**操作系统实验报告**



|  |  |
| --- | --- |
| **姓名：** | **杨瑞基** |
| **学号：** | **19121963** |

**计算机工程与科学学院**

**报告日期 2021年 11 月 11 日**

实验题目: **Linux**操作系统基本命令

姓名：杨瑞基 学号：19121963 实验日期：2021/9/9

实验环境：Ubantu操作系统

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：

过程1：

① 执行pwd查看当前目录。

② 用whoami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。

输入id命令得到用户名、用户标识、组标识

过程2：

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。“>”是一个重定向 命令。

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，注意i节点号，链接计数。

过程3：

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。

过程4：

①chmod 751 mytext.txt ls –l mytext.txt 存取控制模式的表示可用八进制或字符表示

② 执行 chown stud090 mytext.txt

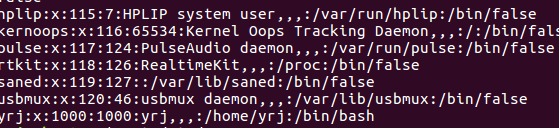
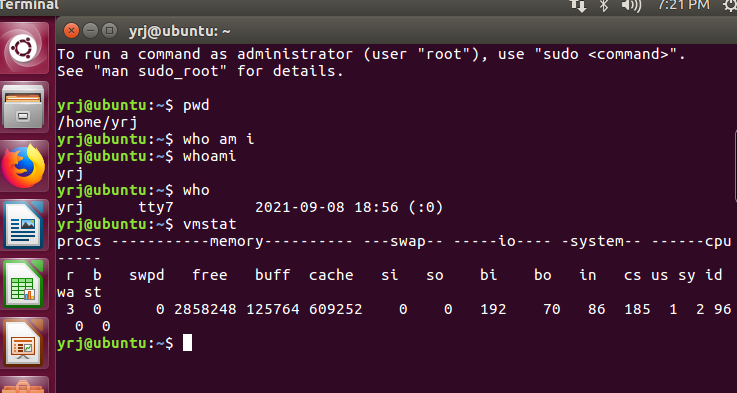
chown stud090 mytext.txt（修改文件所有者为stud090）

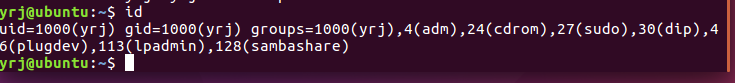
过程5：

① 执行 ps –ef 根据本《实验指导》第二部分介绍的进程管理命令选项，查看当前系统中各个进程的信息。特别注意进程号、父进程号、属主等内容。

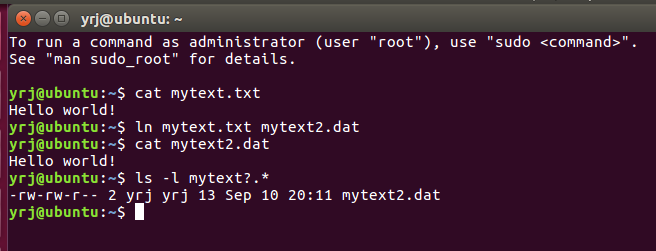
② 执行本《实验指导》第二部分介绍的wait和sleep命令。

结果：

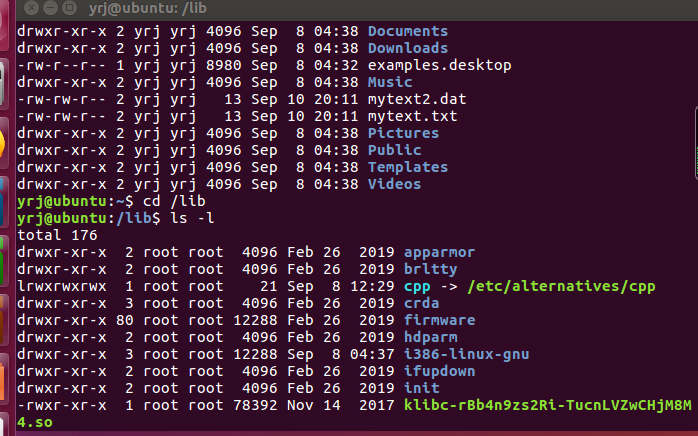




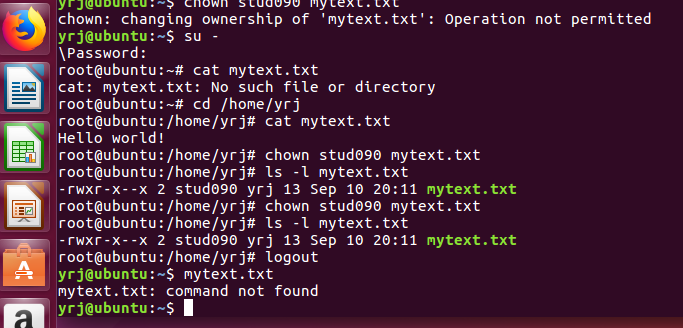
|  |
| --- |
| 思考1：你的用户名、用户标识、组名、组标识是什么？当前你处在系统的哪个位置中？现在有哪些用户和你一块儿共享系统？  我的用户名：yrj, 用户标识：1000，组名：yrj, 组标识：1000. 当前我处在系统的位置为/home/yrj:/bin/bash，没有用户和我一块共享系统 |

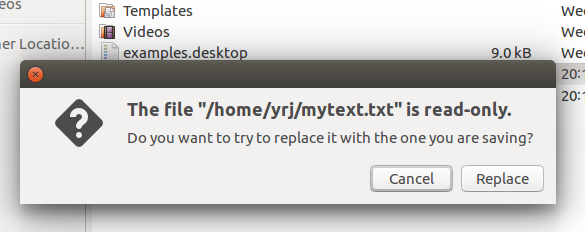


|  |
| --- |
| 思考2：文件链接是什么意思？有什么作用？  答：文件链接是一种在共享文件和访问它的用户的若干目录项之间建立联系的一种方法。作用：给系统中已有的某个文件指定另外一个可用于访问它的名称，对于这个新的文件名，我们可以为之指定不同的访问权限，以控制对信息的共享和安全性问题。 |

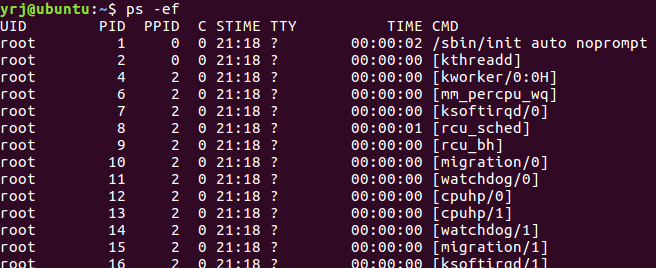


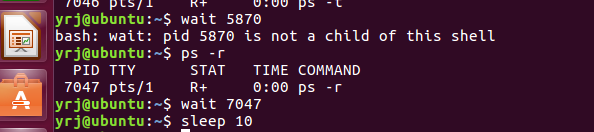
|  |
| --- |
| 思考3：Linux文件类型有哪几种类型？文件的存取控制模式如何描述？  答：Linux文件类型有普通文件(~)、目录文件(d)、块设备特别文件(b)、字符设备特别文件(c)、命名管道文件(p)等。  “存取控制模式”指对不同用户分配不同的操作权。Linux 文件系统将用户分成三类，即文件主、同组人、其他人。每种人可以行使的操作有三种，即读（r）、 写（w）、执行（x）。  权限分别用二进制位来置位，“1”表示有权、“0”表示无权。 |





|  |
| --- |
| 思考4：执行了操作过程4后，，若想再修该文件，看能不能执行。为什么？  答：不能，因为存储控制模式中的751表示：文件主可读写执行，同组成员仅可读和执行，其他人只允许执行，又因为文件mytext.txt的所有者转交给了用户stud090，故不能再修改该文件了. |





|  |
| --- |
| 思考5：系统如何管理系统中的多个进程？进程的家族关系是怎样体现的？有什么用？  答：通过设置了进程号，进程，父进程以及优先级来对系统中的多个进程进行管理；子进程由父进程所创建，父进程的返回值是子进程的进程ID；这样做提高系统的效率。 |

讨论：

|  |
| --- |
| 1. Linux系统命令很多，在手头资料不全时，如何查看命令格式？  答：在终端输入[命令]—help来显示帮助信息。  2. Linux系统用什么方式管理多个用户操作？如何管理用户文件，隔离用户空间？用命令及结果举例说明。  Linux是多用户操作系统。进入系统的用户都必须有个人帐号和口令。其将用户置于登录用户的主目录中(通常在/home 中)，启动一个 Shell进程与用户交互，以此来管理用户文件和隔离用户空间。另外对于用户文件还通过“存取控制模式”对不同用户分配不同的操作权。即文件主、同组人、其他人。每种人可以行使的操作有三种，即读（r）、写（w）、执行（x）。例如通过命令 [chown 用户名 文件名]改变文件的所有者，通过命令[chgrp 组名 文件名]改变文件的组标识，通过命令[chmod 访问模式 文件名|目录名]改变文件权限。  3. 用什么方式查看你的进程的管理参数？这些参数怎样体现父子关系？当结束一个父 进程后其子进程如何处理？用命令及结果举例说明。  答：可以在终端输入ps –help all查看管理参数，这些参数体现了父进程在子进程运行期间处于睡眠状态，以及父进程在子进程退出时收到信号而被唤醒；当结束父进程时，子进程不会结束，子进程会被init进程托管；例如d.sh脚本是一个ping命令，执行d.sh,目前  103310进程为父进程，103344为子进程，kill掉父进程后，最后发现子进程103344被托管到进程1下。  4. Linux 系统“文件”的含义是什么？它的文件有几种类型？如何标识的？  答：“文件”是泛指系统中具有存取控制操作权的程序，目录，文档等；它的文件类型有普通文件，目录文件，块设备特别文件，字符设备文件和命名管道文件等，它们分别通过用符号~、d、b、c、p来表示。  5. Linux 系统的可执行命令主要放在什么地方？找出你的计算机中所有存放系统的可执行命令的目录位置。  答：/bin 或 /usr/bin或/sbin -> /usr/bin 或/usr/bin；我的计算机所有存放系统可执行命令的目录位置为/bin, /usr/bin.  6.Linux系统的设备是如何管理的？在什么地方可以找到描述设备的信息？  答：通过设备识别和设备挂载来进行管理，可以通过lspci -v查看PCI设备，通过more /proc/cpuinf查看cpu信息，more /proc/meminfo查看内存信息，通过fdisk-l查看硬盘分区信息。  7. 画出Linux 根文件系统的框架结构。描述各目录的主要作用。你的用户主目录在哪 里？    /bin 存放必要的用户命令；/boot 引导加载程序使用的静态文件；/dec 设备文件和其它特殊文件；/etc 系统配置文件以及启动文件；/home 用户主目录；/lib 必要的链接库；/mnt 用于暂时安装文件系统；/opt 附件的软件套件；/proc 用于提供内核与进程信息的虚拟文件； /root root用户主目录；/sbin 必要的系统管理员命令；/tmp 暂时性文件；/usr 用于存放大多数用户常用的应用程序和文件；/var 监控程序和工具程序的可变程序。我的用户主目录在/home/yrj中。 8.Linux 系统的Shell是什么？请查找这方面的资料，说明不同版本的Shell的特点。答：Shell是系统的用户界面，提供了用户与内核进行交互操作的一种接口。它接收用户输入的命令并把它送入内核去执行。是在Linux内核与用户之间的解释器程序，现在Linux通常指/bin/bash解释器来负责向内核翻译以及传达用户/程序指令，shell相当于操作系统的“外壳”。大多数linux的shell默认为bash，无多大差别，而ash,zsh,ksh版本会在个别命令上外加一个重写的shell。  9. 下面每一项说明的是哪类文件。 (1）-rwxrw-r-- (5）/etc/passwd (2) /bin (6) crw-rw-rw (3) ttyx3 (7) /usr/lib (4) brw-rw-rw- (8) Linux  答 ：（1）普通文件，（2）目录文件 （3）命名管道文件（4）块设备特别文件（5）普通文件 （6）字符设备特别文件 （7）目录文件（8）目录文件 |

体会：

通过本次实验我了解了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。也在练习Linux系统命令接口的使用过程中一步步提升我对于Linux系统的熟悉程度，同时也在不断思考与讨论中逐步理解Linux的逻辑关系，间接帮助我从一个实际的角度重新认识操作系统，总而言之，在本次的实验中可谓受益匪浅。

