实验题目：**Linux**操作系统基本命令

姓名：王启元 学号：19122925 实验日期：2021年9月9日

**Linux操作系统基本命令**

**实验环境：**

Linux环境：虚拟机：Ubuntu 16.04.3

**实验目的:**

1. 了解Linux运行环境，熟悉交互式分时系统、多用户环境的的运行机制。

2. 练习Linux系统命令接口的使用，学会Linux基本命令、后台命令、管道命令等命令

的操作要点。

**实验内容：**

通过终端或虚拟终端，在基于字符的交互界面中进行Shell的基本命令的操作。

**操作过程：**

登录进入Linux命令操作界面。执行以下各类命令，熟悉Linux用户命令接口。

**1 查看信息**

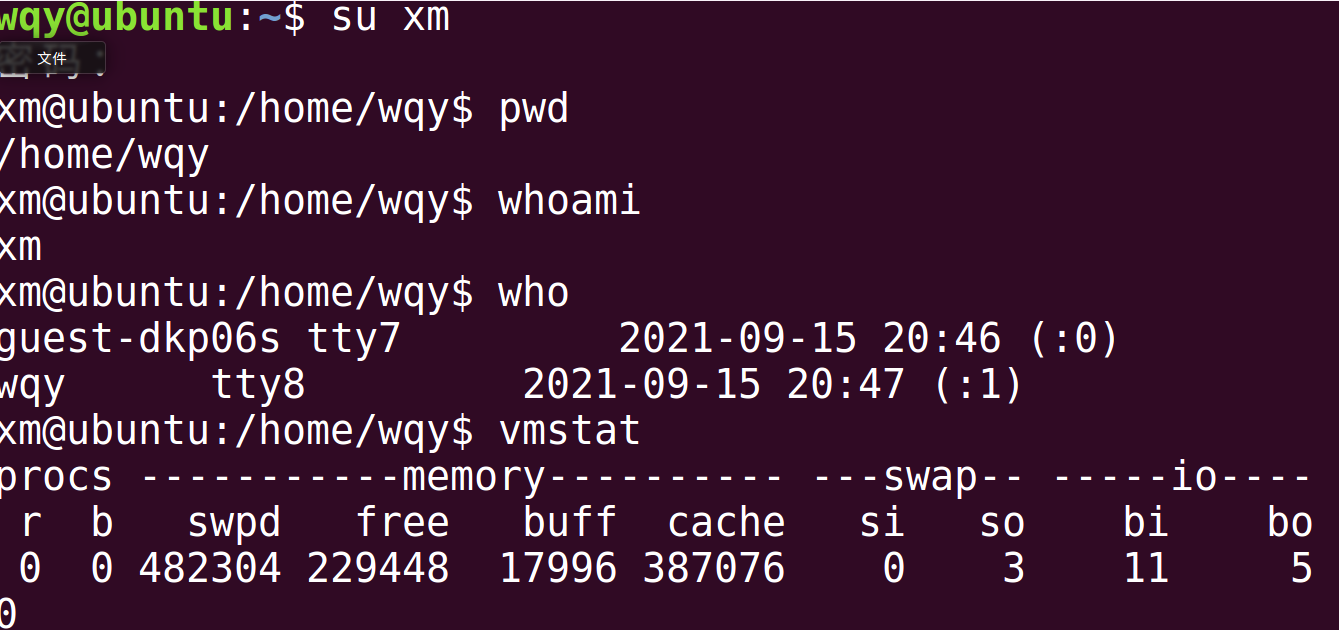
操作：

① 执行pwd查看当前目录。

② 用who am i看看当前用户信息。

③ 通过who看看有谁在系统中。

④ 用vmstat显示系统状态。



**2 文件操作**

操作：

① 执行

cat > mytext.txt

通过键盘输入一些信息，用ctrl+c结束，建立文件mytext.txt。

② 执行

cat mytext.txt

显示文件内容。

③ 执行`

ln mytext.txt mytext2.dat

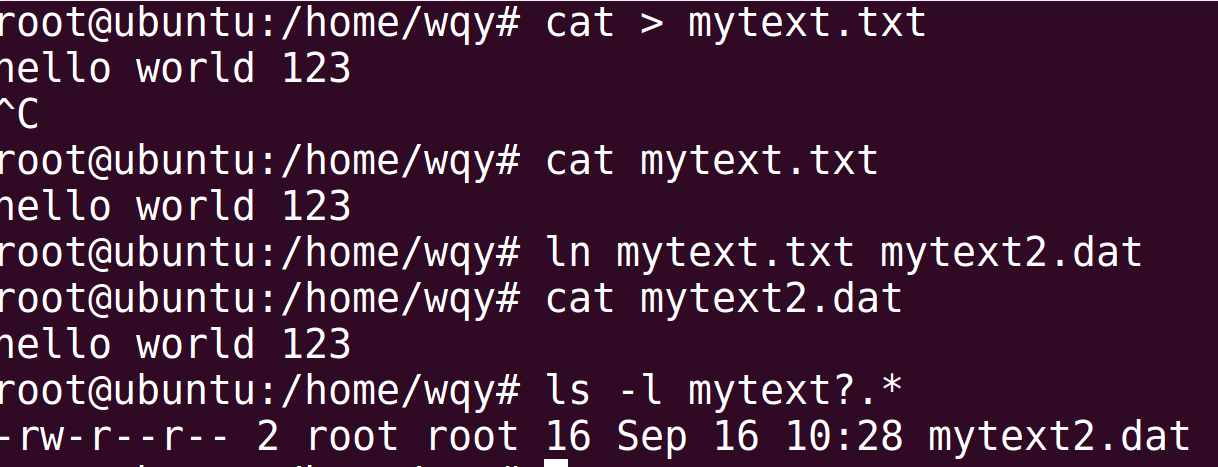
cat mytext2.dat

建立链接

④ 执行

ls –l mytext?.\*

显示文件目录，并注意i节点号，链接计数。



**3 目录操作**

① 执行

ls –l

看看当前目录的内容，请特别注意文件类型、文件的存取控制权限、i节点号、文件

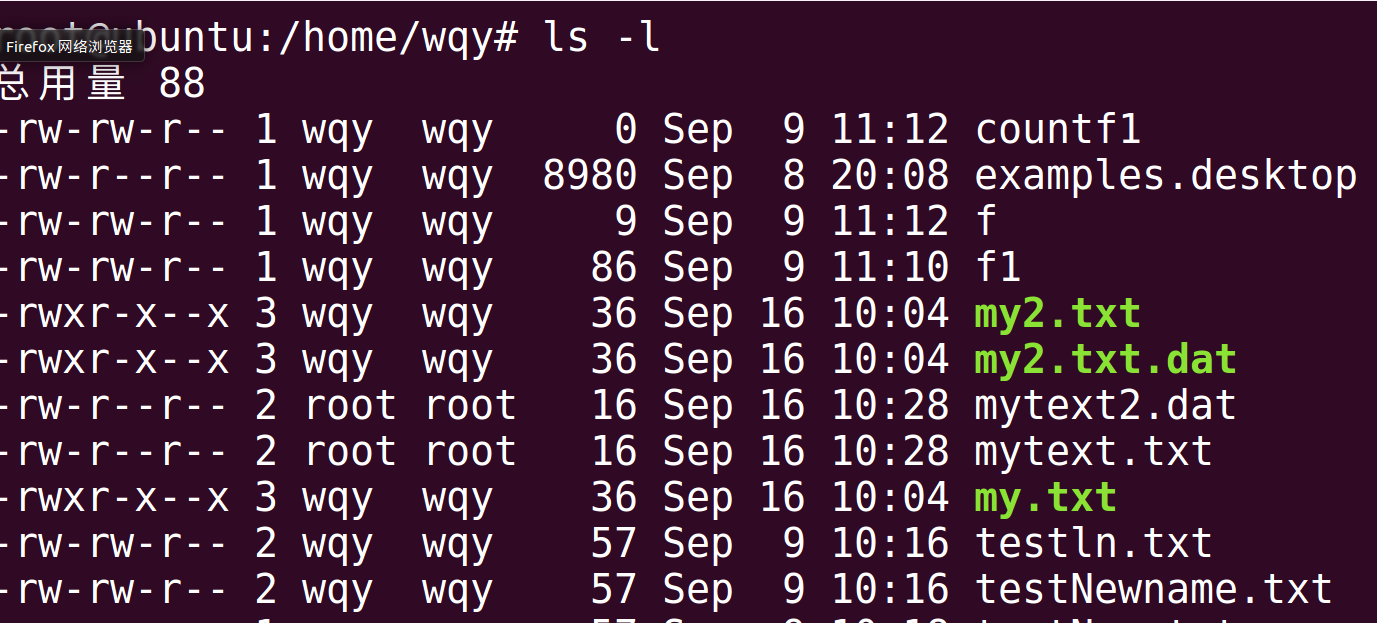
属主、文件属组、文件大小、建立日期等信息。

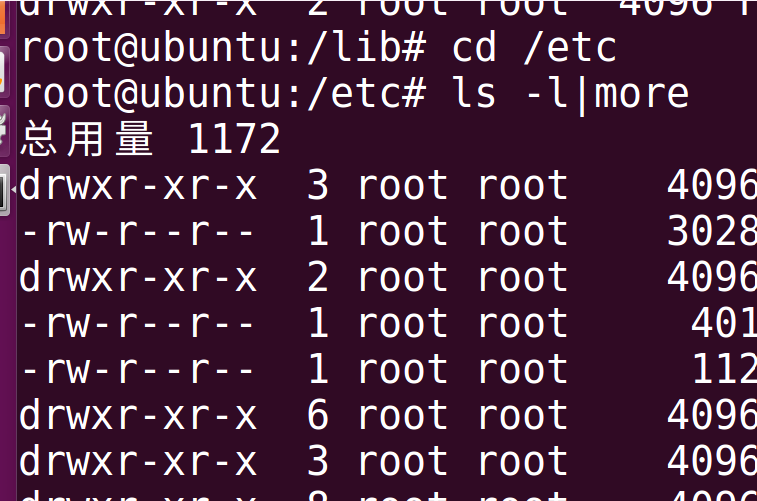
② 执行

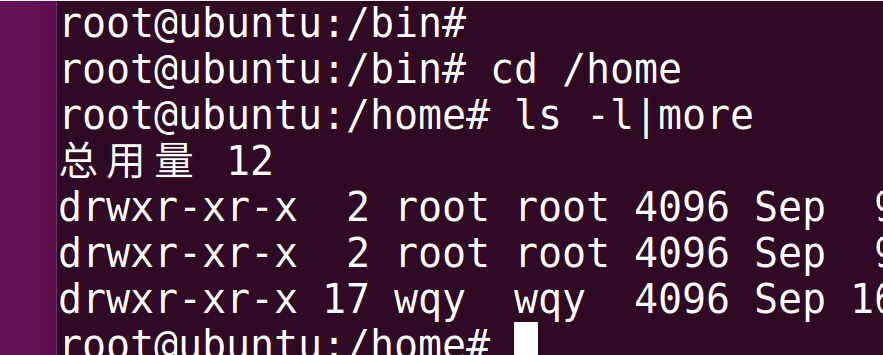
cd /lib ls –l|more

看看/lib目录的内容，这里都是系统函数。再看看/etc，这里都是系统配置用的数据

文件；/bin中是可执行程序；/home下包括了每个用户主目录。







**4 修改文件属性**

操作：

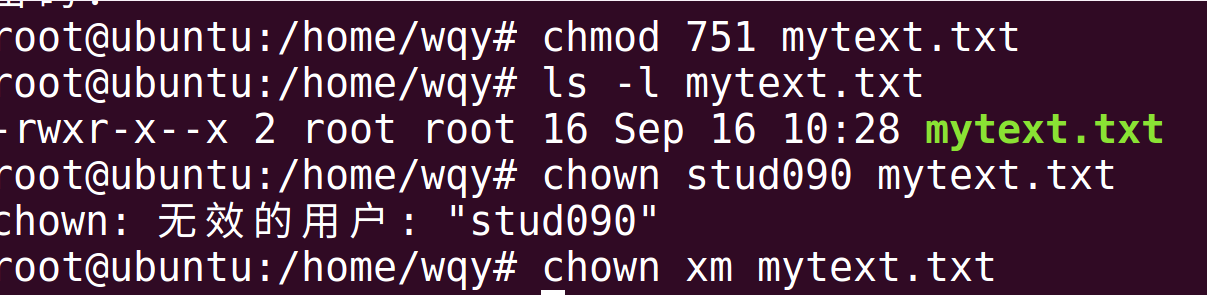
① 执行

chmod 751 mytext.txt

ls –l mytext.txt

② 执行

chown stud090 mytext.txt

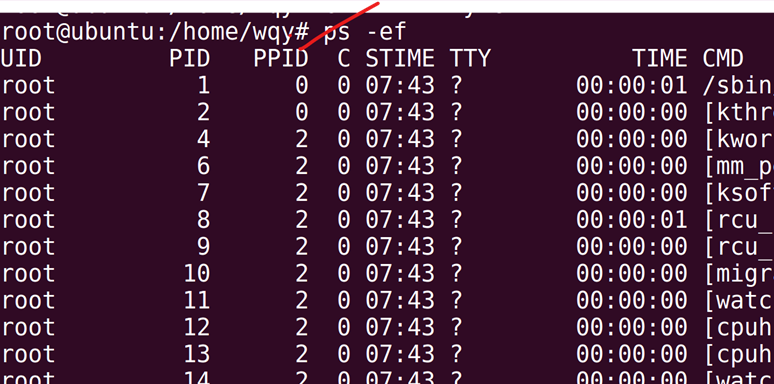


**5 进程管理**

① 执行

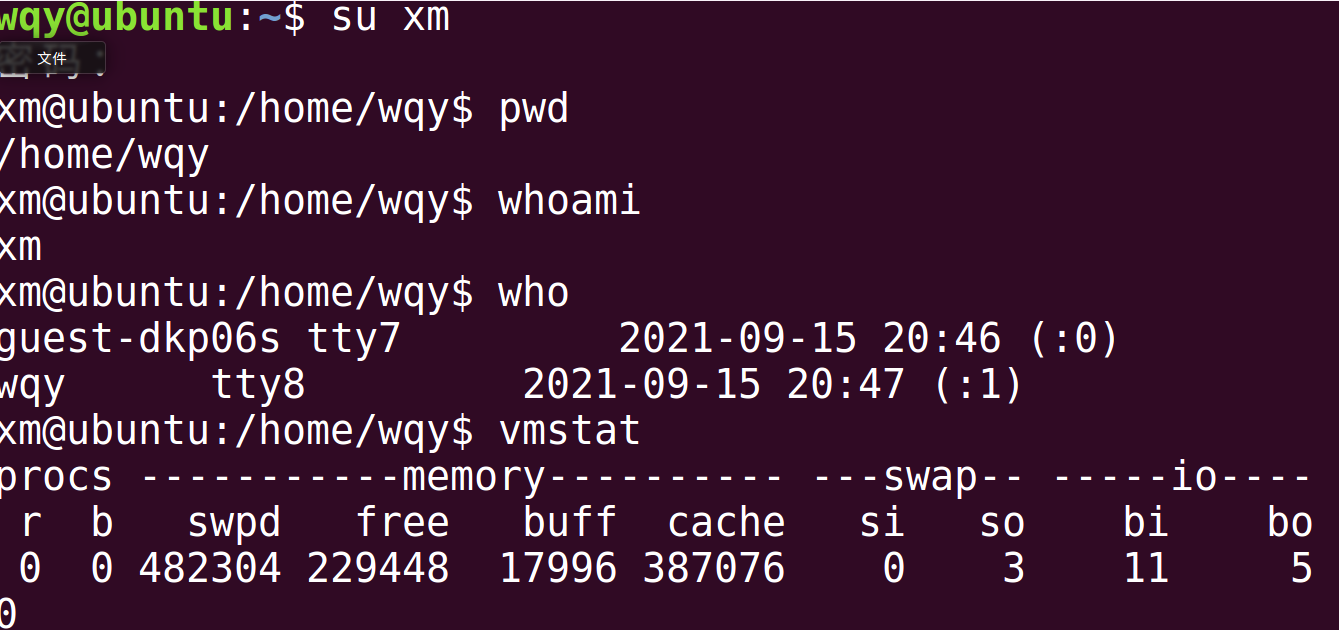
ps –ef

② 执行wait和sleep命令

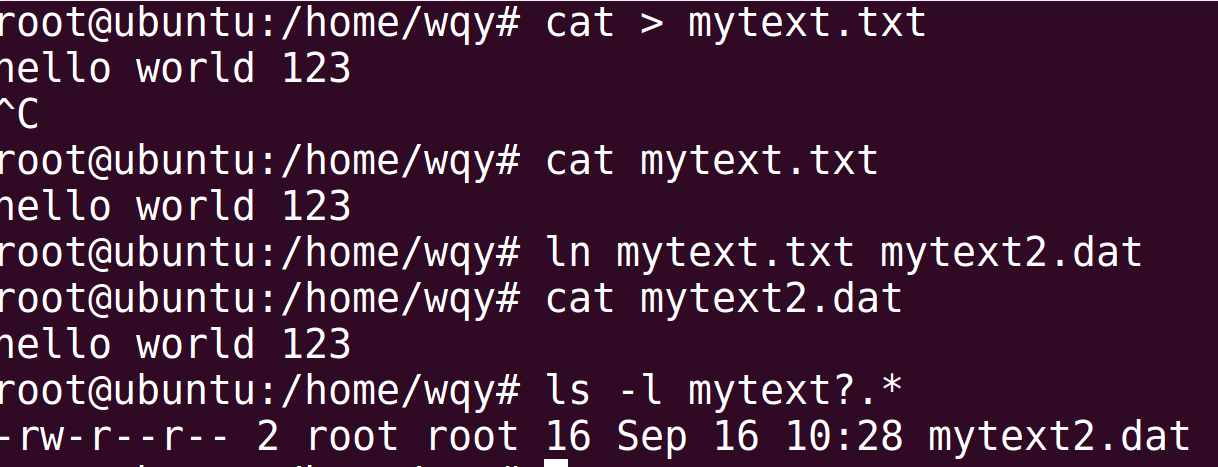


**结果：**

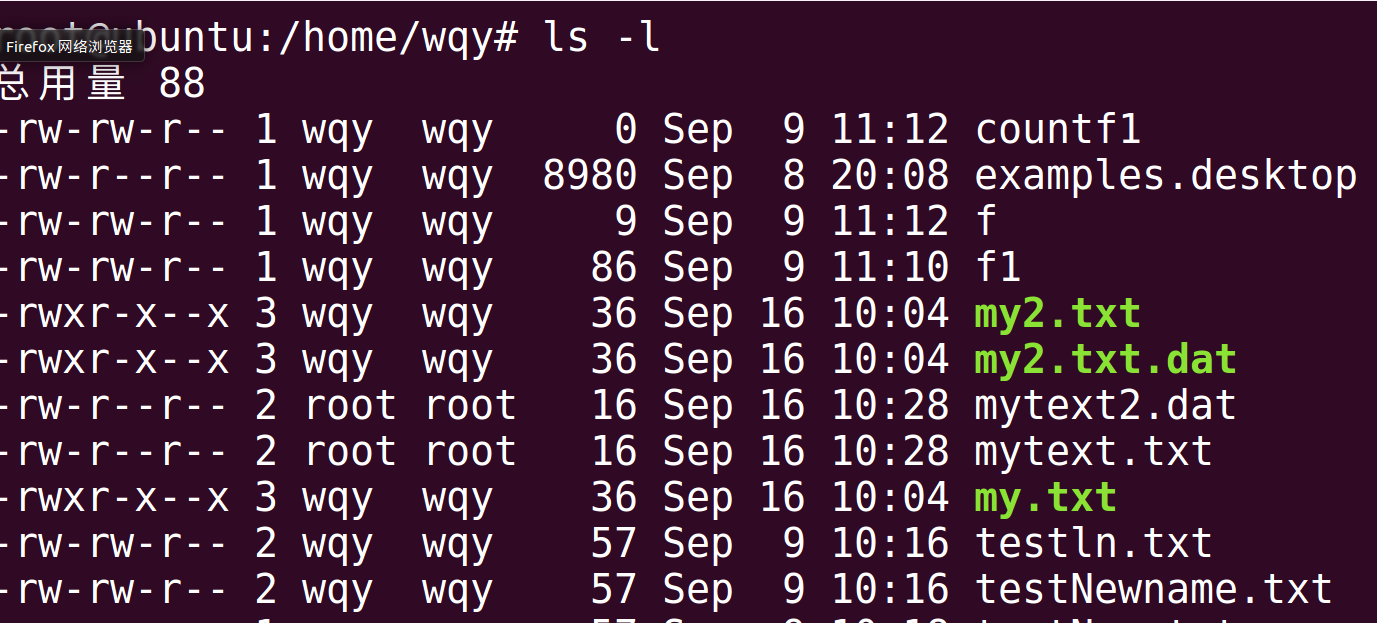
**1 查看信息**

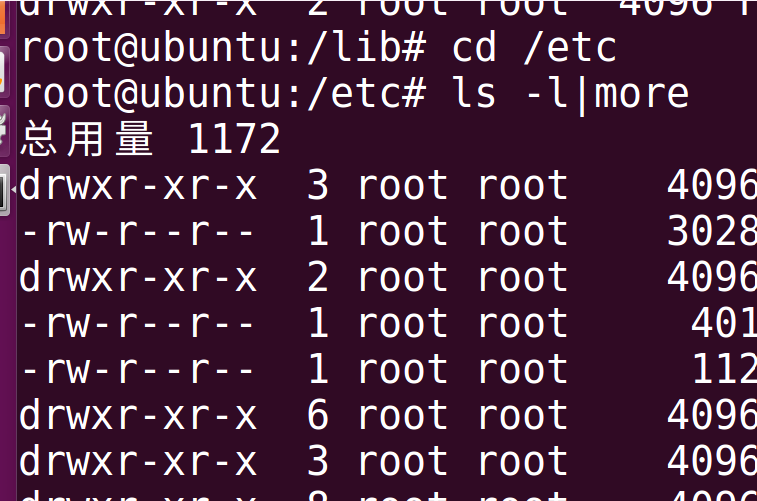


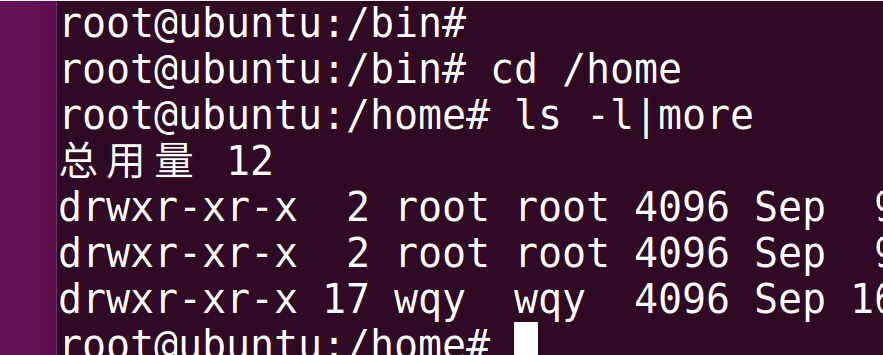
**2 文件操作**



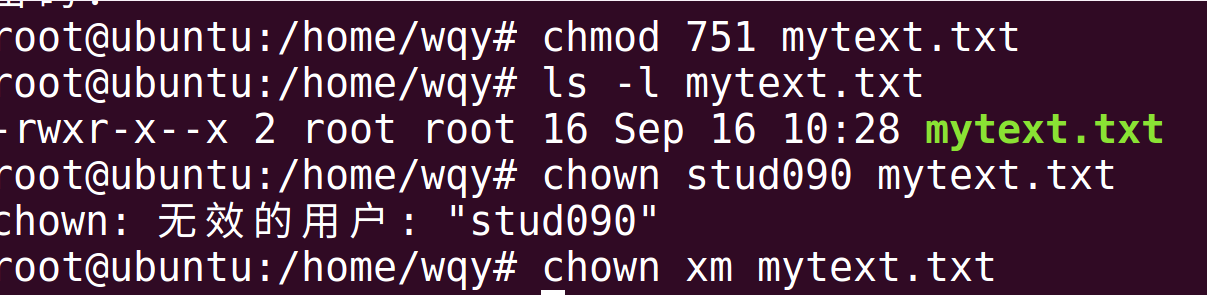
**3 目录操作**



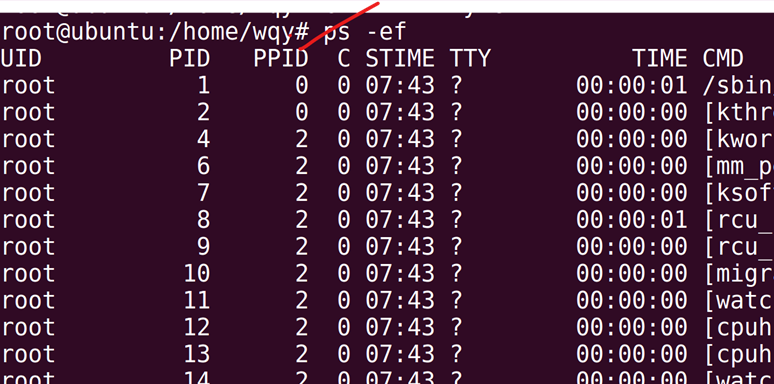




**4 修改文件属性**



**5 进程管理**



**体会：**

linux 是一个开源、免费的操作系统，其稳定性、安全性、处理多并发已经得到业界的认可，目前很多中型，大型甚至是集群项目都在使用 linux,很多软件公司考虑到开发成本都首选 linux,在中国软件公司得到广泛的使用。在这次的实验中，我了解了Linux的基本框架，学习Linux的细节操作，Linux是一门在玩中学，在学中玩的学科，实践很重要，这堂课同时为下次学习提供了基础。

**附录：（源程序）**

**1 查看信息**

pwd

whoami

who

vmstat

**2 文件操作**

cat > mytext.txt

hello world 123

^c

cat mytext.txt

ln mytext.txt mytext2.dat

cat mytext2.dat

ls –l mytext?.\*

**3 目录操作**

ls –l

cd /lib

ls –l|more

cd /etc

ls -l|more

cd /bin

ls -l|more

cd /home

ls -l|more

**4 修改文件属性**

chmod 751 mytext.txt

ls –l mytext.txt

chown xm ,mytext.txt

**5 进程管理**

ps –ef

wait 6

sleep 6

实验题目：**用户界面与 Shell 命令**

姓名：王启元 学号：19122925 实验日期：2021年9月9日

**用户界面与 Shell 命令**

**实验环境：**

Linux环境：虚拟机：Ubuntu 16.04.3

**实验目的:**

（1）掌握图形化用户界面和字符界面下使用 Shell 命令的方法。

（2）掌握 ls、cd 等 Shell 命令的功能。

（3）掌握重定向、管道、通配符、历史记录等的使用方法。

（4）掌握手工启动图形化用户界面的设置方法。的操作要点。

**实验内容：**

通过终端或虚拟终端，在基于字符的交互界面中进行Shell和图形界面基本命令的操作。

**操作过程：**

**1.图形化用户界面下的 Shell 命令操作**

【操作要求 1】显示系统时间，并将系统时间修改为 2011 年 9 月 17 日零点。

【操作步骤】

（1）启动计算机，以超级用户身份登录图形化用户界面。

（2）依次单击顶部面板的「应用程序」菜单=>「附件」=>「终端」，打开桌面环境下的终端工具。

（3）输入命令“date”，显示系统的当前日期和时间。

（4）输入命令“date091700002011”，屏幕显示新修改的系统时间。在桌面环境的终端中执行时显示中文提示信息。

【操作要求 2】切换为普通用户，查看 2011 年 9 月 17 日是星期几。

【操作步骤】

（1）前一操作是以超级用户身份进行的，但通常情况下只有在必须使用超级用户权限的 时候，才以超级用户身份操作。为提高操作安全性，输入“su - helen”命令切换为普通用户 helen。

（2）输入命令“cal 2011”，屏幕上显示出 2011 年的日历，由此可知 2011 年 9 月 17 日是星期日。

【操作要求 3】查看 ls 命令的-s 选项的帮助信息

图 2-2 查看日历

【操作步骤】

方法一：

（1）输入 “man ls” 命令，屏幕显示出手册页中 ls 命令相关帮助信息的第一页，介绍

ls 命令的含义、语法结构以及-a、-A、-b 和-B 等选项的意义。

（2）使用 PgDn 键、PgUp 键以及上、下方向键找到-s 选项的说明信息。

（3）由此可知，ls 命令的-s 选项等同于--size 选项，以文件块为单位显示文件和目录的大小。

（4）在屏幕上的“：”后输入“q”，退 出 ls 命令的手册页帮助信息。

方法二：

（1）输入命令“ls --help”，屏幕显示中文的帮助信息。

（2）拖动滚动条，找到-s 选项的说明信息，由此可知 ls 命令的-s 选项等同于--size 选项， 以文件块为单位列出所有文件的大小，如图 2-3 所示。

（3）在屏幕上的“：”后输入“q”，退 出 ls 命令的手册页帮助信息。

【操作要求 4】查看/etc 目录下所有文件和子目录的详细信息

【操作步骤】

（1）输入命令“cd /etc”，切换到/etc 目录。

（2）输入命令“ls -al”，显示/etc 目录下所有文件和子目录的详细信息。

**2.字符界面下的 Shell 命令操作**

包括 RHEL Server 5 在内的 Linux 系统都具有虚拟终端。虚拟终端为用户提供多个互不干扰、独立工作的工作界面，并且在不同的工作界面可用不同的用户身份登录。也就是说虽然用户只面对一个显示器，但可以切换到多个虚拟终端，好像在使用多个显示器。

RHEL Server 5 具有 7 个虚拟终端，其中第 1 个~第 6 个为字符界面；而第 7 个为图形化用户界面，必须启动图形化用户界面时才存在。各虚拟终端的切换方法为：

从字符界面的虚拟终端到其他虚拟终端：ALT+F1~ ALT+F7

从图形化用户界面到字符界面：CTRL+ALT+F1~CTRL+ALT+F6

【操作要求 1】查看当前目录。

【操作步骤】

（1）启动计算机后默认会启动图形化用户界面，按下 CTRL+ALT+F1 键切换到第 1 个虚拟终端。

（2）输入一个普通用户的用户名（helen）和口令，登录系统。

字符界面下输入口令时，屏幕上不会出现类似“\*”的信息，提高了口令的安全性。

（3）输入命令“pwd”，显示当前目录，相关操作参见如下内容。

