



西安交通大学

XI'AN JIAOTONG UNIVERSITY

## 本科生实验报告

# Linux 操作系统应用开发训练

姓名：杨豪

班级：软件 2101

时间：2022 年 12 月 18 日

# 目录

<b>实验 0: Linux 简介</b>	<b>1</b>
<b>实验 1: Linux 用户、组和进程管理</b>	<b>2</b>
1.1 实验目的	2
1.2 实验内容	2
1.3 题目分析	3
1.4 配置文件修改与运行情况	4
1.4.1 创建用户	4
1.4.2 委托管理任务	4
1.4.3 计划任务	5
1.4.4 系统调用与进程通信	5
1.5 问题与解决方法	9
1.6 实验体会	9
<b>实验 2: Linux 文件管理</b>	<b>10</b>
2.1 实验目的	10
2.2 实验内容	10
2.3 题目分析	10
2.4 配置文件与运行情况	11
2.4.1 public 文件夹权限管理	11
2.4.2 文件权限管理	11
2.4.3 软链接	12
2.4.4 文件系统	12
2.5 问题和解决方法	14
2.6 实验体会	14
<b>实验 3: Linux NFS、Samba 网络服务</b>	<b>15</b>
3.2 实验目的	15
3.2 实验内容	15
3.3 题目分析	15
3.4 配置文件与运行情况	16
3.4.1 本机网络情况	16
3.4.2 NFS	17

3.5 问题和解决方法 . . . . .	19
3.6 实验体会 . . . . .	19
<b>实验 4：综合训练</b>	<b>20</b>
4.1 实验目的 . . . . .	20
4.2 实验内容 . . . . .	20
4.3 题目分析 . . . . .	21
4.3.1 Apache . . . . .	21
4.3.2 收作业服务 . . . . .	22
4.3.3 调度算法 . . . . .	22
4.4 配置文件与运行情况 . . . . .	22
4.4.1 Apache . . . . .	22
4.4.2 收发作业 . . . . .	26
4.4.3 调度算法 . . . . .	27
4.5 脚本源程序清单 . . . . .	28
4.6 问题和解决方法 . . . . .	39
4.7 实验体会 . . . . .	39

# 实验 0: Linux 简介

略

# 实验 1: Linux 用户、组和进程管理

## 1.1 实验目的

熟练掌握 Linux 操作系统的使用，掌握 Linux 的系统的进程管理相关内容，掌握进程之间的通信方式。

进程是操作系统中最重要概念，贯穿始终，也是学习现代操作系统的关键。通过本次实验，要求理解进程的实质和进程管理的机制。在 Linux 系统下实现进程从创建到终止的全过程，从中体会进程的创建过程、父进程和子进程的关系、进程状态的变化、进程之间的同步机制、进程调度的原理和以信号和管道为代表的进程间通信方式的实现。

## 1.2 实验内容

1. 在命令行新建多个普通用户，如 tux, bob, Alice, lily 等，给每个用户创建密码，并将这几个用户分到同一个组 xjtuse 中。再新建两个组 coding 和 testing，使得某些用户也分别为其组用户。在 root 用户和新建用户之间切换，验证用户创建成功与否。（给出相关命令运行结果）
2. 实现 sudo 委托管理任务，给上述某一指定的普通用户赋予创建用户的权限。（给出相关配置文件和命令运行结果）
3. 备份数据是系统应该定期执行的任务，请利用 cron 计划作业在每周五下午 6:10 对某用户（如 tux）主目录下的文件进行备份（可使用 tar 命令）。给出相关运行结果和邮件记录。
4. 编制实现软中断通信的程序。使用系统调用 fork() 创建两个子进程，再用系统调用 signal() 让父进程捕捉键盘上发出的中断信号（即按 delete 键），当父进程接收到这两个软中断的某一个后，父进程用系统调用 kill() 向两个子进程分别发出整数值为 16 和 17 软中断信号，子进程获得对应软中断信号，然后分别输出下列信息后终止：

```
1 Child process 1 is killed by parent !!
2 Child process 2 is killed by parent !!
```

父进程调用 wait() 函数等待两个子进程终止后，输入以下信息，结束进程执行：

```
1 Parent process is killed!!
```

多运行几次编写的程序，简略分析出现不同结果的原因。

5. 编制实现进程的管道通信的程序，使用系统调用 `pipe()` 建立一条管道线，两个子进程分别向管道写一句话：

```
1 Child process 1 is sending a message!
2 Child process 2 is sending a message!
```

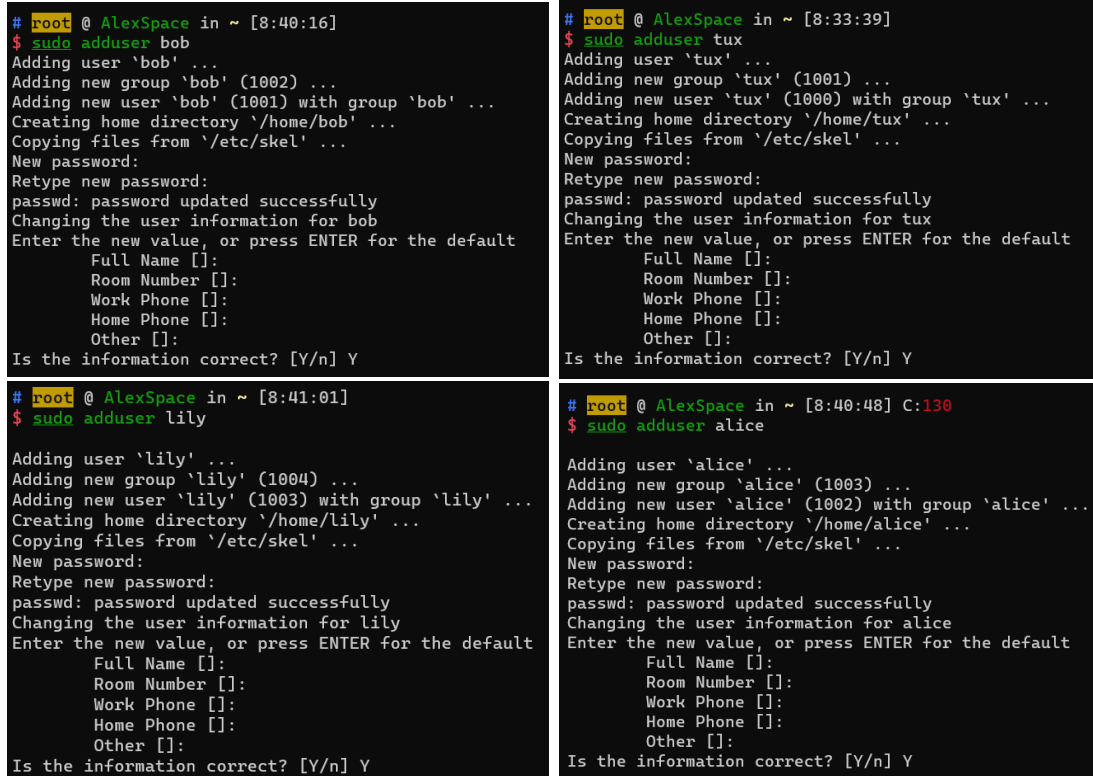
而父进程则从管道中读出来自于两个子进程的信息，显示在屏幕上。要求：父进程先接收子进程 P1 发来的消息，然后再接收子进程 P2 发来的消息。

## 1.3 题目分析

1. 本题目较为简单，利用 `adduser` 创建用户、`groupadd` 创建用户组、`usermod -a -G` 将用户添加入用户组即可，最后用 `groupmems -g -l` 检验实验结果。具体使用方法可以通过 `man` 查阅。
2. 通过修改 `visudo` 配置文件的参数即可实现委托管理任务
3. `cron` 计划任务可以通过 `crontab -e` 编辑配置文件并通过 `crontab -l` 和 `cron.log` 检查
4. 4 和 5 均可通过阅读 `man system call` 完成。第四题需要分别在父进程和两个子进程里使用 `signal` 捕捉信号并处理，为了保证信号的接收需要分别建立若无处理则会死循环的程序，维系该死循环的。第五题需要在父进程和子进程之间利用 `pipe` 创建的管道通信（`read`, `write`）。

## 1.4 配置文件修改与运行情况

### 1.4.1 创建用户



```
# root @ AlexSpace in ~ [8:40:16]
$ sudo adduser bob
Adding user 'bob' ...
Adding new group 'bob' (1002) ...
Adding new user 'bob' (1001) with group 'bob' ...
Creating home directory '/home/bob' ...
Copying files from '/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for bob
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] Y

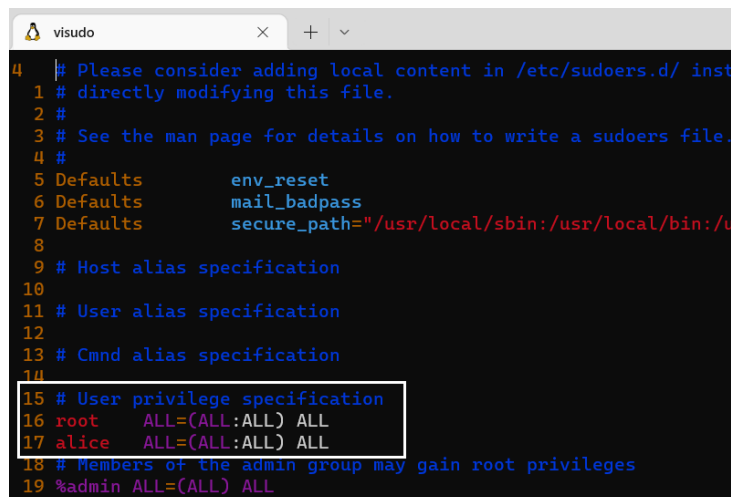
# root @ AlexSpace in ~ [8:41:01]
$ sudo adduser lily
Adding user 'lily' ...
Adding new group 'lily' (1004) ...
Adding new user 'lily' (1003) with group 'lily' ...
Creating home directory '/home/lily' ...
Copying files from '/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for lily
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] Y

# root @ AlexSpace in ~ [8:33:39]
$ sudo adduser tux
Adding user 'tux' ...
Adding new group 'tux' (1001) ...
Adding new user 'tux' (1000) with group 'tux' ...
Creating home directory '/home/tux' ...
Copying files from '/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for tux
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] Y

# root @ AlexSpace in ~ [8:40:48] C:130
$ sudo adduser alice
Adding user 'alice' ...
Adding new group 'alice' (1003) ...
Adding new user 'alice' (1002) with group 'alice' ...
Creating home directory '/home/alice' ...
Copying files from '/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for alice
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] Y
```