附录：（源程序）：本次没有源程序要求。

实验题目: **Linux**操作系统基本命令

姓名：杨瑞基 学号：19121963 实验日期：2021/9/16

实验环境：Ubantu操作系统

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。

实验内容：

1. 图形化用户界面的Shell命令操作

2. 字符界面下的 Shell 命令操作

3. 通配符的使用

4. 设置手工启动图形化用户界面

操作过程：

1. 图形化用户界面的Shell命令操作

A. 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“date 091700002011”，屏幕显示新修改的系统时间。在桌面环境的终端

B. 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

C. 查看 ls 命令的-s 选项的帮助信息

操作步骤：

（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。

D. 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

Linux操作系统具有 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

A. 查看当前目录

操作步骤：

（1）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和口令，登录系统。

（3） 输入命令“pwd”，显示当前目录

B. 用 cat 命令在用户主目录下创建一名为 f1 的文本文件

（1） 输入命令“cat >f1”，屏幕上输入点光标闪烁，依次输入文字内容。

（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。

C. 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

D. 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 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

E. 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

F. 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

G. 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。

A. 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

B. 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

C. 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

注：[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。

D. 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

E. 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s /bin/[!a-u]\*

F. 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。

4.设置手工启动图形化用户界面

A. 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

（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。

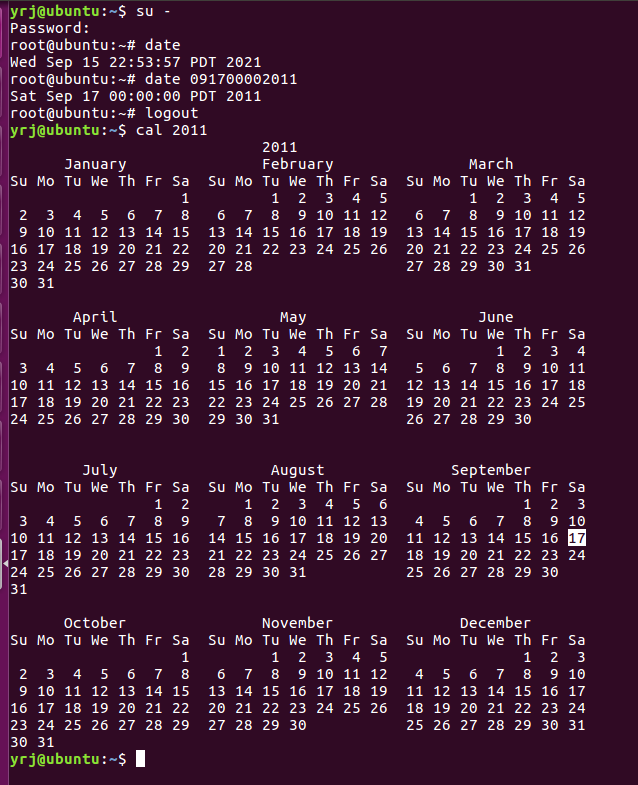
B. 手工启动图形化用户界面。

（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。

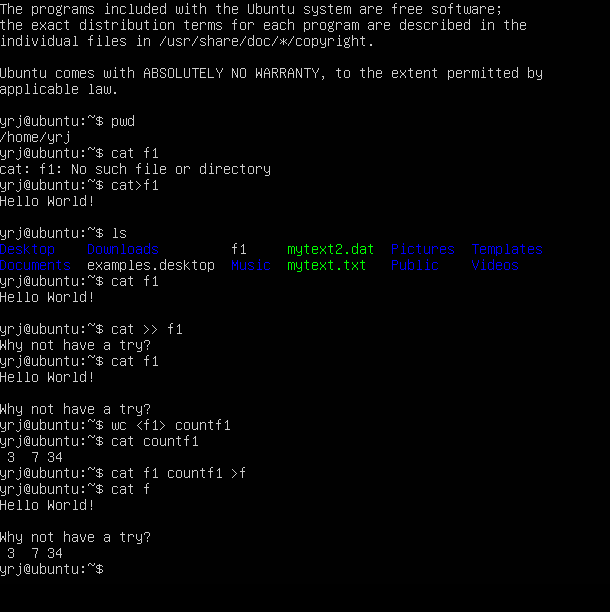
实验结果：

1. 输入命令“cal 2011”，屏幕上显示出 2011 年的日历，从结果可得知2011 年 9 月 17 日是星期六。

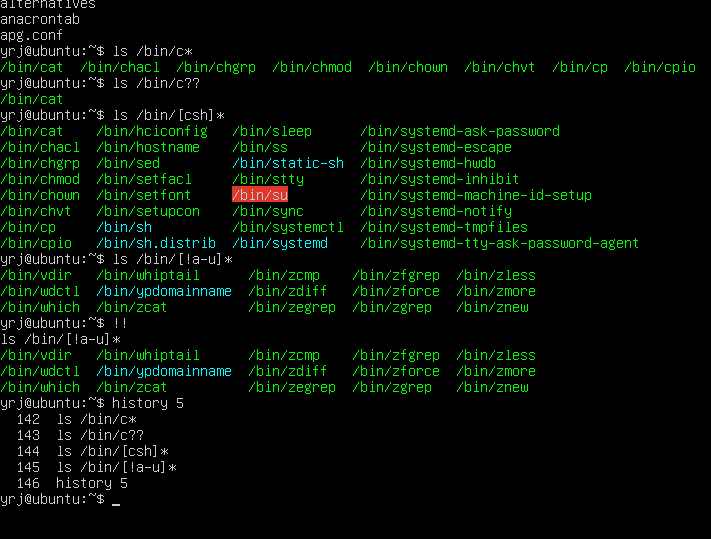


2. 启动计算机后默认会启动图形化用户界面，按下 CTRL+ALT+F1 键切换到第 1 个虚 拟终端。



3. 用 cat 命令在用户主目录下创建一名为 f1 的文本文件，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，将 f1 和 countf1 文件的合并为 f 文件。

4. 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。

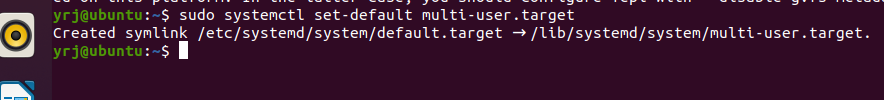


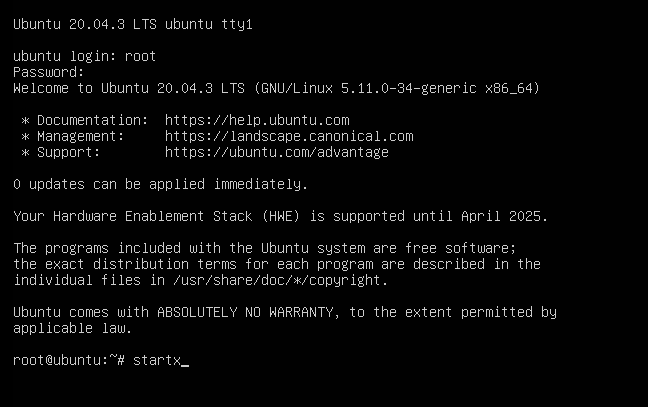
5. 设置开机不启动图形化用户界面，手工启动图形化用户界面。

现ubuntu系统不再提供inittab 文件的编辑。但通过查阅资料得知可通过输入命令来设置开机默认进入命令行吗模式：

1、输入命令：sudo systemctl set-default multi-user.target

2、重启：reboot  
要进入图形界面，只需要输入命令startx  
从图形界面切换回命令行：ctrl+alt+F7





通过startx进入的图形化界面略有些不同，可以明显发现没有用户文件夹与回收站的图表



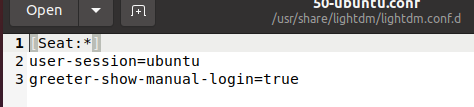
实验中遇到的问题与解决方案：

问题一： Ubuntu不能够直接通过root用户登录进入图形化界面，通过尝试多种网络上提供的方法，最后找到一套极为有效的方式。

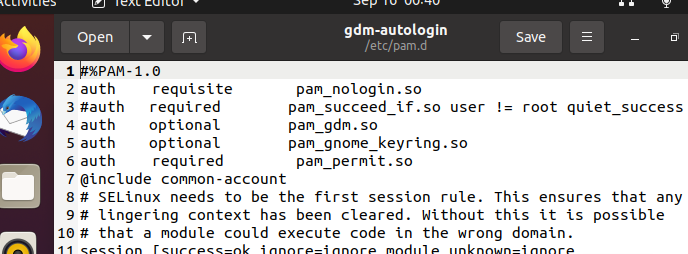
a.修改lightdm.conf.d文件。 设置好root账户和密码后，在命令行输入：sudo gedit /usr/share/lightdm.conf.d/50-ubuntu.conf, 然后编辑 50-ubuntu.conf文件，在文件添加两行代码：

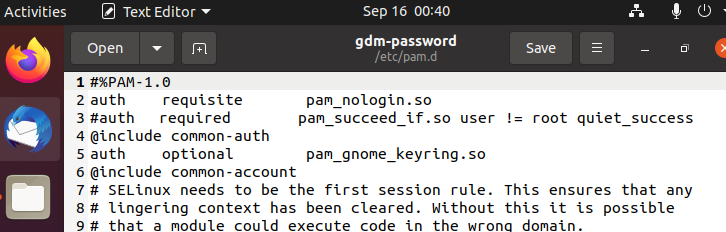
greeter-show-manual-login=true #手工输入登陆系统的用户名和密码

allow-guest=false #不允许guest登录（可选）



b. 修改gdm-autologin、gdm-password文件。分别输入sudo gedit /etc/pam.d/autologin; sudo gedit /etc/pam.d/password; 将两份文件中的第二行代码都注释掉。



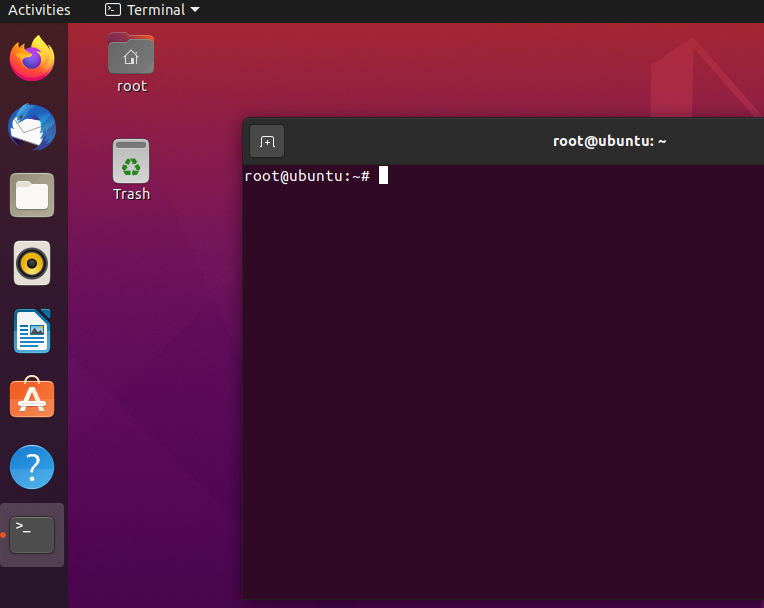


C. 修改.profile文件。输入命令：sudo gedit /usr/share/lightdm/lightdm.conf.d

将文件中mesg n改为tty -s && mesg



重启后即可登录root用户



问题二：由于实验指导书的老旧，如今的ubuntu系统不再适用于哪种方法来设置开机默认进入命令行模式，查阅资料后得到：

一、开机默认进入命令行模式  
1、输入命令：sudo systemctl set-default multi-user.target 2、重启：reboot  
要进入图形界面，只需要输入命令startx  
从图形界面切换回命令行：ctrl+alt+F7

二、开机默认进入图形用户界面  
1、输入命令：sudo systemctl set-default graphical.target 2、重启：reboot  
要进入命令行模式：ctrl+alt+F2  
从命令行切换到图形界面：ctrl+alt+F7

体会：

通过本次实验的学习，我重新认识到了操作系统在文件管理以及命令设置上的逻辑关系，也达到掌握图形化用户界面和字符界面下使用 Shell 命令的方法。 掌握 ls、cd 等 Shell 命令的功能， 掌握重定向、管道、通配符、历史记录等的使用方法， 掌握手工启动图形化用户界面的设置方法的目标。更记忆尤甚的是在面对未知的问题上，积极寻求解决方案，在实现目标的同时，也对最近Linux的更新与优化上的改变有了清楚的认识。