Red Hat Enterprise Linux Server release 5 (Tikanga)

Kernel 2.6.18-8.el5 on an i386

localhost login : helen Password :

Last login: Tue Nov 20 09:28:42 on tty1 pwd

/home/helen

虚拟终端未登录时显示的第一行信息表示当前使用的 Linux 的发行版本是 Red Hat Enterprise Linux Server，版本号为 5，又名 Tikanga。第二行信息显示 Linux 内核版本是2.6.18-8.el5，以及本机的 CPU 型号是 i686。（Linux 将 Intel 奔腾以上级别的 CPU 都表示为 i686。）第三行信息显示本机默认的主机名 localhost。

成功登录系统后，还会显示该用户帐号上次登录系统的时间以及登录的终端号。

【操作要求 2】用 cat 命令在用户主目录下创建一名为 f1 的文本文件，内容为：

Linux is useful for us all.

You can never imgaine how great it is.

【操作步骤】

（1）输入命令“cat >f1”，屏幕上输入点光标闪烁，依次输入上述内容。

使用 cat 命令进行输入时，不能使用左右上下方向键，只能用退格键（Backspace）来删除光标前一位置的字符。并且一旦按下回车键，该行输入的字符就不可修改。

（2）上述内容输入后，按 Enter 键，让光标处于输入内容的下一行，按 CTRL+D 键结束输入。

（3）要查看文件是否生成，输入命令“ls”即可。

（4）输入命令“cat f1”，查 看 f1 文件的内容，相关操作参见如下内容。

cat >f1 Linux is useful for us all.

You can never imagine how great it is. ls

Desktop f1

cat f1 Linux is useful for us all.

You can never imagine how great it is.

【操作要求 3】向 f1 文件增加以下内容：Why not have a try?

【操作步骤】

（1）输入命令“cat >>f1”，屏幕上输入点光标闪烁。

（2）输入上述内容后，按 Enter 键，让光标处于输入内容的下一行，按 CTRL+D 键结束输入。

（3）输入“cat f1”命令，查看 f1 文件的内容，会发现 f1 文件增加了一行，相关操作参见如下内容。

cat >>f1 Why not have a try? cat f1 Linux is useful for us all.

You can never imagine how great it is. Why not have a try?

Shell 命令中可使用重定向来改变命令的执行。此处使用“>>”符号可向文件结尾处追

加内容，而如果使用“>”符号则将覆盖已有的内容。Shell 命令中常用的重定向符号共三个，如下所示：

>：输出重定向，将前一命令执行的结果保存某个文件。如果这个文件不存在，则将创

建此文件；如果这个文件已有内容，则将放弃原有内容。

>>:附加输出重定向，将前一命令执行的结果追加到某个文件。

<：将某个文件交由命令处理。

【操作要求 4】统计 f1 文件的行数，单词数和字符数，并将统计结果存放在 countf1 文件。

【操作步骤】

（1）输入命令“wc <f1> countf1”，屏幕上不显示任何信息。

（2）输入命令“cat countf1”，查看 countf1 文件的内容，其内容是 f1 文件的行数、单词数和字符数信息，即 f1 文件共有 3 行，19 个词和 87 个字符,相关操作参见如下内容。 wc <f1> countf1

cat countf1 3 19 87

【操作要求 5】将 f1 和 countf1 文件的合并为 f 文件

【操作步骤】

（1）输入命令“cat f1 countf1 >f”，将两个文件合并为一个文件。

（2）输入命令“cat f”，查看 f 文件的内容，如下所示。

cat f1 countf1 >f cat f

Linux is useful for us all.

You can never imagine how great it is. Why not have a try?3 19 87

【操作要求 6】分页显示/etc 目录中所有文件和子目录的信息

【操作步骤】

（1）输入命令“ls /etc|more”，屏幕显示出“ls /etc”命令输出结果的第一页，屏幕的最后一行上还出现“--More--”字样，按空格键可查看下一页信息，按 Enter 键可查看下一行信息。

（2）浏览过程中按“q”键，可结束分页显示。

管道符号“|”用于连接多个命令，前一命令的输出结果是后一命令的输入。

【操作要求 7】仅显示/etc 目录中前 5 个文件和子目录。

【操作步骤】

输入命令“ls /etc |head -n 5”, 屏幕显示出“ls /etc”命令输出结果的前面 5 行,相关操作参见如下内容。

ls /etc|head –n 5

a2ps.cfg

a2ps-site.cfg acpi

adjtime aliases

【操作要求 7】清除屏幕内容

【操作步骤】

输入命令“clear”，则屏幕内容完全被清除，命令提示符定位在屏幕左上角。

**3.通配符的使用**

Shell 命令的通配符包括\*、?、[]、-和!，灵活使用通配符可同时引用多个文件方便操作。

\* ：匹配任意长度的任何字符。

? ：匹配一个字符。

[ ] ：表示范围。

- ：通常与[ ]配合使用，起始字符-终止字符构成范围

！ ：表示不在范围，通常也与[ ]配合使用。

【操作要求 1】显示/bin/目录中所有以 c 为首字母的文件和目录

【操作步骤】

输入命令“ls /bin/c\*”，屏幕将显示/bin 目录中以 c 开头的所有文件和目录,相关操作参见如下内容。

ls /bin/c\*

/bin/cat/bin/chmod/bin/cp/bin/csh

/bin/chgrp /bin/chown/bin/cpio /bin/cut

【操作要求 2】显示/bin/目录中所有以 c 为首字母,文件名只有 3 个字符的文件和目录

【操作步骤】

（1）按向上方向键，Shell 命令提示符后出现上一步操作时输入的命令“ls /bin/c\*”。

（2）将其修改为“ls /bin/c??”，按 下 Enter 键，屏幕显示/bin 目录中以 c 为首字母,文件名只有 3 个字符的文件和目录,相关操作参见如下内容。

ls /bin/c??

/bin/cat/bin/csh/bin/cut

Shell 可以记录一定数量的已执行过的命令，当用户需要再次执行时，不用再次输入， 可以直接调用。使用上下方向键，PgUp 或 PgDown 键，在 Shell 命令提示符后将出现已执行过的命令。直接按 Enter 键就可以再次执行这一命令，也可以对出现的命令行进行编辑，修改为用户所需要的命令后再执行。

【操作要求 3】显示/bin 目录中所有的首字母为 c 或 s 或 h 的文件和目录。

【操作步骤】

输入命令“ls /bin/[csh]\*”，屏幕显示/bin 目录中首字母为 c 或 s 或 h 的文件和目录, 相关操作参见如下内容。

ls /bin/[csh]\*

/bin/cat/bin/chown /bin/csh/bin/sed/bin/sh/bin/stty

/bin/chgrp /bin/cp/bin/cut/bin/setfont/bin/sleep /bin/su

/bin/chmod /bin/cpio/bin/hostname /bin/setserial /bin/sort /bin/sync

[csh]\*并非表示所有以 csh 开头的文件，而表示是以 c 或 s 或 h 的文件。另外为避免误解，也可以使用[c,s,h]\*，达到相同的效果。

【操作要求 4】显示/bin/目录中所有的首字母是 v、w、x、y、z 的文件和目录。

【操作步骤】

输入命令“ls /bin/[!a-u]\*”，屏幕显示/bin 目录中首字母是 v~z 的文件和目录,相关操作参见如下内容。

ls /bin/[!a-u]\*

/bin/vi/bin/view/bin/ypdomainname/bin/zcat

【操作要求 5】重复上一步操作

【操作步骤】

输入命令“！！”，自动执行上一步操作中使用过的“ls 见如下内容。

!! ls /bin/[!a-u]\*

/bin/[!a-e]\*”命令,相关操作参

/bin/vi/bin/view/bin/ypdomainname/bin/zcat

用户不仅可利用上下方向键来显示执行过的命令；还可以使用 history 命令查看或调用执行过的命令。history 命令可查看到已执行命令在历史记录列表中的序号，可使用“！ 序

号”命令调用，而“！！”命令则执行最后执行过的那个命令。

【操作要求 6】查看刚执行过的 5 个命令

【操作步骤】

输入命令“history 5”，显示最近执行过的 5 个命令,相关操作参见如下内容。命令编号可能不同。

history 5

15ls /bin/c??