图 1: create 4 users

### 1.4.2 委托管理任务



```
visudo
4 # Please consider adding local content in /etc/sudoers.d/ inst
1 # directly modifying this file.
2 #
3 # See the man page for details on how to write a sudoers file.
4 #
5 Defaults env_reset
6 Defaults mail_badpass
7 Defaults secure_path="/usr/local/sbin:/usr/local/bin:/u
8
9 # Host alias specification
10
11 # User alias specification
12
13 # Cmnd alias specification
14
15 # User privilege specification
16 root ALL=(ALL:ALL) ALL
17 alice ALL=(ALL:ALL) ALL
18 # Members of the admin group may gain root privileges
19 %admin ALL=(ALL) ALL
20
```

图 2: visudo 修改界面

在 visudo 界面修改 alice 权限为管理员

```
# root @ AlexSpace in ~ [9:26:28] C:1
$ su alice
alice@AlexSpace:/root$ su root
Password:
alice@AlexSpace:/root$ sudo su root

# root @ AlexSpace in ~ [9:27:23]
$ su bob
bob@AlexSpace:/root$ sudo su root
[sudo] password for bob:
bob is not in the sudoers file. This incident will be reported.
```

图 3: 验证权限

在 alice 用户下可以通过 sudo 直接切换至 root，而 bob 不行

### 1.4.3 计划任务

为了方便测试，将原题目改为在每个周二的下午 14 点 07 分打包 tux 的用户目录

```
# root @ AlexSpace in ~ [14:13:34]
$ crontab -l
07 14 * * 2 tar -zcf /var/backups/tux.tgz /home/tux/
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
```

图 4: crontab -l

通过 crontab 修改配置文件如上。

```
# root @ AlexSpace in ~ [14:10:34] C:1
$ tail /var/log/cron.log | grep CRON
Nov 4 11:29:01 AlexSpace CRON[4914]: (root) CMD (/bin/ls)
Nov 8 14:07:01 AlexSpace CRON[1123]: (root) CMD (tar -zcf /var/backups/tux.tgz /home/tux/)

# root @ AlexSpace in ~ [14:12:22]
$ l /var/backups | grep tux
-rw-r--r-- 1 root root 2.4K Nov 8 14:07 tux.tgz

# root @ AlexSpace in ~ [14:12:14] C:2
$ tar -tzvf /var/backups/tux.tgz
drwxr-xr-x tux/tux
-rw-r--r-- tux/tux 0 2022-11-04 09:28 home/tux/
-rw-r--r-- tux/tux 86 2022-11-04 18:25 home/tux/.bash_history
-rw-r--r-- tux/tux 887 2022-11-04 08:39 home/tux/.profile
-rw-r--r-- tux/tux 220 2022-11-04 08:39 home/tux/.bash_logout
-rw-r--r-- tux/tux 3771 2022-11-04 08:39 home/tux/.bashrc
```

图 5: create 4 users

### 1.4.4 系统调用与进程通信



Listing 1: oslab1\_1.c

```
1 #include <stdio.h>
2 #include <signal.h>
3 #include <unistd.h>
4 #include <stdlib.h>
5 #include <sys/types.h>
6 #include <wait.h>
7 int flag;
8
9 void stop(){
10     flag = 0;
11 }
12
13 int main(){
14     int pid1, pid2;
15     while((pid1=fork()) == -1);
16     signal(SIGINT, stop);
17     if(pid1>0){
18         while((pid2=fork()) == -1);
19         if(pid2>0){
20             flag = 1;
21             sleep(10);
22             kill(pid1,16);
23             wait(0);
24             kill(pid2,17);
25             wait(0);
26             printf("\n Parent process is killed!!\n");
27             exit(0);
28         }else{
29             flag = 1;
30             signal(17, stop);
31             while(1){
32                 if(flag==0){
33                     printf("\n Child process2 is killed by parent!!\n");
34                     exit(0);
35                 }
```

```
36         }
37     }
38     }else{
39         flag = 1;
40         signal(16,stop);
41         while(1){
42             if(flag == 0){
43                 printf("\n Child process1 is killed by parent!!\n");
44                 exit(0);
45             }
46         }
47     }
48 }
```

Listing 2: oslab1\_2.c

```
1  #include <unistd.h>
2  #include <signal.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <wait.h>
6
7  int pid1,pid2;
8
9  int main(){
10     int fd[2];
11     char OutPipe[100],InPipe[100];
12     pipe(fd);
13     while((pid1=fork())!=-1);
14     if(pid1==0){
15         lockf(fd[1],1,0);
16         sprintf(OutPipe,"\n Child process1 is sending message! \n");
17         ;
18         write(fd[1],OutPipe,50);
19         sleep(5);
20         lockf(fd[1],0,0);
21         exit(0);
22     }else{
```

```
22         while((pid2=fork())== -1);
23         if(pid2==0){
24             lockf(fd[1],1,0);
25             sprintf(OutPipe, "\n Child process2 is sending
26                 message! \n");
27             write(fd[1],OutPipe,50);
28             sleep(5);
29             lockf(fd[1],0,0);
30             exit(0);
31         }else{
32             wait(0);
33             read(fd[0],InPipe,50);
34             printf("%s\n",InPipe);
35             wait(0);
36             read(fd[0],InPipe,50);
37             printf("%s\n",InPipe);
38             exit(0);
39         }
40     }
```

运行结果如下

```
# root @ AlexSpace in ~ [20:04:05]
$ gcc oslab1.c --output lab1_1

# root @ AlexSpace in ~ [20:04:27]
$ gcc oslab1.2.c --output lab1_2

# root @ AlexSpace in ~ [20:04:40]
$ ./lab1_1
^C

Child process1 is killed by parent!!
Child process2 is killed by parent!!

Parent process is killed!!

# root @ AlexSpace in ~ [20:05:07]
$ ./lab1_2

Child process1 is sending message!

Child process2 is sending message!
```

## 1.5 问题与解决方法

在任务三中, 因为 Ubuntu 和 RedHat 系统预设上的区别, cron 任务在开机时不会启动, 也没有发邮件的机制和应用, 除此之外还有日志没记录等问题, 在网络上搜索后找到解决方法: 安装邮件客户端并在每次启动时打开 cron 服务和 rsyslog 服务.

## 1.6 实验体会

本次实验让我熟悉了 Linux 的用户管理与权限分配的指令和机制, 而对定时任务的介绍更让我看到 shell 脚本更多的可行性, 最后的两个任务让我对 Linux 的进程间通信的系统调用有了更深的理解。

# 实验 2：Linux 文件管理

## 2.1 实验目的

熟练掌握 Linux 操作系统的使用，掌握 Linux 的系统的进程管理和文件管理功能。

## 2.2 实验内容

1. 将若干已有用户加入到同一个组 xjtuse 中。在/home 下创建一个共享的公用目录 public，允许 xjtuse 组中的用户对该目录具有读写和执行操作。（给出相关命令及运行结果）
2. 对于 public 目录下的文件，只有文件的拥有者才具有删除文件的权限。（给出相关命令及运行结果）
3. 对于 public 目录下的文件，也可以通过路径/mnt/public 来访问。（给出相关命令及运行结果）
4. 看 Linux 系统磁盘空间的使用情况（给出显示结果），并为/分区创建磁盘配额，使得用户可用空间的软限制为 100M，硬限制为 150M，且每个用户可用的 inodes 的软限制为 100，硬限制为 120。并对磁盘配额情况进行验证测试。（给出相关命令及运行结果）

## 2.3 题目分析

1. 通过实验一的知识即可实现用户加入同一组，通过 chown、chgrp、chmod 等指令即可更改 public 的所属人，所属组和权限，使得 public 文件权限为 drwxrwxr-x，对应用户组为 xjtuse
2. 和第一题一样，通过 chmod 即可修改
3. 通过 ln 建立软连接即可
4. 首先需要准备文件系统，我们通过激活文件系统的限额，并重新将文件系统装入根分区完成文件系统的准备。随后初始化限额系统，成功之后就可以配置和管理用户限额和组限额。

## 2.4 配置文件与运行情况

### 2.4.1 public 文件夹权限管理

```
# root @ AlexSpace in ~ [12:51:43]
$ groupmems -g xjtuse -l
lily tux bob alice

# root @ AlexSpace in /home [12:52:11]
$ mkdir public

# root @ AlexSpace in /home [12:53:22] C:3
$ chgrp xjtuse public

# root @ AlexSpace in /home [12:56:02]
$ l
total 28K
drwxr-xr-x  7 root  root   4.0K Nov 15 12:52 .
drwxr-xr-x 19 root  root   4.0K Nov 15 12:49 ..
drwxr-xr-x  2 alice alice  4.0K Nov  4 09:28 alice
drwxr-xr-x  2 bob  bob   4.0K Nov  4 09:28 bob
drwxr-xr-x  2 lily lily  4.0K Nov  4 08:41 lily
drwxr-xr-x  2 root xjtuse 4.0K Nov 15 12:52 public
drwxr-xr-x  2 tux  tux   4.0K Nov  4 09:28 tux

# root @ AlexSpace in /home [12:56:09]
$ chown alice public

# root @ AlexSpace in /home [12:57:22]
$ chmod g+rxw public

# root @ AlexSpace in /home [12:59:49]
$ l
total 28K
drwxr-xr-x  7 root  root   4.0K Nov 15 12:52 .
drwxr-xr-x 19 root  root   4.0K Nov 15 12:49 ..
drwxr-xr-x  2 alice alice  4.0K Nov  4 09:28 alice
drwxr-xr-x  2 bob  bob   4.0K Nov  4 09:28 bob
drwxr-xr-x  2 lily lily  4.0K Nov  4 08:41 lily
drwxrwxr-x  2 alice xjtuse 4.0K Nov 15 12:52 public
drwxr-xr-x  2 tux  tux   4.0K Nov  4 09:28 tux
```

