任意操作自己编写

的代码，可以查看代码列表，可以查看测试报告；测试人员可以查看和运行提交的代码，可以查看代码工作统计，可以任意操作自己编写的测试报告；**leader** 可以查看查看所有文件且拥有以上两种角色的全部权限。

## 2. 设计决策

基于以上分析得出的题目要求，本组决定使用 **samba** 服务以完成文件共享、文件管理，使用 **shell** 脚本完成自动生成功能，使用 **Linux** 系统文件管理功能实现权限分配。

## 3. 实现过程

为实现上述功能需求，本组进行了以下操作：

- 1) . 下载和配置 **samba** 服务器
- 2) . 添加用户和用户组
- 3) . 创建文件夹并设置权限
- 4) . 编辑自动统计脚本
- 5) . 编辑测试相关脚本
- 6) . 设置脚本计划作业
- 7) . 测试功能是否正常

关于具体功能，通过以下方法进行实现

**代码上传：**通过 **samba** 服务的文件共享功能实现

**代码工作统计：**通过编写脚本 **auto-code-record.sh** 实现

**代码测试：**通过编写脚本 **auto-delete.sh** 实现测试代码从原目录删除的功能

**测试记录生成：**通过编写脚本 **auto-test-record.sh** 实现

以上脚本具体信息将在报告第五部分进行展示

**权限控制：**通过将用户分配到不同组实现角色的分配，为每个组分配相应权限以实现权限控制。在 **samba** 服务中，用户上传的文件默认权限为 **704**，使得开发人员上传的代码只能供自己查看，而其他开发人员不能查看，但测试人员可以查看。

具体权限分配将在报告第四部分进行展示。

## 四. 配置文件关键修改处的说明及运行情况

## 4. 准备阶段

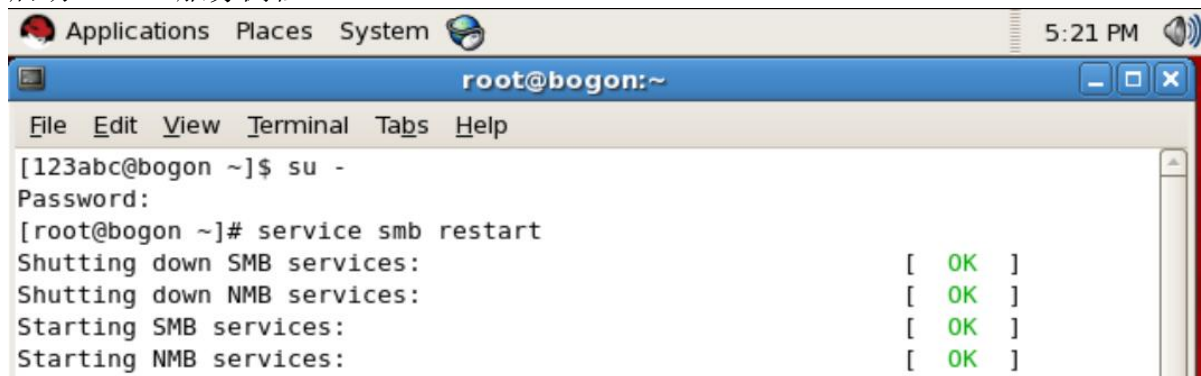
创建必要的用户及组，将创建的用户归入相应的组中：

```
[root@bogon homework]# groupadd developers
[root@bogon homework]# groupadd testers
[root@bogon homework]# useradd developer1 -g developers -m -s /bin/bash
[root@bogon homework]# useradd developer2 -g developers -m -s /bin/bash
[root@bogon homework]# useradd tester -g testers -m -s /bin/bash
[root@bogon homework]# useradd leader -m -s /bin/bash
[root@bogon homework]# usermod -aG developers leader
[root@bogon homework]# usermod -aG testers leader
```

这里创建了开发者用户：**developer1**、**developer2**，将其归入 **developers** 组中；创建了测试者用户：**tester**，将其归入 **testers** 组中；创建了 **leader** 用户，并将其同时放入 **developers** 组和 **testers** 组，保证 **leader** 同时具有两个组的权限。

## 5. Samba 安装和配置

启动 **samba** 服务例程：

A screenshot of a terminal window titled 'root@bogon:~'. The window shows the execution of 'service smb restart'. The output indicates that SMB and NMB services were successfully restarted, with 'OK' status for each step.

```
Applications Places System 5:21 PM
root@bogon:~
File Edit View Terminal Tabs Help
[123abc@bogon ~]$ su -
Password:
[root@bogon ~]# service smb restart
Shutting down SMB services: [ OK ]
Shutting down NMB services: [ OK ]
Starting SMB services: [ OK ]
Starting NMB services: [ OK ]
```

对 **smb.conf** 文件的配置如下：

```
[shared]
comment = Code Sharing
path = /usr/homework/
browseable = yes
read only = no
valid users = @developers, @testers, leader
create mask = 0704
directory mask = 0777
```

