**实验一 Linux操作系统基本命令**

19121525 蒋俊杰 2021年9月9日

**实验环境：**

Vmware worksatation下的Linux虚拟机-abuntu-20.04.3。

**实验目的：**

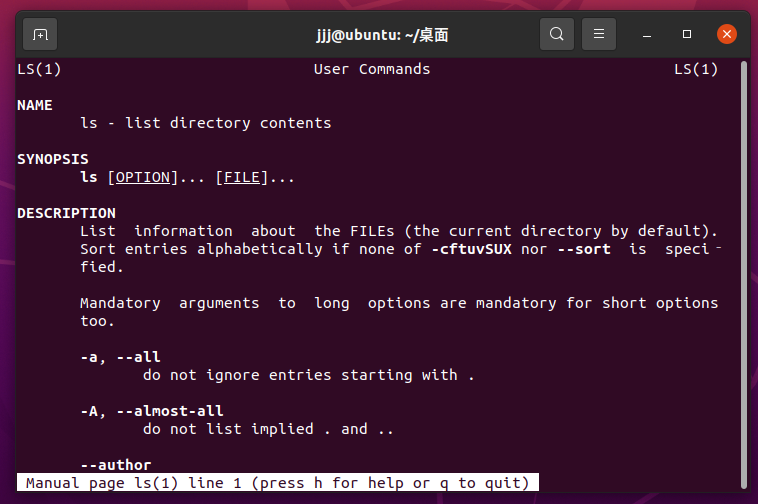
1. 了解Linux运行环境，熟悉交互式分时系统、多用户环境的运行机制。
2. 练习Linux系统命令接口的使用，学会Linux的基本命令。