可以看到修改后 public 的文件权限改变为 drwxrwxr-x，所有人是 alice，组别为 xjtuse

### 2.4.2 文件权限管理

此处以 m.txt 文件为例

```
# root @ AlexSpace in /home/public [13:00:19]
$ touch m.txt

# root @ AlexSpace in /home/public [13:03:19] C:1
$ chmod -R u+rw,g-w,o-w .

# root @ AlexSpace in /home/public [13:03:27]
$ l
total 8.0K
drwxr-xr-x 2 alice xjtuse 4.0K Nov 15 13:00 .
drwxr-xr-x 7 root root 4.0K Nov 15 12:52 ..
-rw-r--r-- 1 root root 0 Nov 15 13:00 m.txt
```

### 2.4.3 软链接

此处未采用要求的/mnt/public 路径, 因为 WSL 中/mnt 指向主机的根目录, 改用将 public/mnt 指向 public

```
# root @ AlexSpace in /home [13:16:24]
$ ln -s public public/mnt

# root @ AlexSpace in /home/public [13:17:12]
$ l
total 8.0K
drwxr-xr-x 2 alice xjtuse 4.0K Nov 15 13:17 .
drwxr-xr-x 7 root root 4.0K Nov 15 12:52 ..
-rw-r--r-- 1 root root 0 Nov 15 13:00 m.txt
lrwxrwxrwx 1 root root 6 Nov 15 13:17 mnt -> public
```

### 2.4.4 文件系统

第一个任务在 WSL+Ubuntu 环境, 其余及以后的任务采用 VMware+Ubuntu 环境

```
# root @ AlexSpace in /home [13:03:32]
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb         251G   5.0G  234G   3% /
tmpfs            6.0G     0  6.0G   0% /mnt/wsl
tools           100G   97G   3.6G  97% /init
none             6.0G     0  6.0G   0% /dev
none             6.0G   4.0K  6.0G   1% /run
none             6.0G     0  6.0G   0% /run/lock
none             6.0G     0  6.0G   0% /run/shm
none             6.0G     0  6.0G   0% /run/user
tmpfs            6.0G     0  6.0G   0% /sys/fs/cgroup
drivers          100G   97G   3.6G  97% /usr/lib/wsl/drivers
lib             100G   97G   3.6G  97% /usr/lib/wsl/lib
C:\              100G   97G   3.6G  97% /mnt/c
D:\             376G  207G  170G  55% /mnt/d
```

图 9: 磁盘使用情况

磁盘配额命令及运行结果如下

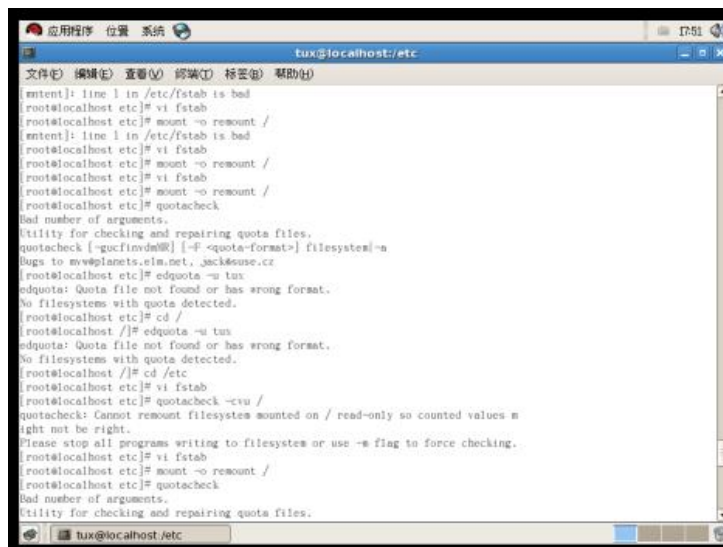
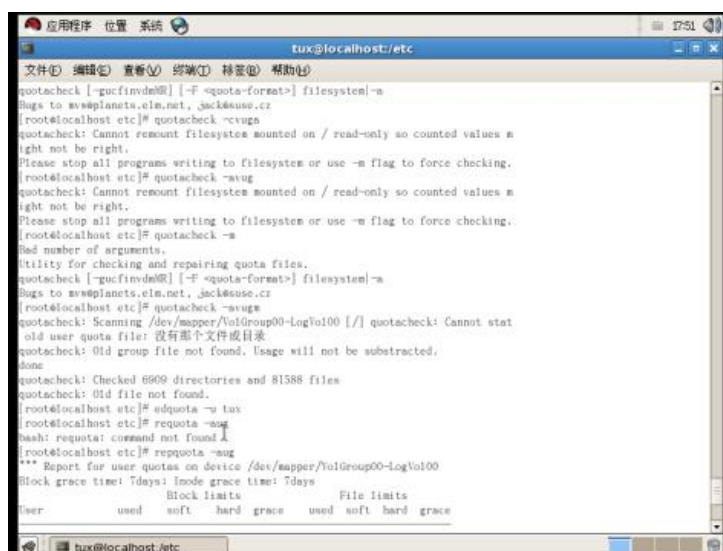
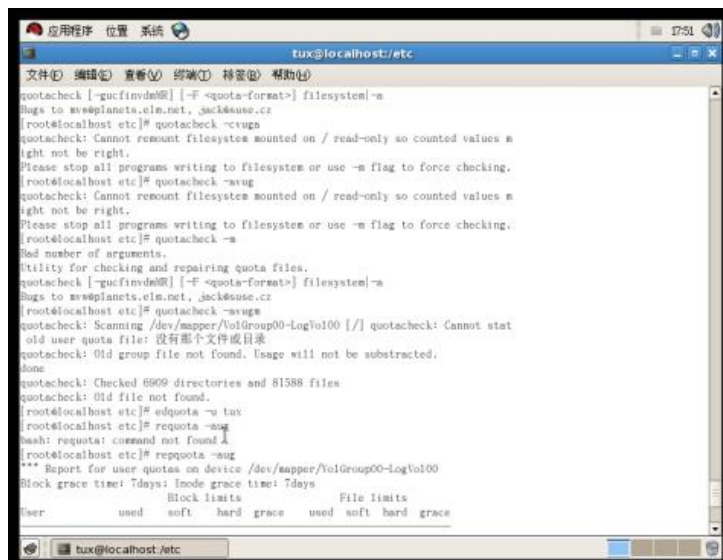


图 10: 运行前

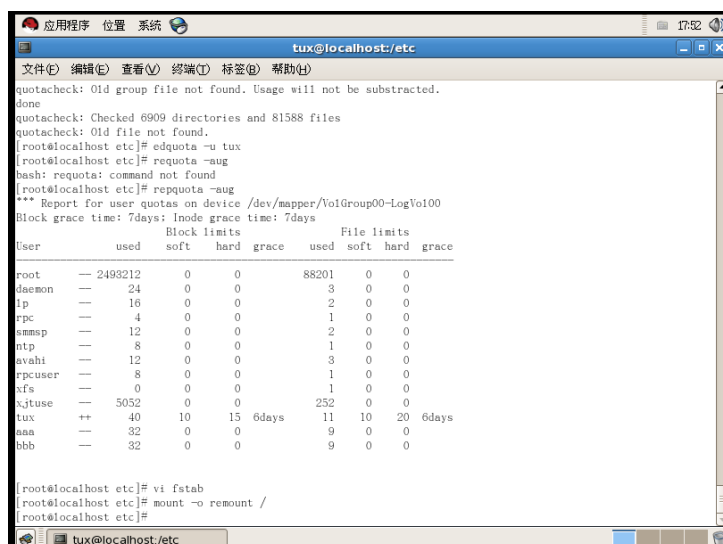






```
tux@localhost:/etc
quotacheck [-gucfinvdm] [-F <quota-format>] filesystem|-m
Bugs to svsw@planets.eln.net, jack@suse.cz
[root@localhost etc]# quotacheck -cvuga
quotacheck: Cannot remount filesystem mounted on / read-only so counted values might not be right.
Please stop all programs writing to filesystem or use -m flag to force checking.
[root@localhost etc]# quotacheck -avug
quotacheck: Cannot remount filesystem mounted on / read-only so counted values might not be right.
Please stop all programs writing to filesystem or use -m flag to force checking.
[root@localhost etc]# quotacheck -m
Red number of arguments.
Utility for checking and repairing quota files.
quotacheck [-gucfinvdm] [-F <quota-format>] filesystem|-m
Bugs to svsw@planets.eln.net, jack@suse.cz
[root@localhost etc]# quotacheck -avug
quotacheck: Scanning /dev/mapper/VolGroup00-LogVol100 [/] quotacheck: Cannot stat old user quota file: 没有那个文件或目录
quotacheck: Old group file not found. Usage will not be subtracted.
done
quotacheck: Checked 6909 directories and 81588 files
quotacheck: Old file not found.
[root@localhost etc]# edquota -u tux
[root@localhost etc]# requota -aug
bash: requota: command not found
[root@localhost etc]# repquota -aug
*** Report for user quotas on device /dev/mapper/VolGroup00-LogVol100
Block grace time: 7days; Inode grace time: 7days
Block limits      File limits
User      used soft hard grace used soft hard grace
root      -- 2493212 0 0      88201 0 0
deemon    -- 24 0 0      3 0 0
lp        -- 16 0 0      2 0 0
rpc       -- 4 0 0      1 0 0
smmmp     -- 12 0 0      2 0 0
ntp       -- 8 0 0      1 0 0
avahi     -- 12 0 0      3 0 0
rpcuser   -- 8 0 0      1 0 0
sfs       -- 0 0 0      1 0 0
xjstuse   -- 5052 0 0      252 0 0
tux       ++ 40 10 15 6days 11 10 20 6days
aaa       -- 32 0 0      9 0 0
bbb       -- 32 0 0      9 0 0

[root@localhost etc]# vi fstab
[root@localhost etc]# mount -o remount /
[root@localhost etc]#
```



```
tux@localhost:/etc
quotacheck: Old group file not found. Usage will not be subtracted.
done
quotacheck: Checked 6909 directories and 81588 files
quotacheck: Old file not found.
[root@localhost etc]# edquota -u tux
[root@localhost etc]# requota -aug
bash: requota: command not found
[root@localhost etc]# repquota -aug
*** Report for user quotas on device /dev/mapper/VolGroup00-LogVol100
Block grace time: 7days; Inode grace time: 7days
Block limits      File limits
User      used soft hard grace used soft hard grace
root      -- 2493212 0 0      88201 0 0
deemon    -- 24 0 0      3 0 0
lp        -- 16 0 0      2 0 0
rpc       -- 4 0 0      1 0 0
smmmp     -- 12 0 0      2 0 0
ntp       -- 8 0 0      1 0 0
avahi     -- 12 0 0      3 0 0
rpcuser   -- 8 0 0      1 0 0
sfs       -- 0 0 0      1 0 0
xjstuse   -- 5052 0 0      252 0 0
tux       ++ 40 10 15 6days 11 10 20 6days
aaa       -- 32 0 0      9 0 0
bbb       -- 32 0 0      9 0 0

[root@localhost etc]# vi fstab
[root@localhost etc]# mount -o remount /
[root@localhost etc]#
```