16ls /bin/[csh]\*

17ls /bin/[!a-u]\*

18ls /bin/[!a-u]\*

19history 5

**4.设置手工启动图形化用户界面**

图形化用户界面可以在启动 RHEL Server 5 时自动启动，也可以在字符界面启动后用“startx”命令 手动启动。/etc/inittab 文件中运行级别（initdefault）的取值决定启动 RHEL Server

5 后是否自动启动图形化用户界面。

RHEL Server 5 默认的运行级别为 5，即自动启动图形化用户界面，如果将其修改为 3

则只提供字符界面。

在实际工作中，对于以担任服务器功能为主的 RHEL Server 5 主机而言，通常运行级别为 3，这样的话系统资源可几乎完全用于提供服务，而不必消耗在图形界面上。

【操作要求 1】设置开机不启动图形化用户界面

【操作步骤】

（1）按下 ALT+F7 键，切换回到图形化用户界面，以超级用户身份登录。

（2）依次单击「应用程序」菜单=>「附件」=>「文本编辑器」，打开 gedit 文本编辑器。

（3）单击工具栏上的「打开」按钮，从「打开文件...」对话框中选择/etc 目录中的 inittab

文件。

（4）将文件中的“id：5：initdefault：”所在行的“5”修改为“3”，修改后的文件。

修改 inittab 文件

（5）单击工具栏上的「保存」按钮，并关闭 gedit。

（6）单击顶部面板的「系统」菜单=>「关机」，弹出对话框，选择「重新启动」，重新启动计算机。

【操作要求 2】手工启动图形化用户界面。

【操作步骤】

（1）计算机重启后只有字符界面可用，输入用户名和相应的口令后，登录 Linux 系统。

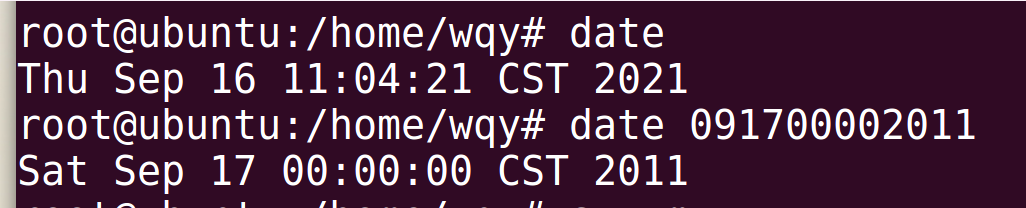
（2）输入命令“startx”，启动图形化用户界面。

（3）单击「系统」菜单=>「注销」，弹出对话框，单击「注销」按钮，返回到字符界面。

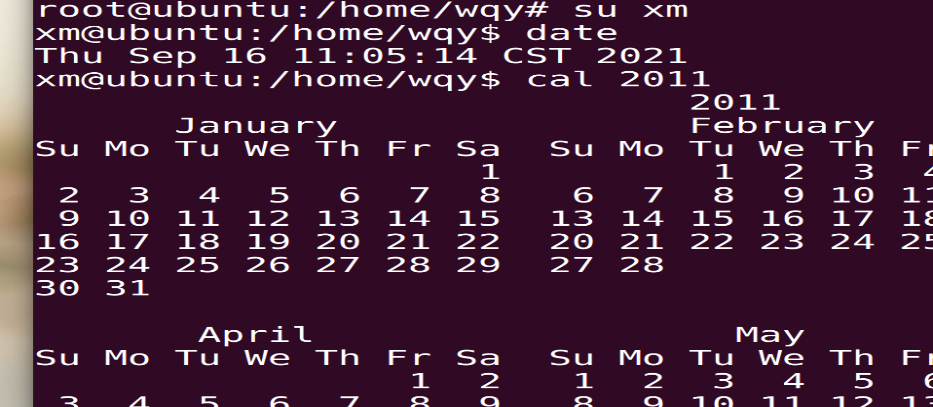
**结果：**

**1 图形化用户界面下shell命令操作**

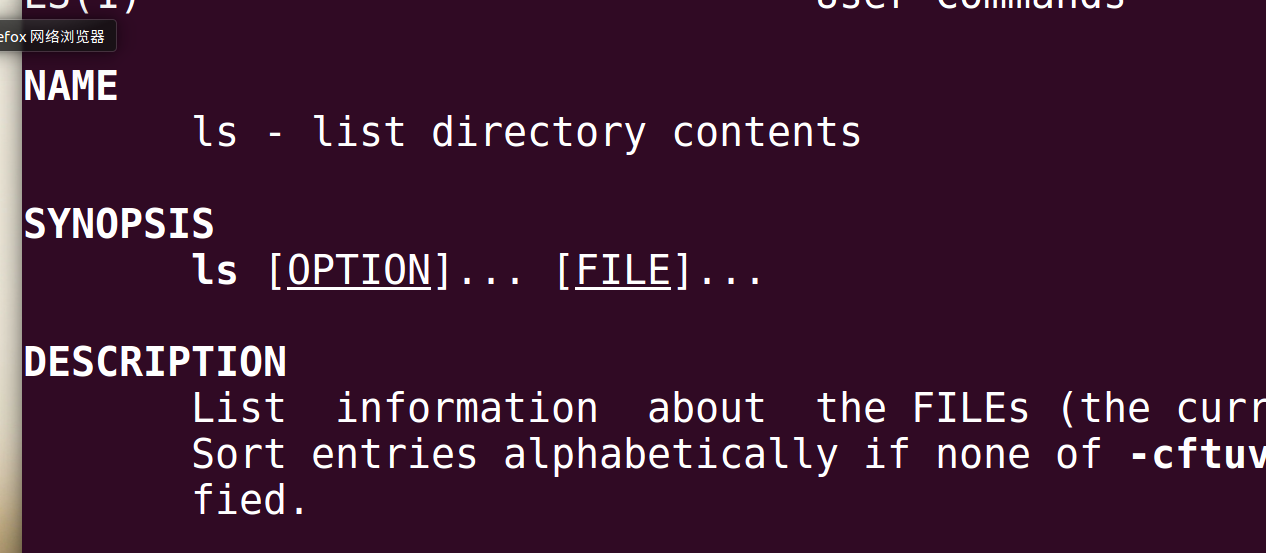
显示系统时间，并将系统时间修改为 2011 年 9 月 17 日零点。



切换为普通用户，查看 2011 年 9 月 17 日是星期几。

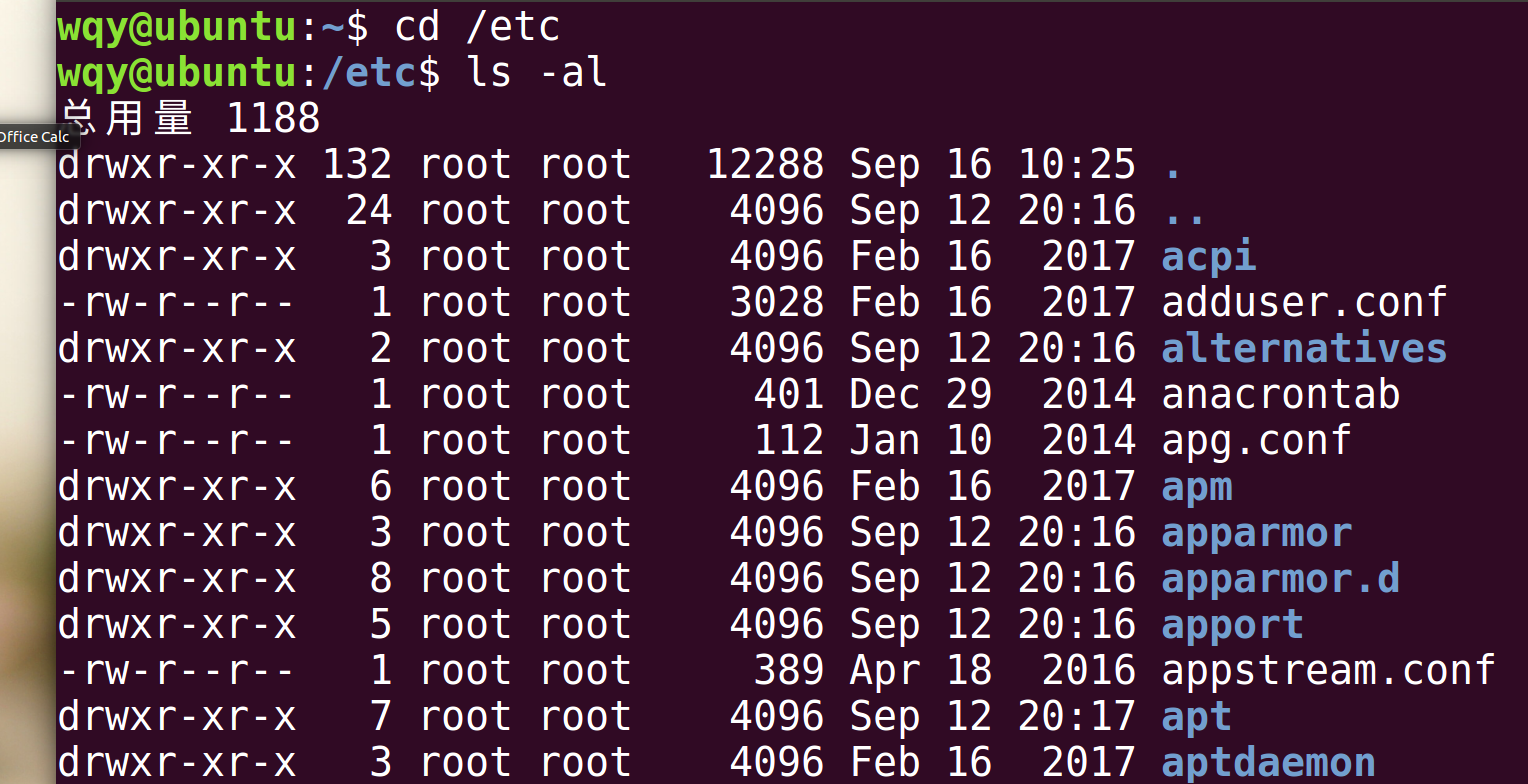
****

查看 ls 命令的-s 选项的帮助信息。



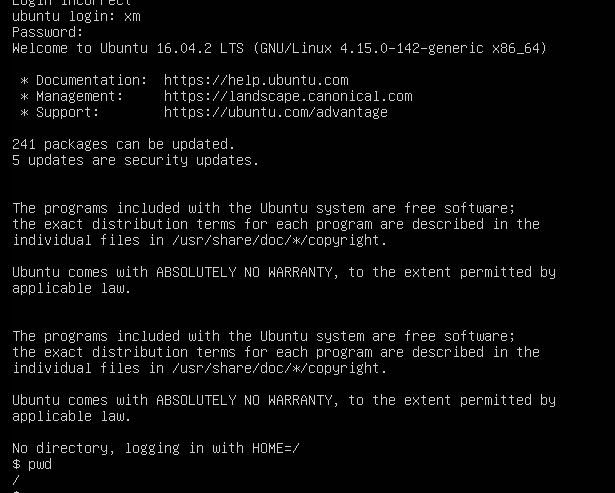


查看/etc 目录下所有文件和子目录的详细信息。

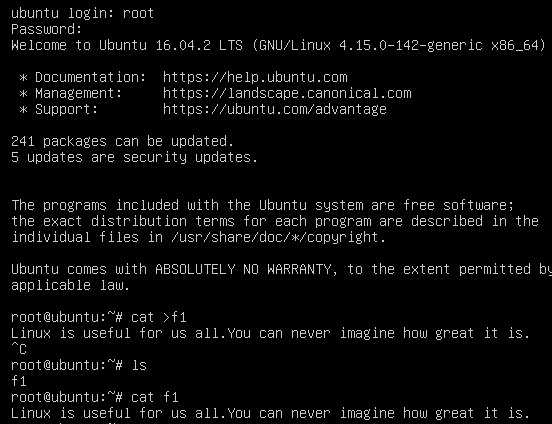


**2 字符界面下的 Shell 命令操作**

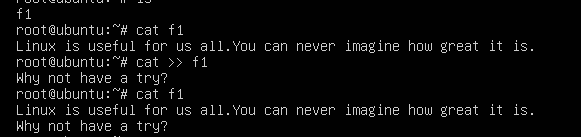
查看当前目录。

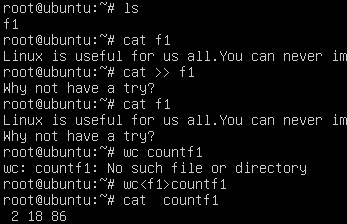


用 cat 命令在用户主目录下创建一名为 f1 的文本文件。

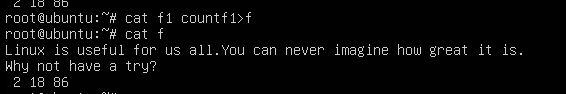


向 f1 文件增加以下内容。





将 f1 和 countf1 文件的合并为 f 文件。



分页显示/etc 目录中所有文件和子目录的信息。



仅显示/etc 目录中前 5 个文件和子目录。



**3 通配符的使用**

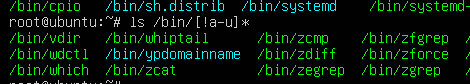
显示/bin/目录中所有以 c 为首字母的文件和目录。



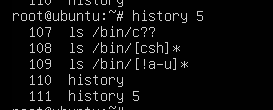
显示/bin/目录中所有以 c 为首字母,文件名只有 3 个字符的文件和目录。



显示/bin 目录中所有的首字母为 c 或 s 或 h 的文件和目录。

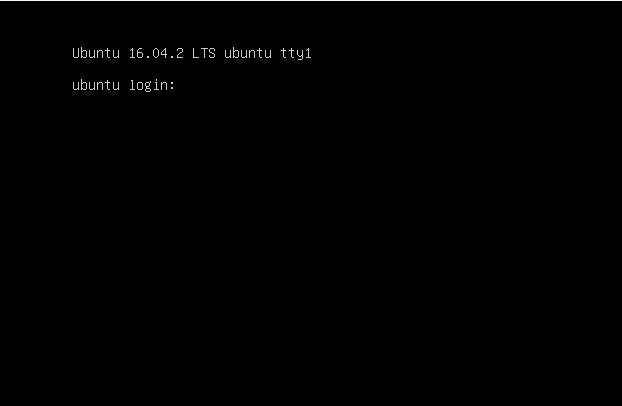
显示/bin/目录中所有的首字母是 v、w、x、y、z 的文件和目录。

查看刚执行过的 5 个命令

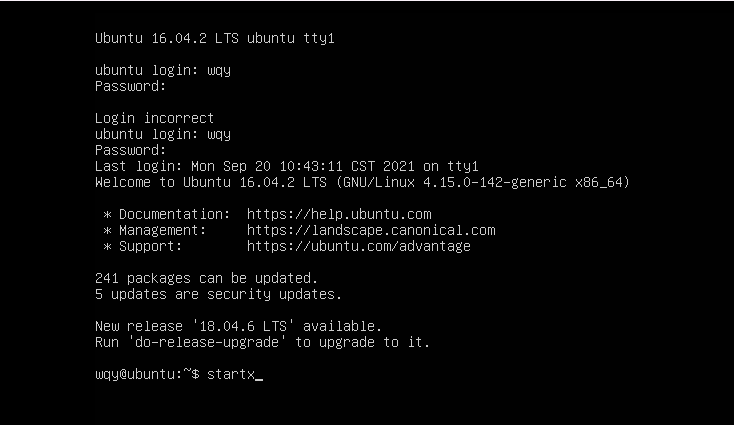


**4 设置手工启动图形化用户界面**

设置开机不启动图形化用户界面



手工启动图形化用户界面。



**体会：**

图形化用户界面（GNOME 和 KDE）下用户操作非常简单而直观，但是到目前为止图形化用户界面还不能完成所有的操作任务。字符界面占用资源少，启动迅速，对于有经验的管理员而言，字符界面下使用 Shell 命令更为直接高效。通过这次课，我更好地了解了图形界面和字符界面，同时更加理解了shell命令是Linux操作系统的灵魂，更加熟练的掌握了图形化界面的操作和shell命令的操作，为下一节课的学习打好了基础。

**附录：（源程序）**

**1 图形化用户界面下的 Shell 命令操作**

date

date 091700002011

cal 2011

man ls

:q

ls --help

q

cd /etc

ls -al

**2 字符界面下的 Shell 命令操作**

cat >f1

cat f1

cat >> f1

wc<f1> countf1

cat countf1

cat f1 countf1>f

ls /etc|more

ls /etc|head -n 5

**3 通配符的使用**

ls /bin/c\*

ls /bin/c??

ls /bin/[csh]\*

ls /bin/[!a-u]\*

history

history 5

**4 设置手工启动图形化用户界面**

sudo systemctl set-default multi-user.target

startx

sudo systemctl set-default graphical.target

实验题目：**进程管理及进程通信**

姓名：王启元 学号：19122925实验日期：2021年10月21日

**进程管理及进程通信**

**实验环境：**

Linux环境：虚拟机：Ubuntu16.04.3

**实验目的:**

利用Linux提供的系统调用设计程序，加深对进程概念的理解。体会系统进程调度的方法和效果。了解进程之间的通信方式以及各种通信方式的使用。

**实验准备：**

复习操作系统课程中有关进程、进程控制的概念以及进程通信等内容（包括软中断

通信、管道、消息队列、共享内存通信及信号量概念）。

熟悉本《实验指导》第五部分有关进程控制、进程通信的系统调用。它会引导你学会怎样掌握进程控制。

阅读例程中的程序段。

**实验方法：**

用vi编写c程序（假定程序文件名为prog1.c）编译程序

$gcc-oprog1.oprog1.c

或$cc-oprog1.oprog1.c

运行

$./prog1.o

**实验内容：**

通过终端或虚拟终端，在基于字符的交互界面中进行Shell的基本命令的操作。

**操作过程：**

用vi编写使用系统调用的C语言程序。

1. 编写程序。显示进程的有关标识（进程标识、组标识、用户标识等）。经过5秒钟后，执行另一个程序，最后按用户指示（如：Y/N）结束操作。

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

#include<sys/wait.h>

intmain()

{

inti,j;

chare;

printf("进程标识：%d\n真实用户标识：%d\n真实组标识：%d\n",getpid(),getuid(),getgid());

i=fork();

if(i==0){

sleep(5);

execlp("ls","ls",(char\*)0);

perror("execerror.\n");

exit(1);

}

j=wait(0);

printf("Doyouwanttoquit,YorN?\n");

scanf("%c",&e);

if(e=='y'){

exit(0);

}

}

**思考:子进程是如何产生的？又是如何结束的?子进程被创建后它的运行环境是怎样建立的？**

答：子进程是通过函数fock()创建的。

通过exit()函数自我结束，如果子进程先退岀，它不会从进程列表里清除在这里子进程向父进程发送SIGCHLD(或SIGCLD)信号给父进程，父进程确认后于进程才会退出。在等待父进程确认期冋，于进程处「''zombie'?犬态。所以我们就需要使用wait()函数。如果调川wait。函数时已经有•个处于''zombie〃状态的子进程，那么函数立即返回的同时该于进程从内存中清除出去；否则，主进程会被挂起，直到其中一个进程退出。直接调用wait。函数有个很明显的缺点就是父进程会被挂起而无法进行其他任务。解决办法就是拦截处理信号SIGCHLD(或

SIGCLD)。

子进程创建后核心将为其分配一个进程表项和进程标识符，检査同时运行的进程数目，并且拷贝进程表项的数据，由子进程继承父进程的所有文件。

1. 参考例程1，编写程序。实现父进程创建一个子进程。体会子进程与父进程分别获得不同返回值，进而执行不同的程序段的方法。

#include<unistd.h>

#include<stdio.h>

#include<stdlib.h>

#include<sys/wait.h>

intmain()

{

inti;

if(i=fork())

{

i=wait(0);

printf("Itisparentprocess.\n");

printf("Thechildprocess,IDnumber%d,isfinished.\n",i);

}

else{

printf("Itischildprocess.\n");

sleep(10);

exit(1);

}

}

思考：子进程是如何产生的？ 又是如何结束的？子进程被创建后它的运行环境是怎

样建立的？

答：子进程是通过函数fork()产生的，通过exit()函数结束。子进程被创建后系统为其分配一个进程表项和进程标识符，检查同时运行的进程数目，并且拷贝进程表项的数据，由子进程继承父进程的所有文件。

1. 参考例程2，编写程序。父进程通过循环语句创建若干子进程。探讨进程的家族树以及子进程继承父进程的资源的关系。

#include<unistd.h>

#include<stdio.h>

#include<stdlib.h>

#include<sys/wait.h>

intmain(){

inti,j;

printf("Mypidis%d,myfather'spidis%d\n",getpid(),getppid());

for(inti=0;i<3;i++)

if(fork()==0)

printf("%dpid=%dppid=%d\n",i,getpid(),getppid());

else{

j=wait(0);

printf("%d:Thechild%disfinished.\n",getpid(),j);

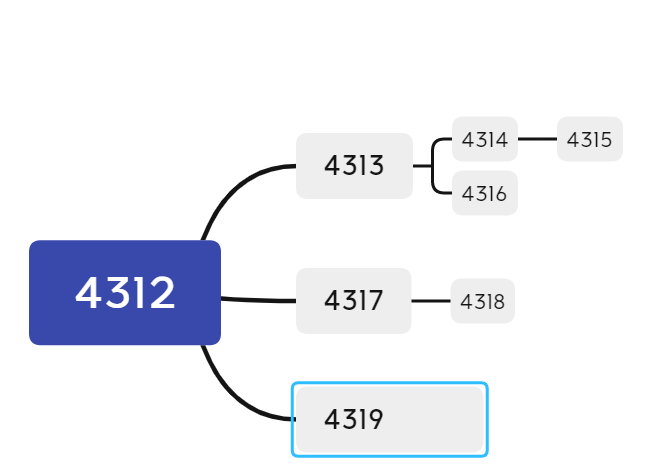
}

return0;

}

1. 画出进程的家族树。子进程的运行环境是怎样建立的？反复运行此程序

看会有什么情况？解释一下。



答：了程序的运行环境是由将其创建的父进程而建立的，反其运行程序会发现何个进程标识号上不断改变，这是因为同一时间许多进程在被创建.