附录：（源程序）：本次没有源程序要求。

实验题目: **Linux**操作系统基本命令

姓名：杨瑞基 学号：19121963 实验日期：2021/9/23

实验环境：Ubantu操作系统

实验目的：

1. 利用Linux提供的系统调用设计程序，加深对进程概念的理解。

2. 体会系统进程调度的方法和效果。

3. 了解进程之间的通信方式以及各种通信方式的使用。

实验内容：

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。

2. 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。

3. 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。

4. 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。

5. 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。

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。

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。

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。

9. 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。

实验结果及思考：

|  |
| --- |
| 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。 |
| 2. 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。    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：通过调用fork()函数把当前进程作为父进程，创建一个子进程。当进程中程序段执行完毕或执行exit(0)后便自动结束。 子进程被创建后系统会将父进程中fork()之后的代码复制进子进程中的代码段中，分配所需的资源后构成运行环境。 |
| 3. 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。    思考：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。 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.  答：①父进程在通过fork()创建子进程后，会将逻辑上在此之后所需运行的代码复制给子进程的代码段，系统在分配需要的资源之后建立起子进程的运行环境。子进程的反复运行此程序发现父进程的PID是同样的，但子进程的PID每一次的PID都不相同。且从每次返回情况的情况上来看，进程之间构成类似于递归样的执行顺序。  ②修改程序如下：   |  | | --- | | #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  int main()  {  int i,j,x=0;  int flag=1;  printf("My pid is %d, my father’s pid is %d\n",getpid(),getppid());  for(i=0; i<3; i++)  {  if(flag ==1)  {  x = fork();  }  if(x==0)  {  printf("%d pid=%d ppid=%d\n",i,getpid(),getppid());  }  if(x!=0&&flag!=0)  {  flag = 0;  j=wait(0);  printf("%d:The child %d is finished.\n",getpid(),j);  }  }  exit(0);  } |   运行结果：    加入变量flag用于标记父进程是否已创建子进程，这样就保证每一个父进程在创建完一个子进程后不会再继续创建子进程  4. 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。    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：多次反复运行此程序可发现子进程执行和终止时间并无顺序或规律所寻，这体现了进程的独立性和异步性，即各自按独立的、不可预知的速度向前推进。 |
| 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。    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：不会，通过实验例程4的接过来看，先执行子程序在其进程中修改了全局公共变量globa和私有变量vari后，再执行父进程，其变量值仍为初始状态。这体现了进程的独立性，且能够保留程序进入就绪状态前的现场运行环境。 |
| 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。    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。  ②修改睡眠时机、睡眠长度，看看会有什么变化。请解释。  ③加锁、解锁起什么作用？不用它行吗？  答：①管道在逻辑上被看作管道文件，在物理上则由文件系统的高速缓冲区构成。管道通信是进程的一种通信方式。Linux 中两个进程可以通过管道来传递消息。管道用 pipe() 系统调用建立。发送信息的进程用 write()把信息写入管道，接收进程用 read()从管道中读取信息。  ②修改睡眠时机、睡眠长度会发现进程苏醒的时机也会变化，进而影响通信的时机。  ③解决临界资源共享的问题，就注释掉加锁和解锁的代码段后的程序运行结果来看，不用它是可行，因为系统执行程序时并未访问到临界资源。 |
| 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。    思考： 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。  答：一个父进程退出，而它的一个或多个子进程还在运行，那么那些子进程将成为孤儿进程。孤儿进程将被init进程所收养，并由init进程对它们完成状态收集工作。 |
| 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。    思考：想一下服务者程序和客户程序的通信还有什么方法可以实现？  答：还可以利用基于网络型的socket套接字的方式建立服务器程序和客户程序的通信。 |
| 9. 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。    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？  答：1. 硬中断是由硬件产生的，软中断由当前正在运行的进程所产生的。  2. 硬中断可以直接中断CPU。软中断并不会直接中断CPU。也只有当前正在运行的代码（或进程）才会产生软中断。 |

研究与讨论

实验结果：

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：系统通过创建、销毁、睡眠、激活、管道通信、中断等方式来管理进程。

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：引用<unistd.h>、<sys/types.h>、<sys/wait.h>等库，如上述实现代码都通过调用这些库所包含的函数来对Linux系统下执行的程序中的进程进行管理。

3. 什么是进程？如何产生的？举例说明。

答：进程是具有独立功能的程序在一个数据集合上运行的过程，它是系统进行资源分配和调度的一个独立单位。通常情况下进程由其父进程创建，例如本次实验中父进程通过fork创建其子进程。

4. 进程控制如何实现的？举例说明。

答：进程控制一般是由OS的内核中的原语来实现的。如OS将利用原语suspend将指定进程或处于阻塞状态的进程挂起。

5. 进程通信方式各有什么特点？

答：1. 管道通信：它是一种半双工的通信方式，数据只能单向流动，而且只能在有父子关系的进程间使用。 2. 信号量：它是一种计数器，可以用来控制多个线程对共享资源的访问，不用于交换大批数据，可用于多线程之间的同步。 3. 信号：这是一种比较复杂的通信方式，可用于通知进程某个事件已经发生。4. 消息队列：消息队列是消息的链表，存放在内核中并由消息队列标识符标识，消息队列克服了信号传递信息少的问题。 5. 共享内存：这是一种

映射一段能被其它进程所访问的内存，这段共享内存由一个进程创建，但多个进程都可以访问。 6. 套接字：可用于不同及其间的进程通信。

6. 管道通信如何实现？该通信方式可以用在何处？

答：管道通信的实现方式是在父进程中创建管道。再创建子进程，由于子进程继承父进程打开的文件描述符，所以父子进程就可以通过创建的管道进行通信。该通信方式可用于父子进程间的通信

7. 什么是软中断？软中断信号通信如何实现

答：软中断是通讯进程之间用来模拟硬中断的 一种信号通讯方式是软件实现的中断,也就是程序运行时其他程序对它的中断。一般情况下，通过系统调用fork()创建两个子进程，再用系统调用signal()让父进程捕捉中断信号，当捕捉到中断信号后，父进程通过调用系统kill()向子进程发出信号，子进程捕捉到信号后便进行中断处理。

**体会**

本次实验通过编程实现的方式来实现系统对进程的管理，增强自身代码能力的同时，也对课堂上所提到进程相关概念和知识有了更加深刻的理解与认识，更为重要的是，在编程实现的过程之中，遇到诸多不熟悉的概念和棘手的问题，都坚持积极地查阅资料加深对进程的认识，也正是通过该实验，让“进程“这一原本只是躺在课本上的概念在我的脑子里已然是一个更加生动且灵活的存在。

**附录（源代码）**

|  |
| --- |
| #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  int main()  {  int i,child\_pid,pid,status;  char yes\_or\_no;  printf("进程标识是 %d, 组标识是 %d，用户标识是 %d\n",getpid(),getgid(),getuid());  sleep(5);  child\_pid=fork(); /\*创建子进程3\*/  if(child\_pid==0)  {  execlp("./jincheng1","./jinghcneg1",(char \*)0); /\*子进程 启动其它程序\*/  perror("exec3 error.\n ");  exit(3);  }  while((pid=wait(&status))!=-1) /\*等待子进程结束\*/  {  if(child\_pid==pid) /\*若子进程 结束\*/  printf("child process terminated with status %d\n",(status>>8));  }  puts("Child processes terminated.");  printf("是否结束程序？");  scanf("%c", &yes\_or\_no);  if (yes\_or\_no == 'Y')  {  puts("Parent processes terminated.");  exit(0);  }  else if (yes\_or\_no == 'N')  {  puts("Parent processes is not terminated.");  }  } |
| #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  /\*例程1：利用fork()创建子进程  用fork()系统调用创建子进程的例子\*/  int main()  {  int i;  if (fork()) {/\*父进程执行的程序段\*/  i = wait(0); /\* 等待子进程结束\*/  printf("It is parent process.\n");  printf("The child process,ID number %d, is finished.\n", i);  }  else  {  printf("It is child process.\n");  sleep(2);  /\*子进程执行的程序段\*/  exit(0); /\*向父进程发出结束信号\*/  }  exit(0);  } |
| /\*例程2：循环调用fork()创建多个子进程。  建立进程树\*/  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  int main()  {  int i,j;  bool flag=false;  printf("My pid is %d, my father’s pid is %d\n",getpid(),getppid());  for(i=0; i<3; i++)  {  if(fork()==0)  {  printf("%d pid=%d ppid=%d\n",i,getpid(),getppid());  }  else  {  j=wait(0);  flag = true;  printf("%d:The child %d is finished.\n",getpid(),j);  }  }  exit(0);  } |
| /\*例程3:创建子进程并用execlp()系统调用执行程序的实验  创建子进程，子进程启动其它程序\*/  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  int main()  {  int child\_pid1,child\_pid2,child\_pid3;  int pid,status;  setbuf(stdout,NULL);  child\_pid1=fork(); /\*创建子进程1\*/  if(child\_pid1==0)  {  execlp("echo","echo","child process 1",(char \*)0); /\*子进程1 启动其它程序\*/  perror("exec1 error.\n ");  exit(1);  }  child\_pid2=fork(); /\*创建子进程2\*/  if(child\_pid2==0)  {  execlp("date","date",(char \*)0); /\*子进程2 启动其它程序\*/  perror("exec2 error.\n ");  exit(2);  }  child\_pid3=fork(); /\*创建子进程3\*/  if(child\_pid3==0)  {  execlp("ls","ls",(char \*)0); /\*子进程3 启动其它程序\*/  perror("exec3 error.\n ");  exit(3);  }  puts("Parent process is waiting for chile process return!");  while((pid=wait(&status))!=-1) /\*等待子进程结束\*/  {  if(child\_pid1==pid) /\*若子进程1 结束\*/  printf("child process 1 terminated with status %d\n",(status>>8));  else  {  if(child\_pid2==pid) /\*若子进程2 结束\*/  printf("child process 2 terminated with status %d\n",(status>>8));  else  {  if(child\_pid3==pid) /\*若子进程3 结束\*/  printf("child process 3 terminated with status %d\n" ,(status>>8));  }  }  }  puts("All child processes terminated.");  puts("Parent process terminated.");  exit(0);  } |
| /\*例程4：观察父、子进程对变量处理的影响  创建子进程的实验。子进程继承父进程的资源，修改了公共变量globa 和私有变  量vari。观察变化情况。\*/  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  int globa=4;  int main()  {  pid\_t pid;  int vari=5;  printf("before fork.\n");  if ((pid=fork())<0) /\*创建失败处理\*/  {  printf("fork error.\n");  exit(0);  }  else  if(pid==0)  { /\*子进程执行\*/  globa++;  vari--;  printf("Child %d changed the vari and globa.\n",getpid());  }  else /\*父进程执行\*/  printf("Parent %d did not changed the vari and globa.\n",getpid());  printf("pid=%d, globa=%d, vari=%d\n",getpid(),globa,vari); /\*都执行\*/  exit(0);  } |
| /\*例程5：管道通信的实验\*/  /\*程序建立一个管道fd\*/  /\*父进程创建两个子进程P1、P2 \*/  /\*子进程P1、P2 分别向管道写入信息\*/  /\*父进程等待子进程结束，并读出管道中的信息\*/  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  int main()  {  int i,r,j,k,l,p1,p2,fd[2];  char buf[50],s[50];  pipe(fd);/\*建立一个管道fd\*/  while((p1=fork())==-1);/\*创建子进程1\*/  if(p1==0)  {  lockf(fd[1],1,0); /\*子进程1 执行\*/  /\*管道写入端加锁\*/  sprintf(buf,"Child process P1 is sending messages! \n");  printf("Child process P1! \n");  write(fd[1],buf,50);  lockf(fd[1],0,0);  /\*信息写入管道\*/  /\*管道写入端解锁\*/  sleep(5);  j=getpid();  k=getppid();  printf("P1 %d is weakup. My parent process ID is %d.\n",j,k);  exit(0);  }  else  {while((p2=fork())==-1);/\*创建子进程2\*/  if(p2==0)  {/\*子进程2 执行\*/  lockf(fd[1],1,0);/\*管道写入端加锁\*/  sprintf(buf,"Child process P2 is sending messages! \n");  printf("Child process P2! \n");  write(fd[1],buf,50);/\*信息写入管道\*/  lockf(fd[1],0,0);/\*管道写入端解锁\*/  sleep(5);  j=getpid();  k=getppid();  printf("P2 %d is weakup. My parent process ID is %d.\n",j,k);  exit(0);  }  else  {l=getpid();  wait(0);  if(r=read(fd[0],s,50)==-1)  printf("Can't read pipe. \n");  else  printf("Parent %d: %s \n",l,s);  wait(0);  if(r=read(fd[0],s,50)==-1)  printf("Can't read pipe. \n");  else  printf("Parent %d: %s \n",l,s);  exit(0);  }  }  } |
| /\*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);  } |
| /\* 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。\*/  #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  int main()  {  int i,j,k;  void func();  signal(18,&func); /\*设置18 号信号的处理程序\*/  if(i=fork())/\*创建子进程\*/  {/\*父进程执行\*/  j=kill(i,18);/\*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(0); /\*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\*/  //exit(0);  }  void func() /\*处理程序\*/  {  int m;  m=getpid();  printf("I am Process %d: It is signal 18 processing function.\n",m);  } |

实验题目: **Linux**进程调度与系统监视

姓名：杨瑞基 学号：19121963 实验日期：2021/10/9

实验环境：Ubantu操作系统

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：

1. 作业和进程的基本管理

2. at 进程调度

3. corn 进程调度

4. 系统性能监视

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。

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

【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。

（1） 输入命令“jobs”，查看当前系统中的所有作业。

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

（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”.