## 2.5 问题和解决方法

实验一开始由于文件系统的不兼容问题 WSL 无法采用,所以重新装了一遍 VMware + Ubuntu 才解决

实验中出现了 quotacheck 指令无法执行的问题,后来通过查询老师的 ppt 发现有时需要添加-m 参数才能成功执行。

## 2.6 实验体会

前几个实验都很简单,但最后一个实验很复杂,再加上环境不兼容,遇到了很多问题,在执行的过程中又出现了如参数不当等的意外情况。所以在这个实验的过程中,我体会到了耐心冷静对于做一件复杂的事情是有着极大的帮助的。

# 实验 3：Linux NFS、Samba 网络服务

## 3.2 实验目的

熟练掌握 Linux 操作系统的使用，掌握 Linux 系统的 NFS 和 Samba 服务的配置和管理。

## 3.2 实验内容

1. 查看系统的网络情况，确保能够在本地网络中联网通信（给出网络接口配置文件和测试结果），获知主机的 IP 地址和主机所在的子网信息。
2. 在本机提供 NFS 服务，请将本地的/home/设为共享目录供指定客户机使用，客户机具有读写权限。给出访问结果。
3. 假设本地网络中大部分客户端是 windows 系统，请建立 Samba 服务器使得客户端能够共享 Linux 服务器的资源，具体要求如下：
  - (a) 创建一个共享文件夹/home/public，使得所有用户都可以匿名访问（可读写）。
  - (b) 每个用户可以访问自己的主目录，且具有完全权限，采用用户验证的方式进行配置；
  - (c) 为用户 tux 和 tom 创建一个共享目录/home/share，可供这两个用户进行文件的共享（可读写）；
  - (d) 测试：使用 smbclient 客户端程序和 windows 客户端分别登录 Samba 服务器，访问服务器中的共享资源。
  - (e) 注：以上所需用户组和用户以及文件夹需要自己创建，并具有适当的权限。实验报告中需要给出配置文件及相关的运行结果。

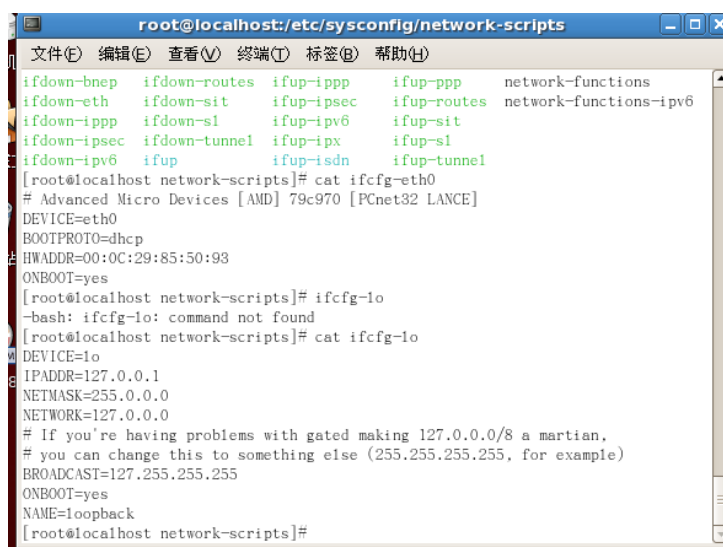
## 3.3 题目分析

1. 网络接口的配置文件在/etc/sysconfig/network-scripts/目录下，本地网络的通信则是采用 ping 命令进行验证
2. 修改配置文件/etc/exports，更新配置文件，启动服务
3. samba

- (a) 修改 smb.conf 文件，客户端访问共享文件夹；
- (b) 配置 smb.conf 文件，为用户设置密码，输入密码访问共享文件夹
- (c) 创建文件夹，并创建文件，配置 smb.conf 文件，分别使用用户 tux，tom 访问共享文件夹

## 3.4 配置文件与运行情况

### 3.4.1 本机网络情况

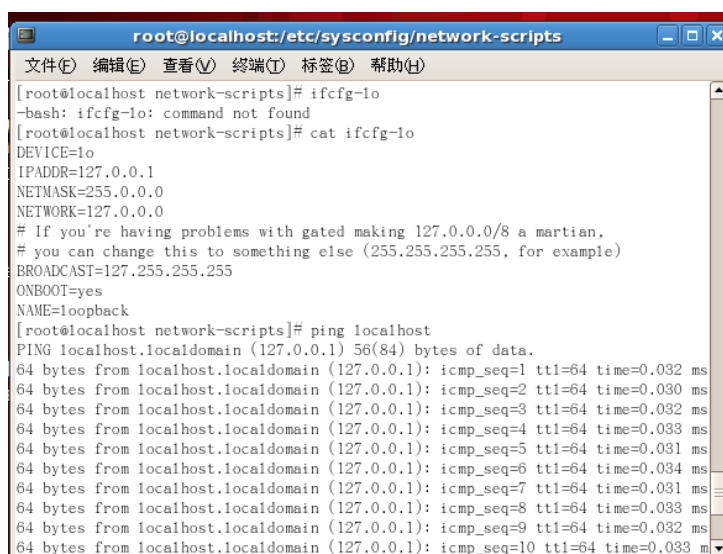


```

root@localhost:/etc/sysconfig/network-scripts
文件(F) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
ifdown-bnep ifdown-routes ifup-ppp ifup-ppp network-functions
ifdown-eth ifdown-sit ifup-ipsec ifup-routes network-functions-ipv6
ifdown-ippv ifdown-sl ifup-ipv6 ifup-sit
ifdown-ipsec ifdown-tunnel ifup-ipv6 ifup-sl
ifdown-ipv6 ifup ifup-isdn ifup-tunnel
[root@localhost network-scripts]# cat ifcfg-eth0
# Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
DEVICE=eth0
BOOTPROTO=dhcp
HWADDR=00:0C:29:85:50:93
ONBOOT=yes
[root@localhost network-scripts]# ifcfg-lo
-bash: ifcfg-lo: command not found
[root@localhost network-scripts]# cat ifcfg-lo
DEVICE=lo
IPADDR=127.0.0.1
NETMASK=255.0.0.0
NETWORK=127.0.0.0
# If you're having problems with gated making 127.0.0.0/8 a martian,
# you can change this to something else (255.255.255.255, for example)
BROADCAST=127.255.255.255
ONBOOT=yes
NAME=loopback
[root@localhost network-scripts]#

```

图 11: 网络接口配置文件



```

root@localhost:/etc/sysconfig/network-scripts
文件(F) 编辑(E) 查看(V) 终端(T) 标签(B) 帮助(H)
[root@localhost network-scripts]# ifcfg-lo
-bash: ifcfg-lo: command not found
[root@localhost network-scripts]# cat ifcfg-lo
DEVICE=lo
IPADDR=127.0.0.1
NETMASK=255.0.0.0
NETWORK=127.0.0.0
# If you're having problems with gated making 127.0.0.0/8 a martian,
# you can change this to something else (255.255.255.255, for example)
BROADCAST=127.255.255.255
ONBOOT=yes
NAME=loopback
[root@localhost network-scripts]# ping localhost
PING localhost.localdomain (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=1 ttl=64 time=0.032 ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=2 ttl=64 time=0.030 ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=3 ttl=64 time=0.032 ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=4 ttl=64 time=0.033 ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=5 ttl=64 time=0.031 ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=6 ttl=64 time=0.034 ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=7 ttl=64 time=0.031 ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=8 ttl=64 time=0.033 ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=9 ttl=64 time=0.032 ms
64 bytes from localhost.localdomain (127.0.0.1): icmp_seq=10 ttl=64 time=0.033 m

