|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Linux操作系统综合实验**  **实验报告** | | | | |
|  | |  |  |  |
| 图片包含 标牌  已生成极高可信度的说明 | | | | |
|  | | |  |  |
| **姓名** |  |  |  |  |
| **班级** |  |  |  |  |
| **学号** |  |  |  |  |
| **日期** | 2023-12-29 |  |  |  |

**目录**

[一． 实验目的 3](#_Toc154749916)

[二． 实验要求 3](#_Toc154749917)

[三． 题目分析及基本设计过程分析 3](#_Toc154749918)

[1. 题目分析 3](#_Toc154749919)

[2. 设计决策 4](#_Toc154749920)

[3. 实现过程 4](#_Toc154749921)

[四． 配置文件关键修改处的说明及运行情况 4](#_Toc154749922)

[1. Samba安装和配置 5](#_Toc154749923)

[2. 文件上传功能 5](#_Toc154749924)

[3. 测试功能 6](#_Toc154749925)

[4. 自动生成文档 7](#_Toc154749926)

[5. 权限控制 8](#_Toc154749927)

[五． 脚本源程序清单 10](#_Toc154749928)

[1. 自动生成代码记录脚本auto-code-test.sh 10](#_Toc154749929)

[2. 自动从目录删除文件的脚本auto-delete.sh 12](#_Toc154749930)

[3. 自动生成测试记录文档脚本auto-test-record.sh 13](#_Toc154749931)

[六． 实验过程中出现的问题及解决方法 14](#_Toc154749932)

[七． 实验体会 15](#_Toc154749933)

[八． 分工情况 15](#_Toc154749934)

# 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。

# 实验要求

1. 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：
2. 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。
3. 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。
4. 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。
5. 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。