（5） 输入命令“kill -9 %1”，终止“vi f4”作业。

（6） 稍等片刻，输入命令“jobs”

（7） 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

（8） 输入命令“who –H”，查看用户信息。

2. at进程调度

at调度：在指定的时间执行一次特定的作业

【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 01012022”，设置 2022 年 1 月 1 日 0 时执行的 at 调度的内容。

（2） 屏幕出现 at 调度的命令提示符“at>”，输入“wall 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。

【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。

光标移动到“at>”提示符的第三行，按下 Ctrl+D 组合键结束输入。

【操作要求 3】查看所有的 at 调度，并删除 22 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。

（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。

（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. corn进程调度

corn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】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。

（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。

（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。

（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。

（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。

4.系统性能监视

【操作要求 1】利用 Shell 命令监视系统性能

（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

（2） 按下 M 键，所有进程按照内存使用率排列

（3） 按下 T 键，所有进程按照执行时间排列

（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） 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

（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，

【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

（3） 「文件系统」选项卡显示全部的文件系统的信息

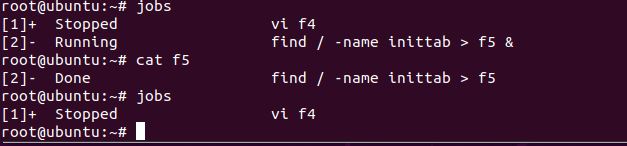
【操作要求 4】利用「系统日志」工具查看系统日志

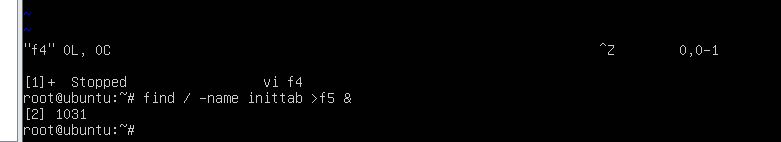
【操作步骤】

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

实验结果：

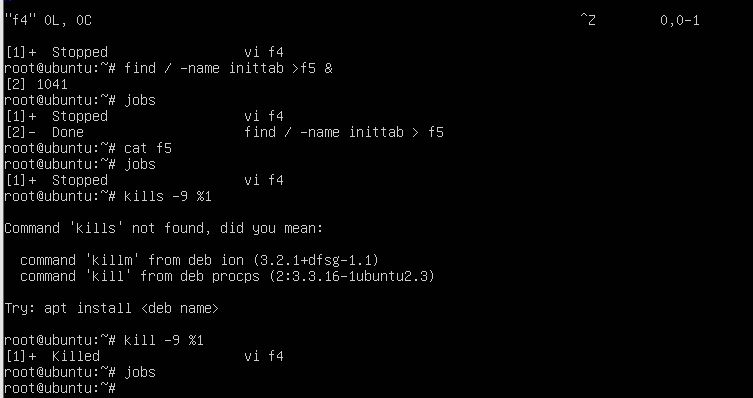
输入命令“find / -name inittab > f5 & “，启动一个后台作业，在显示作业号的同时还显示进程号。



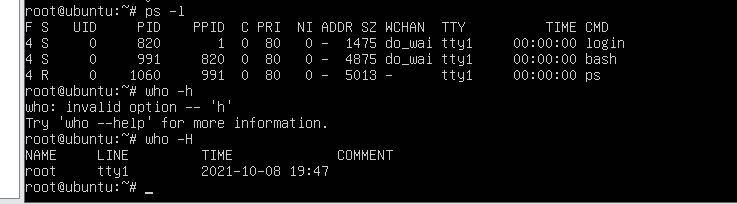


输入命令“kill -9 %1”，终止“vi f4”作业。当再次输入命令“jobs”，查看到当前没有任何作业。

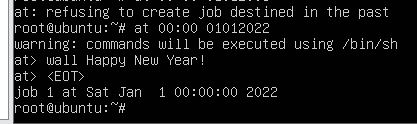




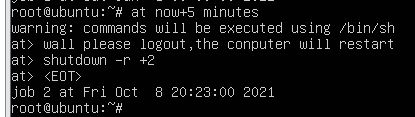
输入命令“ps –l”，查看到进程的相关信息，输入命令“who –H”，查看用户信息。其中NAME、LINE、TIME分别是用户名，用户登录的终端号、用户登录的时间。



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。

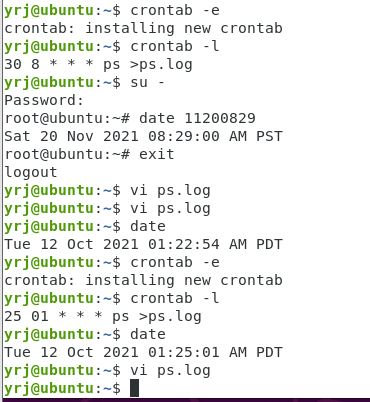


设置一个调度，要求 5 分钟后向所有用户发送系统即将重启的消息，并在 2 分钟后重新启动计算机。

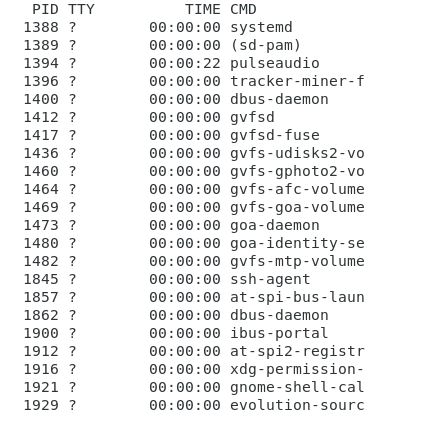




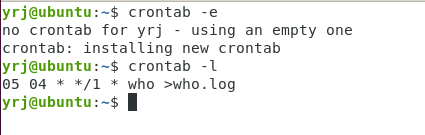
用户设置crontab 调度，要求每天上午 1 点 38 份查看系统的进程状态， 并将查看结果保存于 ps.log 文件。

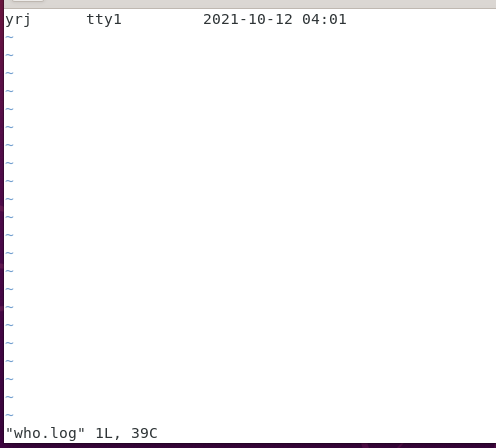


等待1分钟后ps.log 文件，结果如下：

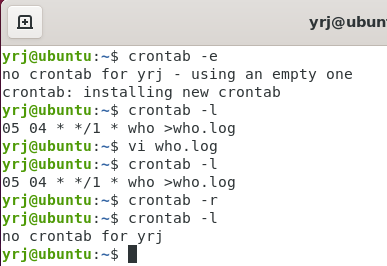


用户添加设置 crontab 调度，要求每隔一个月的4时05分查看正在使用的用户列表。

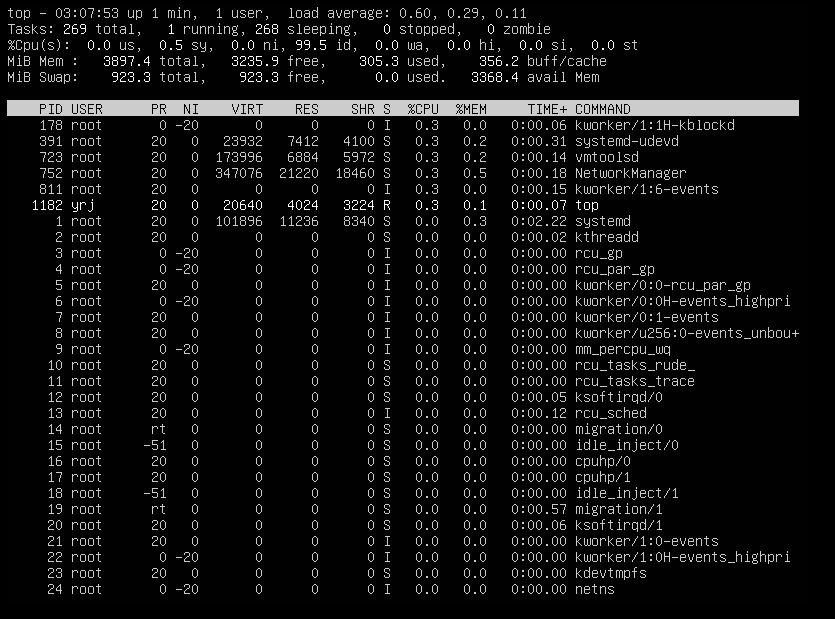




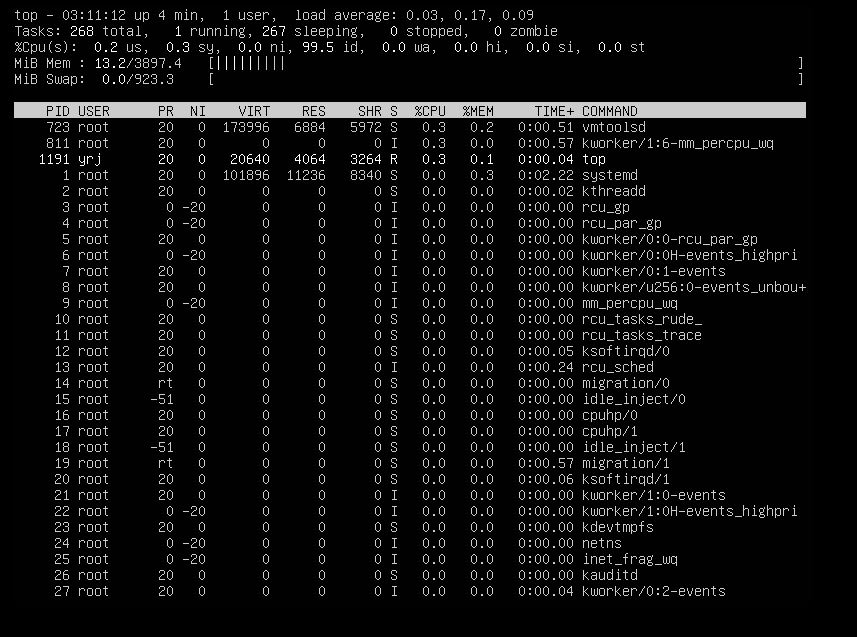
输入命令“crontab -r”，删除 cron 调度内容。



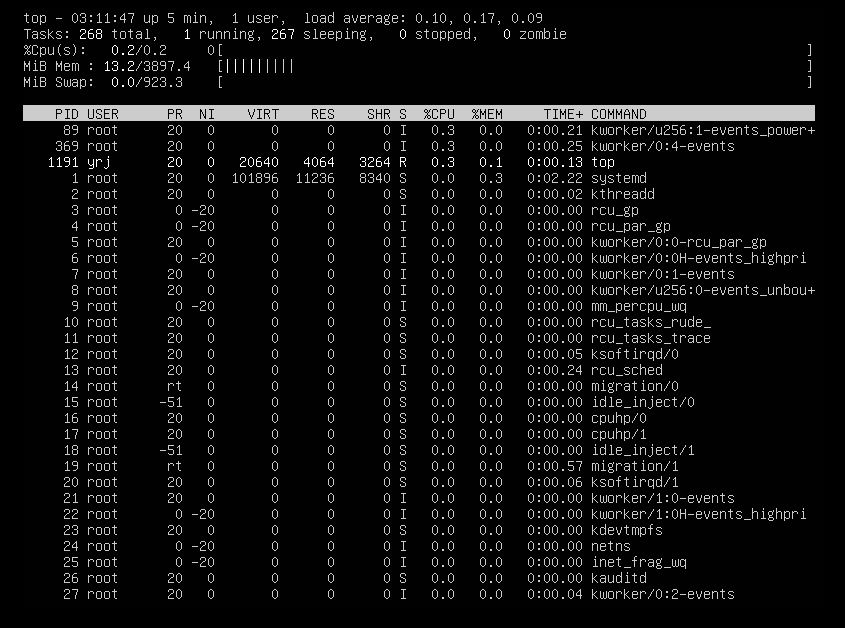
输入命令“top”，屏幕动态显示 CPU 利用率、内存利用率和进程状态等相关信息