```

图 12: ping 命令运行结果

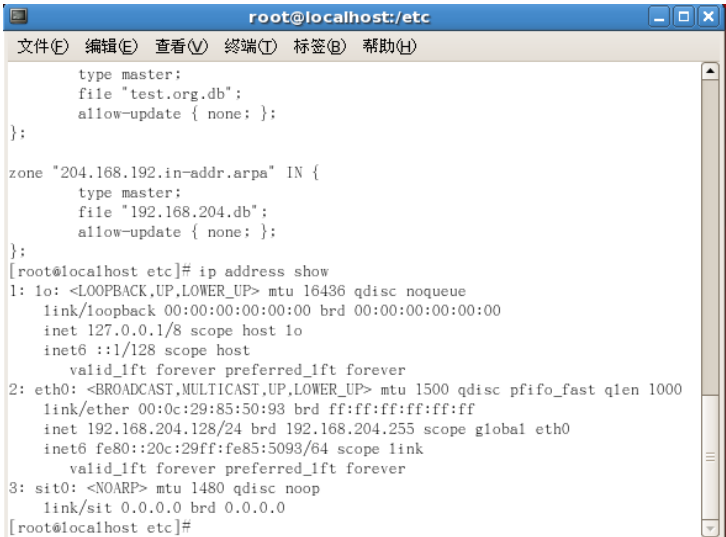


图 13: 主机所在地址和子网信息

3.4.2 NFS

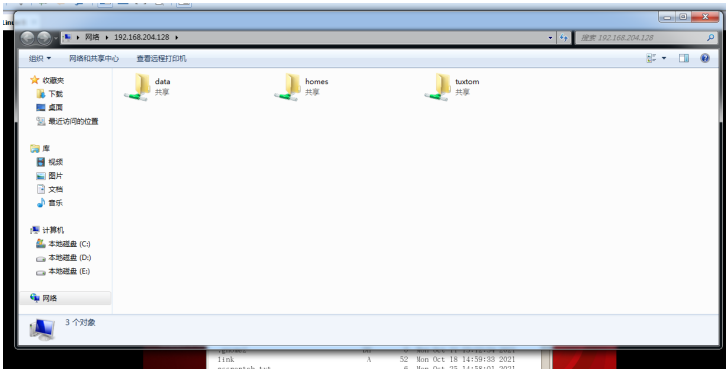


图 14: 连接到 windows

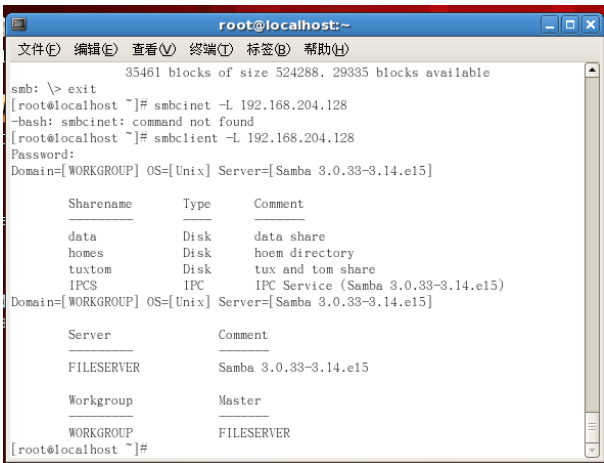


图 15: smbclient -L

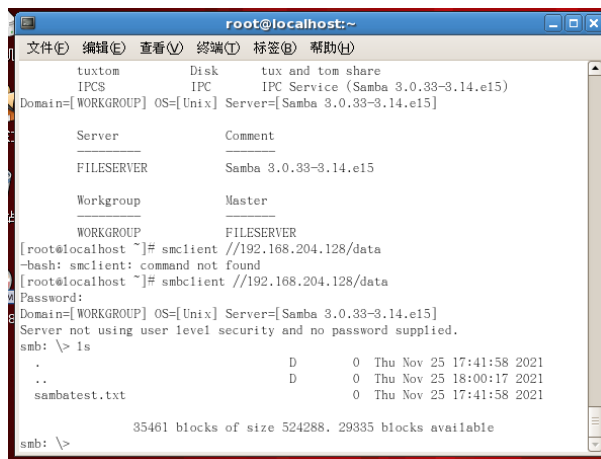
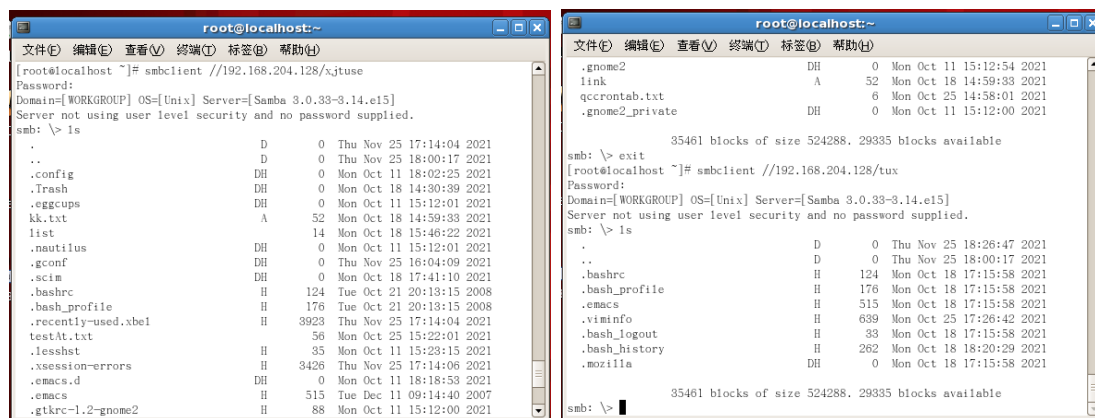
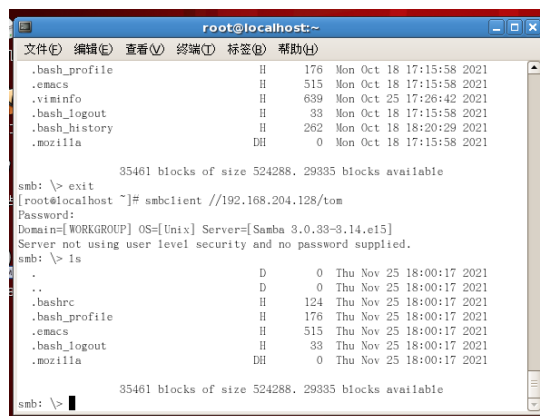


图 16: Data



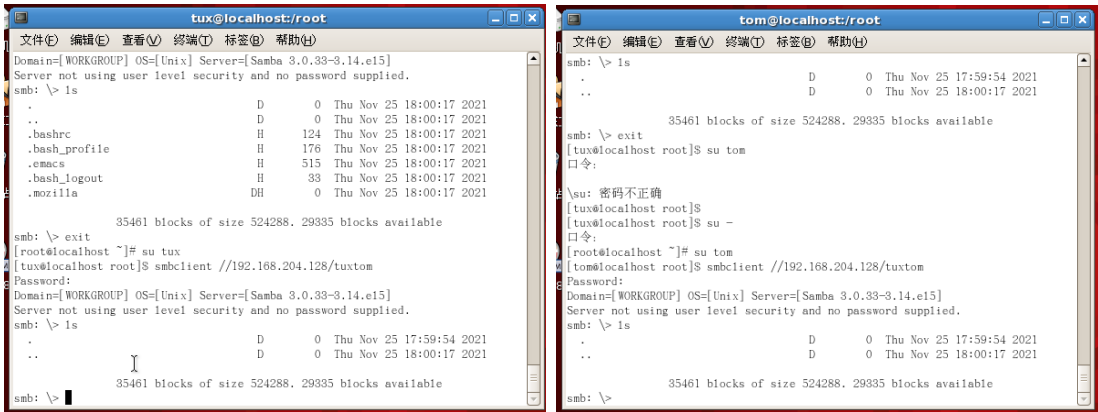
(a) xjtuse

(b) tux



(c) tom

图 17: homes



(a) tux

(b) tom

图 18: share

### 3.5 问题和解决方法

本次实验的过程是较为简单清晰的，注意一些细节上的问题，如文件路径不要出现输入错误或单词拼写错误等，我曾因为这些问题卡了好久。

### 3.6 实验体会

在实验一中，我们了解了 Linux 网络的相关基础知识，并进行了实践，加深了对理论知识的理解；了解到了 samba 的配置方法，并成功完成了实验的要求。

# 实验 4：综合训练

## 4.1 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。

## 4.2 实验内容

准备工作：利用虚拟机安装 Linux 操作系统，在系统中安装适当的软件包以备后续的实验需要，可关闭防火墙和 SeLinux。

### 1. 根据以下要求配置 Apache 服务器：

- (a) 设置 Web 页面的主目录为 `/var/www/web`;
- (b) 设置 Apache 监听的端口号为 8080;
- (c) 建立一个名为 temp 的虚拟目录，其对应的物理路径是 `/var/www/temp`，并对该虚拟目录启用用户认证，只允许用户 tux 和 lily 访问。
- (d) 允许每个用户拥有自己的个人主页。制作你的个人主页，并给出你的个人主页显示结果。（20 分）

### 2. 根据所学内容，使用一种或多种服务（如 ftp、samba、Http 等）搭建一台服务器，支持多用户访问，并能完成下述功能：（50 分）

- (a) 学生用户能实现学生作业的上传，学生以“姓名 + 学号”命名作业。学生能看到作业列表，但是不能下载其他用户的作业。
- (b) 在指定的交作业截止时间到时，编写脚本自动统计交作业的学生名单和人数，生成文档，供教师查看。
- (c) 教师用户能够查看学生提交的作业，在生成的交作业名单中录入成绩，并发布成绩，供学生查看。
- (d) 对于指定格式的作业，编写脚本自动批改作业并在交作业名单中记录成绩。（注：格式可以自己指定，也可做成模板供学生下载使用）
- (e) 教师可以提供课件和参考资料供学生下载。（注：可以根据题目要求，自己创建相应的用户、设定相关目录和权限）