**实验内容：**

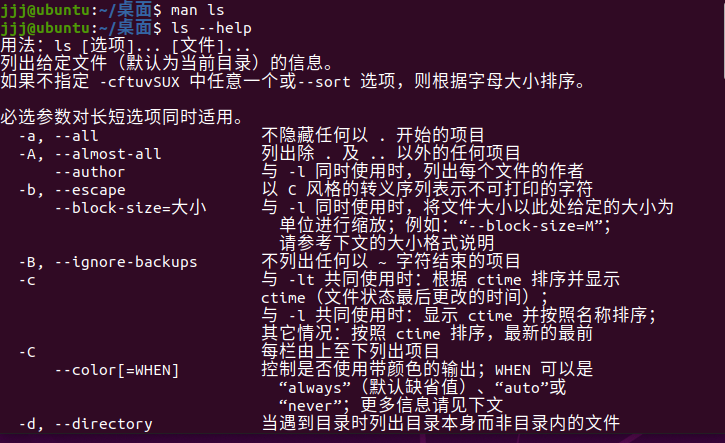
通过虚拟终端，练习Linux中的交互界面里练习shell的基本命令操作。

**操作过程：**

1.man命令：如man -ls 可以查询命令手册：



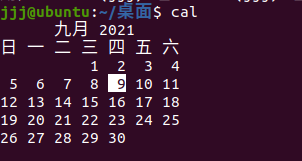
2.—help命令：显示联机状态下的命令帮助手册



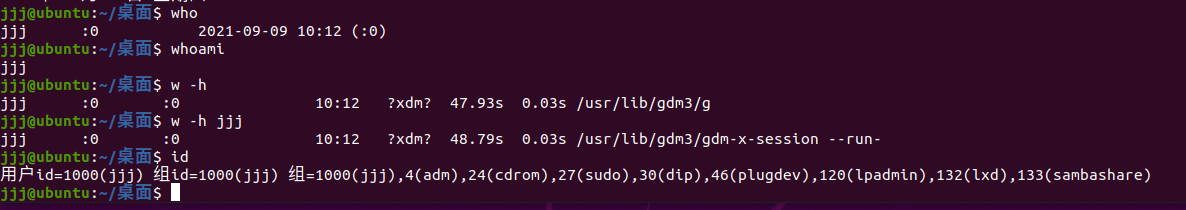
3.pwd：



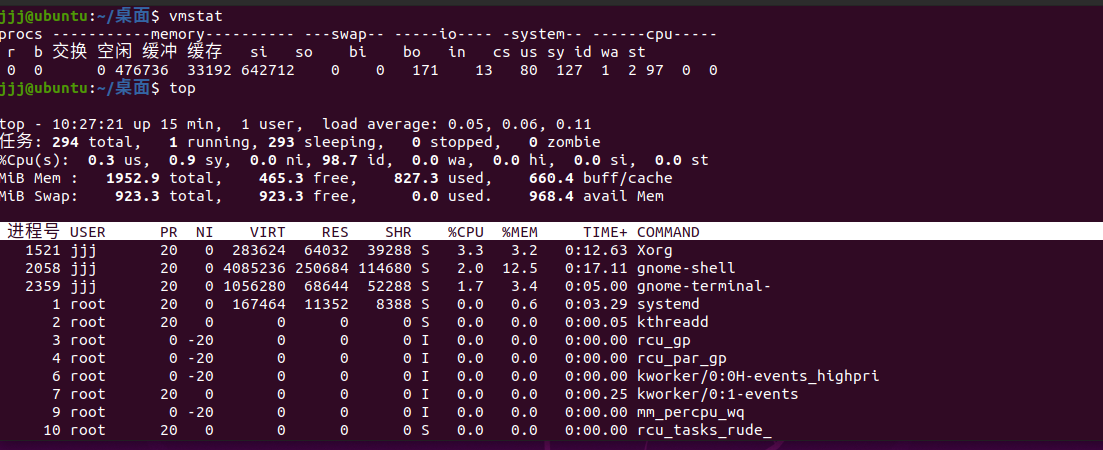
4.date、cal命令：显示系统时间和日历：