**/usr/homework** 是本服务的主目录。在 **samba** 配置中，将这一目录共享给用户组 **developers** 和 **testers**，以及领导者 **leader**。在服务主目录中有以下几个文件夹：

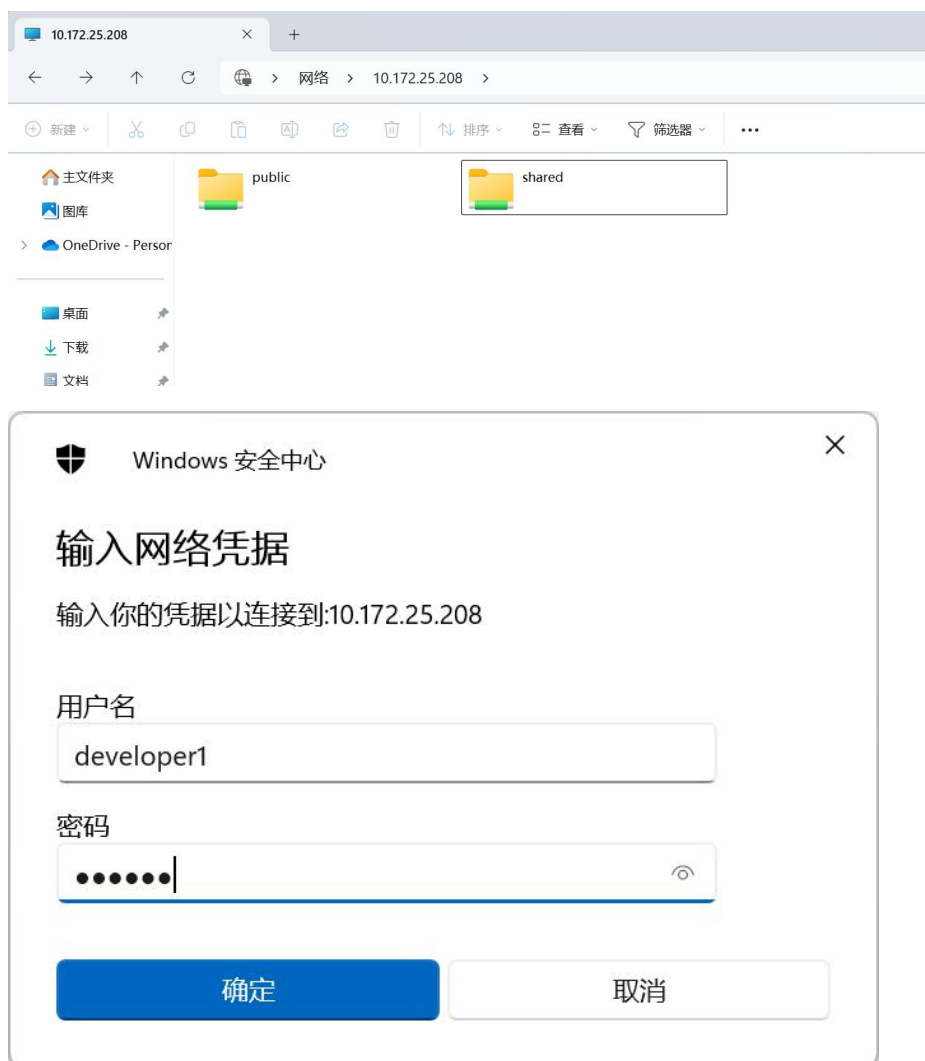
- /code** : 开发人员将代码上传至此文件夹。
- /codetest** : 测试人员测试代码使用的目录。
- /testreport**: 测试人员完成测试后，在此上传测试报告。
- /record**: 系统在此自动生成开发/测试记录。
- /bin**: 系统的核心脚本。

```
[root@bogon homework]# ls
bin  code  codetest  record  testreport
```

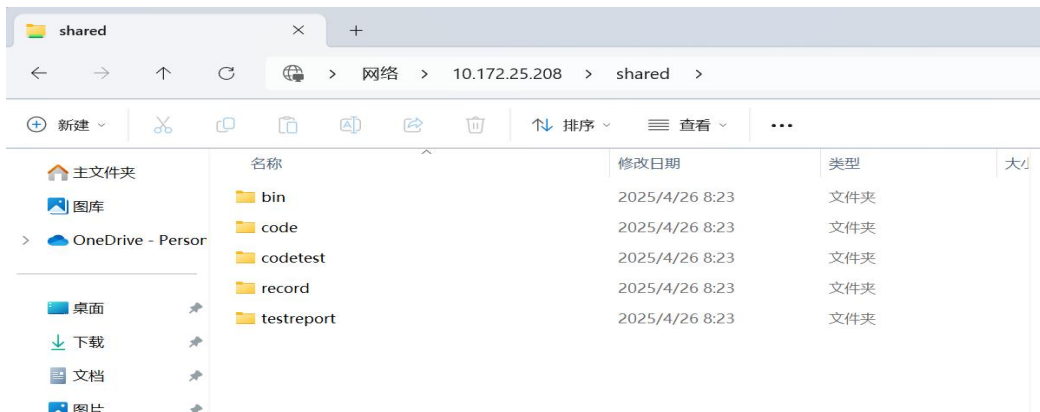
## 6. 文件上传功能

配置好 samba 服务后，可以在 Windows 上通过 \\IP 地址或者在 Linux 中使用 smbclient 连接到 samba 服务。使用开发人员账户登录到 samba 服务器后，可以直接向 code 目录中上传代码。代码应以标准格式命名。