# 题目分析及基本设计过程分析

## 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。

## 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。

## 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

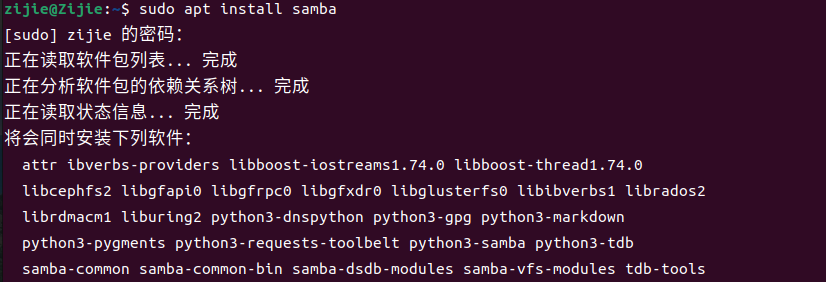
以上脚本具体信息将在报告第五部分进行展示

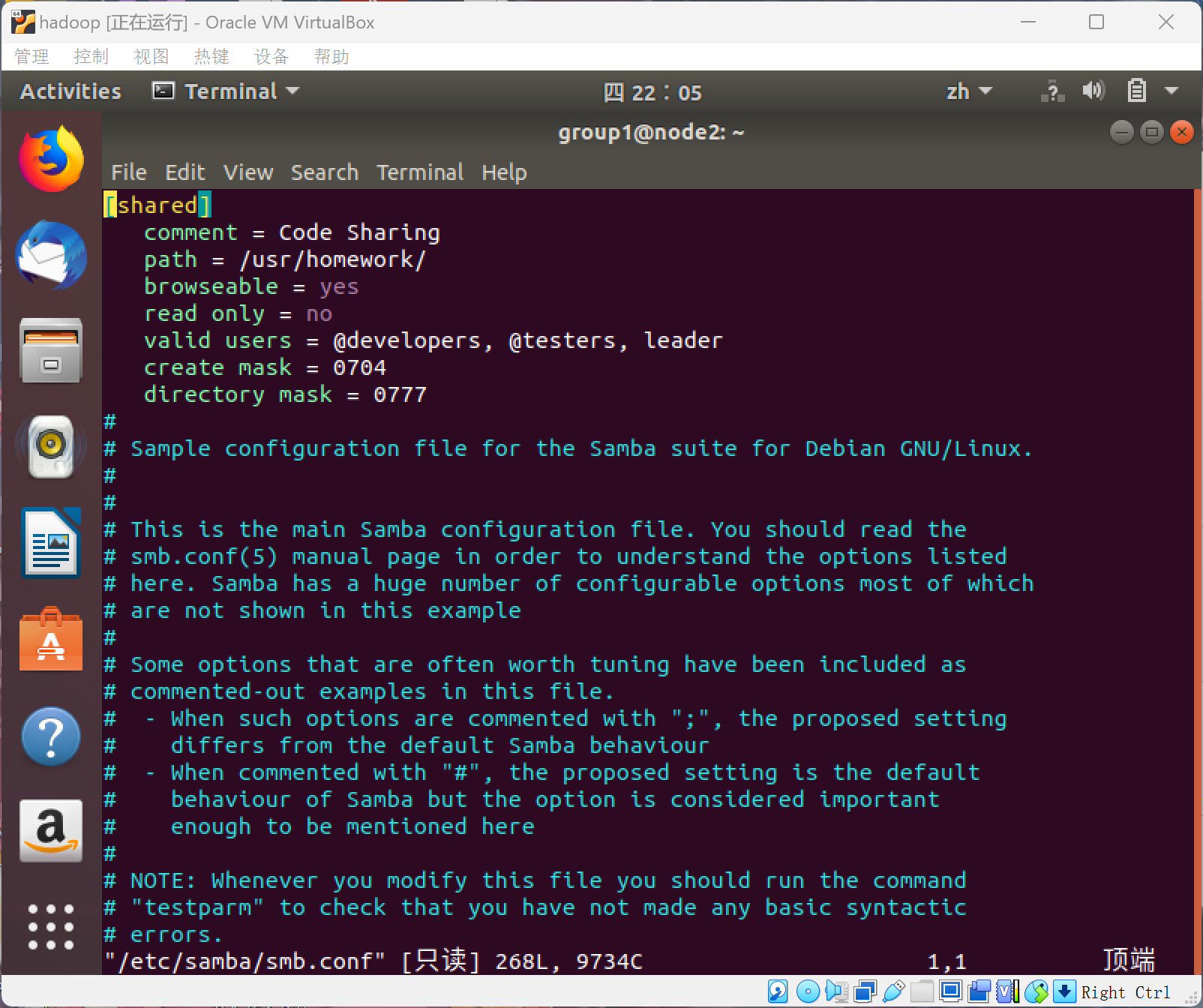
**权限控制：**通过将用户分配到不同组实现角色的分配，为每个组分配相应权限以实现权限控制。在samba服务中，用户上传的文件默认权限为704，使得开发人员上传的代码只能供自己查看，而其他开发人员不能查看，但测试人员可以查看。

具体权限分配将在报告第四部分进行展示。

# 配置文件关键修改处的说明及运行情况

## Samba安装和配置





/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。

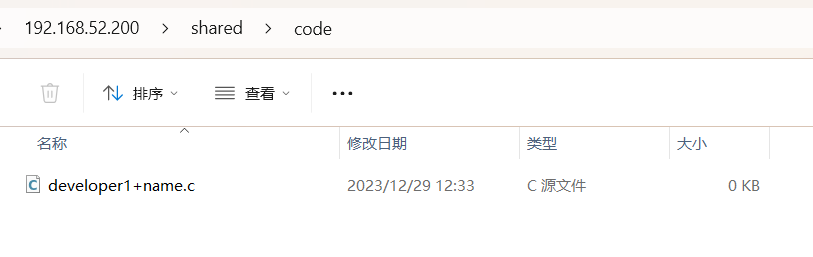
/testreport：测试人员完成测试后，在此上传测试报告。

/record：系统在此自动生成开发/测试记录。

/bin：系统的核心脚本。

## 文件上传功能

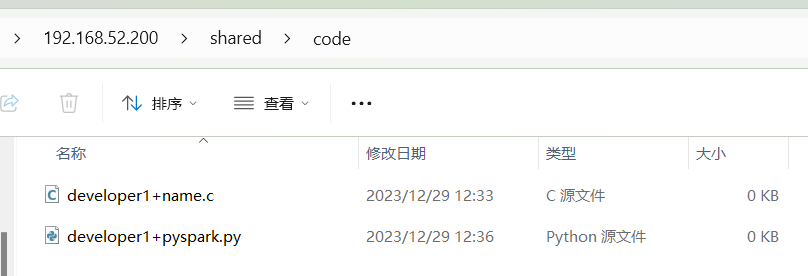
配置好samba服务后，可以在windows上通过\\IP地址或者在linux中使用smbclient连接到samba服务。使用开发人员账户登录到samba服务器后，可以直接向code目录中上传代码。代码应以标准格式命名。