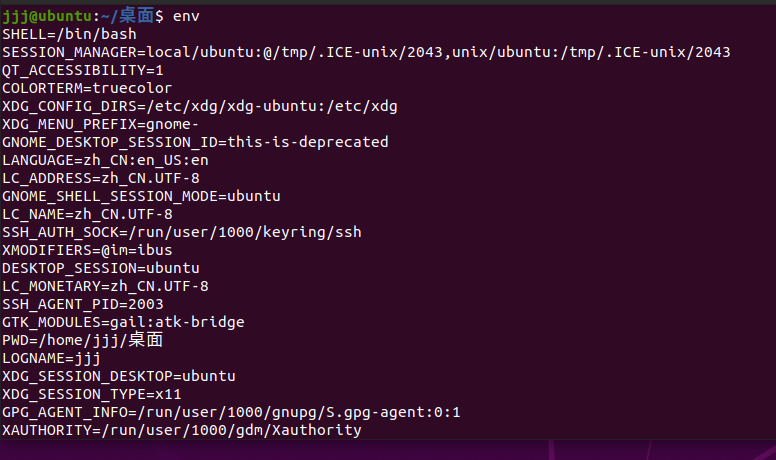


5.who、whoami、id命令：查看系统的用户信息：



6.env、vmstat命令：显示环境变量与查看系统状态：





7.cat> [文件名]:创建一个文件（可以添加路径来指定创建文件的位置）

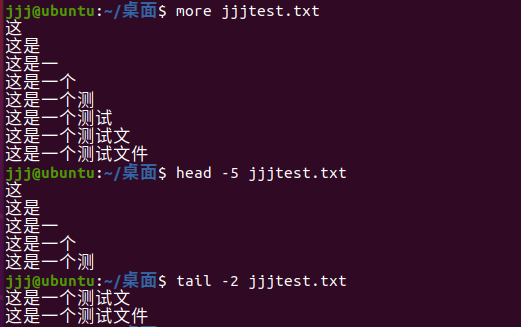


8.more 文件名、head 文件名、tail 文件名：浏览文件内容：

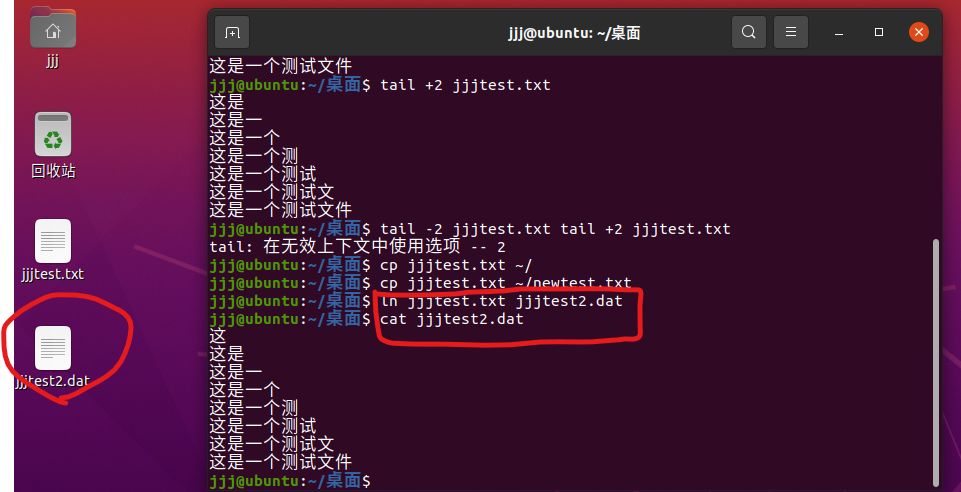