图表 1按 cpu 使用率显示进程信息



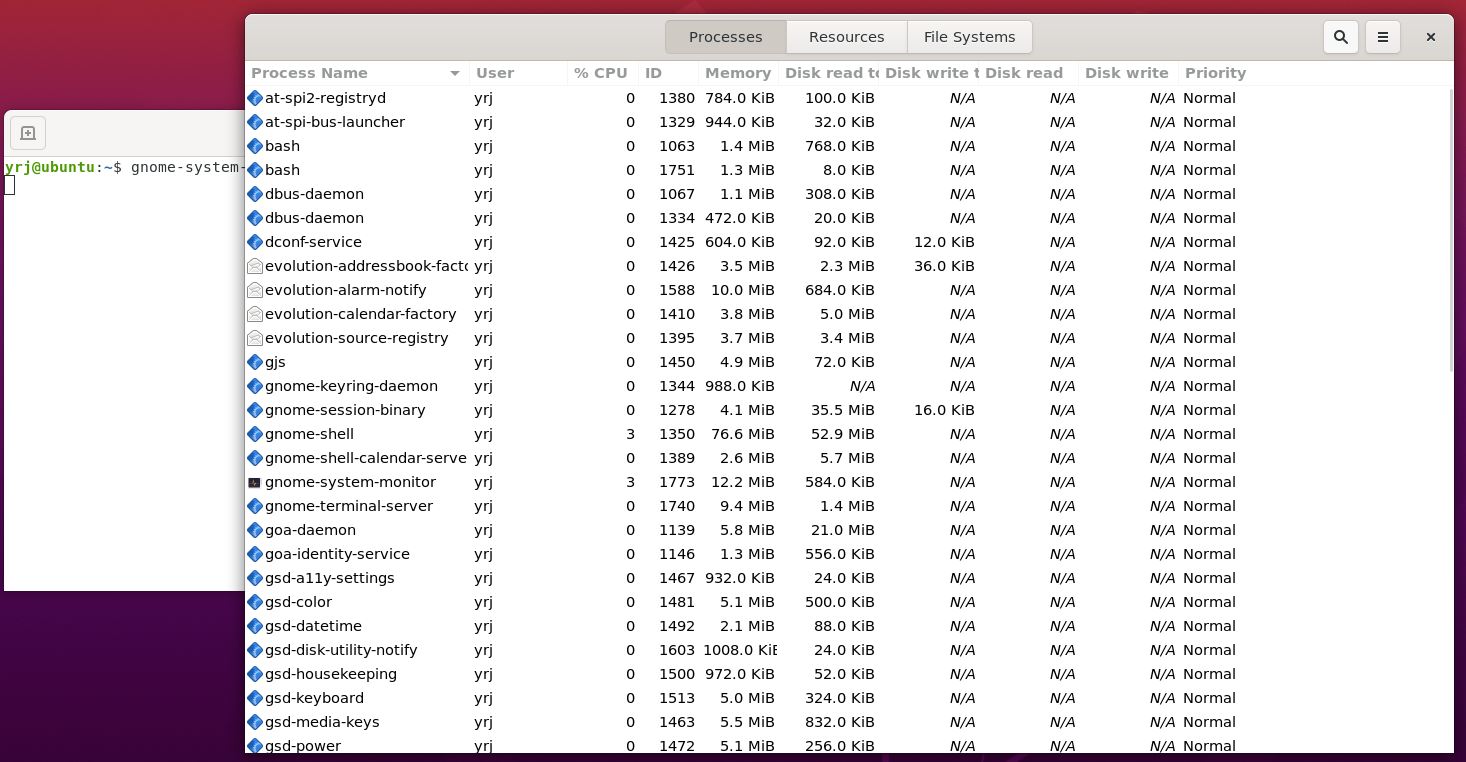
图表 2按下 M 键，所有进程按照内存使用率排列



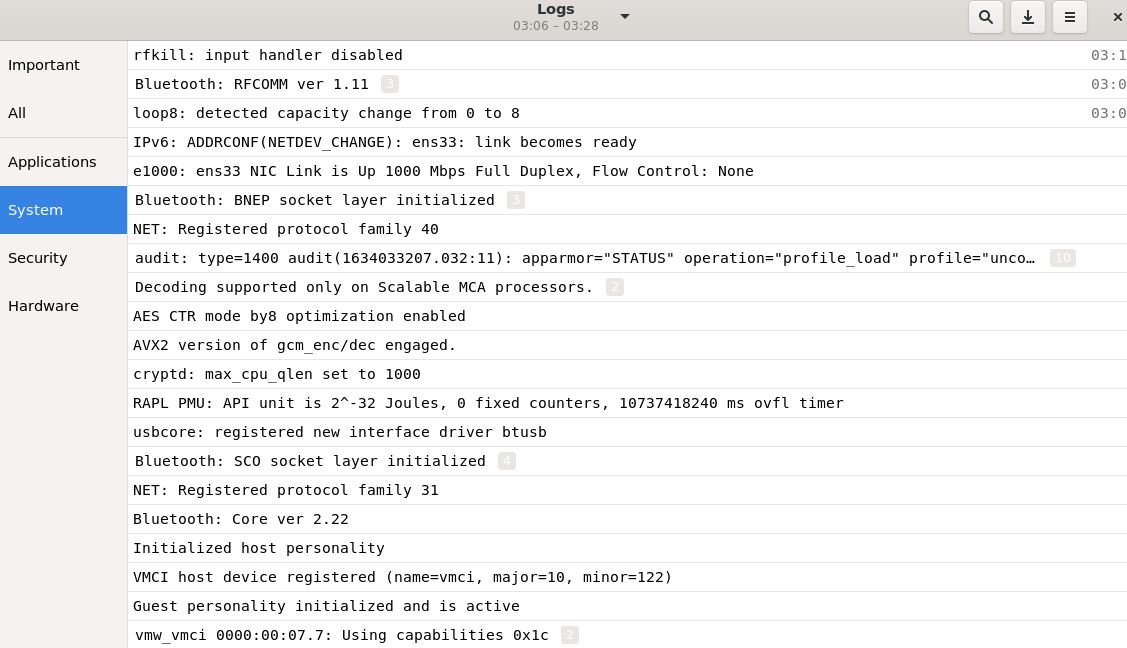
图表 3 按执行时间显示进程信息

利用「系统监视器」工具监视 CPU 使用情况。



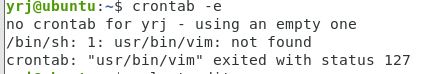


利用「系统日志」工具查看系统日志

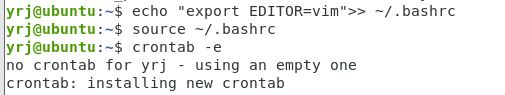


实验问题及解决方法：

实验中在设置crontab的时候，遇到报错crontab: "usr/bin/vim" exited with status 127，无法正常新建crontab配置文件。



查阅资料后找解决方案需要将默认编辑器改为vim，重新设置后便可正常新建crontab配置文件。



附录：本次实验没有源程序要求

实验题目：用户与组群管理

姓名：杨瑞基 学号：19121963 实验日期：2021/10/21

实验环境：Ubantu操作系统

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

4.批量新建多个用户帐号

**操作过程：**

一、**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）进入系统设置[settings]找到Users选项添加用户[Add Users], 新建用户帐号，用户名为 xuser1，口令为“e12ut59er”。

（2）按照同样的方法新建用户xuser2，其口令为“wfu1t28er”。

（3）通过命令 vi /etc/passwd和vi /etc/shadow查看文件，发现文件末尾出现表示 xuser1 和 xuser2户帐号的信息。

(4) 按照同样的方法查看etc目录下的group和gshadow文件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”。

（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。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。

【操作要求3】删除xuser2用户

（1）进入系统设置[settings]找到Users，选中xuser2后点击Remove User

（2）在跳出的提示框内点击Delete Files

（3）返回用户管理界面发现xuser2已消失不见，说明­ xuser2用户已被删除

【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”，单击「确定」按钮，返回「用户管理者」窗口。

（3） 用相同的方法新建 temp 组群。

【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。

【操作要求6】删除 temp 组群

【操作步骤】

（1）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。

**二、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。

（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”。单击「添加」按钮，回到「User Profile Editor」窗口。

【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。

（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，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。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。

（2） 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。

**三、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。

（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”。

（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。

【操作要求3】

【操作步骤】

1. 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）。

【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。

（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】

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

**四、批量新建多个用户账号**

【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，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，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，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。

（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。

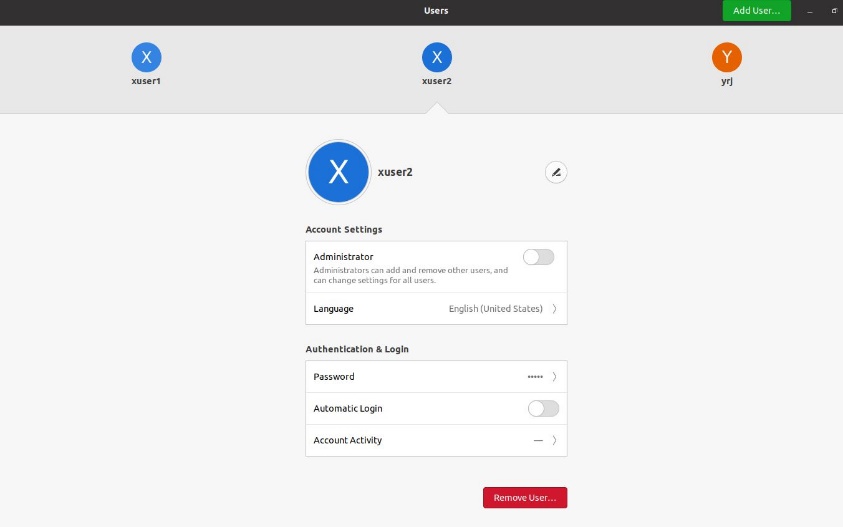
（13） 输入命令“pwconv”，进行 shadow 加密，完成批量创建用户帐号工作。