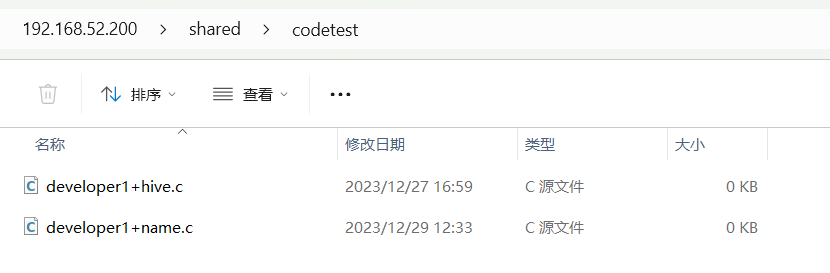
## 测试功能

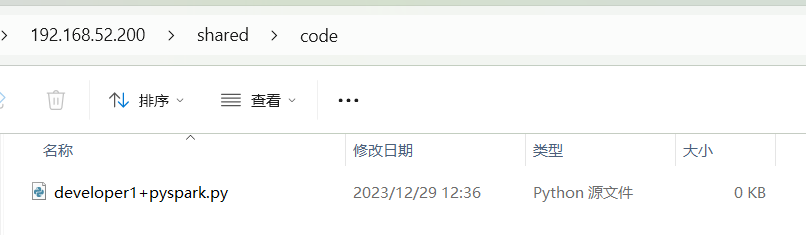
测试人员登录到samba服务器后，可以将开发人员上传至/code的代码复制到/codetest，以进行代码测试。成功复制后，系统的定时任务脚本将自动删除/code中的同名文件。