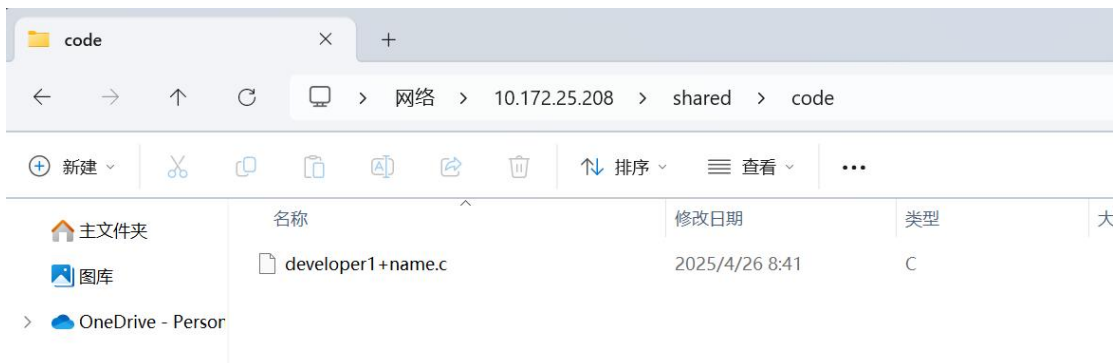
输入正确的用户名和密码访问 shared 目录：



成功访问 shared 目录后可以看到我们为实现一系列功能创建的目录：



在用户 **developer1** 的权限下可以访问 **code** 目录并按命名要求添加代码文件：



**developer1** 无权访问 **codetest** 目录：



## 7. 测试功能

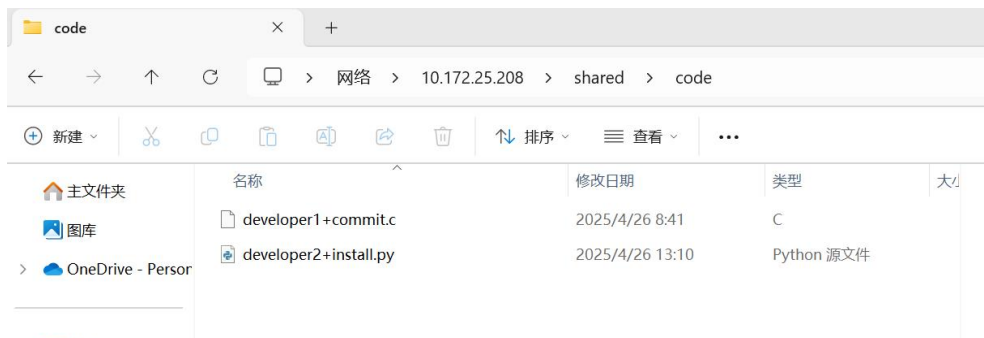
测试人员登录到 **samba** 服务器后，可以将开发人员上传至 **/code** 的代码复制到 **/codetest**，以进行代码测试。成功复制后，系统的定时任务脚本将自动删除 **/code** 中的同名文件。