3. 在 Linux 环境下编写 C 或 C++ 程序实现几种 CPU 调度算法: FCFS、SJF 和优先权调度。在 Linux 下进行编译和运行, 可使用 Makefile 文件实现程序的编译、安装和卸载。并比较这几种 CPU 调度算法的性能, 给出等待时间、周转时间及其平均值。(报告中给出源代码、Makefile 文件、make 运行结果以及程序运行结果)。实验所用测试数据如下表(算法均默认为非抢占)(30 分)

作业 ID	到达时间	执行时间	优先级
1	800	50	0
2	815	30	1
3	830	25	2
4	835	20	2
5	845	15	2
6	700	10	1
7	820	5	0

## 4.3 题目分析

### 4.3.1 Apache

1. 进行 Apache 的安装, 需要注意依赖软件的安装。
2. 找到 Apache 的配置文件, 进行相应的修改。
3. 更改配置文件的根目录为 /var/www/web/。
4. 更改监听的端口号为 8080。
5. 建立虚拟目录, 注意相关的语法。
6. 配置用户认证。
7. 个人主页的配置。
8. 以及 userdir 等的配置。
9. 修改文件的权限: `chmod -R 755`。
10. 在配置文件中写好的密码文件中, 利用 `htpasswd` 命令添加用户并设定密码
11. 启动 `httpd` 服务



### 4.3.2 收作业服务

本实验中我选择了 samba 完成。

1. 安装 samba, 注意依赖软件的安装。
2. 配置 smb.conf 将访问权限改为 user。
3. 配置 student 域, 用于提交作业, 并限制访问对象为 student 组和 teacher 组。
4. 配置 teacher 域, 用于发布成绩和学习资料, 并限制访问对象为 student 组和 teacher 组。
5. 使用 testparm 命令检查配置文件的格式。
6. 编写脚本完成作业情况的收集以及作业的批改。
7. 创建用户和用户组并设定 samba 密码。
8. 修改相关文件夹的权限。
9. 利用 crontab 设置脚本定时执行。
10. 启动 samba 服务

### 4.3.3 调度算法

调度算法在操作系统课程中已经有详尽的介绍, 经过复习后写出相关程序即可, 设计一个 PCB 的数据结构, 最后还要安装 g++ 并编写 make 文件

## 4.4 配置文件与运行情况

### 4.4.1 Apache

配置文件修改

```
#Listen 12.34.56.78:80
Listen 8080

#
# ServerName localhost:8080
```

图 19: 修改端口

```

<Directory "/var/www/web">
#
# Possible values for the Options directive are "None", "All",
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important. Please see
# http://httpd.apache.org/docs/2.2/mod/core.html#options
# for more information.
#
    Options Indexes FollowSymLinks

#
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
    AllowOverride None

#
# Controls who can get stuff from this server.
#
</Directory>

```

图 20: Web 页面主目录

```

Alias /temp/ "/var/www/temp/"
<Directory "/var/www/temp">
    AuthType Basic
    AuthUserFile /etc/httpd/mysecret.pwd
    AuthName "Please Login"
    Require user tux lily
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

<IfModule mod_userdir.c>
#
# UserDir is disabled by default since it can confirm the presence
# of a username on the system (depending on home directory
# permissions).
#
#
# To enable requests to ~/user/ to serve the user's public_html
# directory, remove the "UserDir disable" line above, and uncomment
# the following line instead:
#
    UserDir public_html
</IfModule>

```

图 21: 建立虚拟目录并配置用户认证

```

<Directory /home/*/public_html>
#
    AllowOverride FileInfo AuthConfig Limit
#
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
#
    <Limit GET POST OPTIONS>
#
        Order allow,deny
#
        Allow from all
#
    </Limit>
#
    <LimitExcept GET POST OPTIONS>
#
        Order deny,allow
#
        Deny from all
#
    </LimitExcept>

    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>

```

图 22: 个人主页配置

## 结果展示

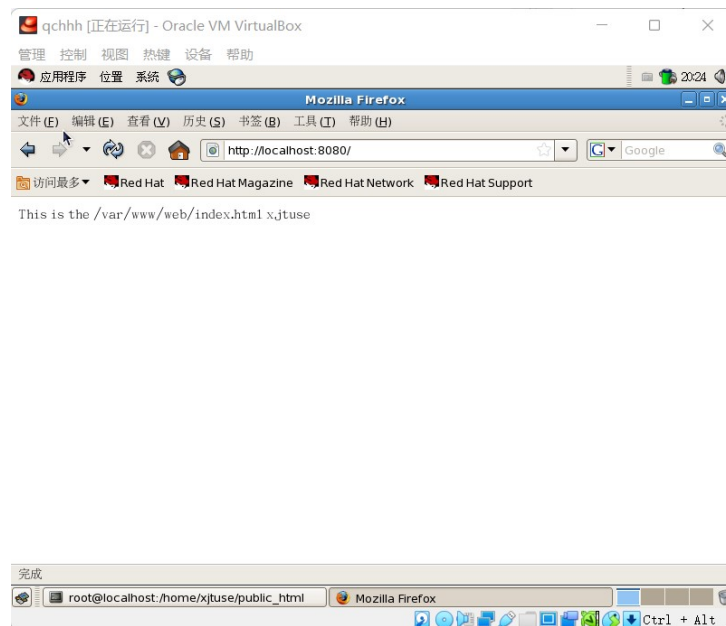


图 23: 主目录访问

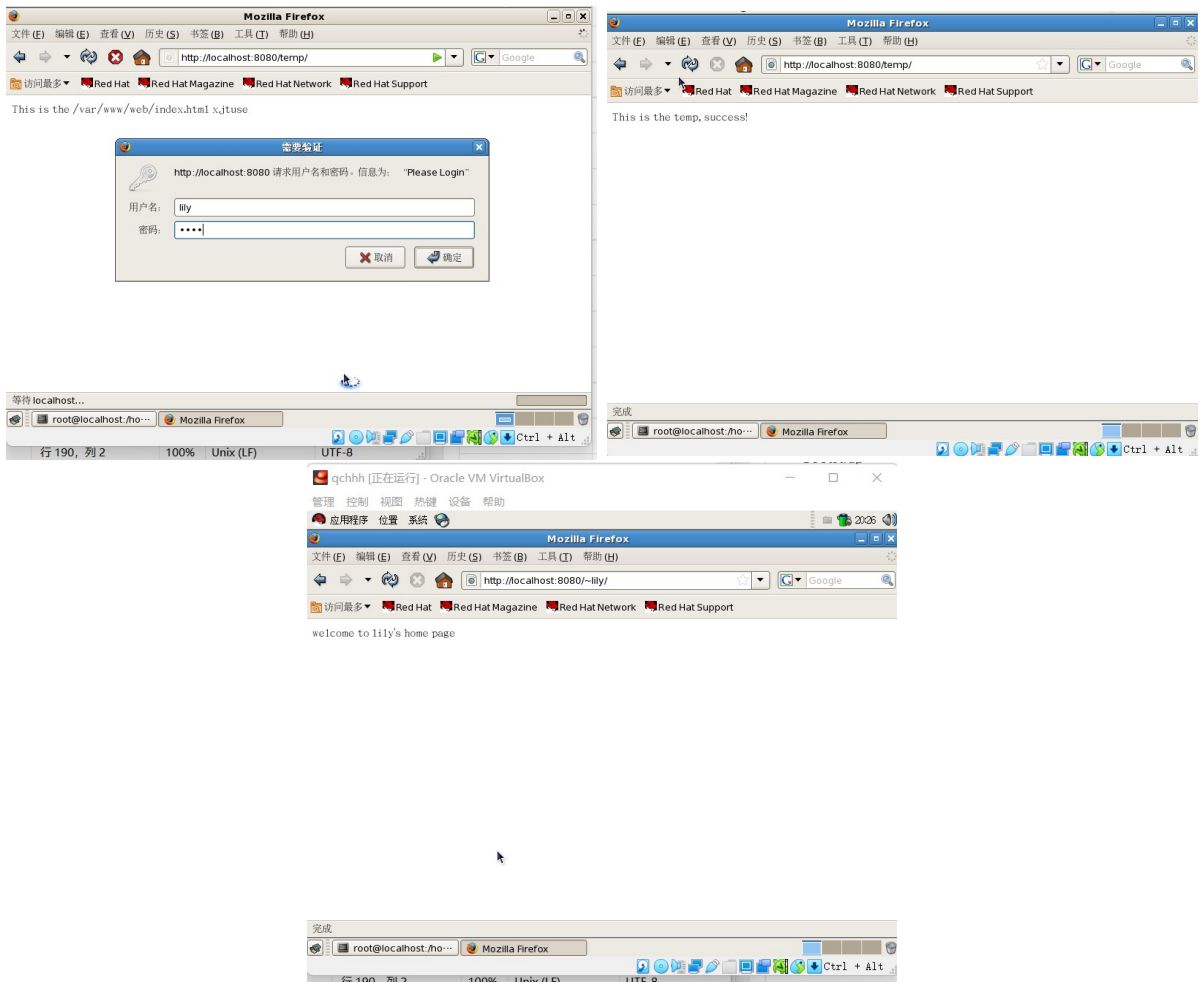


图 24: temp 访问

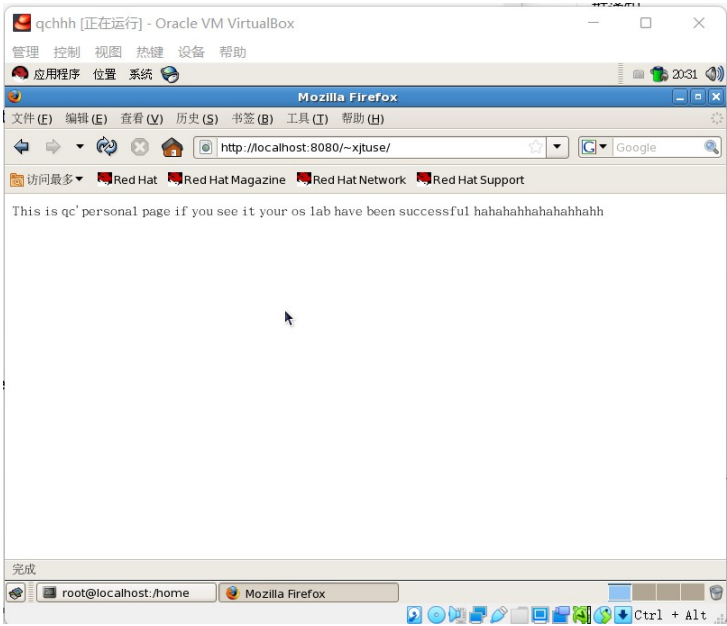


图 25: 个人主页访问

4.4.2 收发作业

配置文件修改

```
[student]
comment = "upload and check your homework"
path = /home/homework/student
writable = yes
public = yes
valid users = @student,@teacher
create mode = 0707
force create mode = 0707

[teacher]
comment = teacher
path = /home/homework/teacher
writable = yes
public = yes
valid users = @student,@teacher
create mode = 0704
force create mode = 0704
```