测试前：

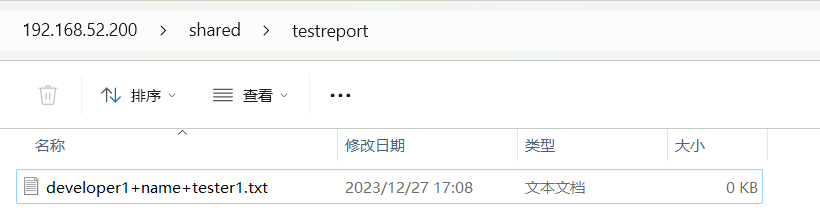


开始测试：



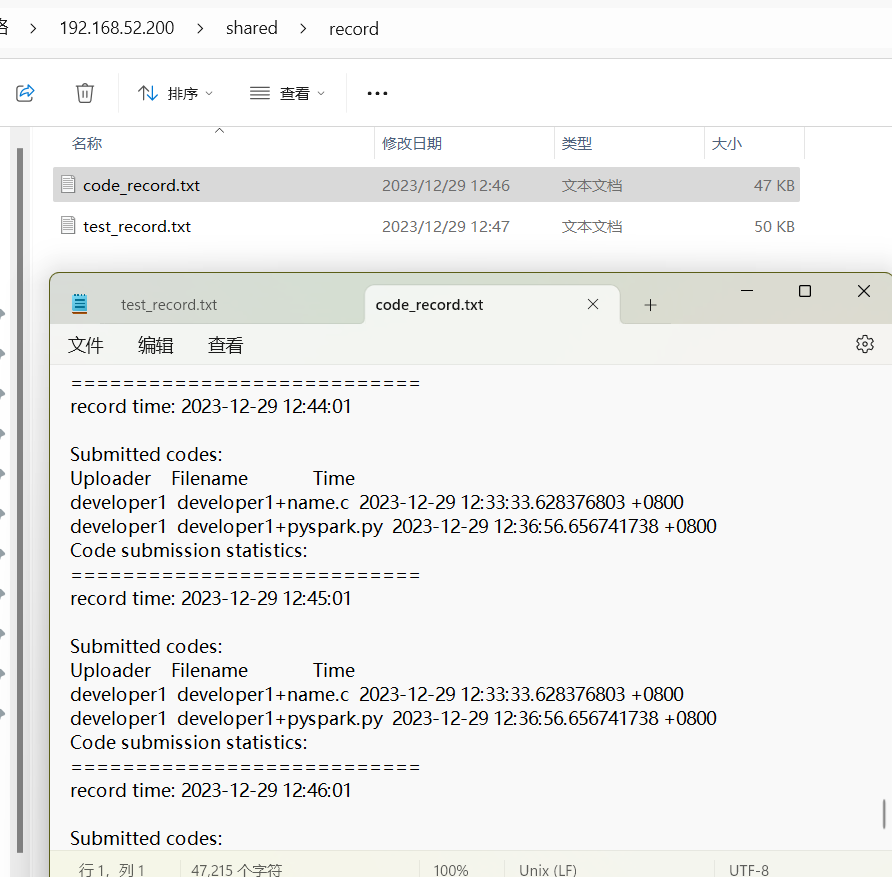


完成测试后，测试人员应上传以标准格式命名的测试报告。

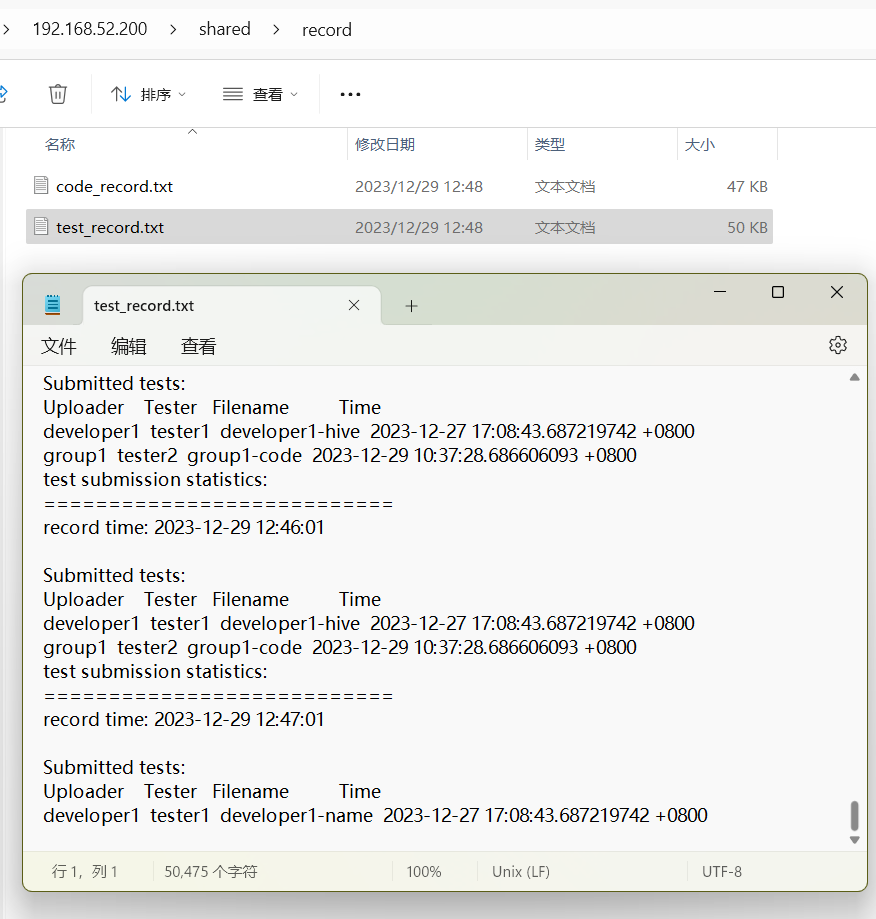


## 自动生成文档

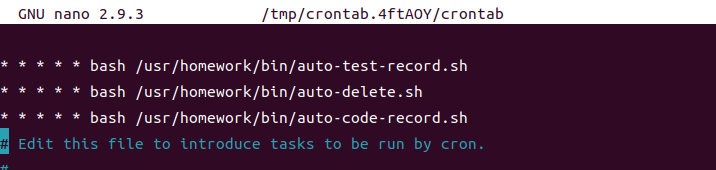
**代码上传记录**



**测试记录**



记录的生成通过定时任务实现，定时任务配置如下：