②修改程序，使运行结果呈单分支结构，即每个父进程只产生一个子进程。画出进程树，解释该程序。

1. 参考例程3编程，使用fork()和exec()等系统调用创建三个子进程。子进程分别启动不同程序，并结束。反复执行该程序，观察运行结果，结束的先后，看是否有不同次序。

#include<unistd.h>

#include<stdio.h>

#include<stdlib.h>

#include<sys/wait.h>

intmain()

{

intchild\_pid1,child\_pid2,child\_pid3;

intpid,status;

setbuf(stdout,NULL);

child\_pid1=fork();

if(child\_pid1==0)

{

execlp("echo","echo","childprocess1",(char\*)0);

perror("exec1error.\n");

exit(1);

}

child\_pid2=fork();

if(child\_pid2==0)

{

execlp("date","date",(char\*)0);

perror("exec2error.\n");

exit(2);

}

child\_pid3=fork();

if(child\_pid3==0)

{

execlp("ls","ls",(char\*)0);

perror("exec3error.\n");

exit(3);

}

puts("Parentsprocessiswaittingforchildprocessreturn!");

while(pid=wait(&status)!=-1){

if(child\_pid1==pid)printf("childprocess1terminatedwithstatus%d\n",(status>>8));

elseif(child\_pid2==pid)printf("childprocess2terminatedwithstatus%d\n",(status>>8));

elseif(child\_pid3==pid)printf("childprocess3terminatedwithstatus%d\n",(status>>8));

}

puts("Allchildprocessesterminated.");

puts("Parentsprocessterminated.");

exit(0);

}

思考：子进程运行其它程序后，进程运行环境怎样变化的？反复运行此程序看会有

什么情况？解释一下。

答：进程运行其他程序后这个进程就完全被新程序代替。并没有产生新进程所以进程标识号不改变除此之外III进程的其它信息代码段数据段栈段等均被新程序的信息所代替。新程序从自己的main。函数开始运行。反复运行此程序发现结朿的先后次序是不可预知的每次运行结果都不一样。原因是当每个子进程运行其他程序时他们的结束随着其他程序的结束而结束所以结束的先后次序在改变。

1. 参考例程4 编程，验证子进程继承父进程的程序、数据等资源。如用父、子进程修改公共变量和私有变量的处理结果；父、子进程的程序区和数据区的位置。

#include<unistd.h>

#include<stdio.h>

#include<stdlib.h>

#include <sys/wait.h>

#include<sys/types.h>

#include <fcntl.h>

int globa=4;

int main()

{

pid\_t pid;

int vari=5;

printf("before fork.\n");

if((pid=fork())<0)

{

printf("fork error.\n");

perror(0);

}

else{

if(pid==0){

globa++;

vari--;

printf("Child %d changed the varl and globa.\n",getpid());

}

else{

wait(0);

printf("Parent %d did not changed the varl and globa.\n",getpid());

}

}

printf("pid = %d,globa = %d,varl = %d \n",getpid(),globa,vari);

exit(0);

return 0;

}

思考：子进程被创建后，对父进程的运行环境有影响吗？解释一下。

答：子进程被创建后，对父进程的运行环境没有影响，因为当子进程在运行时，他有自己的代码段和数据段，这些都可以做修改，但是父进程的代码段和数据段是不会随着子进程的数据段和代码段的改变而改变。

6. 参照《实验指导》第五部分中“管道操作的系统调用”。复习管道通信概念，参考例程5，编写一个程序。父进程创建两个子进程，父子进程之间利用管道进行通信。要求能显示父进程、子进程各自的信息，体现通信效果。/\* 父进程向子进程发送18 号软中断信号后等待。子进程收到信号，执行指定的程

序，再将父进程唤醒。\*/

main()

{

int i,j,k;

int func();

signal(18,func()); /\*设置18 号信号的处理程序\*/

第22页 共 103 页

if(i=fork())

{

j=kill(i,18);

/\*创建子进程\*/

/\*父进程执行\*/

/\*向子进程发送信号\*/

printf("Parent: signal 18 has been sent to child %d,returned %d.\n",i,j);

k=wait(); /\*父进程被唤醒\*/

printf("After wait %d,Parent %d: finished.\n",k,getpid());

}

else

{ /\*子进程执行\*/

sleep(10);

printf("Child %d: A signal from my parent is recived.\n",getpid());

} /\*子进程结束，向父进程发子进程结束信号\*/

}

func() /\*处理程序\*/

{ int m;

m=getpid();

printf("I am Process %d: It is signal 18 processing function.\n",m);

}

思考：①什么是管道？进程如何利用它进行通信的？解释一下实现方法。

答：管道是指能够连接一个写进程和读进程，并允许它们以生产者-消费者方式进行通信的一个共享文件，又称pipe文件。由写进程从管道的入端将数据写入管道，而读进程则从管道出端读出数据进行通信。

②修改睡眠时机、睡眠长度，看看会有什么变化。请解释。

修改睡眠时机和睡眠长度都会引起都会引起进程唤醒的时间不一，因为睡眠时机决定了进程在何时睡眠，睡眠长度决定进程在何时唤醒。

③加锁、解锁起什么作用？不用它行吗？

加锁、解锁是为了解决临界资源的共享问题。不用它将会引起无法有效管理数据，即数据会被修改导致读错了数据。

7. 编程验证：实现父子进程通过管道进行通信。进一步编程，验证子进程结束，由父进

程执行撤消进程的操作。测试父进程先于子进程结束时，系统如何处理“孤儿进程”

的。只要在父进程的后面加上wait()函数，然后打印“子进程已结束”，一旦子进程结束，父进程撤销进程。当父进程先子进程终止时，所有子进程的父进程改变为init进程称为进程由init领养。

8. 编写两个程序一个是服务者程序，一个是客户程序。执行两个进程之间通过消息机制 通信。消息标识MSGKEY 可用常量定义，以便双方都可以利用。客户将自己的进程 标识（pid）通过消息机制发送给服务者进程。服务者进程收到消息后，将自己的进程 号和父进程号发送给客户，然后返回。客户收到后显示服务者的pid 和ppid，结束。 以下例程6 基本实现以上功能。这部分内容涉及《实验指导》第五部分中“IPC系统 调用”。先熟悉一下，再调试程序。

/\*客户程序\*/

/\*The client send a message to server,and receives another message from

server\*/

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/wait.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/msg.h>

#define MSGKEY 75

struct msgform /\*定义消息结构\*/

{

long mtype;

char mtext[256];

};

int main()

{

struct msgform msg;

int msgqid,pid,\*pint;

msgqid=msgget(MSGKEY,0777);/\*建立消息队列\*/

pid=getpid();

pint=(int \*)msg.mtext;

\*pint=pid;

msg.mtype=1;/\*定义消息类型\*/

msgsnd(msgqid,&msg,sizeof(int),0);/\*发送消息\*/

msgrcv(msgqid,&msg,256,pid,0);/\*接受从服务者发来的消息\*/

printf("Clint : receive from pid %d\n",\* pint);

exit(0);

}

/\*例程6：消息通信的实验\*/

/\*客户进程向服务器进程发出信号，服务器进程接收作出应答，并再向客户返回消

息。\*/

/\*===============================\*/

/\*服务者程序\*/

/\*The server receives the message from client,and answer a message\*/

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/wait.h>

#include<sys/types.h>

#include<sys/ipc.h>

#include<sys/msg.h>

#define MSGKEY 75

struct msgform

{ /\*定义消息结构\*/

long mtype;

char mtext[256];

}msg;

int msgqid;

int main()

{

int i,pid,\* pint;

void cleanup();

for(i=0;i<20;i++) /\*设置软中断信号的处理程序\*/

signal(i,&cleanup);

msgqid=msgget(MSGKEY,0777|IPC\_CREAT); /\*建立消息队列\*/

for(;;) /\*等待接受消息\*/

{

msgrcv(msgqid,&msg,256,1,0); /\* 接受消息\*/

pint=(int \*)msg.mtext;

pid=\*pint;

printf("Server :receive from pid %d\n",pid);/\*显示消息来源\*/

msg.mtype=pid;

\*pint=getpid(); /\*加入自己的进程标识\*/

msgsnd(msgqid,&msg,sizeof(int),0); /\*发送消息\*/

}

exit(0);

}

void cleanup()

{

msgctl(msgqid,IPC\_RMID,0);

exit(0);

}

9.这部分内容涉及《实验指导》第五部分中“有关信号处理的系统调用”。编程实现软

中断信号通信。父进程设定软中断信号处理程序，向子进程发软中断信号。子进程收

到信号后执行相应处理程序。

例程7：软中断信号实验

/\* 父进程向子进程发送18 号软中断信号后等待。子进程收到信号，执行指定的程

序，再将父进程唤醒。\*/

main()

{

int i,j,k;

int func();

signal(18,func());

/\*设置18 号信号的处理程序\*/

if(i=fork())

{

j=kill(i,18);

/\*创建子进程\*/

/\*父进程执行\*/

/\*向子进程发送信号\*/

printf("Parent: signal 18 has been sent to child %d,returned %d.\n",i,j);

k=wait();

/\*父进程被唤醒\*/

printf("After wait %d,Parent %d: finished.\n",k,getpid());

}

else

{

/\*子进程执行\*/

sleep(10);

printf("Child %d: A signal from my parent is recived.\n",getpid());

} /\*子进程结束，向父进程发子进程结束信号\*/

}

func()

/\*处理程序\*/

{ int m;

m=getpid();

printf("I am Process %d: It is signal 18 processing function.\n",m);

}

思考：这就是软中断信号处理，有点儿明白了吧？讨论一下它与硬中断有什么区

别？看来还挺管用，好好利用它。

答：①硬中断是由外部事件引起的因此具有随机性和突发性；软中断是执行中断指令产生的，无面外部施加中断请求信号，因此中断的发生不是随机的而是由程序安排好的。

②硬中断的中断响应周期，CPU需要发中断回合信号（NMI不需要），软中断的中断响应周期，CPU不需发中断回合信号。

③硬中断的中断号是由中断控制器提供的（NMI硬中断中断号系统指定为02H）；软中断的中断号由指令直接给出，无需使用中断控制器。

④硬中断是可屏蔽的（NMI硬中断不可屏蔽），软中断不可屏蔽。

**研究与讨论**

1. 讨论Linux 系统进程运行的机制和特点，系统通过什么来管理进程？

答：进程提供了两种优先级，一种是普通的进程优先级，第二个是实时优先级。前者适用SCHED\_NORMAL调度策略，后者可选SCHED\_FIFO或SCHED\_RR调度策略。任何时候，实时进程的优先级都高于普通进程，实时进程只会被更高级的实时进程抢占，同级实时进程之间是按照FIFO（一次机会做完）或者RR（多次轮转）规则调度的。

2. C 语言中是如何使用Linux 提供的功能的？用程序及运行结果举例说明。

答： Linux OS系统中主要用GCC编译器来编译C语言代码，主要步骤是程序运行、连接、开展编译工作、程序输入等。

3. 什么是进程？如何产生的？举例说明。

答：进程是进程实体的运行过程，是系统进行资源分配和调度的一个独立单位。

产生步骤：

（1）申请空白PCB。为新进程申请获得惟一的数字标识符，并从PCB集合中索取一个空白PCB。

（2）为新进程分配资源。为新进程的程序和数据以及用户栈分配必要的内存空间。

1. 初始化进程控件块。

（4）将新进程插入就绪队列，如果进程就绪队列能够接纳新进程，便将新进程插入就绪队列。

4. 进程控制如何实现的？举例说明。

答：为了描述和控制进程的运行，系统为每个进程定义了一个数据结构——进程控制块PCB，它是进程实体的一部分，是操作系统中最重要的记录型数据结构。PCB中记录了操作系统所需的、用于描述进程的当前情况以及控制进程运行的全部信息。当系统创建一个新进程时，就为它建立了一个PCB；进程结束时又回收其PCB，进程于是也随之消亡。

5. 进程通信方式各有什么特点？用程序及运行结果举例说明。

答：共享存储器系统：相互通信的进程共享某些数据结构或共享存储区，进程之间能够通过这些空间进行通信。消息传递系统：进程间的数据交换是以格式化的消息为单位的。管道通信系统：能有效地传送大量数据，因而又被引入到许多其它操作系统中。

6. 管道通信如何实现？该通信方式可以用在何处？

答：向管道提供输入的发送进程，以字符流形式将大量的数据送入管道；而接受管道输出的接收进程，则从管道中接收数据。由于发送进程和接收进程是利用管道进行通信的，故又称为管道通信。由于它能有效地传送大量数据，因而又被引入到许多其它操作系统中。

7. 什么是软中断？软中断信号通信如何实现？

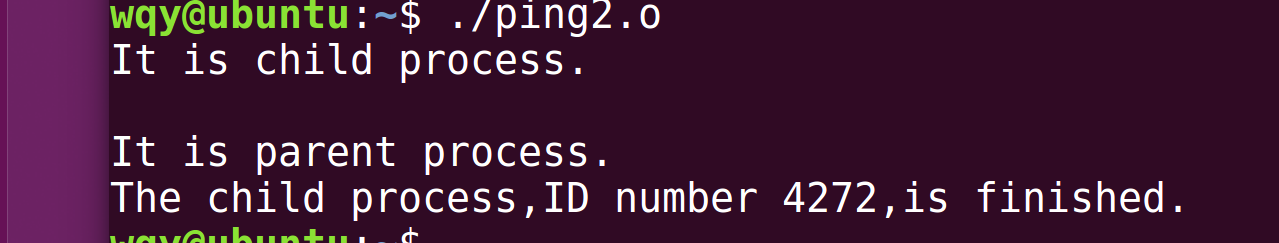
答：软中断是利用硬件中断的概念，用软件方式进行模拟，实现宏观上的异步执行效果。软中断是CPU根据软件的某条指令或者软件对标志寄存器的某个标志位的设置而产生的

**结果：**

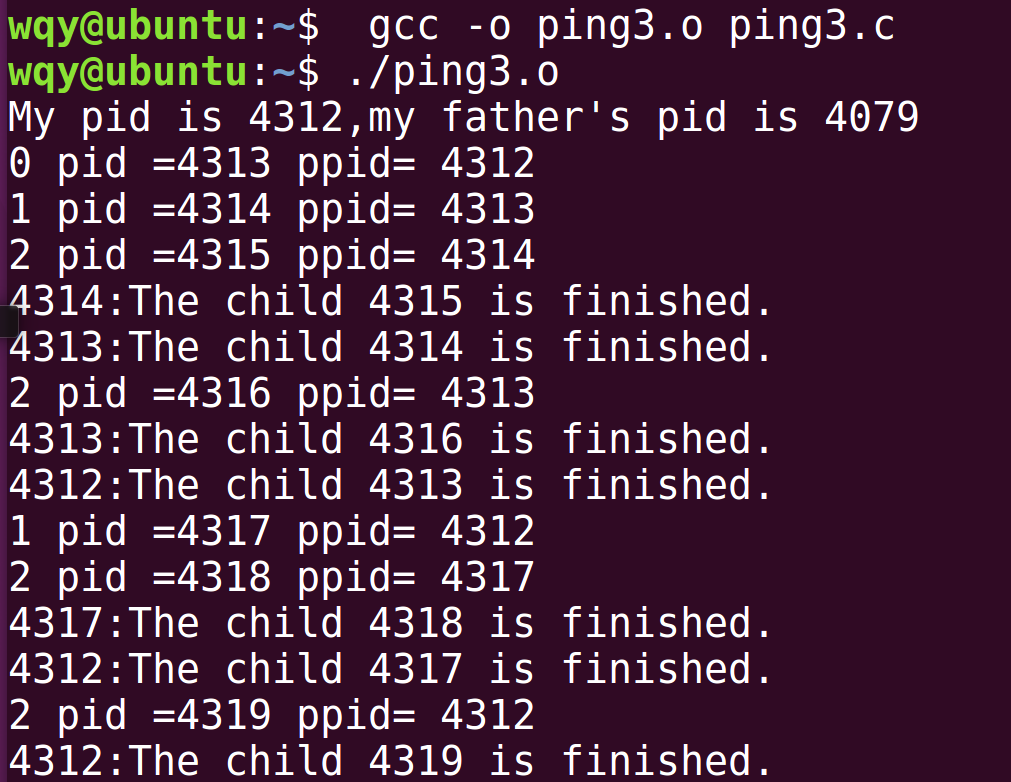
操作一：



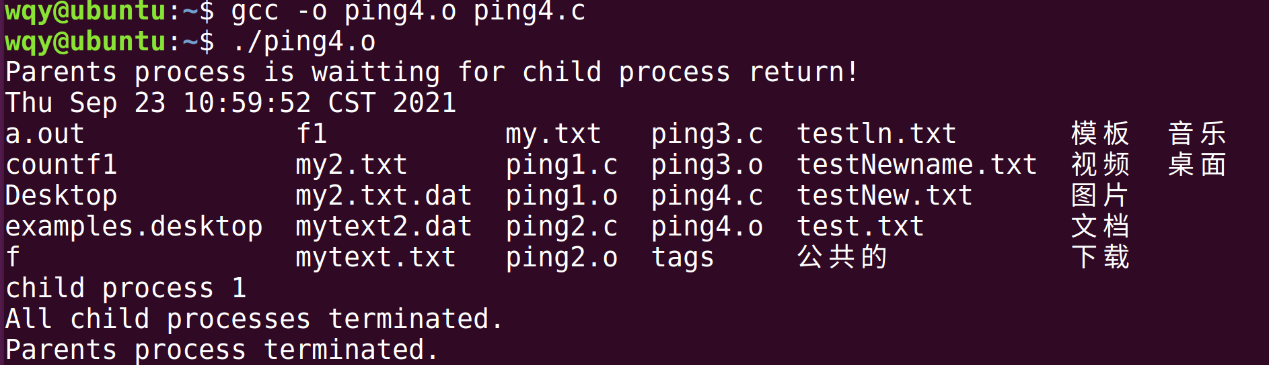
操作二：



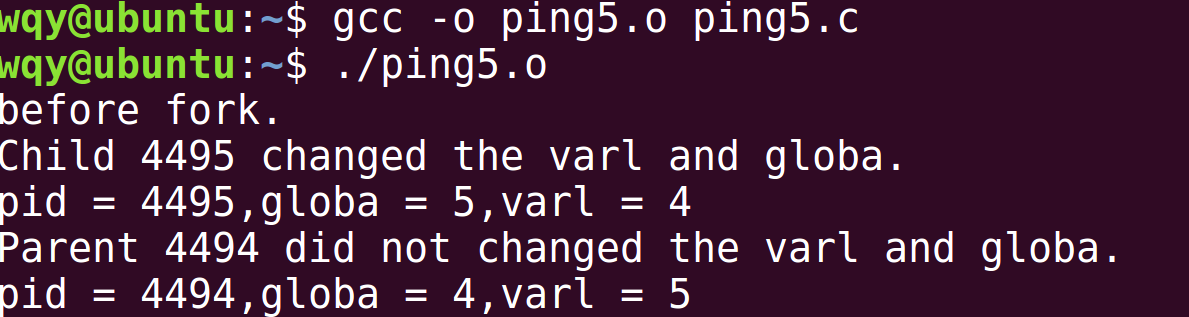
操作三：



操作四：



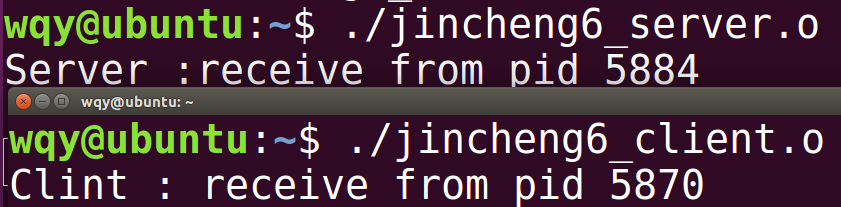
操作五：



操作六：



操作七：



操作八：



**体会：**

在本次实验中，我了解了很多关于进程的知识，对课本内容有了更深刻的理解，真正的把课本知识落实到实处，了解了关于linux的调度和监控知识，并且了解了在linux下，进行c语言编程。

实验题目：**LINUX进程调度与系统监视**

姓名：王启元 学号：19122925 实验日期：2021年10月21日

**LINUX进程调度与系统监视**

**实验环境：**

Linux环境：虚拟机：Ubuntu 16.04.3

**实验目的:**

（1）熟练掌握手工启动前后台作业的方法。

（2）熟练掌握进程与作业管理的相关 Shell 命令。

（3）掌握 at 调度和 cron 调度的设置方法。

（4）了解进行系统性能监视的基本方法。

**实验内容：**

通过终端或虚拟终端，在基于字符的交互界面中进行Shell和图形界面基本命令的操作。

**操作过程：**

**1.作业和进程的基本管理**

【操作要求 1】先在前台启动 vi 编辑器并打开 f4 文件，然后挂起，最后在后台启动一个查找 inittab 文件的 find 作业，find 的查找结果保存到 f5。