因为head与tail命令类似于文件内容的切片浏览，所以我后来增加了一开始所创建的txt文件中的内容，让显示的结果更加直观：

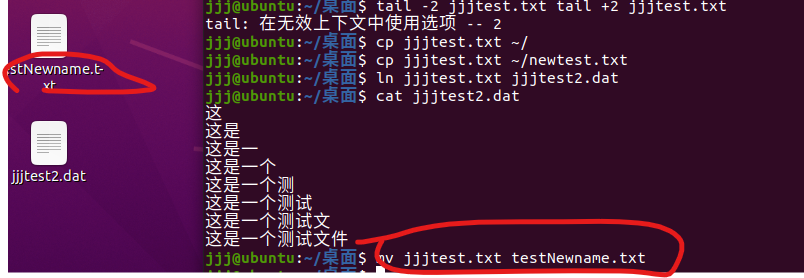


9.cp、ln命令：复制文件与创建文件链接：

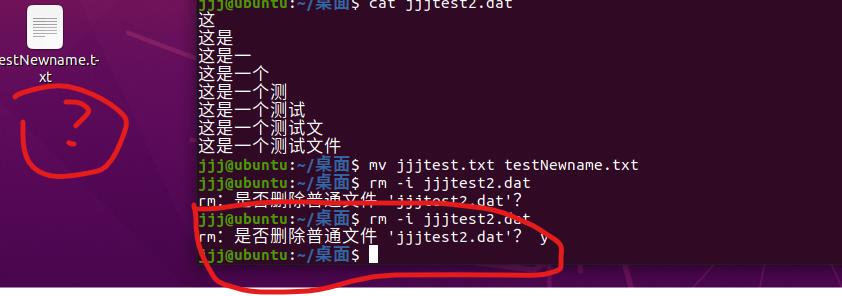




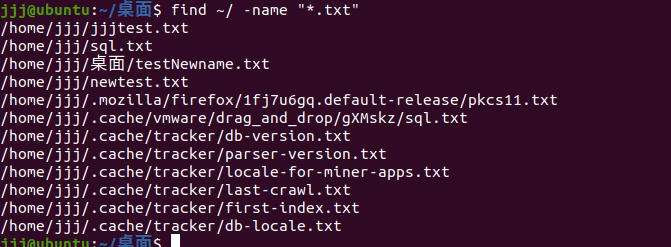
10.mv命令：移动或者重命名文件：



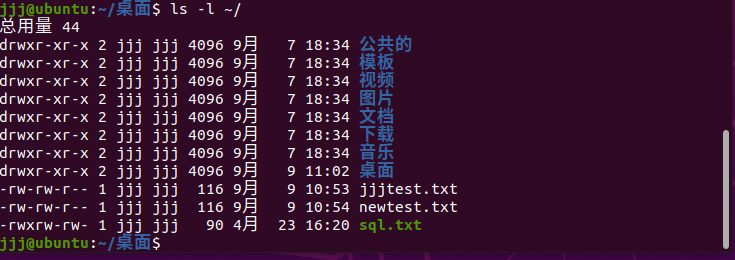
11.rm命令：删除文件：

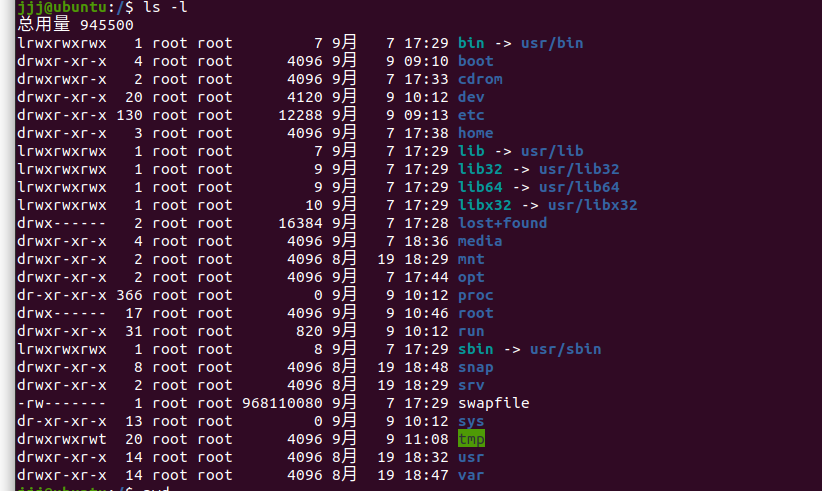