## 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：

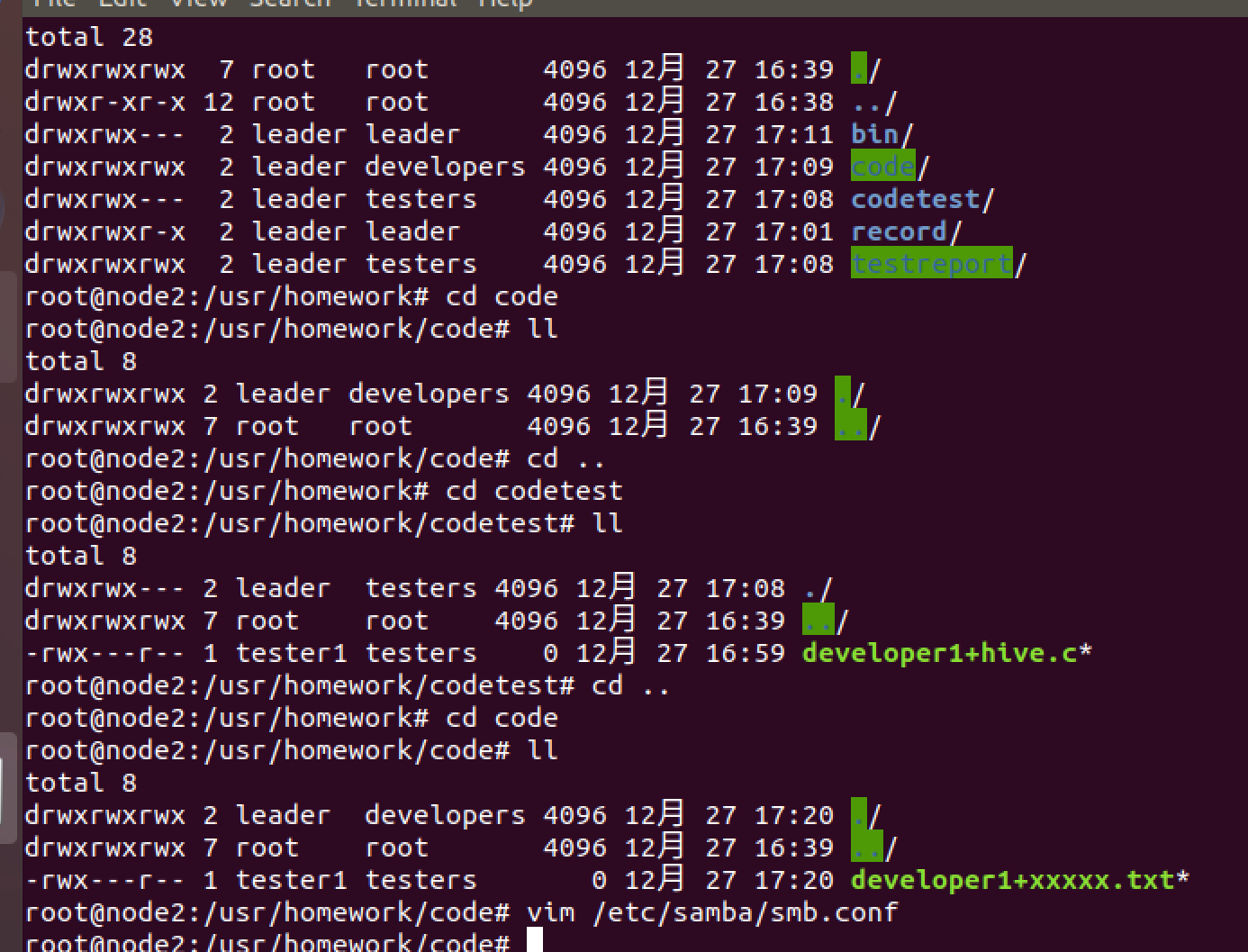
所有文件夹的所有者均为leader，保证了领导者对本系统的最高权限。

/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。

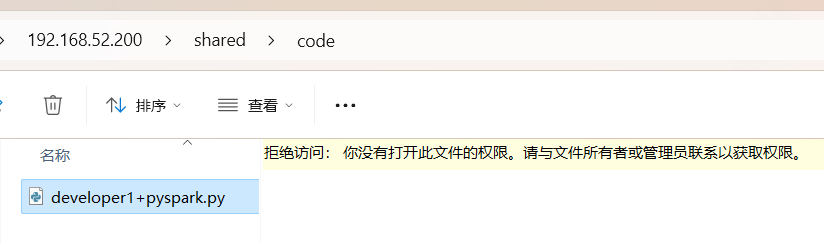
/testreport ： 该文件夹可供所有人查看，但仅供测试人员修改。