测试前：

**tester** 能够进入 **codetest** 目录，该目录当前为空：



当前 `code` 目录下有两个文件：



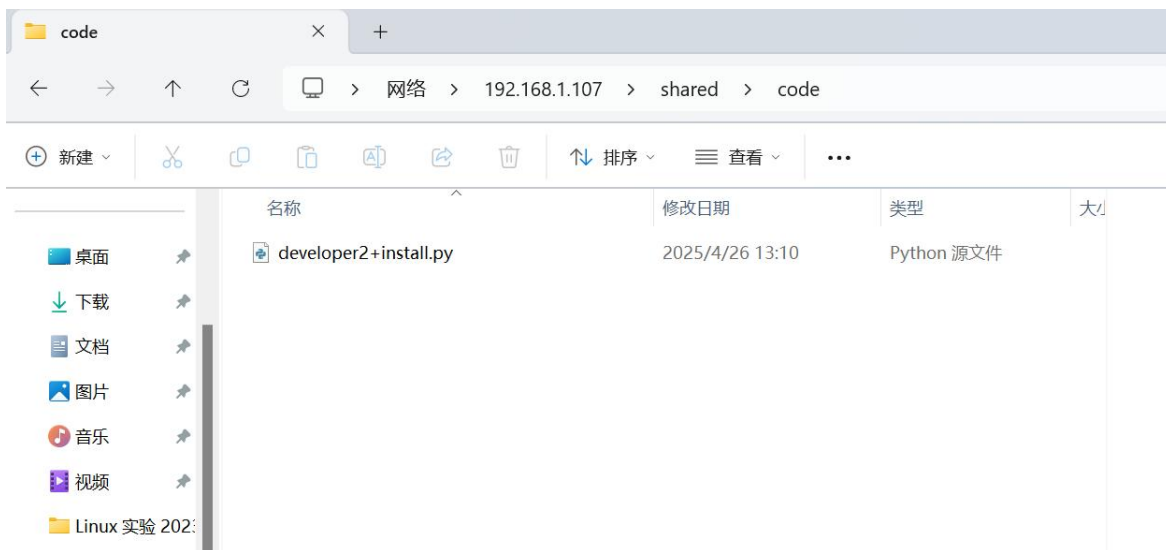
开始测试：

将 `developer1+commit.c` 复制到 `codetest` 目录下：

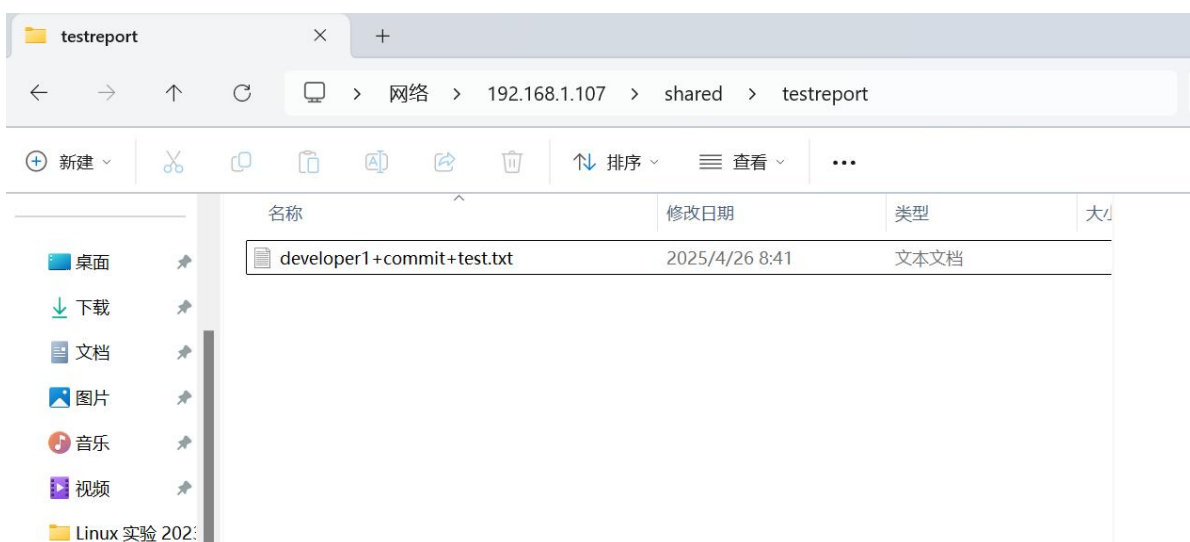


可以看到 `code` 目录下的 `developer1+commit.c` 文件被删除





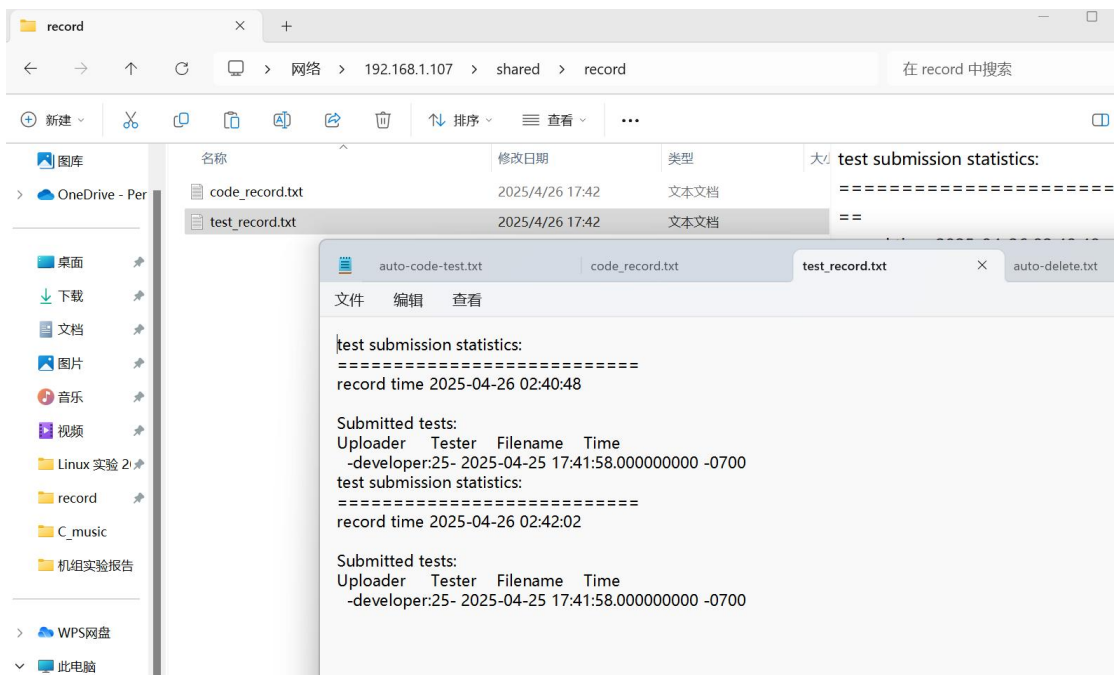
完成测试后，测试人员应上传以标准格式命名的测试报告：



## 8. 自动生成文档

代码上传：





记录的生成通过 `crontab -e` 设置定时任务实现，定时任务配置如下：

```

c[root@bogon ~]# crontab -l
* * * * * bash /usr/homework/bin/auto-test-record.sh
* * * * * bash /usr/homework/bin/auto-delete.sh
* * * * * bash /usr/homework/bin/auto-code-test.sh