12.find命令：查找文件：

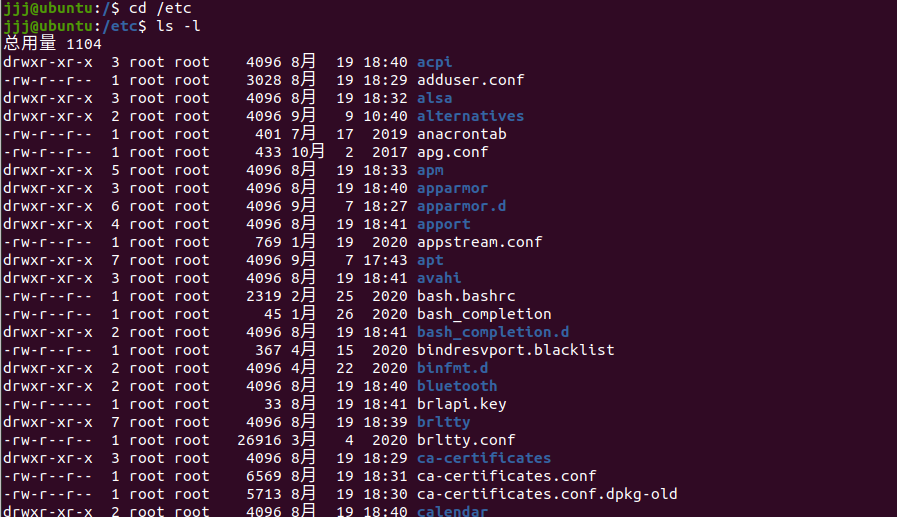


13.ls命令：列出目录

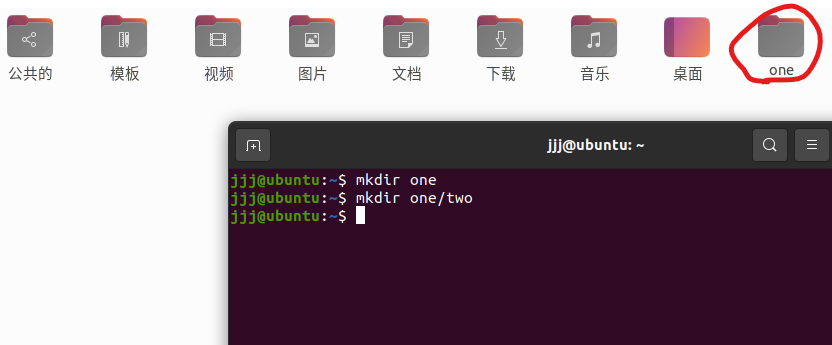


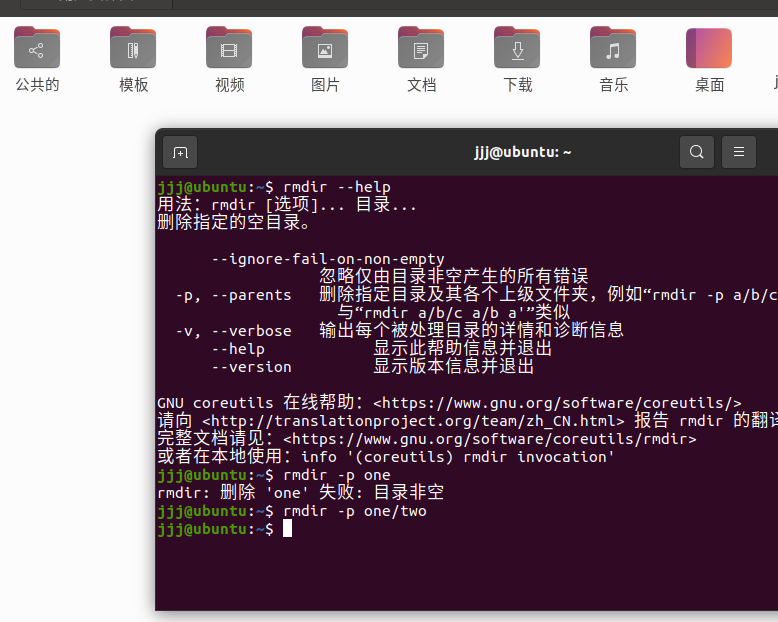


14.cd命令：改变当前目录：



15.mkdir、rmdir命令：创建与删除命令。





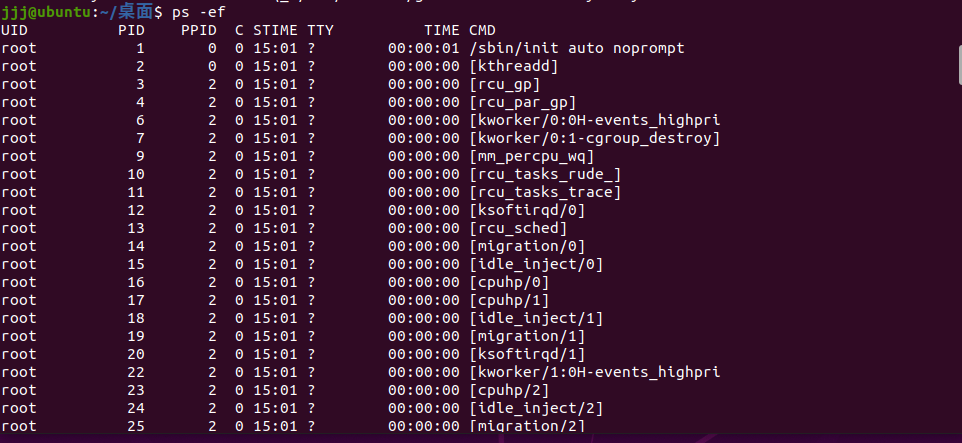
16.修改文件属性的命令：chown、chgrp、chmod:







17.ps命令：查看进程、wait命令和sleep命令：等待进程完成与挂起一段时间

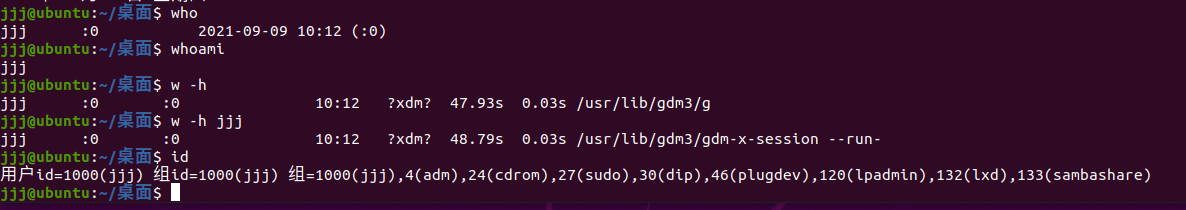


我编写了一个shell来实现对wait命令的应用，在运行这个脚本的时候可以明显看到因为使用了sleep和wait命令所以回显的1,3,4和5不是同时显示在屏幕上的。



**思考题：**

1.我的用户名为：jjj，用户id为1000，组名为jjj，组标识为1000；当前处于系统的桌面目录下，只有我自己一个用户使用系统：





2.文件链接是将多个新的文件名与一个文件实体建立连接但这不是一个独立的文件（即所谓的硬连接，硬链接只是分区表中的一个条目，它指向磁盘上的某个区域，表示该区域已经被分配给文件。），作用是可以让你同时在多个位置显示一个文件或者目录。

3.Linux的文件分为七种类型：

普通文件（-）：我们用 ls -lh 来查看某个文件的属性，可以看到有类似 -rw-r--r-- ，值得注意的是第一个符号是 - ，这类文件的删除方式是用rm 命令。

目录（d）：当我们在某个目录下执行，看到有类似 drwxr-xr-x ，这样的文件就是目录，目录在Linux是一个比较特殊的文件。注意它的第一个字符是d。创建目录的命令可以用 mkdir 命令，或cp命令，cp可以把一个目录复制为另一个目录。删除用rm 或rmdir命令。

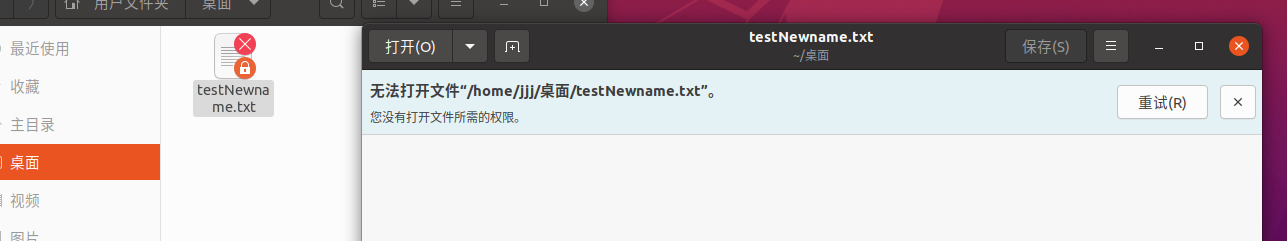
字符设备(c)和块设备（b）：这个种类的文件，是用mknode来创建，用rm来删除。这些文件是和内核相关联的。

套接口文件（s）：这个文件的属性的第一个字符是 s。

符号链接（l）：当我们查看文件属性时，会看到有类似 lrwxrwxrwx,注意第一个字符是l，这类文件是链接文件。是通过ln -s 源文件名 新文件名 。

管道（p）：管道文件只能以读写方式打开。

4.在使用了chown、chgrp、chmod一系列命令改变了文件的权限，所有者和组标识后就不能再修改文件了，因为当前的用户已经没有权限了：



5.系统使用ps命令来管理进程，可以进行通过-ef等参数来进行显示，kill命令来传送命令或者结束进程、wait命令等待进程完成等待。子进程是依赖于父进程的，父进程复制自己的地址空间(fork)创建一个新的(子)进程结构。每个新进程分配一个唯一的进程 ID (PID)，满足跟踪安全性之需。PID 和 父进程 ID (PPID)是子进程环境的元素，任何进程都可以创建子进程，所有进程都是第一个系统进程（ID=1）的后代。还可以通过pstree命令来查看进程树。这样的家族关系有利于用户对进程进行管理。