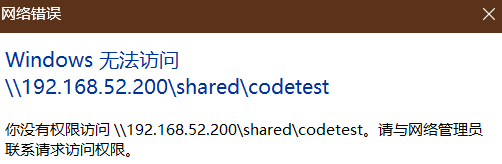
/record 和/bin ： 可供所有人查看，但仅供系统和领导者修改。



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，



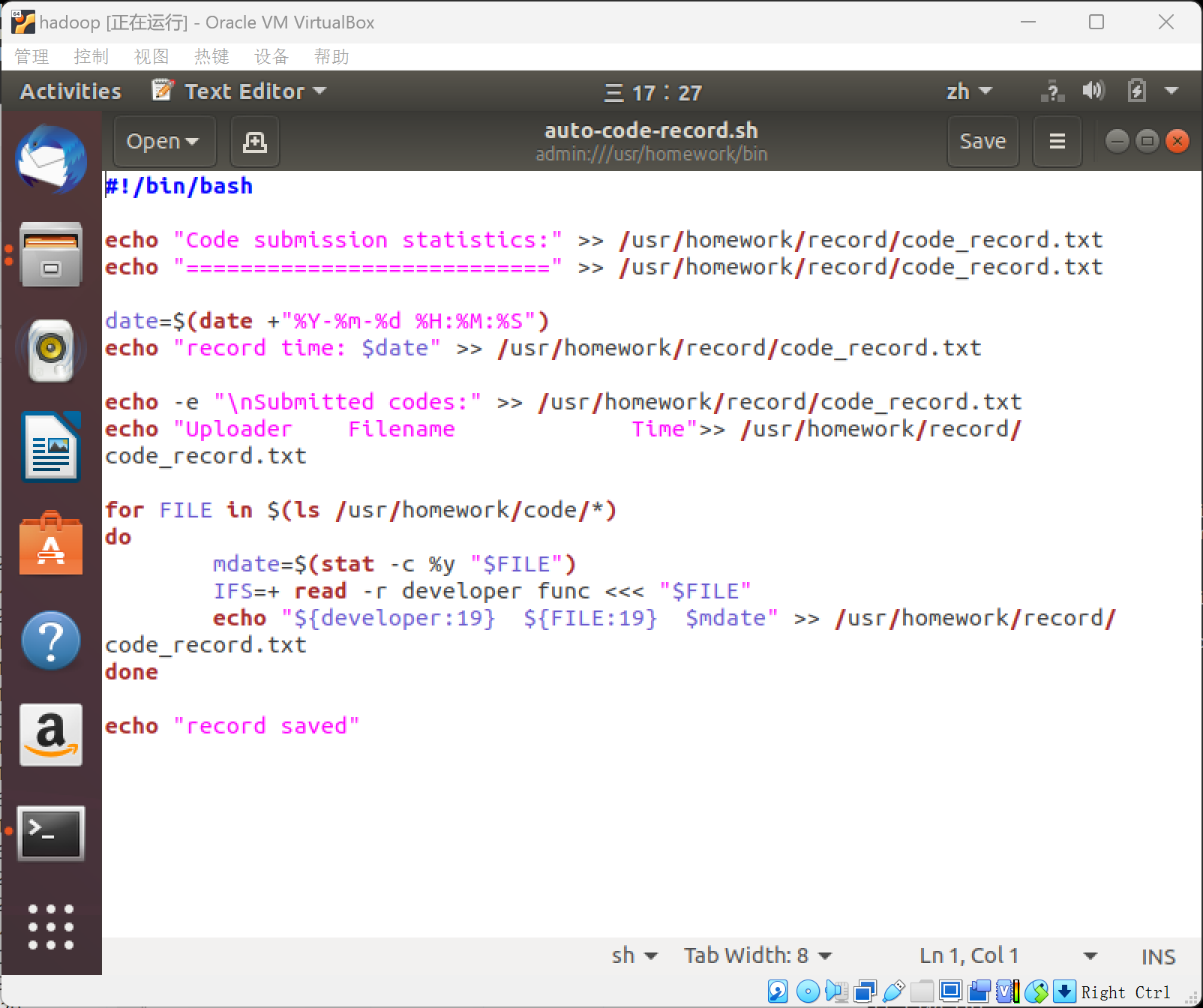


满足题目的权限需求。

# 脚本源程序清单

## 自动生成代码记录脚本auto-code-test.sh

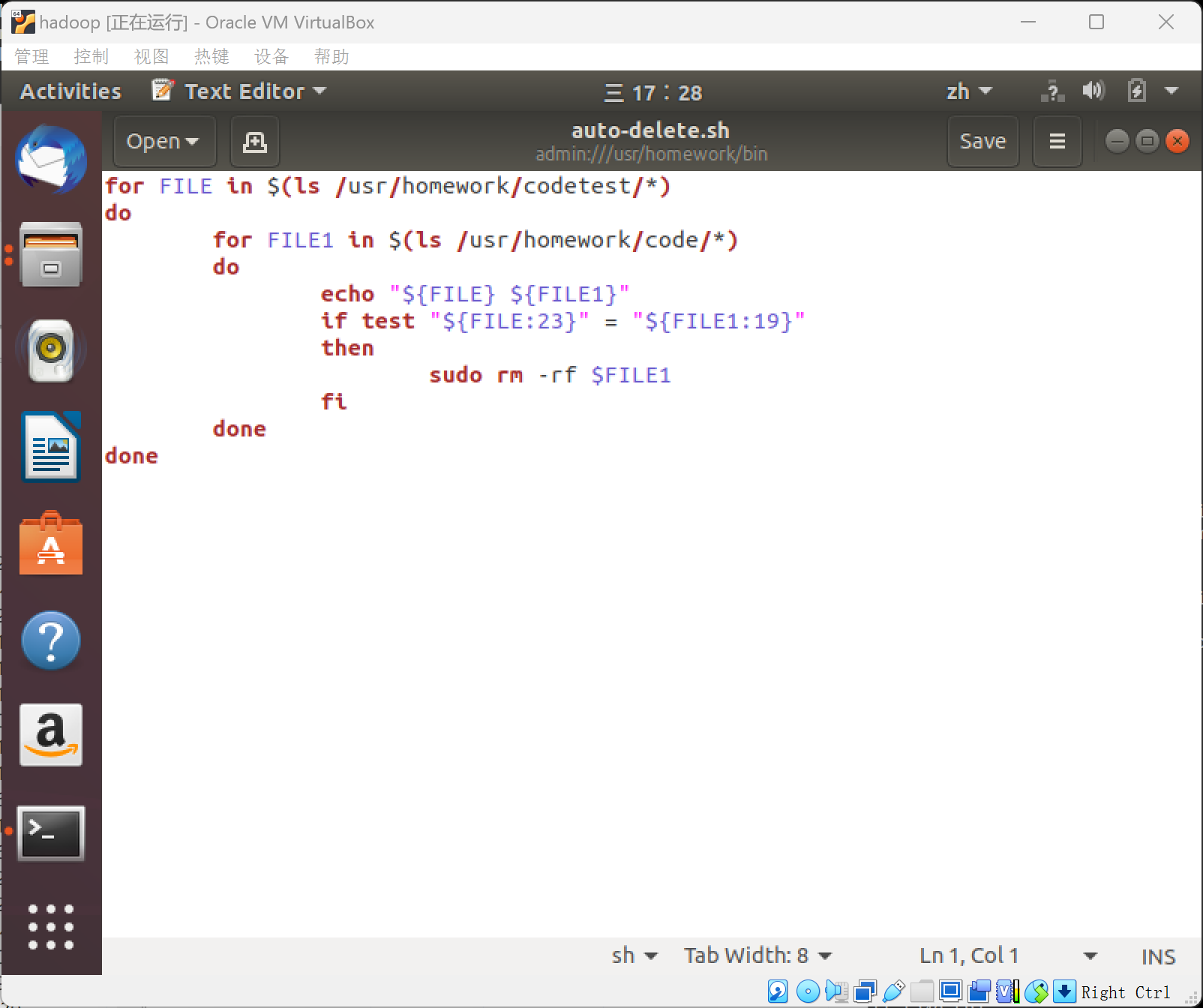
截图如下



**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}  $mdate" >> /usr/homework/record/code\_record.txt  
# 将变量 developer、FILE 和 mdate 的值输出到 /usr/homework/record/code\_record.txt 文件中，并追加到文件末尾  
# "${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