```

## 9. 权限控制

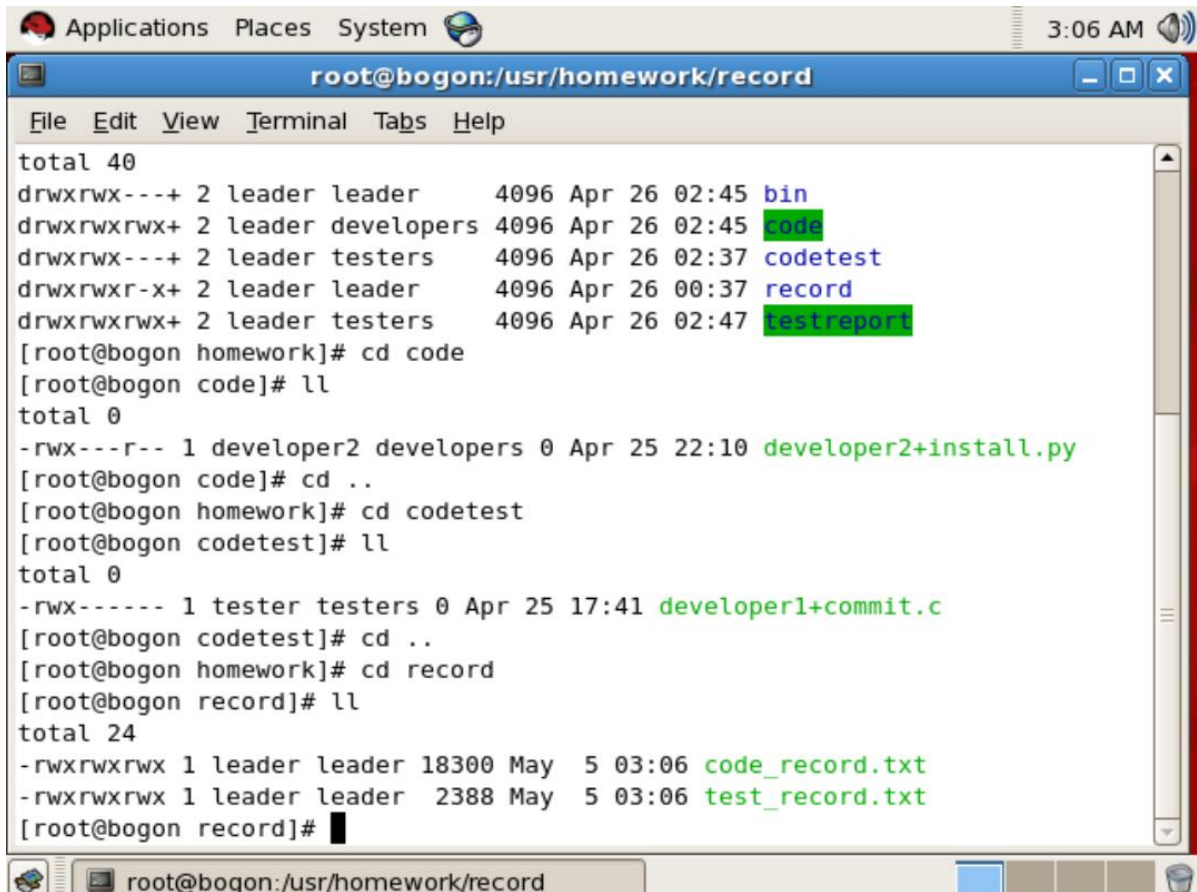
在本系统中，文件夹和文件均设置了权限以符合相关需求，如图所示。说明如下：  
所有文件夹的所有者均为 **leader**，保证了领导者对本系统的最高权限。

**/code**：该文件夹可供所有人查看，但仅供开发者修改。

**/codetest**：该文件夹只对测试人员开放。

**/testreport**：该文件夹可供所有人查看，但仅供测试人员修改。

**/record** 和 **/bin**：可供所有人查看，但仅供系统和领导者修改。



```
Applications Places System 3:06 AM
root@bogon:/usr/homework/record
File Edit View Terminal Tabs Help
total 40
drwxrwx---+ 2 leader leader 4096 Apr 26 02:45 bin
drwxrwxrwx+ 2 leader developers 4096 Apr 26 02:45 code
drwxrwx---+ 2 leader testers 4096 Apr 26 02:37 codetest
drwxrwxr-x+ 2 leader leader 4096 Apr 26 00:37 record
drwxrwxrwx+ 2 leader testers 4096 Apr 26 02:47 testreport
[root@bogon homework]# cd code
[root@bogon code]# ll
total 0
-rwx---r-- 1 developer2 developers 0 Apr 25 22:10 developer2+install.py
[root@bogon code]# cd ..
[root@bogon homework]# cd codetest
[root@bogon codetest]# ll
total 0
-rwx----- 1 tester testers 0 Apr 25 17:41 developer1+commit.c
[root@bogon codetest]# cd ..
[root@bogon homework]# cd record
[root@bogon record]# ll
total 24
-rwxrwxrwx 1 leader leader 18300 May 5 03:06 code_record.txt
-rwxrwxrwx 1 leader leader 2388 May 5 03:06 test_record.txt
[root@bogon record]#
```

对于开发人员上传的文件，在 `samba` 配置中已将默认权限配置为 `704`，因此开发人员 `developer1` 上传的文件属于 `developer1:developers`，其他开发者只能看到文件名但不能访问文件内容，测试人员可以查看文件内容但不能修改。同理，测试人员上传的测试报告可供开发人员查看。