（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。

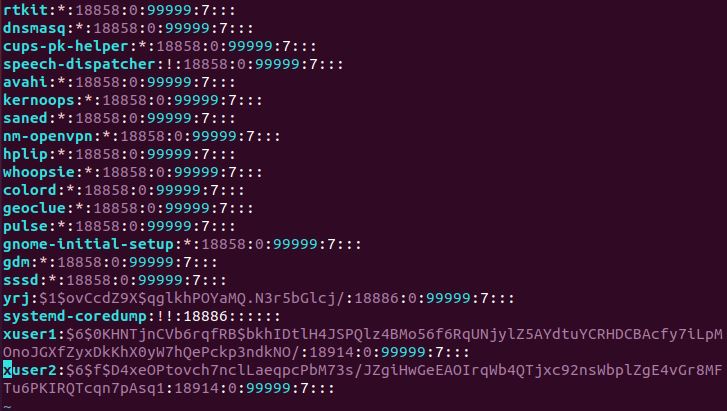
（15） 可尝试以新建的用户名登录，并应该及时修改用户的口令。

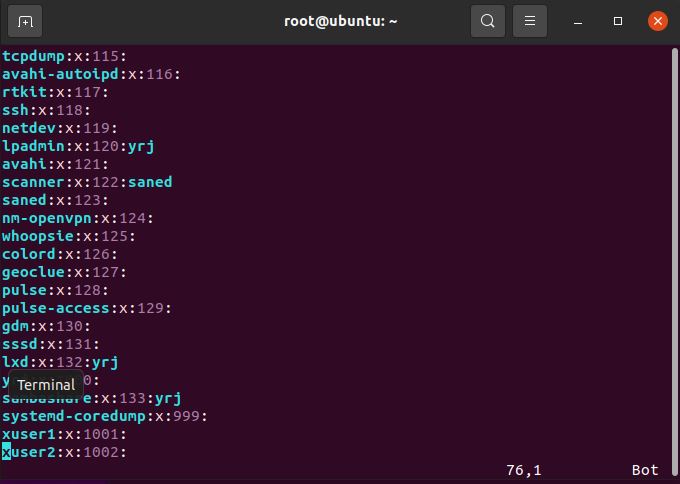
**实验结果**

创建新用户xuser1和xuser2如下：

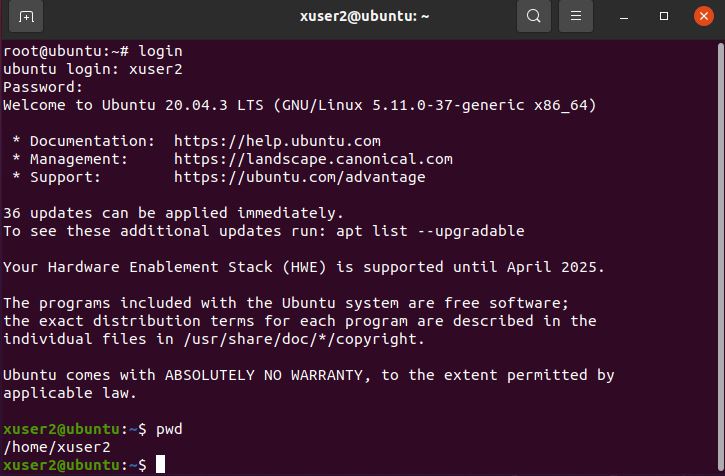
****

发现etc目录下的passwd和shadow文件多了有关xuser1和xuser2的相关内容

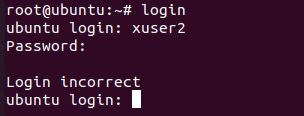


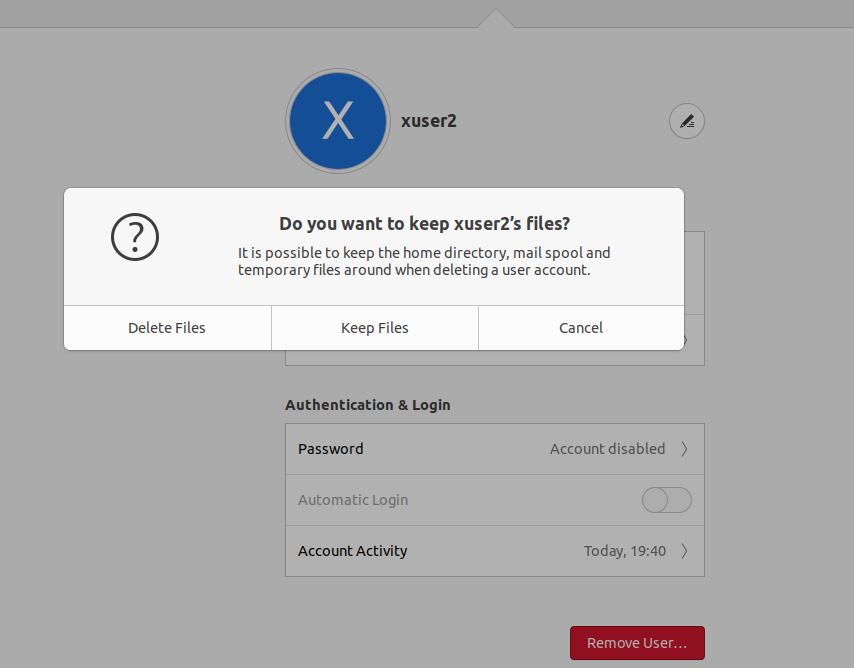


切换到第 2 个虚拟终端，输入用户名 xuser2 和相应的口令可登录 Linux 系统，说明新建用户操作已成功。

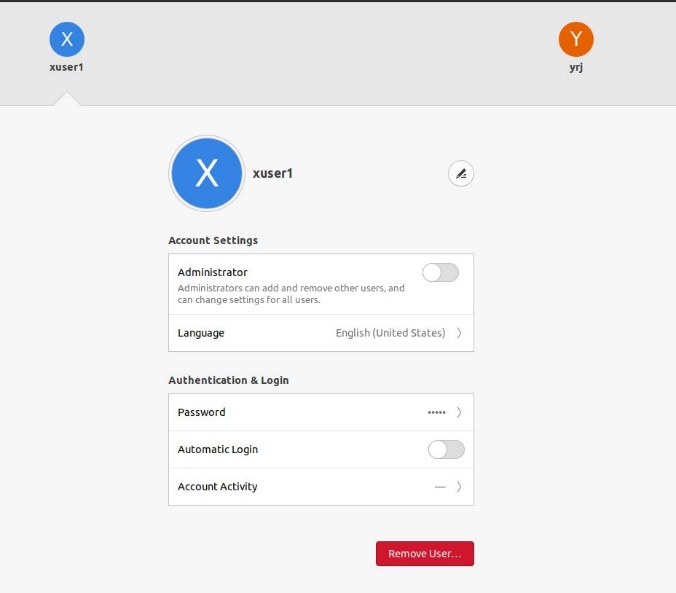


锁定用户后发现xuser2 用户无法登录 Linux 系统，说明 xuser2 用户账号的确已被锁定。

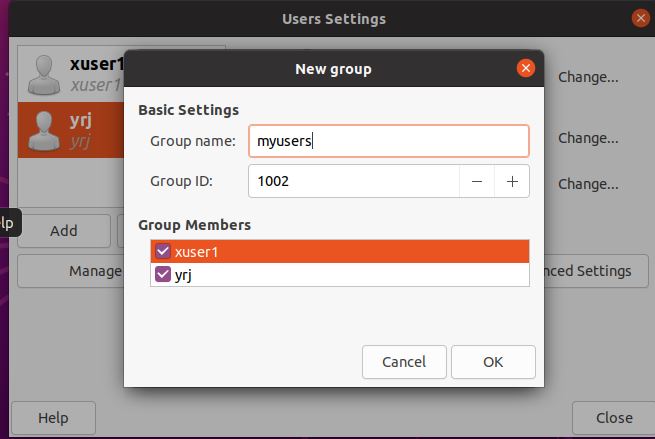




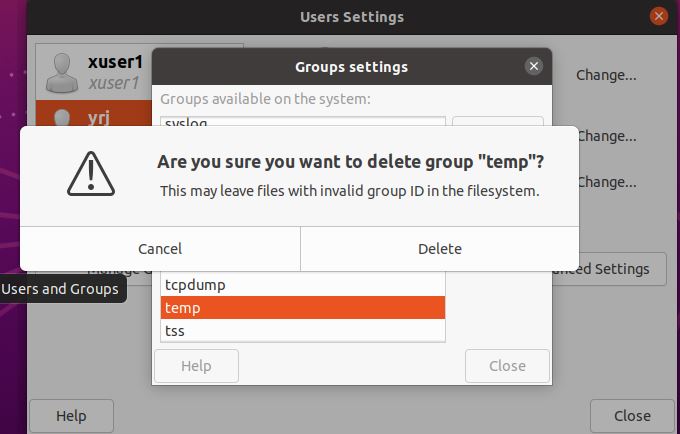
返回用户管理界面发现xuser2已消失不见，说明­ xuser2用户已被删除



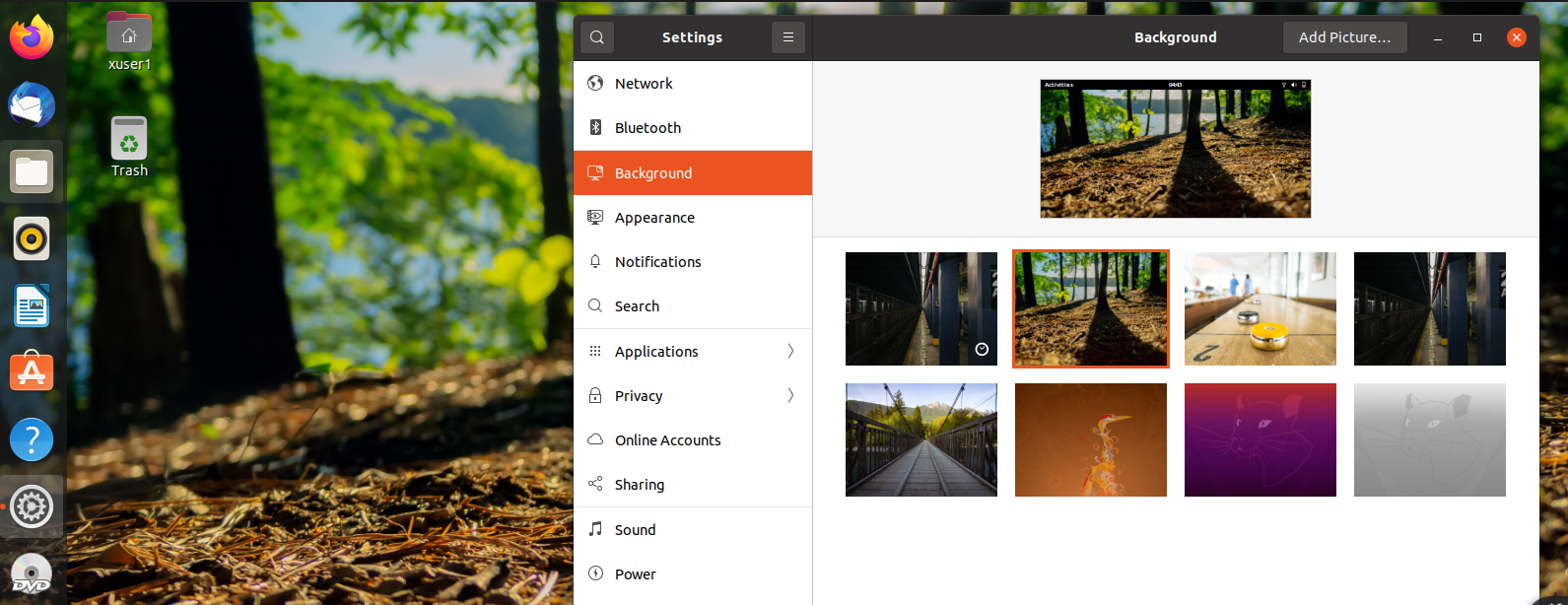
修改 myusers 组群属性，将 xuser1 和 helen 用户加入 myusers 组群：