图 26: 通过 create mode 的设定完成学生不能下载其他学生的要求

```
[root@localhost ~]# groups ab
ab : student
[root@localhost ~]# groups cd
cd : student
[root@localhost ~]# groups teacher1
teacher1 : teacher
[root@localhost ~]#
```

图 27: 用户分组设定

```
#!/bin/bash
#Statistic the homework
path=/home/homework/student/
count=0
homeworks=$(ls $path)
declare -a ans=(1 2 3 4)
for hw in $homeworks
do
let count=$((count + 1))
done
echo "===== " >> statistic.txt
echo "The number of student who submitted homework was $count" >> statistic.txt
echo "===== " >> statistic.txt
echo "The list of student who submitted homework" >> statistic.txt
echo "===== " >> statistic.txt
for hw in $homeworks
do
newPath=$path$hw
if [ $(ls $newPath) = "xxx" ];
then
res=0
index=0
while read line
do
if [ $line = ${ans[$index]} ];
then
let res=$((res + 1))
fi
let index=$((index + 1))
done < $newPath
```

图 28: 作业统计脚本

```
let index=$((index + 1))
done < $newPath
echo "${hw%.*}: $res" >> statistic.txt
else
echo "${hw%.*}" >> statistic.txt
fi
```

图 29: 自动批改脚本

效果展示

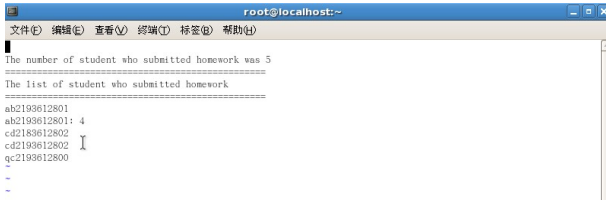


图 30: 提交作业名单和自动批改结果

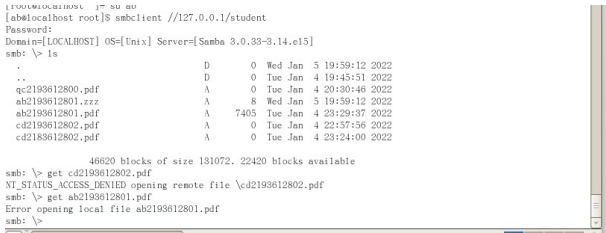


图 31: smbclient 访问 student

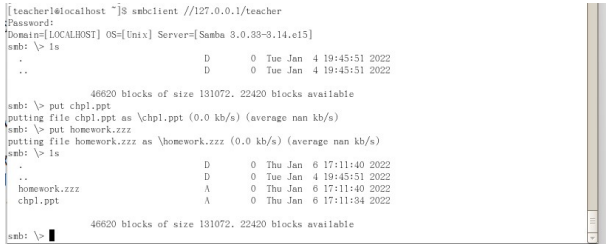


图 32: smbclient 访问 teacher

4.4.3 调度算法

配置文件



图 33: makefile

运行情况

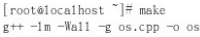


图 34: make 运行情况

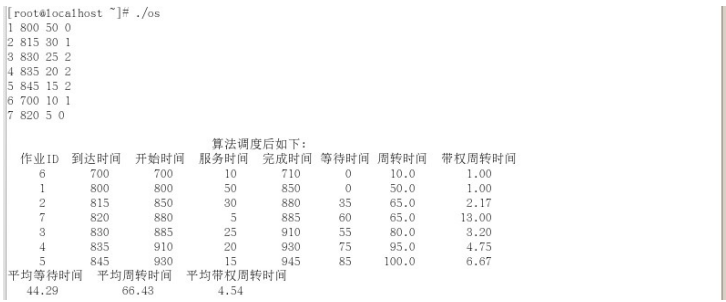


图 35: FCFS

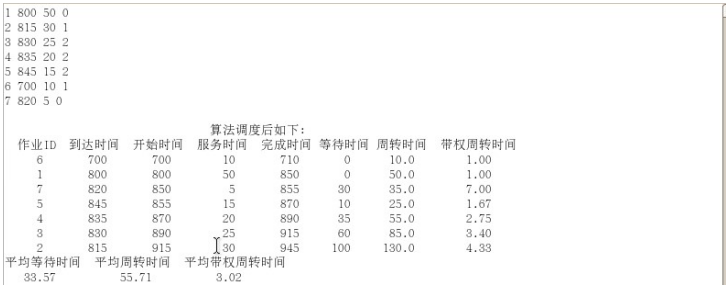


图 36: SJF

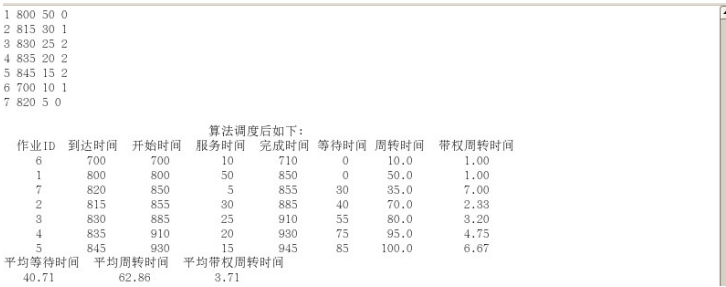


图 37: 优先权调度

## 4.5 脚本源程序清单

实验一无需脚本

实验二所需脚本如下:

Listing 3: static.sh

```
1 #!/bin/bash
2 #statistic the homework
3 #作业提交的路径
4 path=/home/homework/student/
5 #记录作业个数
```

```

6 count=0
7 #读取文件夹中的文件名称
8 homeworks=$(ls $path)
9 #利用数组记录自动批改作业的标准答案
10 declare -a ans=(1 2 3 4)
11 #开始循环读取文件夹下的作业以获得其总数
12 for hw in $homeworks; do
13     let count="$count + 1"
14 done
15 #在文件中输出已提交作业的份数
16 echo "===== " >>statistic.txt
17 echo "The number of student who submitted homework was $count" >>statistic.
    txt
18 echo "===== " >>statistic.txt
19 #开始输出学生名单
20 echo "The list of student who submitted homework" >>statistic.txt
21 echo "===== " >>statistic.txt
22 for hw in $homeworks; do
23     newPath=$path$hw
24     #对于文件格式为.zzz的作业进行自动批改
25     if [ ${hw:0-3} = "zzz" ]; then
26         res=0
27         index=0
28         while read line; do
29             #若结果匹配, 则分数加一
30             if [ $line = ${ans[$index]} ]; then
31                 let res="$res + 1"
32             fi
33             let index="$index + 1"
34         done <$newPath
35         #输出分数和学生名单
36         echo "${hw%.*}: $res" >>statistic.txt
37     else
38         #对于非自动批改格式的作业则直接输出名单
39         echo "${hw%.*}" >>statistic.txt
40     fi

```



41 done

实验三源码如下:

Listing 4: last.cpp

```

1  #include<stdlib.h>
2  #include<stdio.h>
3  #include<iostream>
4  #include<assert.h>
5
6  using namespace std;
7  #define TIME_SLICE 2
8
9  typedef struct PCB
10 {
11     char name[10]; //此为进程id
12     char state;    //进程状态w/r
13     int Arrivetime; //进程到达时间
14     int BeginTime; //进程开始时间
15     int FinishTime; //进程结束时间
16     int ServerTime; //进程服务时间
17     double waitTime; //等待时间
18     double Average_waitTime; //平均等待时间
19     float wholeTime; //周转时间
20     float Weight_wholeTime; //带权周转时间
21     double Average_wholeTime; //平均周转时间
22     double Average_weight_wholeTime; //带权平均周转时间
23
24     int RunTime; //已经占用cpu时间
25     int NeedTime; //还需要cpu时间
26
27     int Prio; //优先级
28     struct PCB* next=NULL;
29     struct PCB* next2=NULL;
30 }pcb, * Pcb;
31
32 int Proc_Num = 0; //进程数目
33

```

```

34 void Show(Pcb proc)
35 {
36     assert(proc != NULL);
37     double sum_waitTime = 0;
38     double sum_wholeTime = 0;
39     double sum_weight_wholetime = 0;
40     pcb* p = proc;
41     while (p != NULL)
42     {
43         sum_waitTime += p->FinishTime - p->Arrivetime - p->
            ServerTime;
44         sum_wholeTime += p->wholeTime;
45         sum_weight_wholetime += p->Weight_wholetime;
46         p = p->next;
47     }
48     double Average_waitTime = sum_waitTime / Proc_Num;
49     double Average_wholeTim = sum_wholeTime / Proc_Num;
50     double Average_weight_wholetime = sum_weight_wholetime / Proc_Num;
51     printf("  作业ID  到达时间  开始时间  服务时间  完成时间  等待时间
        周转时间  带权周转时间\n");
52     while (proc != NULL)
53     {
54         printf("%6s    %6d    %6d    %6d    %6d %6d %8.1f    %8.2
            f\n",
55             proc->name,
56             proc->Arrivetime, proc->BeginTime, proc->ServerTime
            ,
57             proc->FinishTime, proc->FinishTime - proc->
                Arrivetime - proc->ServerTime, proc->wholeTime,
                proc->Weight_wholetime);
58         proc = proc->next;
59     }
60     printf("平均等待时间  平均周转时间  平均带权周转时间  \n");
61     printf("    %.2f    %.2f    %.2f\n", Average_waitTime,
        Average_wholeTim,
62         Average_weight_wholetime);