登录到用户 `developer2` 时：

### 记事本

Z:\code\developer1+name.c

你没有权限打开该文件，请向文件的所有者或管理员申请权限。

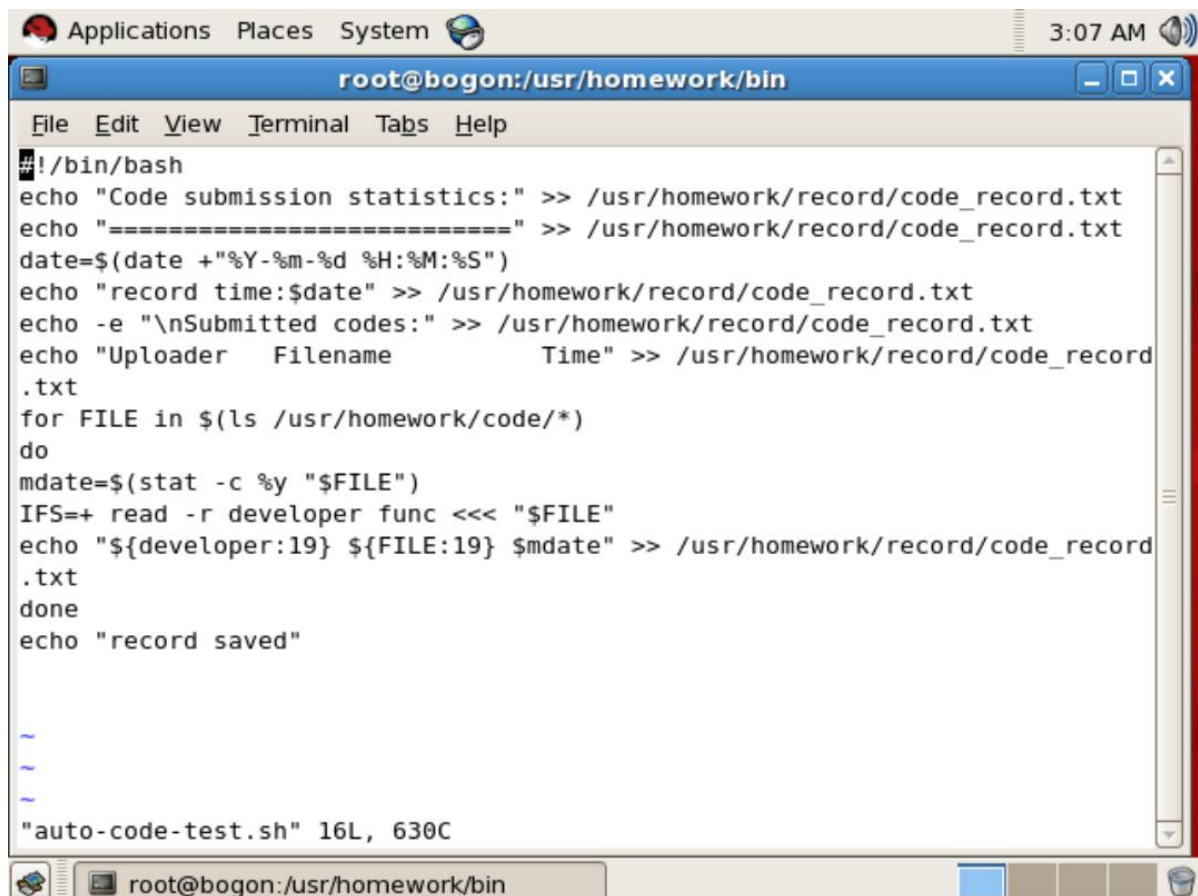
确定

满足题目的权限需求。

## 五. 脚本源程序清单

### 1. 自动生成代码记录脚本 `auto-code-test.sh`

截图如下



The screenshot shows a terminal window titled 'root@bogon:/usr/homework/bin'. The window contains a shell script being executed. The script starts with a shebang line, followed by several echo statements and a loop that processes files in the directory /usr/homework/code/. The script ends with an echo statement indicating the record is saved. The terminal output shows the script's progress, including the current time and the list of files being processed.

```
#!/bin/bash
echo "Code submission statistics:" >> /usr/homework/record/code_record.txt
echo "======" >> /usr/homework/record/code_record.txt
date=$(date +"%Y-%m-%d %H:%M:%S")
echo "record time:$date" >> /usr/homework/record/code_record.txt
echo -e "\nSubmitted codes:" >> /usr/homework/record/code_record.txt
echo "Uploader    Filename          Time" >> /usr/homework/record/code_record
.txt
for FILE in $(ls /usr/homework/code/*)
do
mdate=$(stat -c %y "$FILE")
IFS+= read -r developer func <<< "$FILE"
echo "${developer:19} ${FILE:19} $mdate" >> /usr/homework/record/code_record
.txt
done
echo "record saved"
```