【操作步骤】

（1）以超级用户（root）身份登录到 RHEL Server 5 字符界面。

（2）输入命令“vi f4”，在前台启动 vi 文本编辑器并打开 f4 文件。

（3）按下 Ctrl+Z 组合键，暂时挂起“vi f4”作业，屏幕显示该作业的作业号。

[1]+ stopped vim f4

（4） 输入命令“find / -name inittab > f5 &”，启动一个后台作业，如下所示。在显

示作业号的同时还显示进程号。

[root@localhost ~]# find / -name inittab >f5 &

[2] 2619

【操作要求 2】查看当前作业、进程和用户信息，并对作业进行前后台切换。

【操作步骤】

输入命令“jobs”，查看当前系统中的所有作业。

[root@localhost ~]# jobs

[1]+ Stopped vi 4

[2]- Running find / -name inittab > f5

由此可知“vi f4”作业的作业号为 1，已经停止。“find / -name inittab > f5 &”

作业的作业号为 2，正在运行。

（2） 输入命令“fg 2”，将“find / -name inittab > f5 &”作业切换到前台。屏幕显

示出“find / -name inittab > f5”命令，并执行此命令。稍等片刻，作业完成后屏幕

再次出现命令提示符。

（3） 输入命令“cat f5”，查看“find / -name inittab > f5”命令的执行结果。

（4） 再次输入命令“jobs”，可发现当前系统中的只有一个已停止的作业“vi f4”。

（5） 输入命令“kill -9 %1”，终止“vi f4”作业。

[root@localhost ~]# jobs

[1]+ Stopped vi f4

[root@localhost ~]# kill -9 %1

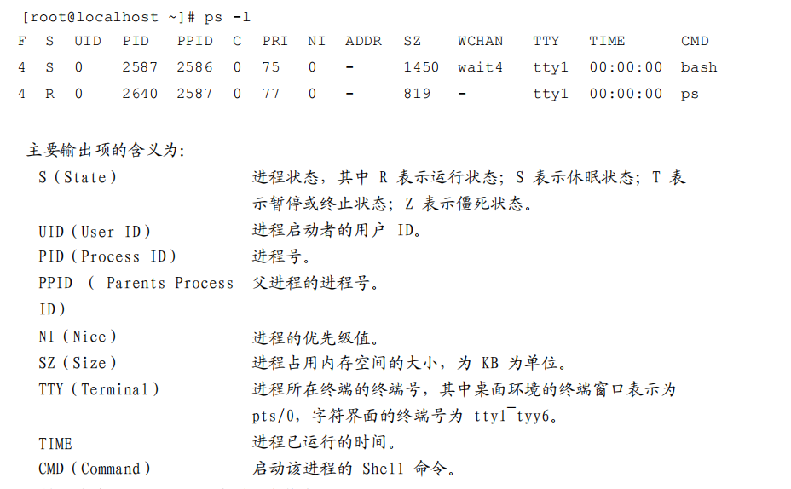
使用 kill 命令时“-9”选项可强制性中止进程或作业。

（6） 稍等片刻，输入命令“jobs”，查看到当前没有任何作业。

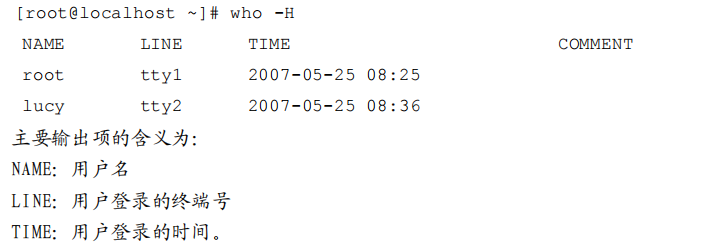
[root@localhost ~]# jobs

[root@localhost ~]#

1. 输入命令“ps –l”，查看进程的相关信息，显示出的信息类似如下信息。



1. 输入命令“who –H”，查看用户信息。



1. **at 进程调度**

所谓进程调度就是设定某个指定的作业在固定的时间、或者固定的频率，或者系统空闲

时自动执行的操作。根据作业要求执行的条件不同，可选中不同的调度方式。

at 调度：在指定的时间执行一次特定的作业。

batch 调度：在系统空闲时执行一次特定的作业。

cron 调度：每到指定的时间就执行特定的作业，可执行多次

【操作要求 1】设置一个调度，要求在 2008 年 1 月 1 日 0 时，向所有用户发送新年快乐的问候。

【操作步骤】

（1）超级用户输入命令“at 00:00 01012008”，设置 2008 年 1 月 1 日 0 时执行的 at 调度的内容。

at 调度的时间表示方法如下所示：

（1）HH:MM：即小时:分钟，如 09:17，采用 24 小时计时制。

（2）数字 AM/PM：采用 12 小时计时制，如 3am。 （3）MMDDYY 或 MM/DD/YY 或 DD.MM.YY：指定具体的日期，必须写在具体时间之后。

（4）now+时间间隔：指定距离现在的时间，时间单位为 minutes（分钟），hours（小

时），day（天），week（星期）。

（5）具体时间：today（今天）、tomorrow（明天）、midnight（深夜）、noon（中午）

teatime（下午 4 点），Tuesday（周二），July 11（7 月 11 日）。（2）输入一个普通用户的用户名（helen）和口令，登录系统。

字符界面下输入口令时，屏幕上不会出现类似“\*”的信息，提高了口令的安全性。

（2） 屏幕出现 at 调度的命令提示符“at>”，输入“wall Happy New Year!”，向所有用户发送消息。

（3） 光标移动到“at>”提示符的第三行，按下 Ctrl+D 组合键结束输入。根据调度设置的时间，最后显示出作业号和将要运行的时间。

[root@localhost ~]# at 00:00 01012008

at>wall Happy New Year！

at><EOT>

job 1 at 2008-01-01 00:00

【操作要求 2】设置一个调度，要求 5 分钟后向所有用户发送系统即将重启的消息，并在 2

分钟后重新启动计算机。

【操作步骤】

（1） 超级用户输入命令“at now +5 minutes”，设置 5 分钟后执行的 at 调度的内容。

（2） 屏幕出现 at 调度的命令提示符“at>”，输入“wall please logout; the computer will

restart.”，向所有用户发送消息。

（3） 在“at>”提示符的第二行输入“shutdown –r +2”，系 统 2 分钟后将重新启动。

“shutdown –r +2”命令与“reboot +2”命令效果相同，都是在 2 分钟后重新启动。

【操作步骤】

（1） 超级用户输入命令“at now +5 minutes”，设置 5 分钟后执行的 at 调度的内容。

（2） 屏幕出现 at 调度的命令提示符“at>”，输入“wall please logout; the computer will

restart.”，向所有用户发送消息。

（3） 在“at>”提示符的第二行输入“shutdown –r +2”，系 统 2 分钟后将重新启动。

“shutdown –r +2”命令与“reboot +2”命令效果相同，都是在 2 分钟后重新启动。

（4） 光标移动到“at>”提示符的第三行，按下 Ctrl+D 组合键结束输入。最后显示作业号

和运行时间。

[root@localhost ~]# at now+5 minutes

at>wall please logout,the computer will restart

at>shutdown –r +2

at><EOT>

job 2 at 2007-10-13 12:07

【操作要求 3】查看所有的 at 调度，并删除 08 年 1 月 1 日执行的调度任务。

【操作步骤】

（1） 输入“atq”命令，查看所有的 at 调度，显示出作业号、将在何时运行以及 at 调度

的设定者。

[root@localhost ~]# atq

1 at 2008-01-01 00:00 a root

2 at 2007-10-13 12:07 a root

（2） 输入“atrm 1”命令删除作业号为 1 的 at 调度，并再次输入“atq”命令查看剩余的

所有 at 调度内容。

[root@localhost ~]# atrm 1

[root@localhost ~]# atq

2 at 2007-10-13 12:07 a root

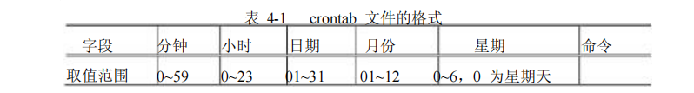
（3） 5 分钟后系统将自动运行作业号为 2 的 at 调度内容。先向所有用户发送消息，然后

再等 2 分钟重新启动。

3.cron 进程调度

cron 调度的内容共有 6 个字段，从左到右依次为分钟、小时、日期、月份、星期和命令，

如表 4-1 所示。



在设置 cron 调度时，所有的字段都不能为空，字段之间用空格分开，如果不指定字段

内容，则使用“\*”符号。

使用“-”符号表示一段时间。如果在日期栏中输入“1-5”则表示每个月前 5 天每天都

要执行该命令。

使用“,”符号表示指定的时间。如果在日期栏中输入“5,15,25”则表示每个月的 5 日、

15 日和 25 日都要执行该命令。

使用“/”符号表示间隔频率，如果在小时栏中输入“\*/2”，表示 某 2 小时执行一次该命

令。

【操作要求 1】helen 用户设置 crontab 调度，要求每天上午 8 点 30 份查看系统的进程状态

【操作步骤】

（1） 以普通用户 helen 登录，并输入命令“crontab –e”，新建一个 crontab 配置文件。

（2） 屏幕出现 vi 编辑器，按下“

i”，进入输入模式，输入“30 8 \* \* \* ps >ps.log ”。

（3） 按下 Esc 键退出 vi 的文本输入模式，并按下“：”键切换到最后行模式，输入“wq”，

保存并退出编辑器，显示“crontab: installing new crontab”信息。

（4） 输入命令“crontab -l”，查 看 helen 用户的 cron 调度内容。

[helen@localhost ~]$ crontab –l

30 8 \* \* \* ps >ps.log（5） 为立即查看到 crontab 调度的结果，切换为超级用户，并适当修改系统时间，如修改

为 8 点 29 分。最后退回到 helen 用户。

[helen@localhost ~]$ su –

Password:

[root@localhost ~]# date 11200829

Tue Nov 20 08:29:00 CST 2007

[root@localhost ~]# exit

[helen@localhost ~]$

（6） 等待 1 分钟后，查看 ps.log 文件的内容，如果显示出正确的内容，那么说明 crontab

调度设置成功。

【操作要求 2】helen 用户添加设置 crontab 调度，要求每三个月的 1 号零时查看正在使用的

用户列表。

【操作步骤】

（1） 再次输入命令“crontab –e”，出 现 vi 编辑器，按下“

i”，屏幕进入文本输入模式。

（2） 在原有内容之后，另起一行，输入“0 0 \* \*/3 \* who >who.log ”。

（3） 最后保存并退出 vi 编辑器。

（4） 为立即查看到 crontab 调度的结果，切换为超级用户，并适当修改系统时间，如修改

为 3 月 31 日 23 点 59 分。最后退回到 helen 用户。

[helen@localhost ~]$ su –

Password:

[root@localhost ~]# date 03312359

Sat Nov 20 23:59:00 CST 2007

[root@localhost ~]# exit

[helen@localhost ~]$

（5） 等待 1 分钟后，查看 who.log 文件的内容，如果显示出正确的内容，那么说明新增

加的 crontab 调度设置成功。

【操作要求 3】查看 cron 调度内容，最后删除此调度。

【操作步骤】

（1） 输入命令“crontab -l”，查看 cron 调度内容。

[helen@localhost ~]$ crontab -l

30 8 \* \* \* ps >ps.log

0 0 \* \*/3 \* who >who.log

（2） 输入命令“crontab -r”，删除 cron 调度内容。

（3） 再次输入命令“crontab -l”，此时无 cron 调度内容。

[helen@localhost ~]$ crontab –r

[helen@localhost ~]$ crontab -l

no crontab for helen

1. 系统性能监视

【操作要求 1】利用 Shell 命令监视系统性能

【操作步骤】

（1） 输入命令“top”，屏幕动态显示 CPU 利用率、内存利用率和进程状态等相关信息，

类似图 4-1。

1. 按下 M 键，所有进程按照内存使用率排列，类似图 4-2。

（3） 按下 T 键，所有进程按照执行时间排列，类似图 4-3。

（4） 最后按下 P 键，恢复按照 CPU 使用率排列所有进程。

（5） 按下 CTRL+C 组合键结束 top 命令。

【操作要求 2】利用「系统监视器」工具监视 CPU 使用情况。

【操作步骤】

（1） 启动 GNOME 桌面环境，依次单击「系统」菜单=>「管理」=>「系统监视器」，打

开「系统监视器」窗口。

（2） 自动显示「资源」选项卡，查看当前 CPU、内存和交换分区、网络历史的使用情况，

如图 4-4 所示。

【操作要求 2】利用「系统监视器」查看当前所有的进程，要求显示出启动进程的用户。

【操作步骤】

（1） 在「系统监视器」窗口单击「进程列表」选项卡，默认显示当前用户启动的所有进

程。单击「查看」菜单，选中「所有的进程」单选按钮，并选中「依赖关系」复选框，

则显示系统中所有的进程，如图 4-5 所示。

（2） 单击「编辑」菜单中的「首选项」，弹出「系统监视器首选项」对话框。在「进程」

选项卡，选中「进程域」栏的「用户」复选框，要求显示出启动进程的用户，如图 4-6

所示。单击「关闭」按钮，显示进程的各种信息。

【操作要求 3】利用「系统监视器」查看所有的文件系统

【操作步骤】

（1） 在「系统监视器」窗口单击「文件系统」选项卡，显示当前 RHEL Server 5 系统中

主要的文件系统。

（2） 单击「编辑」菜单中的「首选项」，弹出「系统监视器首选项」对话框。在「文件系

统」选项卡，选中「显示全部文件系统」复选框，要求显示出全部的文件系统，如图

4-7 所示。最后单击「关闭」按钮。

（3） 「文件系统」选项卡显示全部的文件系统的信息，如图 4-8 所示。

【操作要求 4】利用「系统日志」工具查看系统日志

系统日志文件都保存于/var/log 目录中，包括以下重要的日志文件：

boot.log 记录系统引导的相关信息

cron 记录 cron 调度的执行情况

dmesg 记录内核启动时的信息，主要包括硬件和文件系统的启动信息

maillog 记录邮件服务器的相关信息

messages 记录系统运行过程的相关信息，包括 I/O、网络等

rpmpkgs 记录已安装的 RPM 软件包信息

secure 记录系统安全信息

Xorg.0.log 记录图形化用户界面的 Xorg 服务器的相关信息

【操作步骤】

超级用户依次单击「系统」菜单=>「管理」=>「系统日志」，打开「系统日志」窗口。

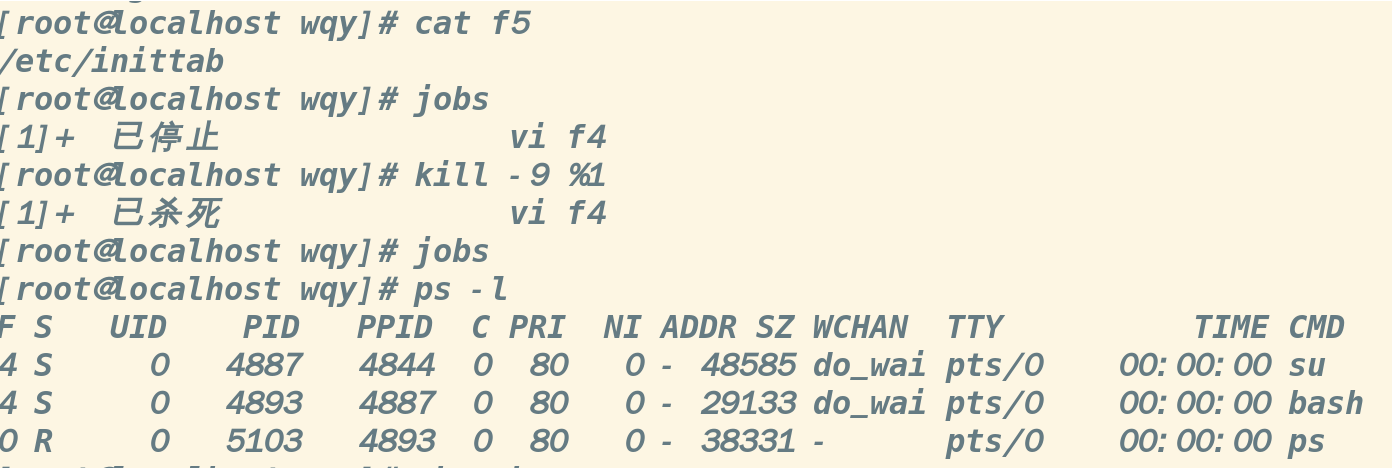
可分别查看各类系统日志，如图 4-9 所示。

**结果：**

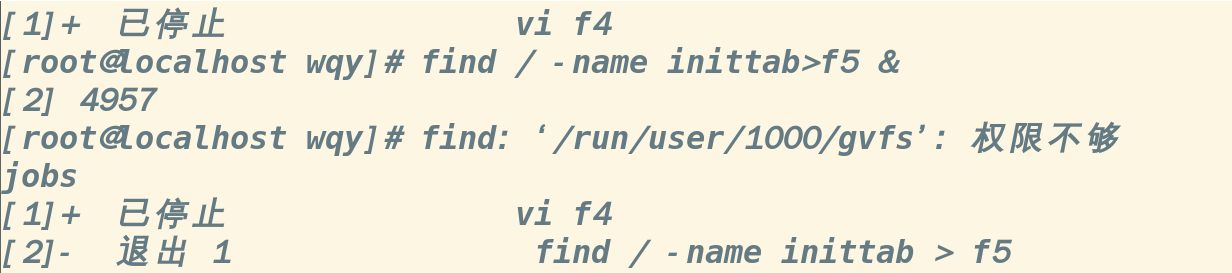
**1 作业和进程的基本管理**

【操作要求 1】先在前台启动 vi 编辑器并打开 f4 文件，然后挂起，最后在后台启动一个查

找 inittab 文件的 find 作业，find 的查找结果保存到 f5。



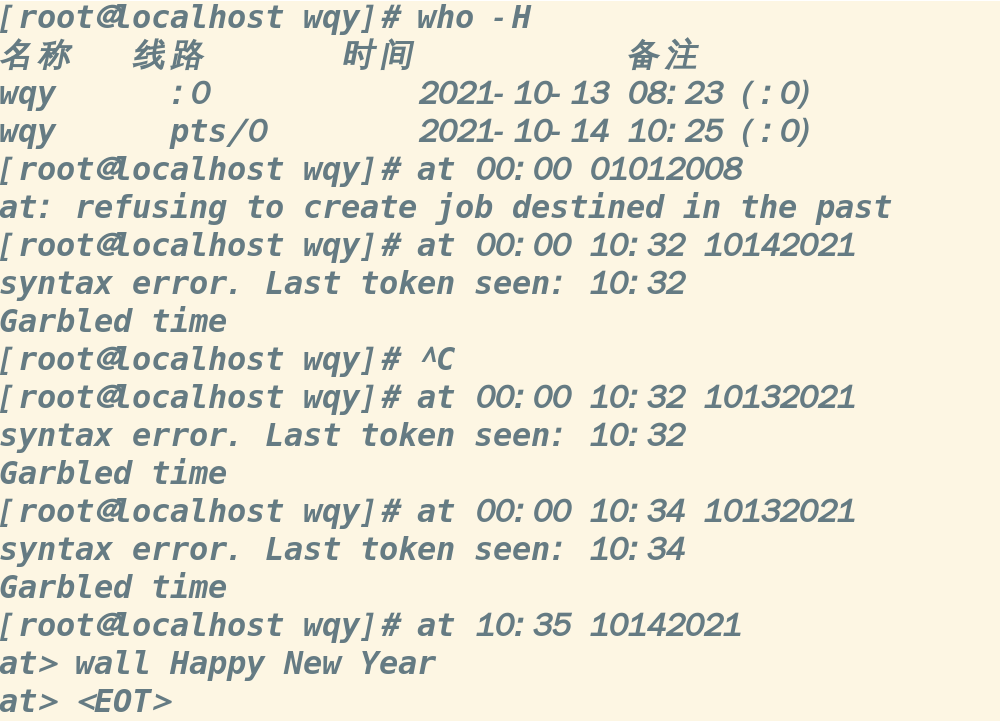
【操作要求 2】查看当前作业、进程和用户信息，并对作业进行前后台切换。



**2 at进程调度**

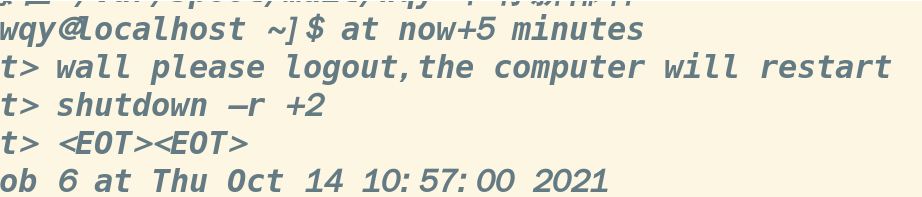
【操作要求 1】设置一个调度，要求在 2008 年 1 月 1 日 0 时，向所有用户发送新年快乐的