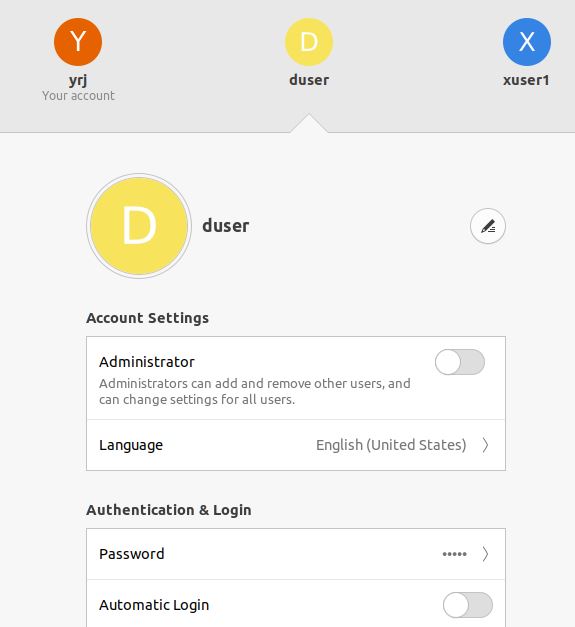
删除 temp 组群：



在用户管理界面中更改桌面backgroud

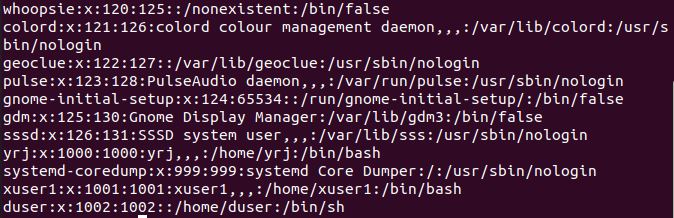


新建一名为 duser 的用户

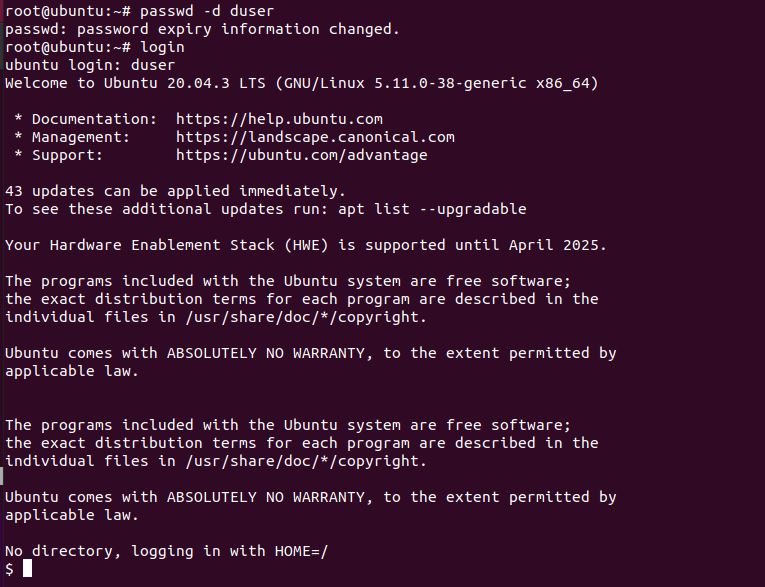


查看 duser 用户的相关信息

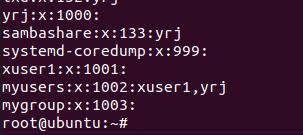




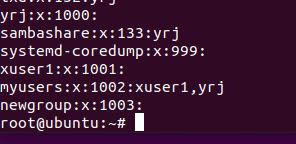
duser无密登录



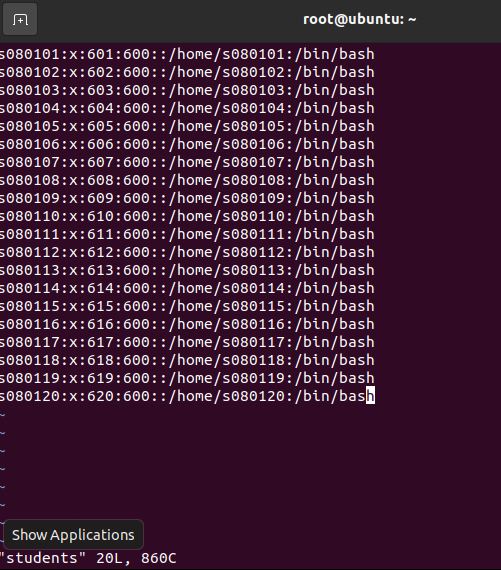
新建组群 mygroup，查看其相关文件信息



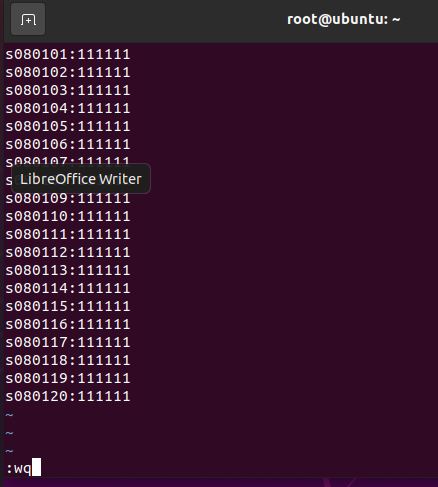
查看更名为newgroup后的相关文件信息



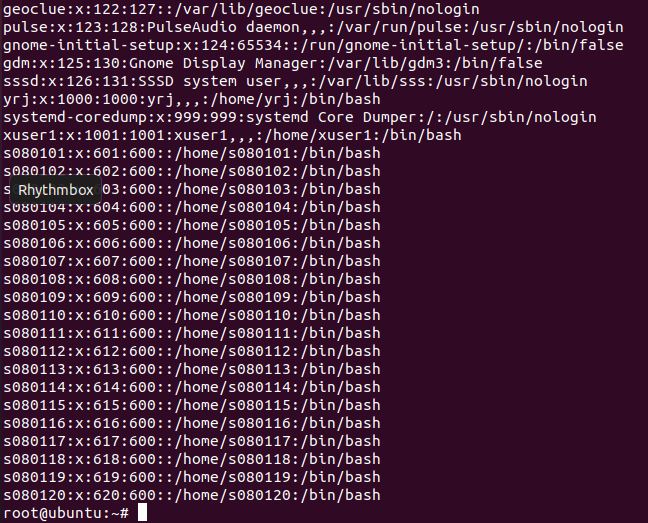
students文件信息



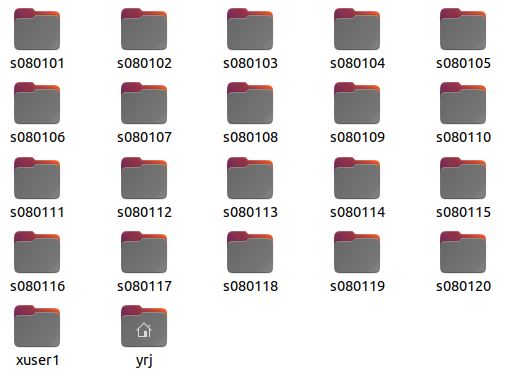
stu-passwd文件信息



批量新建多个用户账号



查看批量建立文件后的home文件夹



**体会：**

通过本次实验，我理解了/etc/passwd 和/etc/group 文件的含义、掌握桌面环境下管理用户与组群的方法、掌握利用 Shell 命令管理用户与组群的方法并掌握批量新建用户帐号的步骤和方法。同时在使用vi命令时熟悉了文本编辑器的使用。

**附录：（源程序）**

本实验无源程序。

实验题目：SHELL编程

姓名：杨瑞基 学号：19121963 实验日期：2021/11/4

实验环境：Ubantu操作系统

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

实验步骤

**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。

**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. #没有参数

或 $sh prog1.

在提示符后键入命令行：

$./prog1.h Theodore #引用$1 参数的效果

**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。

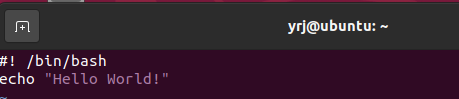
**9.** 用vi 编写《实验指导》“第四部分 Shell 程序设计”中的例4、例5、例6、例7，掌握控 制语句的用法，观察运行结果。

**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。

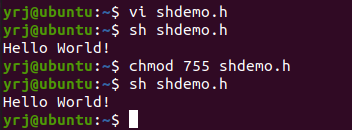
**11.** 编程，在屏幕上显示用户主目录名（HOME）、命令搜索路径（PATH），并显示由位 置参数指定的文件的类型和操作权限。

实验结果：

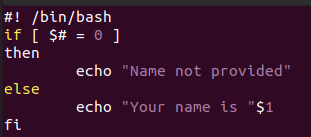
**1.** 用vi文本编辑器创建并编写文件shdemo.h, 并将文 件以文本格式保存在相应的目录中。其内容如下：



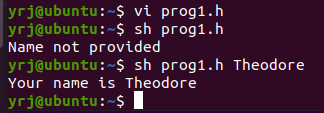
**2.** 用 chmod 将文件的权限设置为可执行模式后的执行结果



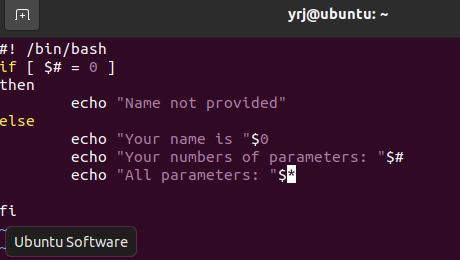
**3.** 用vi编写如下程序以练习内部变量和位置参数的用法：



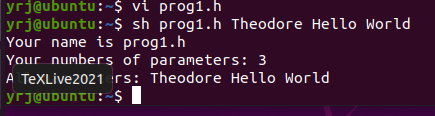
上述程序显示结果如下：



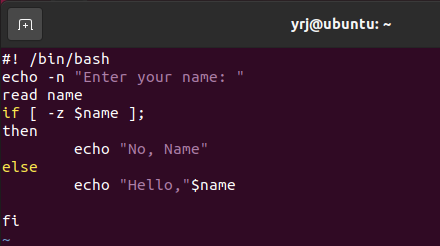
**4.** 修改程序prog1.h，显示参数个数、程序名字，并逐个显示参数，其代码如下：