其具体注释如下，其中代码部分标红

`#!/bin/bash`

# 指定使用 Bash 解释器来执行该脚本

`echo "Code submission statistics:" >> /usr/homework/record/code_record.txt`

#将 "Code submission statistics:" 输出

到 /usr/homework/record/code\_record.txt 文件中，并追加到文件末尾

`echo "======" >> /usr/homework/record/code_record.txt`

# 将 "======" 输出

到 /usr/homework/record/code\_record.txt 文件中，并追加到文件末尾

`date=$(date +"%Y-%m-%d %H:%M:%S")`

# 获取当前时间，并将其格式化成 "年-月-日 时:分:秒" 的形式，赋值给变量 date

`echo "record time: $date" >> /usr/homework/record/code_record.txt`

```

# 将 "record time: " 和变量 date 的值输出
到 /usr/homework/record/code_record.txt 文件中，并追加到文件末尾

echo -e "\nSubmitted codes:" >> /usr/homework/record/code_record.txt

# 将 "\nSubmitted codes:" 输出到 /usr/homework/record/code_record.txt 文件中，并
追加到文件末尾。

echo "Uploader Filename Time">> /usr/homework/record/code_record.txt

#将 "Uploader Filename Time" 输出到 /usr/homework/record/code_record.txt 文件中，
并追加到文件末尾

for FILE in $(ls /usr/homework/code/*)
do

# 遍历 /usr/homework/code/ 目录下的所有文件

mdate=$(stat -c %y "$FILE")

# 获取文件的修改时间，并将其格式化成 "年-月-日 时:分:秒" 的形式，赋值给变量 mdate

IFS=+ read -r developer func <<< "$FILE"

# 将文件名按照 "+" 分隔成两部分，分别赋值给变量 developer 和 func

echo "${developer:19} ${FILE:19} $mdate" >> /usr/homework/record/code_recor
d.txt

#将变量 developer、FILE 和 mdate 的值输出
到 /usr/homework/record/code_record.txt 文件中，并追加到文件末尾

# "${developer:19}" 表示从字符串 developer 的第 19 个字符开始截取，
"${FILE:19}" 同理

done

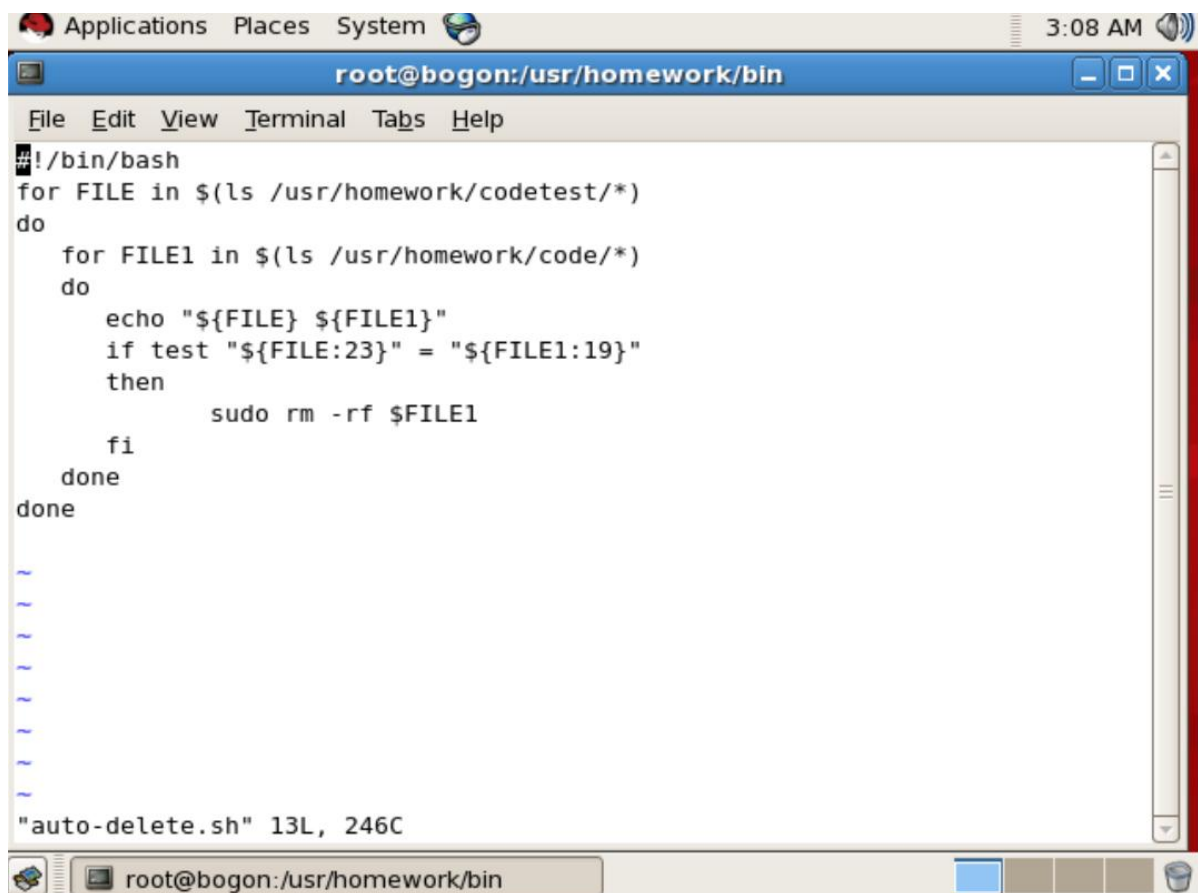
echo "record saved"

# 输出 "record saved" 字符串到终端，表示记录已保存

```

## 2. 自动从目录删除文件的脚本 auto-delete.sh

截图如下



具体注释如下

```
#!/bin/bash
```

# 指定使用 Bash 解释器来执行该脚本

```
for FILE in $(ls /usr/homework/codetest/*)
```

```
# 遍历 /usr/homework/codetest/ 目录下的所有文件
```

do

```
for FILE1 in $(ls /usr/homework/code/*)
```

```
# 遍历 /usr/homework/code/ 目录下的所有文件
```

do

```
echo "${FILE} ${FILE1}"
```

```
# 输出变量 FILE 和 FILE1 的值到终端
```

```
if test "${FILE:23}" = "${FILE1:19}"
```