问候。

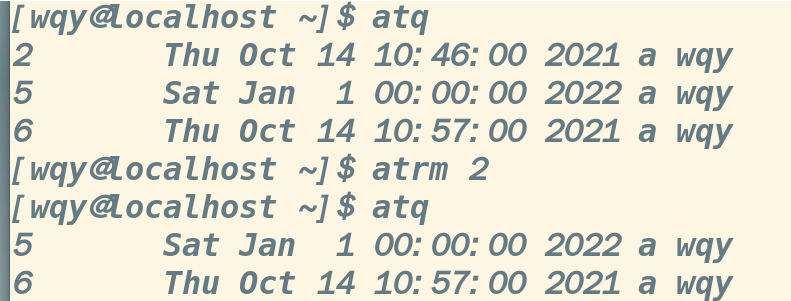


【操作要求 2】设置一个调度，要求 5 分钟后向所有用户发送系统即将重启的消息，并在 2

分钟后重新启动计算机。



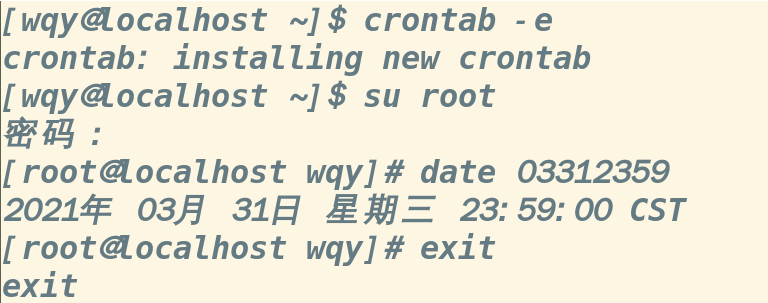
【操作要求 3】查看所有的 at 调度，并删除 08 年 1 月 1 日执行的调度任务。



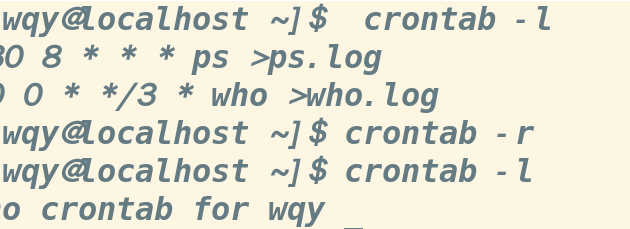
3.cron 进程调度

【操作要求 1】helen 用户设置 crontab 调度，要求每天上午 8 点 30 份查看系统的进程状态，

并将查看结果保存于 ps.log 文件。

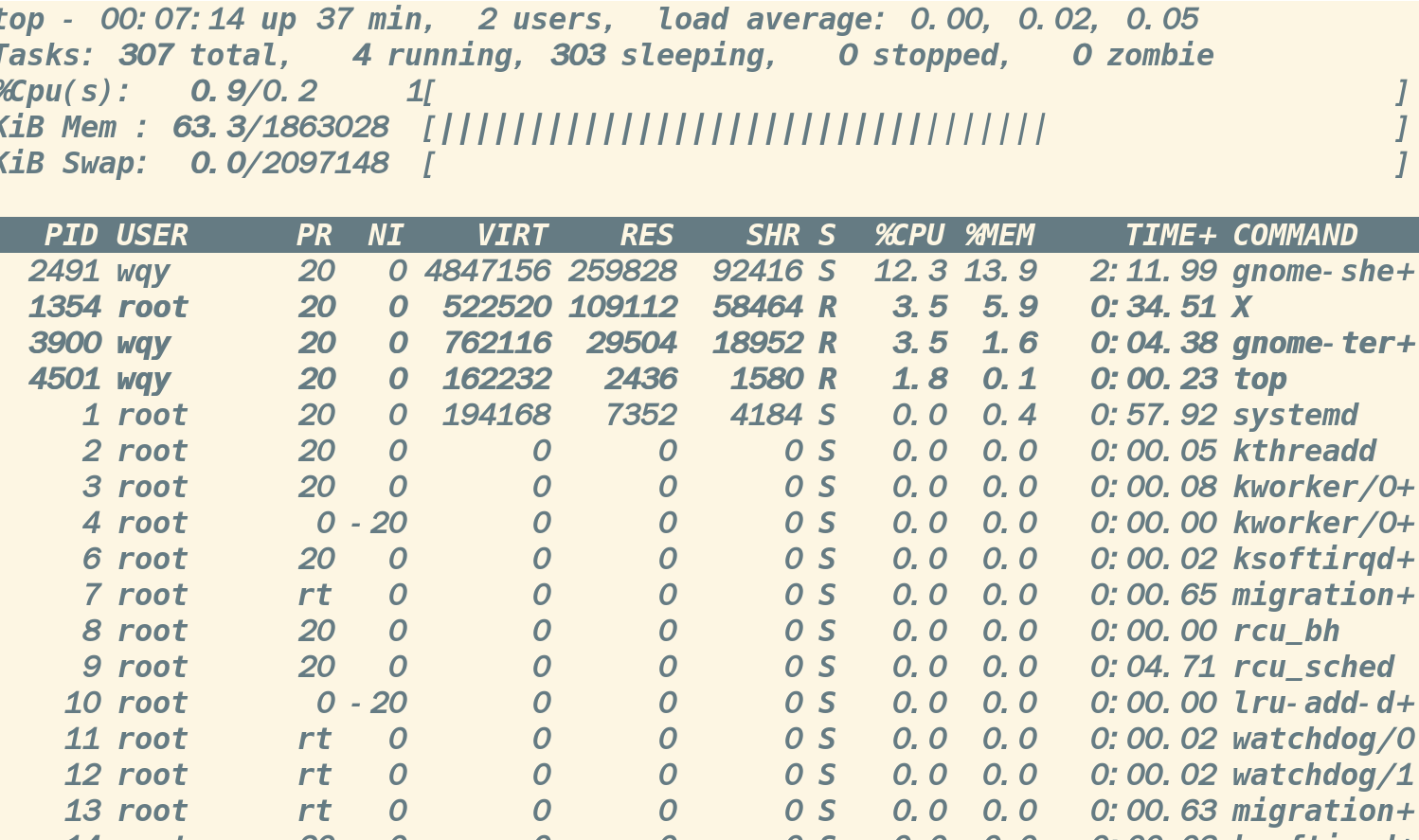


【操作要求 3】查看 cron 调度内容，最后删除此调度。

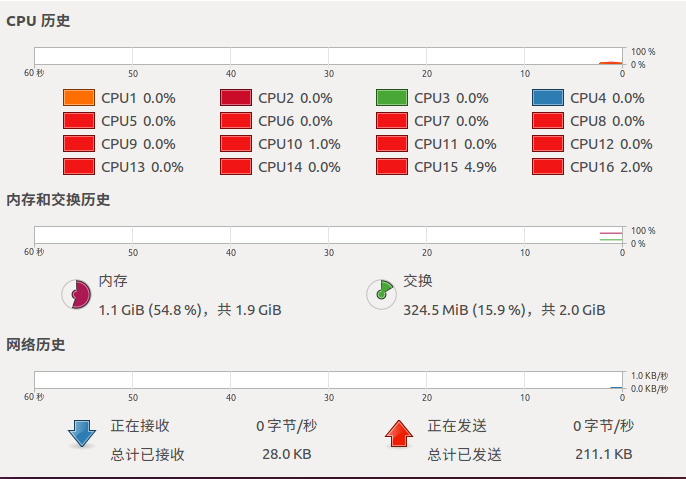


**4** 系统性能监视

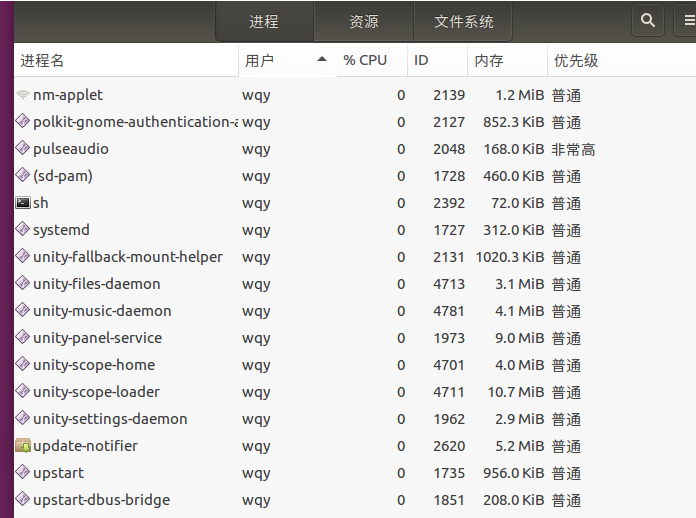
【操作要求 1】利用 Shell 命令监视系统性能



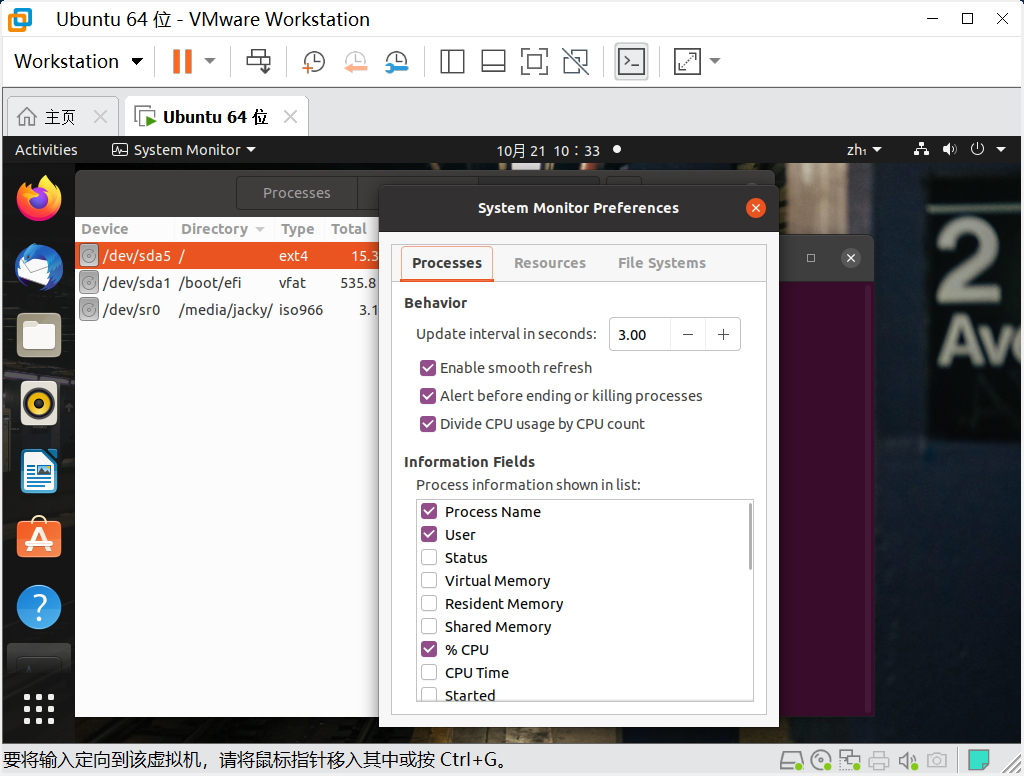
【操作要求 2】利用「系统监视器」工具监视 CPU 使用情况。

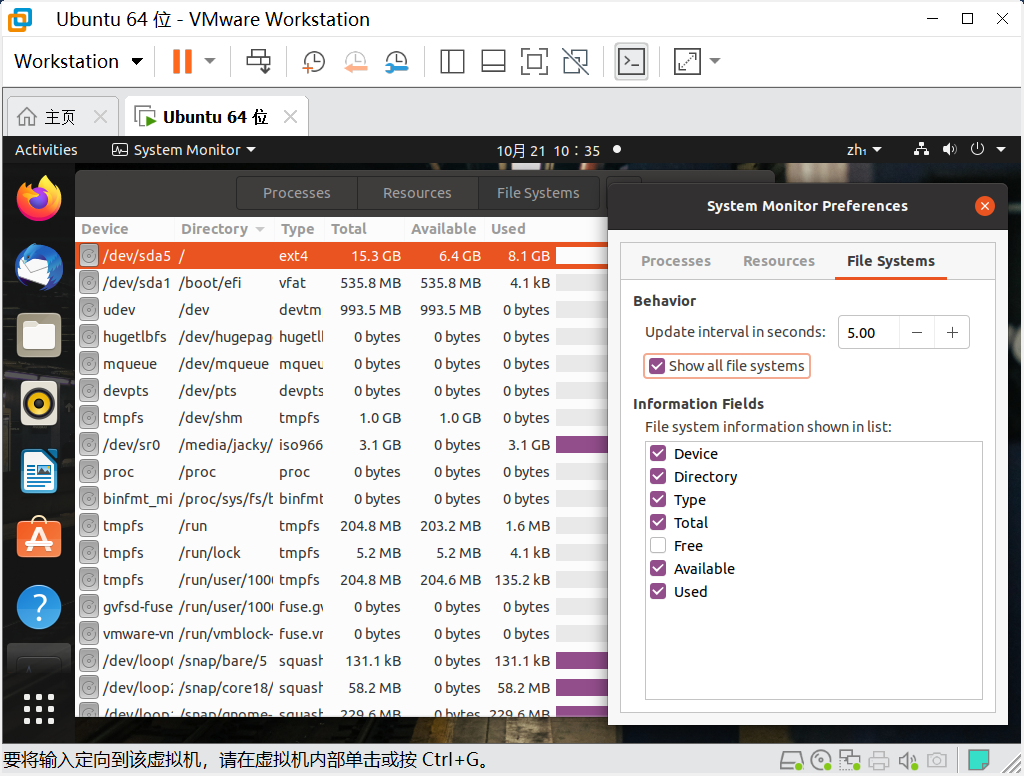


【操作要求 3】利用「系统监视器」查看当前所有的进程，要求显示出启动进程的用户。

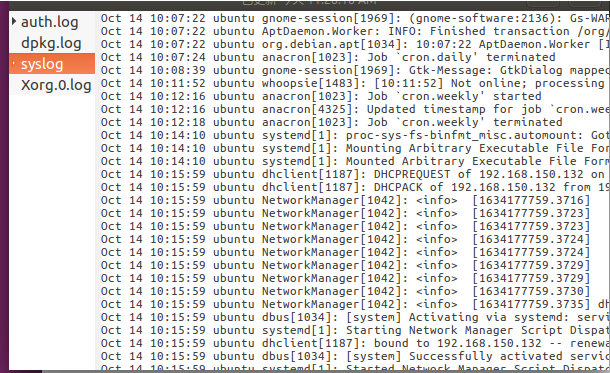


【操作要求 3】利用「系统监视器」查看所有的文件系统





【操作要求 4】利用「系统日志」工具查看系统日志



**体会：**

。

在本次作业中，我熟练掌握了手工启动前后台作业的方法，了解了有关进程和作业管理的相关shell命令，掌握了at调度和cron调度的相关方法，这是linux命令，的基础，这让我对与下一次的实验有了更加充分的准备。

实验题目：用户与组群管理

姓名：王启元 学号：19122925 实验日期：2021年10月21日

用户与组群管理

**实验环境：**

Linux环境：虚拟机：Ubuntu 16.04.3

**实验目的:**

1.理解/etc/passwd 和/etc/group 文件的含义。

2.掌握桌面环境下管理用户与组群的方法。

3.掌握利用 Shell 命令管理用户与组群的方法。

4.掌握批量新建用户帐号的步骤和方法。

**实验内容：**

1.桌面环境下管理用户与组群

2.编辑用户配置文件

3.利用 Shell 命令管理用户与组群

**操作过程：**

**1.桌面环境下管理用户与组群**

【操作要求 1】新建两个用户帐号，其用户名为 xuser1 和 xuser2，口令为“e12ut59er”和

“wfu1t28er”。

【操作步骤】

（1） 以超级用户身份登录 X Window 图形化用户界面，依次单击「系统」菜单=>「管理」

=>「用户和组群」，启动「用户管理者」窗口，如图 5-1 所示。

（2） 单击工具栏上的「添加用户」按钮，出现「创建新用户」窗口。在「用户名」文本

框中输入用户名“xuser1”，在「口令」文本框中输入口令“e12ut59er”，在「确认口令」

文本框中再次输入口令“e12ut59er”，如图 5-2 所示，然后单击「确定」按钮，返回「用

户管理者」窗口。

（3） 用同样的方法新建用户 xuser2，完成后「用户管理者」窗口如图 5-3 所示。

（4） 依次单击顶部面板的「应用程序」=>「附件」=>「文本编辑器」，启动 gedit 文本编

辑器，打开/etc/passwd 和/etc/shadow 文件将发现文件的末尾出现表示 xuser1 和 xuser2

用户帐号的信息。打开/etc/group 和/etc/gshadow 文件将发现文件末尾出现表示 xuser1

和 xuser2 私人组群的信息。

（5） 按下 CTRL+ALT+F2 组合键切换到第 2 个虚拟终端，输入用户名 xuser2 和相应的口

令可登录 Linux 系统，说明新建用户操作已成功。

（6） 输入“pwd”命令，屏幕显示用户登录后进入用户主目录“/home/xuser2”，操作内容如下所示。

localhost login : xuser2

Password :

[xuser2@localhost ~]$ pwd

/home/xuser2

（7） 输入“exit”命令，xuser2 用户退出登录。

（8） 按下 ALT+F7 组合键返回 GNOME 桌面环境。

【操作要求 2】锁定 xuser2 用户帐号

【操作步骤】

（1） 在「用户管理者」窗口选中 xuser2 用户帐号，单击工具栏上的「属性」按钮，打开

「用户属性」窗口。

（2） 选中「帐号信息」选项卡让「本地口令被锁」复选框被选中，如图 5-4 所示。单击「确定」按钮，返回「用户管理者」窗口。

（3） 按下 CTRL+ALT+F2 组合键，再次切换到第 2 个虚拟终端，输入用户名 xuser2 和相

应的口令，发现 xuser2 用户无法登录 Linux 系统，说明 xuser2 用户账号的确已被锁定，

操作内容如下所示：

localhost login : xuser2

Password :

Login incorrect

（4） 按下 ALT+F7 组合键再次返回 GNOME 桌面环境。

【操作要求 3】删除 xuser2 用户

【操作步骤】

（1） 在「用户管理者」窗口，单击「编辑」菜单的「首选项」，弹出「首选项」对话框，不选中「隐藏系统用户和组」复选框，如图 5-5 所示，最后单击「关闭」按钮。此时「用户」选项卡中显示包括超级用户和系统用户在内的所有用户，如图 5-6 所示。

（2） 在「搜索过滤器」文本框中输入“x\*”并按下 Enter 键，则仅显示以 x 为首字母的用户，如图 5-7 所示。

（3） 选中 xuser2 用户，单击工具栏上的「删除」按钮，弹出对话框，如图 5-8 所示，单击「是」按钮，返回「用户管理者」窗口，发现 xuser2 用户已被删除。

（4） 在「搜索过滤器」文本框中输入“\*”并按下 Enter 键，则显示所有用户。

【操作要求 4】新建两个组群，分别是 myusers 和 temp。

【操作步骤】

（1） 在「用户管理者」窗口选中「组群」选项卡，当前显示出所有组群。

（2） 单击工具栏上的「添加组群」按钮，出现「创建新组群」对话框。在「组群名」文

本框中输入“myusers”，如图 5-9 所示，单击「确定」按钮，返回「用户管理者」窗口。

1. 用相同的方法新建 temp 组群，完成后「用户管理者」窗口如图 5-10 所示。

【操作要求 5】修改 myusers 组群属性，将 xuser1 和 helen 用户加入 myusers 组群。

【操作步骤】

（1） 从「组群」选项卡中选择 myusers 组群，单击工具栏上的「属性」按钮，弹出「组

群属性」窗口。

（2） 选择「组群用户」选项卡，选中 helen 和 xuser1 前的复选框，设置 helen 用户和 xuser1

用户的 myusers 组群的成员，如图 5-11 所示。单击「确定」按钮，返回「用户管理者」窗口，如图 5-12 所示。

【操作要求 6】删除 temp 组群

【操作步骤】

从「组群」选项卡中选择 temp 组群，单击工具栏上的「删除」按钮，出现确认对话框，单击「是」按钮即可。

1. **编辑用户配置文件**

【操作要求 1】新建用户配置文件 myusers-profile

【操作步骤】

（1） 依次单击「系统」菜单=>「管理」=>「用户配置文件编辑器」，打开「**User Profile Editor**」

窗口，如图 5-13 所示。

（2） 单击「添加」按钮，弹出「**Add Profile**」窗口，在「**Profile name**」文本框中输入用户配置文件名“myusers-profile”，如图 5-14 所示。单击「添加」按钮，回到「**User ProfileEditor**」窗口，如图 5-15 所示

【操作要求 2】设置 myusers-profile 用户配置文件的内容：应用程序的默认字体为中易宋体

18030，桌面背景为花园。

【操作步骤】

（1） 在「**User Profile Editor**」窗口选中“myusers-profile”文件，单击「编辑」按钮，出现「编辑配置文件 **myusers-profile**」窗口。

「编辑配置文件 myusers-profile」窗口与当前实际的桌面几乎一样，利用这个窗口的

系统菜单可设置用户配置文件的实际内容。

（2） 在「编辑配置用户」窗口中依次单击「系统」菜单=>「首选项」=>「字体」，打开「字体首选项」对话框，如图 5-16 所示。单击应用程序字体的字体列表，出现「拾取字体」对话框（参见图 5-17），从 「字体族」选择“中易宋体 18030”，并单击「确定」按钮。

（3） 回到「字体首选项」对话框，如图 5-18 所示，此时窗口中的字体发生变化，单击窗口右上角的关闭按钮，关闭此对话框。

（4） 在「编辑配置用户」窗口中依次单击「系统」菜单=>「首选项」=>「桌面背景」，打开「桌面背景首选项」对话框，选择“花园”。此时「编辑配置用户」窗口的桌面也发生变化，如图5-19 所示。最后单击「关闭」按钮。

（5） 单击「编辑配置用户」窗口「配置文件」菜单的「保存」项，保存用户配置文件的

修改内容。最后单击「编辑配置用户」窗口右上角的关闭按钮，回到「**User Profile Editor**」

窗口。

【操作要求 3】设置 xuser1 的用户配置文件为 myusers-profile

【操作步骤】

（1） 在「**User Profile Editor**」窗口选中 myusers-profile 文件，单击「**Users**」按钮，出现

「配置文件 **myusers-profile** 的用户」对话框。

1. 选中 xuser1 用户的复选框，如图 5-20 所示，最后单击「关闭」按钮。

（3） 单击「系统」菜单的「注销」项，超级用户退出 GNOME 桌面环境。

（4） 以 xuser1 用户登录，并启动 GNOME 桌面环境，查看应用程序的字体和桌面环境。

GNOME 的用户配置文件编辑功能允许超级用户创建多种用户配置方案，分配给不同类型

的普通用户。

**3.利用 Shell 命令管理用户与组群**

【操作要求 1】新建一名为 duser 的用户，其口令是“tdd63u2”，主要组群为 myusers。

【操作步骤】

（1） 按下 CTRL+ALT+F3 组合键，切换到第 3 个虚拟终端，以超级用户身份登录。

（2） 输入命令“useradd -g myusers duser”，建立新用户 duser，其主要组群是 myusers。

（3） 为新用户设置口令，输入命令“passwd duser”，根据屏幕提示输入两次口令，最后屏幕提示口令成功设置信息，如下所示。

[root@localhost ~]# useradd –g myusers duser

[root@localhost ~]# passwd duser

Changing password for user duser.