## 自动从目录删除文件的脚本auto-delete.sh

截图如下

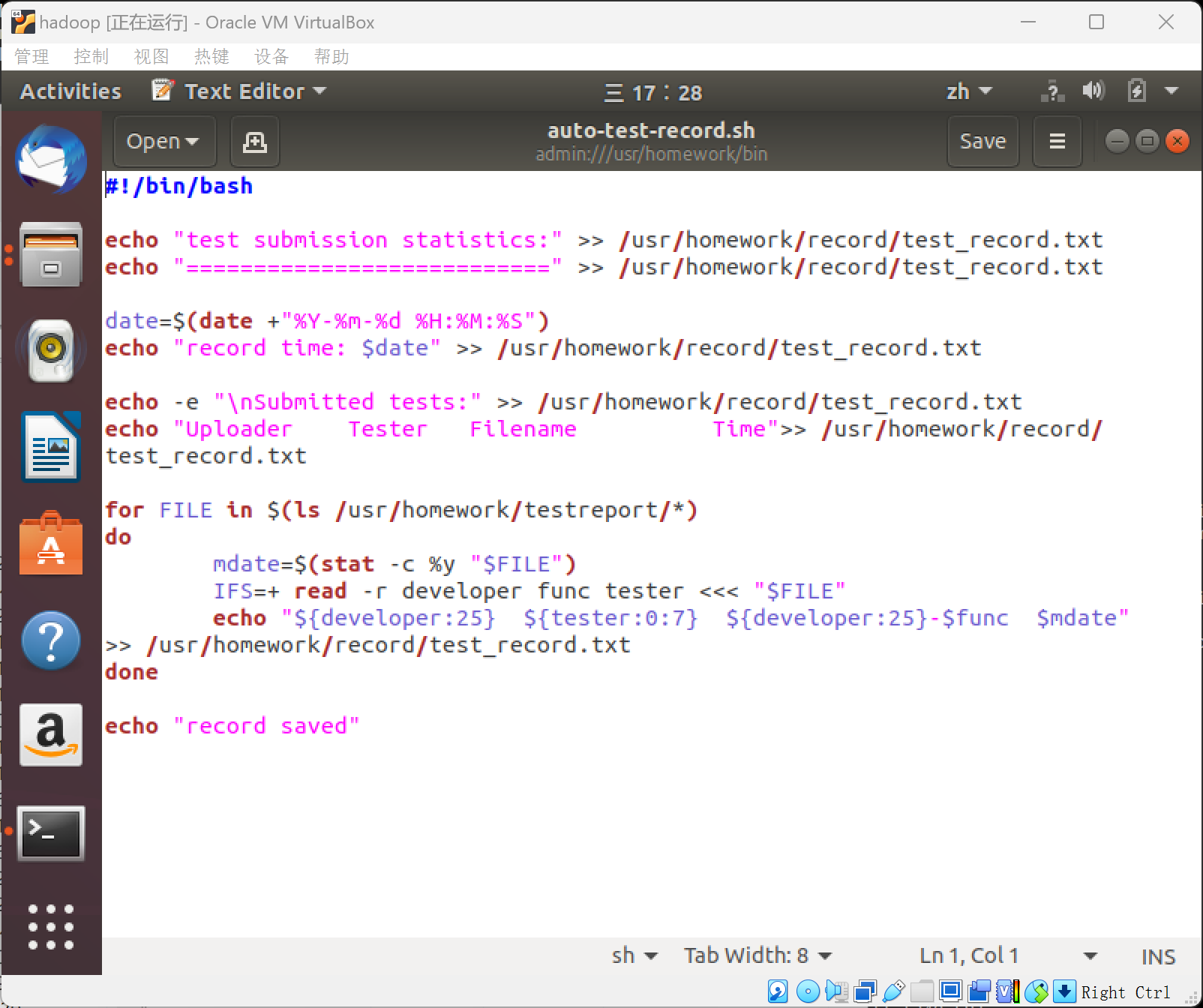


**具体注释如下**

#!/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

## 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

# 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：

1. 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。
2. 在脚本遍历文件时，发现取出的用户名最后多了一位冒号。这是因为对linux脚本编写不熟悉，采取了与python相似的语法在for循环后加了冒号，导致取出的文件名末尾都被添加了冒号。
3. 通过文件名自动生成记录的脚本中，运行报错，发现是取出文件名后几位（去除前面的路径部分）所使用的语法不对，在linux脚本中，${a:3:4}表示从a的第三位向后取4位。

# 实验体会

在本次实验中，本组完成了实验要求，使用samba服务和shell脚本实现了文件共享、文件管理、自动生成和权限控制功能。为完成本次实验，本组多次在samba服务配置与脚本的编写上进行了多次尝试，最终完成任务，通过反复实践更加熟练地掌握了samba服务的配置文件编写与shell脚本的语法，能够更好地运行Linux系统地文件管理功能，将前四次实验中习得的知识进行了汇总与运用，能够将这些知识投入实际运用中，增进了本组成员对Linux系统的理解，为未来进一步运用Linux系统打下了基础。