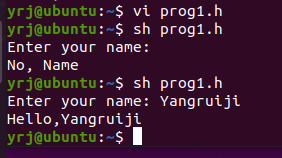


上述程序显示结果如下：

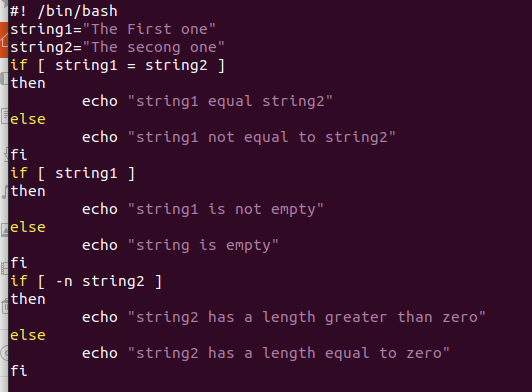


**5.** 修改程序prog1.h，用read命令接受键盘输入。若没有输入显示第一种 提示，否则第二种提示。其代码如下所示：

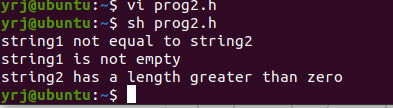


上述程序的运行结果：  


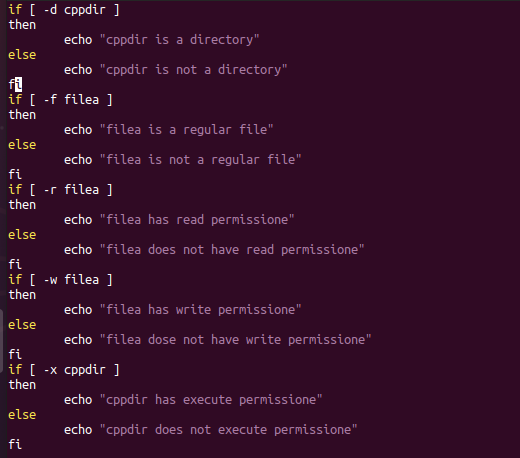
**6.** 例程2的代码：



例程2的运行结果：



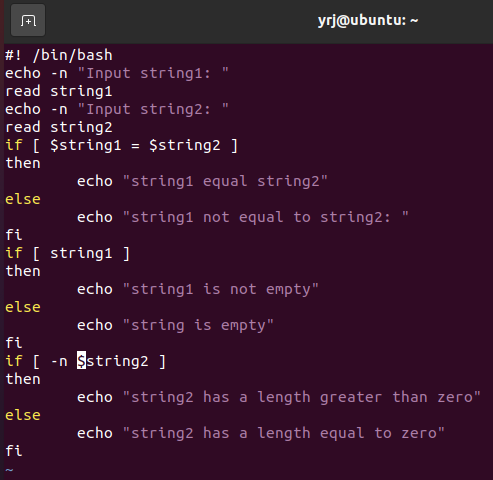
例程3的代码：



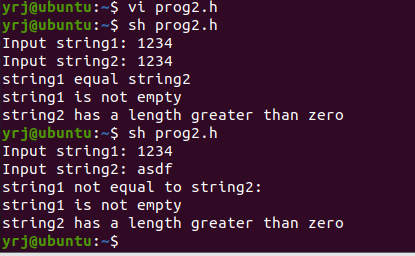
例程3的运行结果：



**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，代码如下。

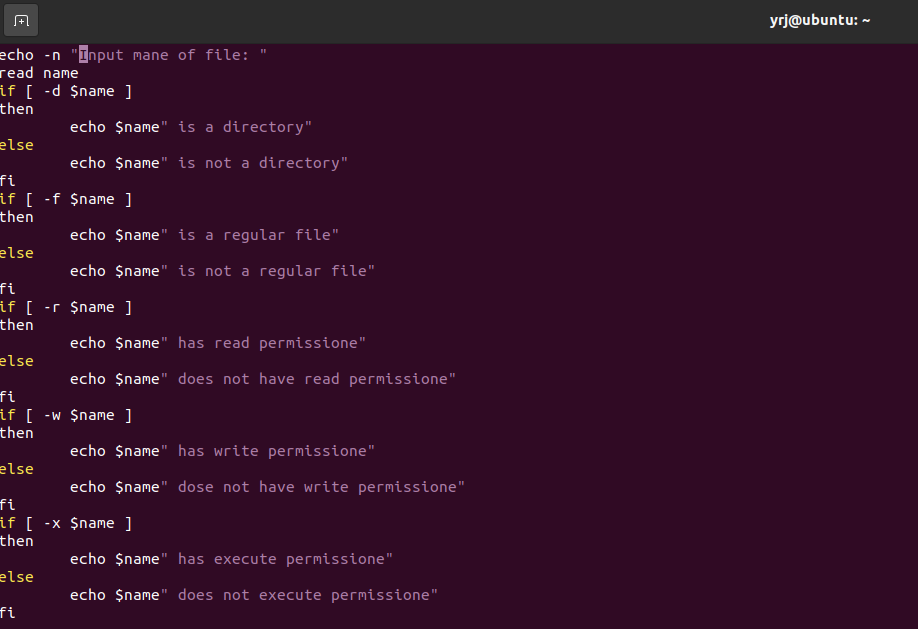


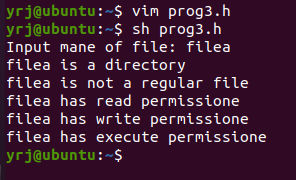
上述程序运行结果如下：



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。其代码如下：

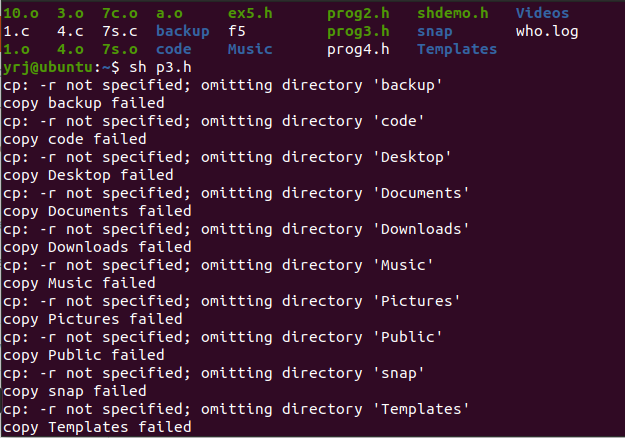
上述程序运行结果如下：



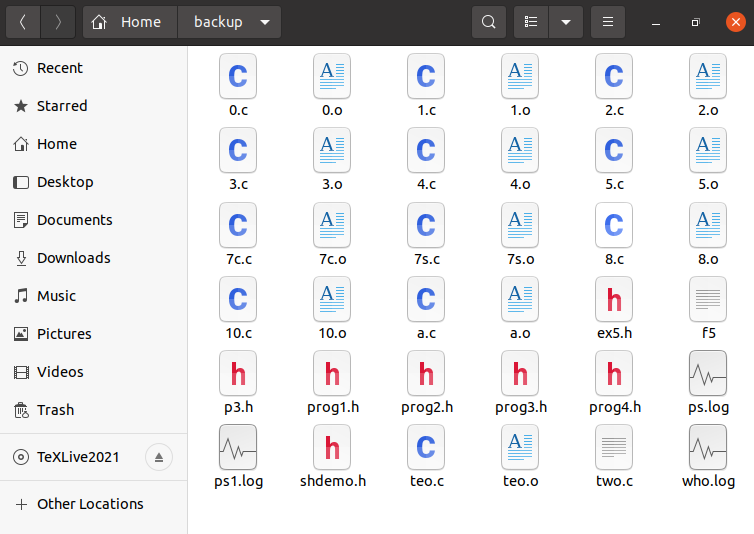


例4代码及运行结果如下：

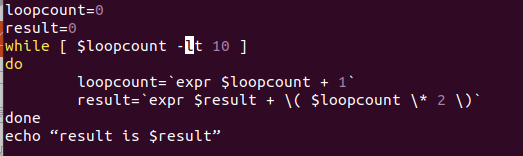




成功将文件复制进backup目录中

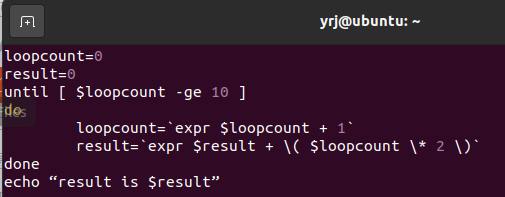


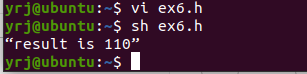
例5代码及运行结果如下：



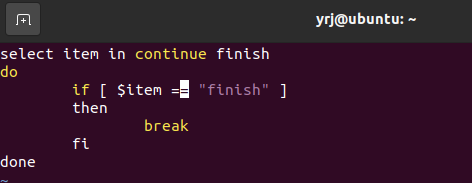


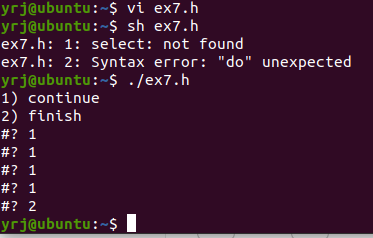
例6代码及运行结果如下：



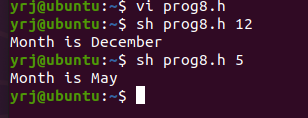


例7代码及运行结果如下：

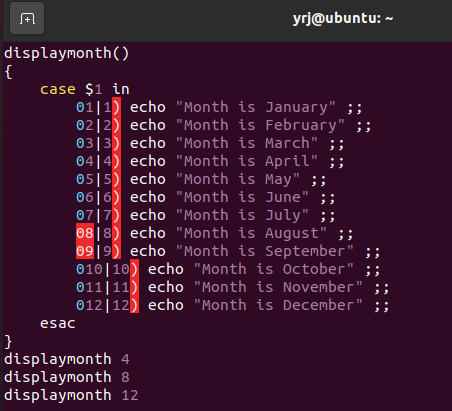


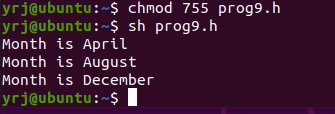


例8代码及运行结果如下：

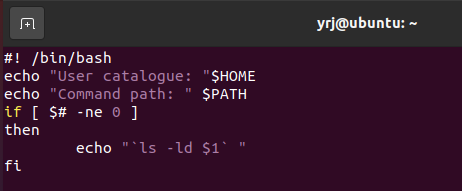


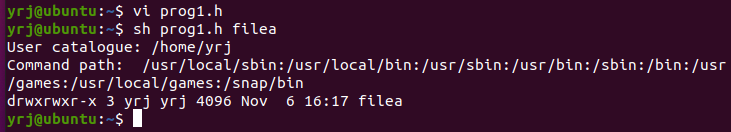
例9代码及运行结果如下：





**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





思考题：

1. 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：插入方式用于编辑文本内容时，转义命令方式用于移动、删除、替换等特殊功能时末行命令方式用于用户对文件进行全局操作时，主要是全局查找、替换、文件读、写等。

通过输入i, I, a, A, o, O的命令来进入插入模式，用<ESC>命令来退出输入模式。

可通过键入命令 :w或 :w newfile以及 : w! newfilename的方式将文件存盘。

思考题：到此为止你对Shell 有所认识了吧？怎么样？自己再编两个程序：

* 1. 做个批处理程序，体会一下批处理概念。
  2. 做个菜单，显示系统环境参数。将此程序设置为人人可用。

|  |
| --- |
| #1. 批处理程序      #2. 做个菜单，显示系统环境参数 |

讨论题：

1. Linux的Shell有什么特点？

答：

（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？

答：Shell程序就是一个包含若干行Shell或Linux命令的文件，像编写高级语言的程序一样，编写一个Shell程序需要一个文本编辑器，如vi和vim等。在文本编辑环境下，依据Shell的语法规则，输入一些Shell/Linux命令行，形成一个完整的程序文件。用chmod将文件的权限设置为可执行模式，如若文件名为shedemo.h，则命令如下：$chmod 755 shdemo.h（文件组可读、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 或 $sh shdemo.h 或$./shdemo.h

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

我们在Shell环境(提示符为$)下输入启动vi命令，进入编辑器时，就是处于该模式下。在该模式下，用户可以输入各种合法的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，所输入的命令并不在屏幕上显示出来。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当作文件保存起来，并将其显示在屏幕上。在文本输入过程中，若想回到命令模式下，按Esc键即可。

3、末行模式

vi有一个专门的“转义”命令，可访问很多面向行的Ex命令。在命令模式下，用户按“：”键即可进入末行模式，此时vi会在显示窗口的最后一行显示一个“：”作为末行模式的提示符，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。末行命令执行完后，vi自动回到命令模式。

4. 编写一个具有以下功能的Shell程序。

(1) 把当前目录下的文件目录信息输出到文件 filedir.txt 中；

(2) 在当前目录下建立一个子目录，目录名为 testdir2 ；

(3) 把当前目录下的所有扩展名为 c 的文件以原文件名复制到子目录testdir2中；

(4) 把子目录中的所有文件的存取权限改为不可读。（提示：用 for 循环控制语句实现，循环的控制列表用 ’ls’ 产生。）

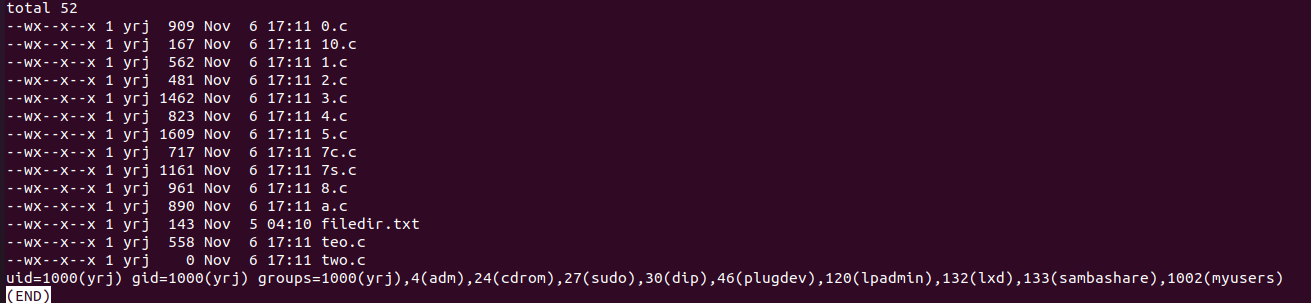
(5) 在把子目录 testdir2 中所有文件的目录信息追加到文件 filedir.txt 中；

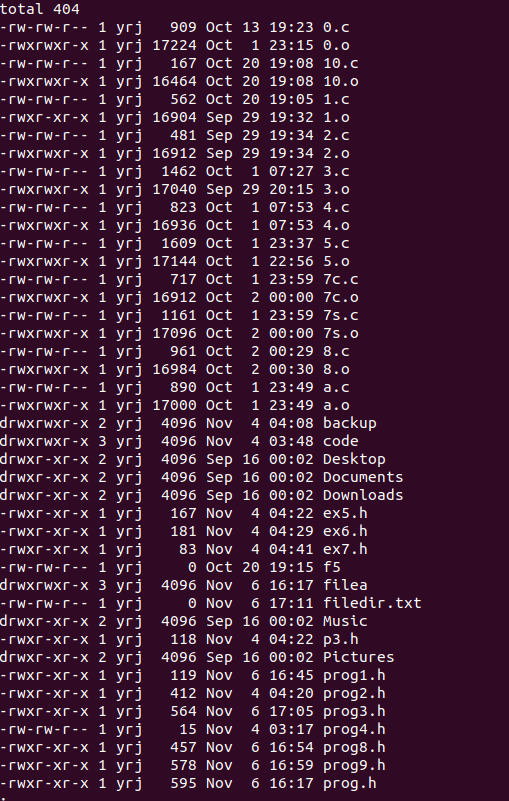
(6) 把你的用户信息追加到文件 filedir.txt 中；

(7) 分屏显示文件 filedir.txt



执行结果如下：





|  |
| --- |
| #! /bin/bash  ls -o>filedir.txt  mkdir testdir5  chmod 755 testdir5  cp $( find \*.c) testdir5  cd testdir5  for filename in `ls`  do  chmod 311 $filename  done  cd  cd /home/yrj  ls -o testdir5>>filedir.txt  id>>filedir.txt  less filedir.txt |

**体会：**

通过此次实验，我学习使用了vim编写Shell程序、将程序文件设置为可执行文件（用chmod命令）并在命令行方式中运行Shell程序。过程中也遇到了许多语句上的小错误，但是通过寻找资料和不断试错，最终也都得到了结果。这次的实验让我对于使用vi编程的理解变得更加深刻，也更加熟悉如何使用linux的文本编辑器，同时也了解了变量、参数、操作符和重复语句等的概念和操作。

附录（程序）

|  |
| --- |
| #修改程序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。  #! /bin/bash  echo -n "Enter your name: "  read name  if [ -z $name ];  then  echo "No, Name"  else  echo "Hello,"$name  fi |

|  |
| --- |
| #例程2  #! /bin/bash  string1="The First one"  string2="The secong one"  if [ string1 = string2 ]  then  echo "string1 equal string2"  else  echo "string1 not equal to string2"  fi  if [ string1 ]  then  echo "string1 is not empty"  else  echo "string is empty"  fi  if [ -n string2 ]  then  echo "string2 has a length greater than zero"  else  echo "string2 has a length equal to zero"  fi |
| #例程3  if [ -d cppdir ]  then  echo "cppdir is a directory"  else  echo "cppdir is not a directory"  fi  if [ -f filea ]  then  echo "filea is a regular file"  else  echo "filea is not a regular file"  fi  if [ -r filea ]  then  echo "filea has read permissione"  else  echo "filea does not have read permissione"  fi  if [ -w filea ]  then  echo "filea has write permissione"  else  echo "filea dose not have write permissione"  fi  if [ -x cppdir ]  then  echo "cppdir has execute permissione"  else  echo "cppdir does not execute permissione"  fi |
| #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  #! /bin/bash  echo -n "Input string1: "  read string1  echo -n "Input string2: "  read string2  if [ $string1 = $string2 ]  then  echo "string1 equal string2"  else  echo "string1 not equal to string2: "  fi  if [ string1 ]  then  echo "string1 is not empty"  else  echo "string is empty"  fi  if [ -n $string2 ]  then  echo "string2 has a length greater than zero"  else  echo "string2 has a length equal to zero"  fi |
| #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。  cho -n "Input mane of file: "  read name  if [ -d $name ]  then  echo $name" is a directory"  else  echo $name" is not a directory"  fi  if [ -f $name ]  then  echo $name" is a regular file"  else  echo $name" is not a regular file"  fi  if [ -r $name ]  then  echo $name" has read permissione"  else  echo $name" does not have read permissione"  fi  if [ -w $name ]  then  echo $name" has write permissione"  fi  if [ -x $name ]  then  echo $name" has execute permissione"  else  echo $name" does not execute permissione"  fi |
| #例程4  for filename in `ls`  do  cp $filename backup/$filename  if [ $? -ne 0 ]  then  echo "copy $filename failed"  fi  done |
| #例程5  loopcount=0  result=0  while [ $loopcount -lt 10 ]  do  loopcount=`expr $loopcount + 1`  result=`expr $result + \( $loopcount \\* 2 \)`  done  echo “result is $result” |
| #例程 6  loopcount=0  result=0  until [ $loopcount -ge 10 ]  do  loopcount=`expr $loopcount + 1`  result=`expr $result + \( $loopcount \\* 2 \)`  done  echo “result is $result” |
| #例程 7  select item in continue finish  do  if [ $item = "finish" ]  then  break  fi  done |
| #例程 8  case $1 in  01|1) echo "Month is January" ;;  02|2) echo "Month is February" ;;  03|3) echo "Month is March" ;;  04|4) echo "Month is April" ;;  05|5) echo "Month is May" ;;  06|6) echo "Month is June" ;;  07|7) echo "Month is July" ;;  08|8) echo "Month is August" ;;  09|9) echo "Month is September" ;;  010|10) echo "Month is October" ;;  011|11) echo "Month is November" ;;  012|12) echo "Month is December" ;;  esac |
| #例程 9  displaymonth()  {  case $1 in  01|1) echo "Month is January" ;;  02|2) echo "Month is February" ;;  03|3) echo "Month is March" ;;  04|4) echo "Month is April" ;;  05|5) echo "Month is May" ;;  06|6) echo "Month is June" ;;  07|7) echo "Month is July" ;;  08|8) echo "Month is August" ;;  09|9) echo "Month is September" ;;  010|10) echo "Month is October" ;;  011|11) echo "Month is November" ;;  012|12) echo "Month is December" ;;  esac  }  displaymonth 4  displaymonth 8  displaymonth 12 |
| #**11.** 编程，在屏幕上显示用户主目录名（HOME）、命令搜索路径（PATH），并显示由位 置参数指定的文件的类型和操作权限  #! /bin/bash  echo "User catalogue: "$HOME  echo "Command path: " $PATH  if [ $# -ne 0 ]  then  echo "`ls -ld $1` "  fi |
| #做个菜单，显示系统环境参数。  select cho in env finish  do  if [ $cho = "finish" ]  then  echo "It is finished"  break  else  $cho  fi  done |