New UNIX password:

Retype new UNIX password:

passwd: all authentication tokens updated successfully.

设置用户口令时输入的口令在屏幕上并不显示出来，而输入两次的目的在于确保口令没

有输错。

（4） 输入命令“cat /etc/passwd”，查看 /etc/passwd 文件的内容，发现文件的末尾增加 duser

用户的信息。

（5） 输入命令“

cat /etc/ group”，查 看/etc/ group 文件的内容，发现文件内容未增加。

（6） 按下 ALT+F4 组合键，切换到第 4 个虚拟终端，输入 dusr 用户名和口令可登录 Linux

系统。

（7） 输入“exit”命令，duser 用户退出登录。

【操作要求 2】将 duser 用户设置为不需口令就能登录。

【操作步骤】

（1） 按下 ALT+F3 组合键，切换到正被超级用户使用的第 3 个虚拟终端。

（2） 输入命令“passwd -d duser”，如下所示。

[root@localhost ~]# passwd –d duser

Removing password for user duser.

passwd: Success

在实际应用中此功能必须谨慎使用，因为不需口令的用户帐号一旦被窃可能造成系统入

侵的严重结果。

（3） 按下 ALT+F3 组合键，再次切换到第 3 个虚拟终端，在“Login：”后输入用户名“duser”，

按下 Enter 键就直接出现 Shell 命令提示符，说明 duser 用户不需口令即可登录，如下所示。

localhost login : duser

Last login : Thu Nov 22 04:06:15 on tty4

[duser@localhost ~] $

【操作要求 3】查看 duser 用户的相关信息

【操作步骤】

在第 3 个虚拟终端输入命令“id duser”，显示 duser 用户的用户 ID（UID）、主要组群的名称和 ID（GID），如下所示。

[root@localhost ~]# id duser

uid=502(duser)

gid=502(myusers)

groups=502(myusers) context=system\_u:

system\_r:unconfined\_t: SystemLow-SystemHigh

【操作要求 4】从普通用户 duser 切换为超级用户

【操作步骤】

（1） 第 4 个虚拟终端当前的 Shell 命令提示符为“$”，表明当前用户是普通用户。

（2） 输入命令“ls /root”，屏幕上没有出现/root 目录中文件和子目录的信息，而是出现提示信息，提示当前用户没有查看/root 目录的权限。

（3） 输入命令“su -”或者是“su - root”，屏幕提示输入口令，此时输入超级用户的口令，验证成功后 Shell 提示符从“$”变为“#”，说明已从普通用户转换为超级用户。

（4） 再次输入命令“ls /root”，可查看/root 目录中文件和子目录的信息，相关操作如下所示。

[duser@localhost ~]$ ls /root

ls : /root : Permission denied

[duser@localhost ~]$ su –

Password :

[root@localhost ~]# ls /root

anaconda-ks.cfg Desktop install.log install.log.syslog

（5） 输入“exit”命令，回到普通用户的工作状态。

（6） 输入“exit”命令，duser 用户退出登录。

【操作要求 5】一次性删除 duser 用户及其工作目录

【操作步骤】

（1） 按下 ALT+F3 组合键，切换到正被超级用户使用的第 3 个虚拟终端。

（2） 输入命令“userdel –r duser”，删除 duser 用户。

处于登录状态的用户不能删除。如果在新建这个用户时还创建了私人组群，而该私人组群当前又没有其他用户，那么在删除用户的同时也将一并删除这一私人组群。

（3） 输入命令“cat /etc/passwd”，查 看/etc/passwd 文件的内容，发现 duser 的相关信息

已消失。

（4） 输入命令“ls /home”，发 现 duser 的主目录/home/duser 也不复存在。

【操作要求 6】新建组群 mygroup

【操作步骤】

（1） 在超级用户的 Shell 提示符后输入命令“groupadd mygroup”，建 立 mygroup 组群。

（2） 输入命令“cat /etc/group”，发 现 group 文件的末尾出现 mygroup 组群的信息。

（3） 输入命令“cat /etc/gshadow”，发 现 gshadow 文件的末尾也出现 mygroup 组群的信息。

【操作要求 7】将 mygroup 组群改名为 newgroup

【操作步骤】

（1） 输入命令“groupmod –n newgroup mygroup”，其中–n 选项表示更改组群的名称。

（2） 输入命令“cat /etc/group”，查 看组群信息，发现原来 mygroup 所在行的第一项变为“newgroup”。

【操作要求 8】删除 newgroup 组群

【操作步骤】

超级用户输入“groupdel newgroup” 命令，删除 newgroup 组群。

**4.批量新建多个用户帐号**

【操作要求】为全班同学 20 位同学创建用户帐号，用户名为“s”+学号的组合，其中班级名册中第一位同学的学号为 080101。所有同学都属于 class0801 组群。所有同学的初始口令为 111111。

【操作步骤】

（1） 以超级用户身份登录，输入命令“groupadd

-g 600 class0801”（假设值为 600的 GID 未被使用），新建全班同学的组群 class0801，

（2） 输入命令“vi student”，新建用户信息文件。

（3） 按下“i”键，切换为 vi 的文本编辑模式，输入第一行信息： s080101:x:601:600::/home/

s080101:/bin/bash”。

（4） 按下 ESC 键，切换到命令行模式，拖动鼠标，将整行选中，如图 5-21 所示，然后按下字母键 y 两次。也就是将当前选中的行放到 vi 的暂存区域（类似于 Windows 的剪贴板）。

（5） 然后按下字母键 p，就复制一行信息，如图 5-22 所示，重复此操作 19 次，然后部分修改每位同学用户信息不同的地方。

（6） 最后编辑完成的文件，如图 5-23 所示，为节约篇幅仅显示前 10 位同学信息。最后保存并退出 vi。

（7） 输入命令“vi stu-passwd”，新建用户口令文件。

（8） 按下“i”键，切换为 vi 的文本编辑模式，输入第一行信息：“s080101:111111”，即所有同学的初始口令为 111111。按下 ESC 键，切换到命令行模式，拖动鼠标，将整行选中，然后按下字母键 y 两次，复制行。

（9） 连续按 p 键 19 次，就可复制出 19 行信息，然后修改成正确的用户名，如图 5-24 所示。

（10） 输入命令“newusers < students”，批量新建用户帐号。

（11） 输入命令“pwunconv”，暂时取消 shadow 加密。

（12） 输入命令“chpasswd <stu-passwd”，批量新建用户的口令。

（13） 输入命令“pwconv”，进行 shadow 加密，完成批量创建用户帐号工作。

（14） 输入命令“cat /etc/passwd”，查看/etc/passwd 文件将发现所有的用户帐号均已建立。

（15） 可尝试以新建的用户名登录，并应该及时修改用户的口令。

使用此方法批量创建的用户登录时的命令提示符，不是默认的[用户名@localhost

~]$，而是-bash-3.1$。如果希望使用默认的命令提示符，可将采用 useradd 命令新建的用

户，如 helen 的用户主目录中的.bash\_profile 和.bashrc 文件复制到批量创建的用户主目录即可。

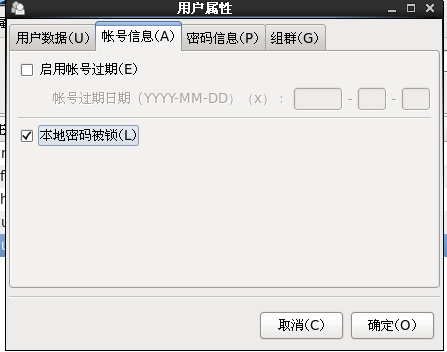
**结果：**

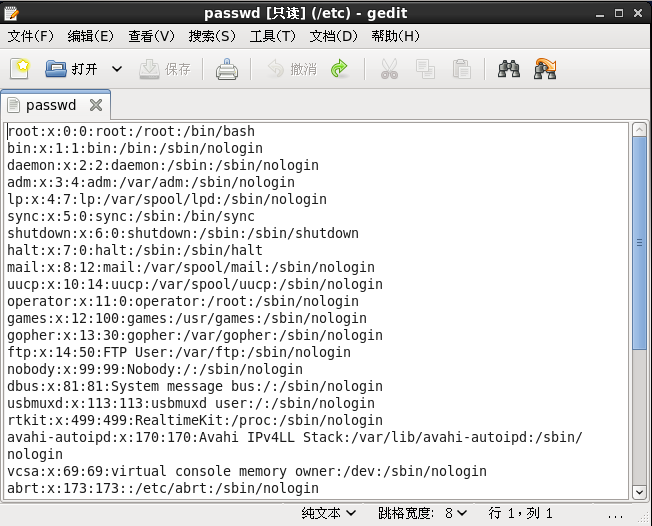
**1 桌面环境下管理用户与组群**

【操作要求 1】新建两个用户帐号，其用户名为 xuser1 和 xuser2，口令为“e12ut59er”和

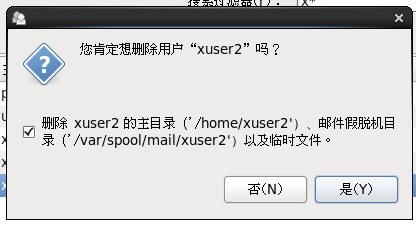
“wfu1t28er”。

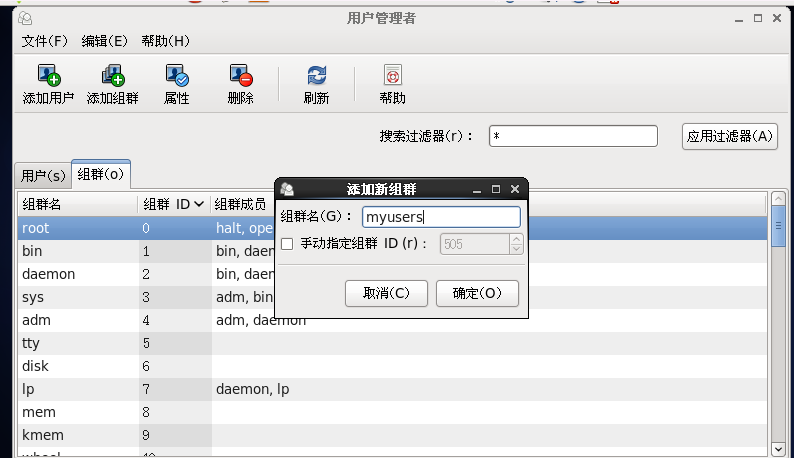


****

【操作要求 3】删除 xuser2 用户

****

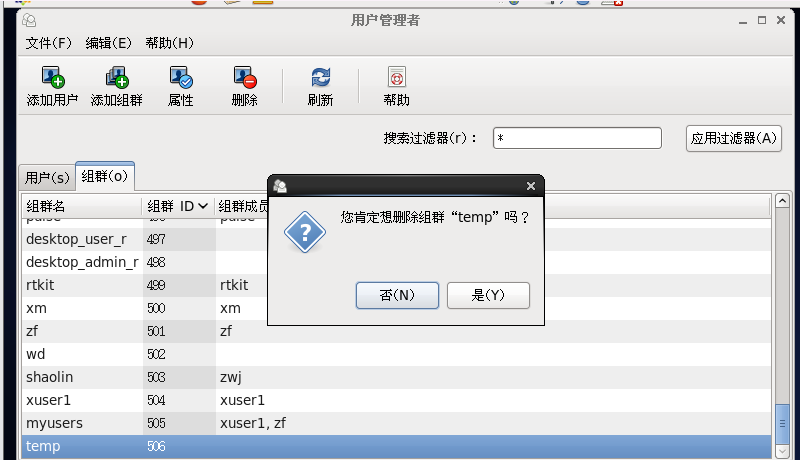
【操作要求 4】新建两个组群，分别是 myusers 和 temp。



【操作要求 5】修改 myusers 组群属性，将 xuser1 和 helen 用户加入 myusers 组群。



【操作要求 6】删除 temp 组群



**2 编辑用户配置文件**

【操作要求 1】新建用户配置文件 myusers-profile

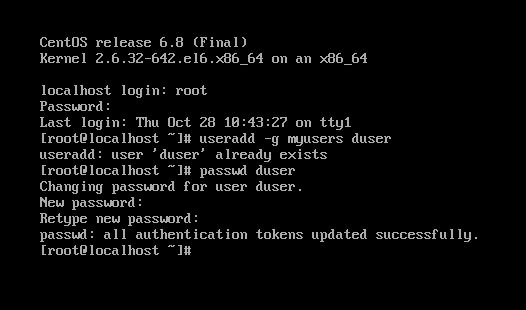
【操作要求 2】设置 myusers-profile 用户配置文件的内容：应用程序的默认字体为中易宋体

18030，桌面背景为花园。

【操作要求 3】设置 xuser1 的用户配置文件为 myusers-profile

**3.利用 Shell 命令管理用户与组群**

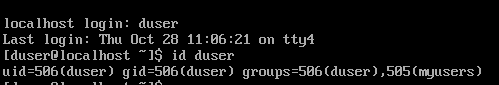
【操作要求 1】新建一名为 duser 的用户，其口令是“tdd63u2”，主要组群为 myusers。



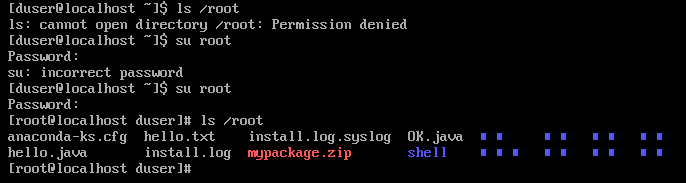
【操作要求 2】将 duser 用户设置为不需口令就能登录。



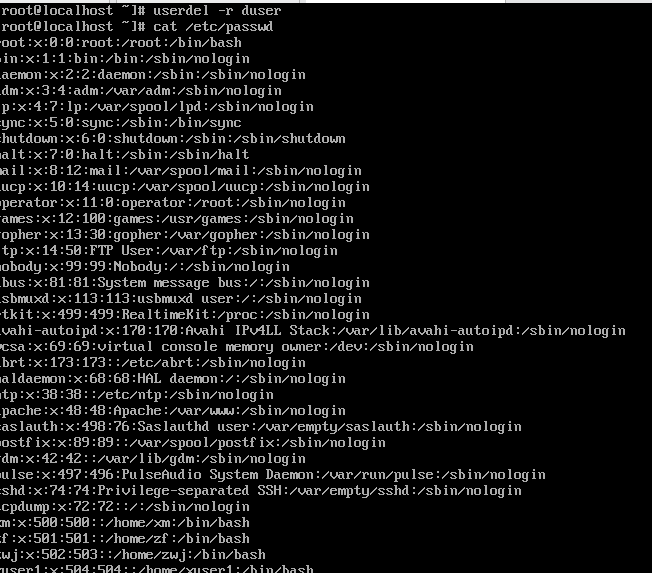
【操作要求 3】查看 duser 用户的相关信息



【操作要求 4】从普通用户 duser 切换为超级用户



【操作要求 5】一次性删除 duser 用户及其工作目录



【操作要求 6】新建组群 mygroup



【操作要求 7】将 mygroup 组群改名为 newgroup

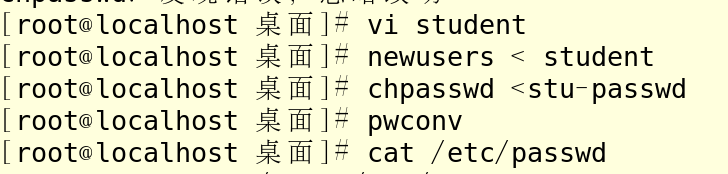
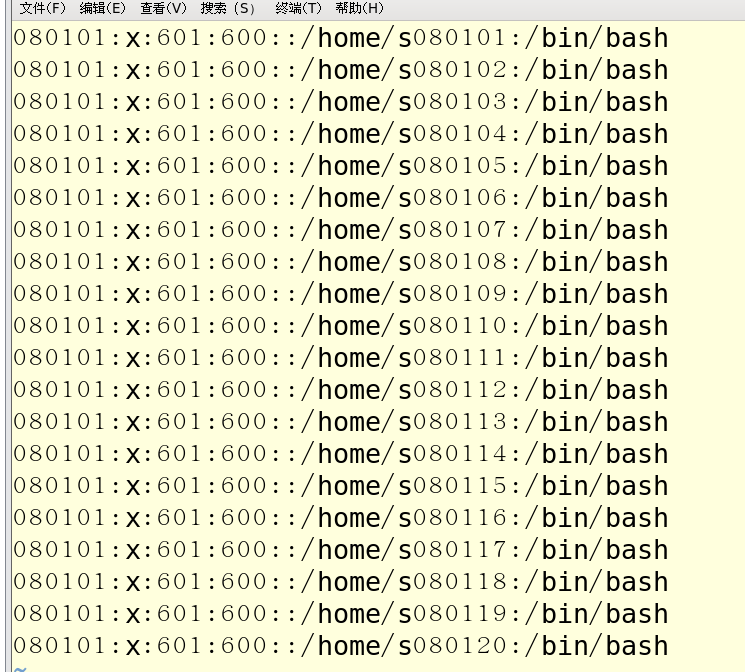


【操作要求 8】删除 newgroup 组群



**4.批量新建多个用户帐号**

【操作要求】为全班同学 20 位同学创建用户帐号，用户名为“s”+学号的组合，其中班级名册中第一位同学的学号为 080101。所有同学都属于 class0801 组群。所有同学的初始口令为 111111。