### # 判断变量 FILE 和 FILE1 的文件名是否相同



then

```
sudo rm -rf $FILE1
```

# 如果文件名相同，则删除 /usr/homework/code/ 目录下的文件

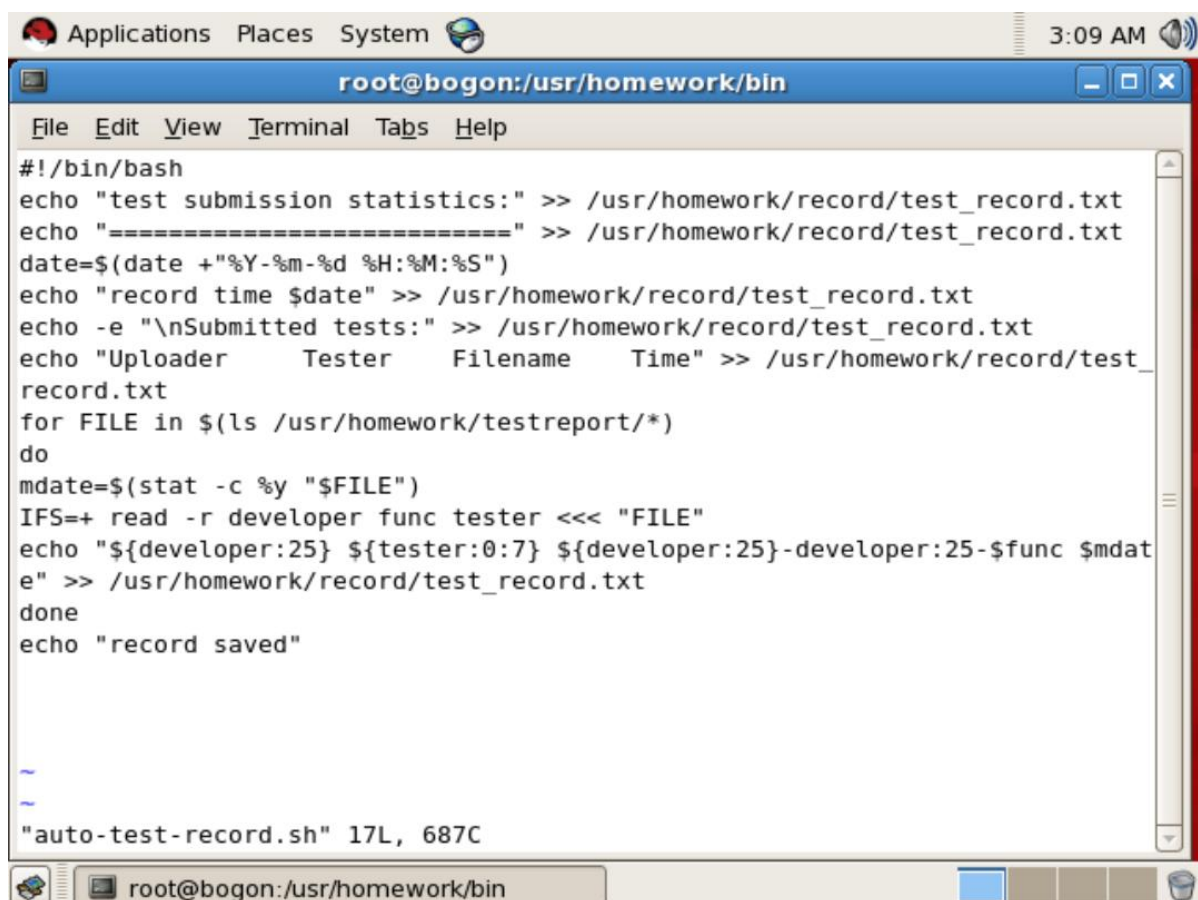
fi

done

done

### 3. 自动生成测试记录文档脚本 auto-test-record.sh

截图如下



具体注释信息如下

```
#!/bin/bash
```

指定使用 Bash 解释器来执行该脚本

```
echo "test submission statistics:" >> /usr/homework/record/test_record.txt
```

将 "test submission statistics:" 这个字符串输出



到 /usr/homework/record/test\_record.txt 文件中，并追加到文件末尾

```
echo "===== " >> /usr/homework/record/test_record.txt
```

将 "===== " 输出

到 /usr/homework/record/test\_record.txt 文件中，并追加到文件末尾

```
date=$(date +"%Y-%m-%d %H:%M:%S")
```

获取当前时间，并将其格式化成 "年-月-日 时:分:秒" 的形式，赋值给变量 date

```
echo "record time: $date" >> /usr/homework/record/test_record.txt
```

将 "record time: " 和 变量 date 的值 输出

到 /usr/homework/record/test\_record.txt 文件中，并追加到文件末尾

```
echo -e "\nSubmitted tests:" >> /usr/homework/record/test_record.txt
```

将 "\nSubmitted tests:" 输出到 /usr/homework/record/test\_record.txt 文件中，并追加到文件末尾。

```
echo "Uploader Tester Filename Time">> /usr/homework/record/test_record.txt
```

将 "Uploader Tester Filename Time" 输出

到 /usr/homework/record/test\_record.txt 文件中，并追加到文件末尾

```
for FILE in $(ls /usr/homework/testreport/*)
```

遍历 /usr/homework/testreport/ 目录下的所有文件

```
do
```

```
mdate=(stat -c %y "(stat-cFILE)")
```

# 获取文件的修改时间，并将其格式化成 "年-月-日 时:分:秒" 的形式，赋值给变量 mdate

```
IFS=+ read -r developer func tester <<< "FILE"
```

# 将文件名按照 "+" 分隔成三部分，分别赋值给变量 developer、func 和 tester

```
echo "${developer:25} ${tester:0:7} {developer:25}-developer:25-func mdate" >> /usr/homework/record/test_record.txt
```

# 将 变量 developer 、 tester 、 func 和 mdate 的值 输出到 /usr/homework/record/test\_record.txt 文件中，并追加到文件末尾

# "{developer:25}" 表示从字符串 developer 的第 25 个字符开始截取，

"\${tester:0:7}" 表示从字符串 `tester` 的第 0 个字符开始截取，截取 7 个字符

`done`

`echo "record saved"`

输出 `"record saved"` 字符串到终端，表示记录已保存

## 六. 实验过程中出现的问题及解决方法

实验过程中遇到的问题有：

由于 `samba` 配置时有误，导致任意用户登录到 `samba` 服务时，均不能正常访问共享文件，修改 `samba` 配置后解决。

在脚本文件时，一开始在 `windows` 下编写后，将文件直接放入共享文件夹后再 `linux` 下复制后脚本无法运行。检查后发现 `windows` 下文档的换行键与 `linux` 不同，无法识别导致编译错误，在 `linux` 环境下重新书写一遍后编译成功。

## 七. 实验体会

在本次实验中，本组顺利完成了实验任务，借助 `Samba` 服务和 `Shell` 脚本实现了文件共享、管理、自动生成以及权限控制等功能。在实验过程中，本组在 `Samba` 服务配置和脚本编写方面经历了多次尝试与调整，最终成功完成了任务。通过这一过程，本组成员对 `Samba` 配置文件的编写和 `Shell` 脚本的语法有了更深入的理解和更熟练的掌握，进一步提升了在 `Linux` 系统中进行文件管理的能力。同时，本组将前四次实验所学的知识进行了整合与应用，不仅巩固了理论知识，还提高了实际操作能力。此次实验不仅加深了本组成员对 `Linux` 系统的理解，还为未来进一步运用 `Linux` 系统奠定了坚实的基础。

## 八. 分工情况

史佳鑫：`samba` 服务器搭建、自动生成代码记录脚本编写

熊永轩：自动从目录删除文件的脚本、自动生成测试记录文档脚本、实验报告