```

```

63 }
64 Pcb PCB_Create()//创建输入链表
65 {
66     Proc_Num = 7;
67     pcb* _head = NULL;
68     pcb* _tail = NULL;
69     if (Proc_Num > 6000) return NULL;
70     for (int i = 1; i <= Proc_Num; i++){
71         pcb* new_proc = (pcb*)malloc(sizeof(pcb));
72         assert(NULL != new_proc);
73         cin >> new_proc->name >> new_proc->Arrivetime >> new_proc->
            ServerTime >> new_proc->Prio;
74         new_proc->next = NULL;
75         if (NULL == _head){
76             _tail = new_proc;
77             _head = new_proc;
78         }
79         else{
80             _tail->next = new_proc;
81             _tail = new_proc;
82         }
83     }
84 }
85 return _head;
86 }
87 Pcb Sort_Arrivetime(Pcb list)//FCFS排序
88 {
89     assert(NULL != list);
90     pcb* new_head = (pcb*)malloc(sizeof(pcb));
91     assert(NULL != new_head);
92     new_head->Arrivetime = 0;
93     new_head->ServerTime = 0;
94     new_head->next = NULL;
95     pcb* head = NULL;
96     while (list != NULL){
97         pcb* cur = list;

```

```

98         list = list->next;
99         cur->next = NULL;
100
101         if (new_head->next == NULL)
102             new_head->next = cur;
103         else{
104             pcb* ptr = new_head;
105             for (ptr; ptr->next != NULL; ptr = ptr->next);
106
107             if (cur->Arrivetime >= ptr->Arrivetime)
108                 ptr->next = cur;
109
110             else{
111                 pcb* p = new_head;
112                 while (cur->Arrivetime > p->next->
113                     Arrivetime)
114                     p = p->next;
115
116                 cur->next = p->next;
117                 p->next = cur;
118             }
119         }
120     return new_head->next;
121 }
122 bool write(Pcb list) {
123     while (list != NULL) {
124         if (list->state != 'W') {
125             return true;
126         }
127         list = list->next;
128     }
129     return false;
130 }
131 Pcb SortSJF(Pcb list) {
132     pcb* head = Sort_Arrivetime(list);

```

```

133     pcb* fhead = head;
134     head->state = 'W';
135     int end_time = head->Arrivetime + head->ServerTime;
136     pcb* temp = head->next;
137     while (write (fhead)) {
138         int flag = 0;
139         pcb* temp2 = temp;
140         pcb* temp3 = temp;
141         while (temp!= NULL) {
142             if (end_time >= temp->Arrivetime&&temp->state!='W')
143             {
144                 if (flag == 0) {
145                     temp2 = temp;
146                     temp3 = temp;
147                     flag++;
148                 }
149                 else {
150                     temp2->next2 = temp;
151                     temp2 = temp2->next2;
152                 }
153             }
154             temp = temp->next;
155         }
156         temp2->next2 = NULL;
157         pcb* min = temp3;
158         while (temp3 != NULL) {
159             if (temp3->ServerTime < min->ServerTime) {
160                 min = temp3;
161             }
162             temp3 = temp3->next2;
163         }
164         min->state = 'W';
165         pcb* temp4 = head->next;
166         if (temp4 != min) {
167             pcb* temp5 = min->next;
168             head->next = min;

```

```

168         min->next = temp4;
169         pcb* temp6 = temp4;
170         if (temp6 == NULL) {
171             return fhead;
172         }
173         while (temp6->next != NULL) {
174             if (temp6->next == min) {
175                 temp6->next = temp5;
176             }
177             temp6 = temp6->next;
178             if (temp6 == NULL) break;
179         }
180     }
181     temp = head->next;
182     temp = temp->next;
183     head = head->next;
184     if (head->Arrivetime < end_time) {
185         end_time = end_time + head->ServerTime;
186     }
187     else {
188         end_time = head->Arrivetime + head->ServerTime;
189     }
190     temp = head->next;
191 }
192 return fhead;
193 }
194 Pcb SortPrc(Pcb list) {
195     pcb* head = Sort_Arrivetime(list);
196     pcb* fhead = head;
197     head->state = 'W';
198     int end_time = head->Arrivetime + head->ServerTime;
199     pcb* temp = head->next;
200     while (write(fhead)) {
201         int flag = 0;
202         pcb* temp2 = temp;
203         pcb* temp3 = temp;

```

```

204         while (temp != NULL) {
205             if (end_time >= temp->Arrivetime && temp->state !=
                'W') {
206                 if (flag == 0) {
207                     temp2 = temp;
208                     temp3 = temp;
209                     flag++;
210                 }
211                 else {
212                     temp2->next2 = temp;
213                     temp2 = temp2->next2;
214                 }
215             }
216             temp = temp->next;
217         }
218         temp2->next2 = NULL;
219         pcb* min = temp3;
220         while (temp3 != NULL) {
221             if (temp3->Prio < min->Prio) {
222                 min = temp3;
223             }
224             temp3 = temp3->next2;
225         }
226         min->state = 'W';
227         pcb* temp4 = head->next;
228         if (temp4 != min) {
229             pcb* temp5 = min->next;
230             head->next = min;
231             min->next = temp4;
232             pcb* temp6 = temp4;
233             if (temp6 == NULL) {
234                 return fhead;
235             }
236             while (temp6->next != NULL) {
237                 if (temp6->next == min) {
238                     temp6->next = temp5;

```

```

239         }
240         temp6 = temp6->next;
241         if (temp6 == NULL) break;
242     }
243 }
244     temp = head->next;
245     temp = temp->next;
246     head = head->next;
247     if (head->Arrivetime < end_time) {
248         end_time = end_time + head->ServerTime;
249     }
250     else {
251         end_time = head->Arrivetime + head->ServerTime;
252     }
253     temp = head->next;
254 }
255 return fhead;
256 }
257
258 Pcb End_list(Pcb plist) {//最终链表
259     assert(NULL != plist);
260     int begin_time = plist->Arrivetime;
261     plist->BeginTime = begin_time;
262     int end_time = begin_time + plist->ServerTime;
263     plist->FinishTime = end_time;
264     plist->wholeTime = (float)(plist->FinishTime - plist->Arrivetime);
265     plist->Weight_wholetime = (float)(plist->wholeTime / plist->
        ServerTime);
266     plist->state = 'W';
267     plist->RunTime = 0;
268     pcb* ptr = plist->next;
269     while (ptr != NULL)
270     {
271         if (ptr->Arrivetime <= end_time) {
272             ptr->BeginTime = end_time;
273             ptr->FinishTime = end_time + ptr->ServerTime;

```



```

274         end_time += ptr->ServerTime;
275     }
276     else {
277         ptr->BeginTime = ptr->Arrivetime;
278         ptr->FinishTime = ptr->Arrivetime + ptr->ServerTime
                ;
279         end_time = ptr->FinishTime;
280     }
281     ptr->wholeTime = (float)(ptr->FinishTime - ptr->Arrivetime)
                ;
282     ptr->Weight_wholetime = (float)(ptr->wholeTime / ptr->
                ServerTime);
283     ptr->state = 'W';
284     ptr->RunTime = 0;
285     ptr = ptr->next;
286 }
287
288 return plist;
289 }
290
291
292 void FCFS()//先来先服务
293 {
294     pcb* head = PCB_Create();
295     printf("\t\t\t\t算法调度后如下:\n");
296     pcb* end_head = Sort_Arrivetime(head);
297     struct PCB* list = End_list(end_head);
298     Show(list);
299 }
300 void SJF()
301 {
302     pcb* head = PCB_Create();
303     printf("\t\t\t\t算法调度后如下:\n");
304     pcb* end = SortSJF(head);// Sort_SJF(head);
305     pcb* list = End_list(end);
306     Show(list);

```

```
307 }
308 void PrioCreate()
309 {
310     pcb* head = PCB_Create();
311     printf("\t\t\t\t算法调度后如下:\n");
312     pcb* end = SortPrc(head);
313     pcb* list = End_list(end);
314     Show(list);
315 }
316
317 int main()
318 {
319     FCFS();
320     SJF();
321     PrioCreate();
322     return 0;
323 }
```

## 4.6 问题和解决方法

### 1. 在实验一的过程中:

- 出现 forbidden 页面: 通过更改文件夹以及文件的权限得以改正。
- 用户认证配置不成功: 注意 userdir 的修改以及浏览器的缓存等问题。

### 2. 在实验二的过程中:

- 对于用户下载的限制: 通过 create mode 来实现的, 同时在运行之前修改相关文件夹和文件的权限。
- 脚本的编写过程中发现 while read line 使用 read 会不生效, 所以将代码结构进行了改进。

## 4.7 实验体会

1. 实验一深化了我对于 apache 的了解, 并将 apache 配置的相关作用并将其赋予了实际作用。在这个过程中深化了我对于 Linux 系统的使用和网络相关知识的理解。

2. 实验二加强了我关于 samba 服务的配置相关的知识, 提升了我解决问题的能力。同时提升了我对于 shell 脚本的理解以及实际操作的能力。
3. 实验三加强了我对于 c++ 程序的编写能力尤其是结构体和指针方面, 同时了解了各种 CPU 调度算法。并成功使用了 Makefile。