**体会：**本实验学习的是用户和组群管理方面的知识，在此次实验中我学习到了/etc/passwd 和/etc/group 文件的含义，掌握了桌面环境下和利用Shell命令管理用户与组群的方法以及批量新建用户帐号的步骤和方法。其中我觉得有些命令是极有裨益的，比如useradd、addgroup、gshadow等等。在实验过程中也是不可避免的出现了一些问题，大多都是由于使用虚拟机的版本与环境和实验指导书不同造成的，但是通过上网查阅资料等方法都能成功解决。总的而言，此次实验对我受益匪浅。

**实验题目：进程管理及进程通信**

**姓名：王启元 学号：19122925 实验日期：2021.9.23**

**实验环境：**

VMware Workstation、Linux Ubuntu

**实验目的：**

1.掌握vi 的三种工作方式，熟悉vi 编辑程序的使用。

2.学习Shell 程序设计方法。掌握编程要领。

**实验内容：**

1.学习使用vi 编辑程序。

2.编写Shell 程序。

3.将程序文件设置为可执行文件（用chmod 命令）。

4.在命令行方式中运行Shell 程序

**操作过程：**

**1.** 按本《实验指导》第三部分的内容。熟悉vi 的三种工作方式。熟悉使用各种编辑功能。

 思考：试一试vi 的三种工作方式各用在何时？用什么命令进入插入方式？怎样退出插入方式？文件怎样存盘？注意存盘后的提示信息。

答：三种工作方式用在何时：1.插入方式 ：插入方式可用插入命令（见表 2）的任何一个进入，屏幕下方有一行”------Insert------”字样表示。插入时用户输入的任何字符都认为是文本的内容，且可以用退格键来纠正错误。在插入方式中按一下<ESC>，即退出插入方式，进入转义命令方式。

1. 转义命令方式：刚进入 vi 或退出插入方式，即为转义命令方式。这时键入的任何字符转义为特殊功能，如：移动、删除、替换等。大多数转义命令由一个或两个字母组成，操作时没有提示符，而且输入命令不需要按<ENTER>。
2. 末行命令方式：在转义命令方式中，按冒号“：”就进入末行命令方式。屏幕最末一行的行首显示冒号作为命令提示。命令行输入后按<ENTER>开始执行。此时用户可进行文件的全局操作，如：全局查找、替换、文件读、写等。

插入命令：i text <ESC> 在光标前插入新文本，ENTER 可重起一行

I text <ESC> 在当前行起始处插入新文本

a text <ESC> 在光标后输入新的文本

A text <ESC> 在当前行末尾输入新的文本

o text <ESC> 在当前行下产生新的一行并输入文本

O text <ESC> 在当前行上产生新的一行并输入文本

退出插入方式：Esc

文件怎样存盘：w

2. 创建和执行Shell 程序

用前面介绍的Vi 或其他文本编辑器编写Shell 程序，并将文件以文本文件方式保存在相应的目录中。 用chmod 将文件的权限设置为可执行模式，如若文件名为shdemo.h,则命令如下：

$ chmod 755 shdemo.h (文件主可读、写、执行，同组人和其他人可读和执行)

在提示符后执行Shell 程序：

$ shdemo.h （直接键入程序文件名执行）

或 $ sh shdemo.h （执行Shell 程序）

或 $ .shdemo.h （没有设置权限时可用点号引导）

3. 用vi 编写《实验指导》“第四部分Shell 程序设计”中的例1（假设文件名为prog1.h），练习内部变量和位置参数的用法。

用chmod将文件的权限设置为可执行模式，并在提示符后键入命令行：

$./prog1.

或 $sh prog1.

屏幕显示:

Name not provided

在提示符后键入命令行：

$./prog1.h Theodore

屏幕显示:

Your name is Theodore

4. 进一步修改程序prog1.h，要求显示参数个数、程序名字，并逐个显示参数。

5. 修改例1程序（即上面的 prog1.h），用read命令接受键盘输入。若没有输入显示第一种

提示，否则第二种提示。

6. 用vi 编写《实验指导》“第四部分 Shell 程序设计”中的例2、例3，练习字符串比较运

算符、数据比较运算符和文件运算符的用法，观察运行结果。

7. 修改例2程序，使在程序运行中能随机输入字符串，然后进行字符串比较。

8. 修改例3程序，使在程序运行中能随机输入文件名，然后进行文件属性判断。

9. 用vi 编写《实验指导》“第四部分 Shell 程序设计”中的例4、例5、例6、例7，掌握控

制语句的用法，观察运行结果。

10. 用vi 编写《实验指导》“第四部分 Shell 程序设计”中的例8 及例9掌握条件语句的用

法，函数的用法，观察运行结果。

11. 编程，在屏幕上显示用户主目录名（HOME）、命令搜索路径（PATH），并显示由位

置参数指定的文件的类型和操作权限。

**讨论：**

**1.** Linux 的Shell 有什么特点？

（1）把已有命令进行适当组合构成新的命令。

（2）提供了文件名扩展字符（通配符，如\* 、 ?、 [ ]），使得用单一的字符串可以匹配多个文件名，省去键入一长串文件名的麻烦。

（3）可以直接使用Shell的内置命令,而不需创建新的进程，如Shell中提供的cd、echo、exit、pwd、kill等命令。为防止因某些Shell不支持这类命令而出现麻烦，许多命令都提供了对应的二进制代码，从而也可以在新进程中运行。

（4）Shell允许灵活地使用数据流，提供通配符、输入/输出重定向、管道线等机制，方便了模式匹配、I/O处理和数据传输。

（5）结构化的程序模块，提供了顺序流程控制、条件控制、循环控制等。

（6）Shell提供了在后台执行命令的能力。

（7）Shell提供了可配置的环境，允许创建和修改命令、命令提示符和其它的系统行为。

（8）Shell提供了一个高级的命令语言，能够创建从简单到复杂的程序。这些Shell程序称为Shell脚本，利用Shell脚本，可把用户编写的可执行程序与Unix命令结合在一起，当作新的命令使用，从而便于用户开发新的命令。

**2.** 怎样进行Shell编程？如何运行？有什么条件？

在linux中使用vi或者vim进行shell编程。使用语句./文件名便可运行，条件是要将文件转为可执行模式，即使用语句：chmod 755 文件名。

**3.** vi 编辑程序有几种工作方式？查找有关的详细资料，熟练掌握屏幕编辑方式、转移

命令方式以及末行命令的操作。学习搜索、替换字符、字和行，行的复制、移动，

以及在vi中执行Shell命令的方式。

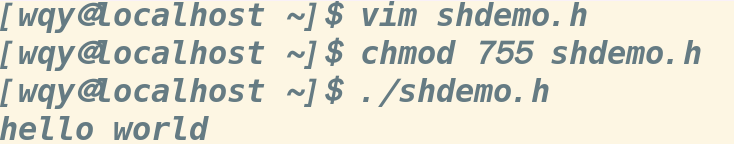
1.命令行模式:该模式是进入vi编辑器后的默认模式。任何时候，不管用户处于何种模式，按下Esc键即可进入命令模式。在该模式下，用户可以输入 vi命令，用于管理自己的文档。此时从键盘上输入的任何字符都被当做编辑命令来解释。若输入的字符是合法的vi命令，则vi在接受用户命令之后完成相应的动作。但需注意的是，所输入的命令并不回显在屏幕上。若输入的字符不是vi的合法命令，vi会响铃报警。

2.文本输入模式:在命令模式下输入插入命令i、附加命令a、打开命令o、修改命令c、取代命令r或替换命令s都可以进入文本输入模式。在该模式下，用户输入的任何字符都被vi当做文件内容保存起来，并将其显示在屏幕上。在文本输入过程中，若想末行模式:末行模式也称ex转义模式。在命令模式下，用户按“:”键即可进入末行模式下，此时vi会在显示窗口的最后一行（通常也是屏幕的最后一行）显示一个“:”作为末行模式的说明符，等待用户输入命令。多数文件管理命令都是在此模式下执行的（如把编辑缓冲区的内容写到文件中等）。末行命令执行完后，vi自动回到命令模式。如果要从命令模式转换到编辑模式，可以键入命令a或者i。如果需要从文本模式返回，则按 Esc键即可。在命令模式下输入“:”即可切换到末行模式，然后输入命令。

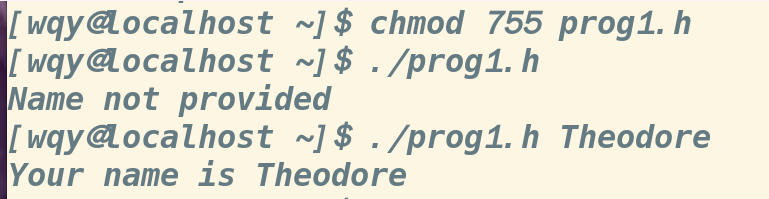
3.末行模式:末行模式也称ex转义模式。在命令模式下，用户按“:”键即可进入末行模式下，此时vi会在显示窗口的最后一行（通常也是屏幕的最后一行）显示一个“:”作为末行模式的说明符，等待用户输入命令。多数文件管理命令都是在此模式下执行的（如把编辑缓冲区的内容写到文件中等）。末行命令执行完后，vi自动回到命令模式。如果要从命令模式转换到编辑模式，可以键入命令a或者i。如果需要从文本模式返回，则按 Esc键即可。在命令模式下输入“:”即可切换到末行模式，然后输入命令。

**结果：**

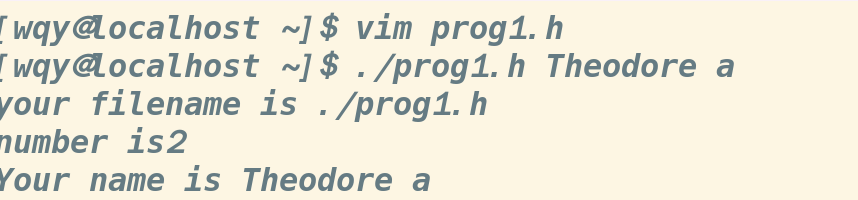
操作一用前面介绍的Vi 或其他文本编辑器编写Shell 程序，并将文件以文本文件方式保存在相应的目录中。



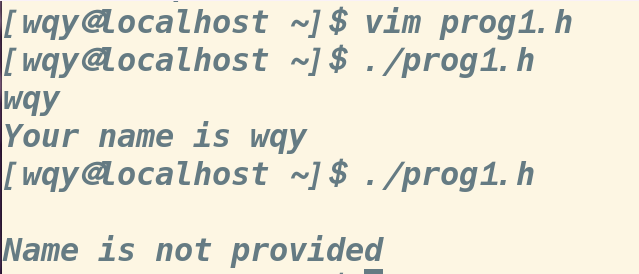
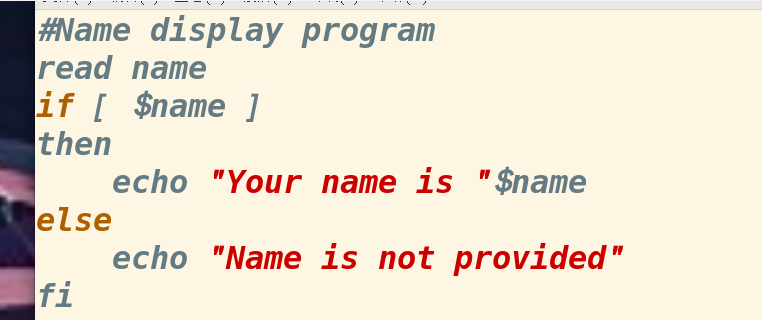
操作二用vi 编写《实验指导》“第四部分Shell 程序设计”中的例1（假设文件名为prog1.h），练习内部变量和位置参数的用法



操作三进一步修改程序prog1.h，要求显示参数个数、程序名字，并逐个显示参数。

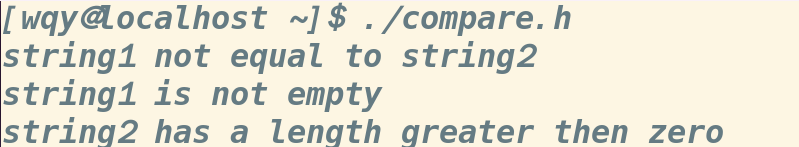


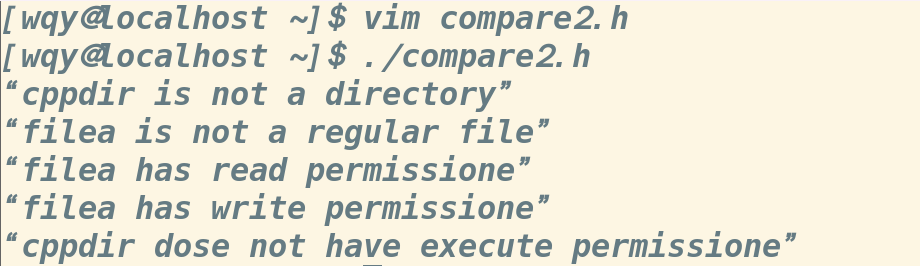
操作四修改例1程序（即上面的 prog1.h），用read命令接受键盘输入。若没有输入显示第一种提示，否则第二种提示。

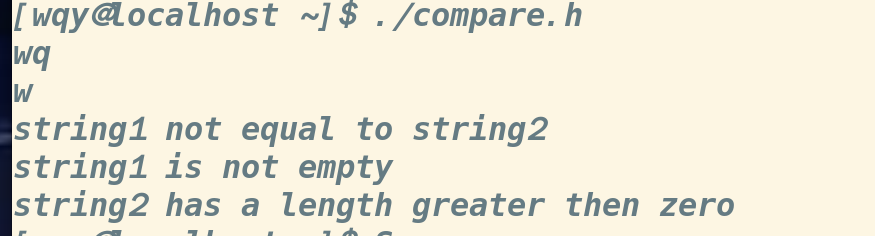
操作五用vi 编写《实验指导》“第四部分 Shell 程序设计”中的例2、例3，练习字符串比较运

算符、数据比较运算符和文件运算符的用法，观察运行结果。

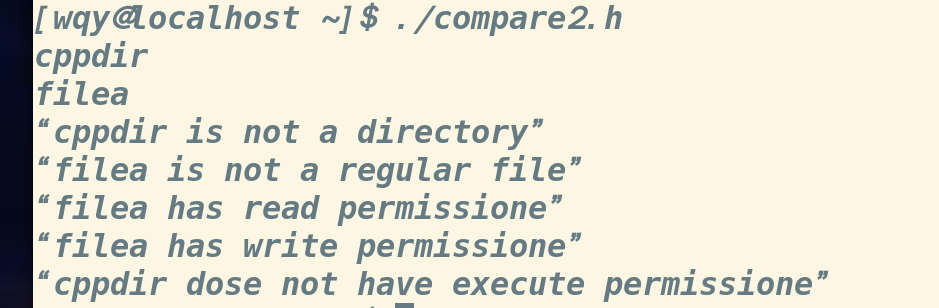




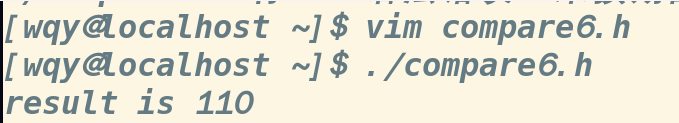
操作六修改例2程序，使在程序运行中能随机输入字符串，然后进行字符串比较。

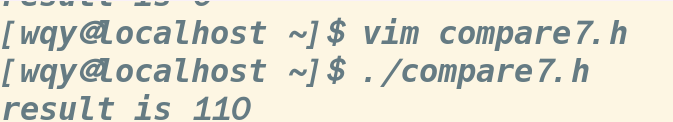


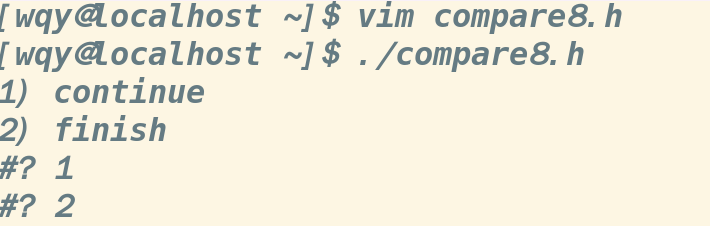
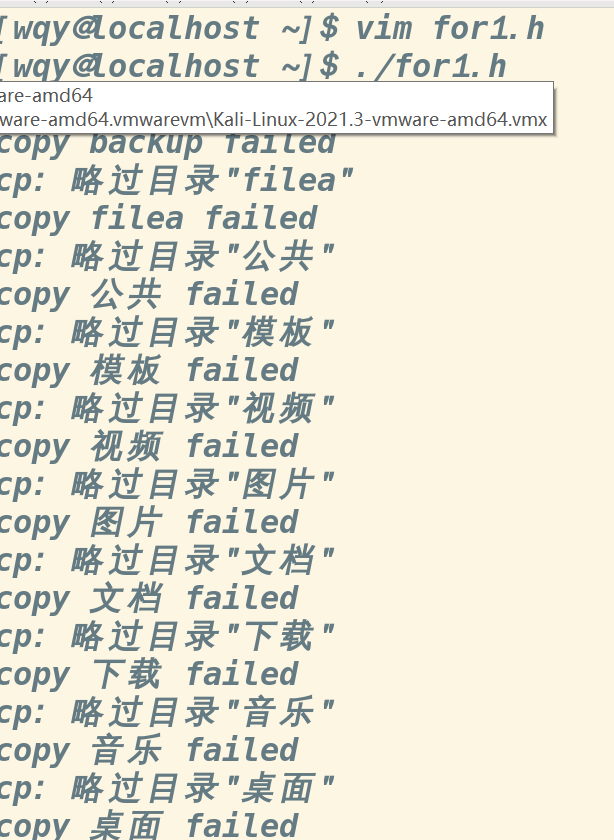
操作七修改例3程序，使在程序运行中能随机输入文件名，然后进行文件属性判断。



**操作八**用vi 编写《实验指导》“第四部分 Shell 程序设计”中的例4、例5、例6、例7，掌握控制语句的用法，观察运行结果。

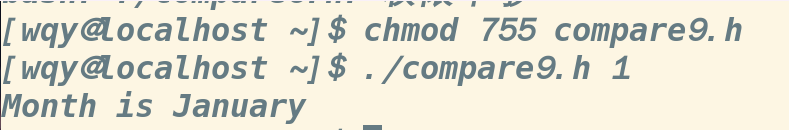


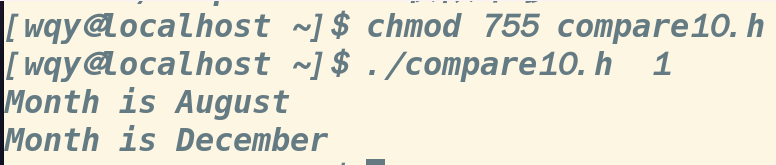




**操作九**用vi 编写《实验指导》“第四部分 Shell 程序设计”中的例8 及例9掌握条件语句的用

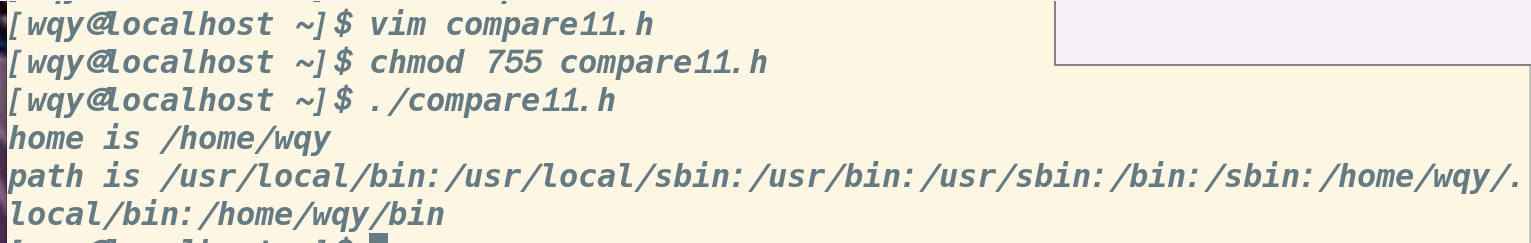
法，函数的用法，观察运行结果。





**操作十**编程，在屏幕上显示用户主目录名（HOME）、命令搜索路径（PATH），并显示由位

置参数指定的文件的类型和操作权限。



**体会：**

shell简单、灵活，对于linux系统的自动化管理非常有帮助，可以让系统管理员的工作变得容易、简单、高效。shell在linux很多地方都用到，熟悉shell命令，可以帮助解决一些故障问题。对于一个系统运维工程师来说是必要的。而且面试都会涉及到shell编程的内容。但是我们考虑到效率时，从来不用Shell编程，而更多的考虑C++和java。因为它不具有数据结构，多维数组的支持，对于我来说，本节课收获还挺多的。