**讨论题**：

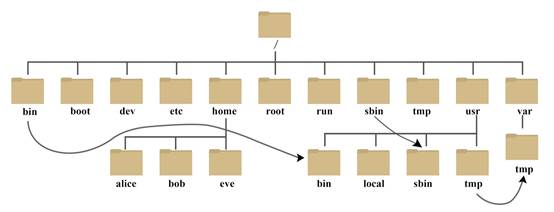
1. 使用命令：[需要查找的命令名] –help或者man来查看相关资料。

2. Useradd chmod,chgrp等。当使用ls -l命令时可以看到文件权限的分配，可以在文件创建者的身份或者root情况下通过chmod修改文件的权限,chgrp可以修改文件的所属组，chown修改文件的所有者和所属组。

3. Ps查看进程管理参数，pstree以树的形式查看；当父进程先结束，内核会检查哪些子进程的父进程为该进程，将其父进程修改为init进程，并由init进程对他们完成状态搜集过程。4. Linux系统中“一切都是文件”，即可以通过文本查看各项信息。Linux文件类型有：普通文件[-]，目录文件[d]，链接文件[l]，设备文件[b]，管道文件[p]，套接字文件[s]。Linux的文件标识符，相当于一个指针。

5. Bin /usr/bin

6. 在Linux中以设备文件来进行管理，在media目录中。

7.  树状目录结构：  
·        /bin：  
bin是Binaries (二进制文件)的缩写,这个目录存放着最经常使用的命令.

·        /boot：  
这里存放的是启动Linux时使用的一些核心文件，包括一些连接文件以及镜像文件。

·        /dev：  
dev是Device(设备)的缩写,该目录下存放的是Linux的外部设备，在Linux中访问设备的方式和访问文件的方式是相同的。

·        /etc：  
etc是Etcetera(等等)的缩写,这个目录用来存放所有的系统管理所需要的配置文件和子目录。

·        /home：  
用户的主目录，在Linux中，每个用户都有一个自己的目录，一般该目录名是以用户的账号命名的，如上图中的alice、bob和eve。

·        lib：  
lib是Library(库)的缩写这个目录里存放着系统最基本的动态连接共享库，其作用类似于Windows里的DLL文件。几乎所有的应用程序都需要用到这些共享库。

·        /lost+found：  
这个目录一般情况下是空的，当系统非法关机后，这里就存放了一些文件。

·        /media：  
linux系统会自动识别一些设备，例如U盘、光驱等等，当识别后，Linux会把识别的设备挂载到这个目录下。

·        /mnt：  
系统提供该目录是为了让用户临时挂载别的文件系统的，我们可以将光驱挂载在/mnt/上，然后进入该目录就可以查看光驱里的内容了。

·        /opt：  
opt是optional(可选)的缩写，这是给主机额外安装软件所摆放的目录。比如你安装一个ORACLE数据库则就可以放到这个目录下。默认是空的。

·        /proc：  
proc是Processes(进程)的缩写，/proc是一种伪文件系统（也即虚拟文件系统），存储的是当前内核运行状态的一系列特殊文件，这个目录是一个虚拟的目录，它是系统内存的映射，我们可以通过直接访问这个目录来获取系统信息。这个目录的内容不在硬盘上而是在内存里，我们也可以直接修改里面的某些文件，比如可以通过下面的命令来屏蔽主机的ping命令，使别人无法ping你的机器：

8. 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”。

9. （1）普通文件（2）目录文件（3）普通文件（4）接口设备（5）普通文件（6）串行端口设备（7）目录文件（8）普通文件

**实验二 用户界面与shell命令**

19121525 蒋俊杰 2021年9月16日

**实验环境：**

Vmware worksatation下的Linux虚拟机-abuntu-20.04.3。

**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。

**实验内容：**

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。

Shell 命令是 Linux 操作系统的灵魂，灵活运用 Shell 命令可完成操作系统所有的工作。

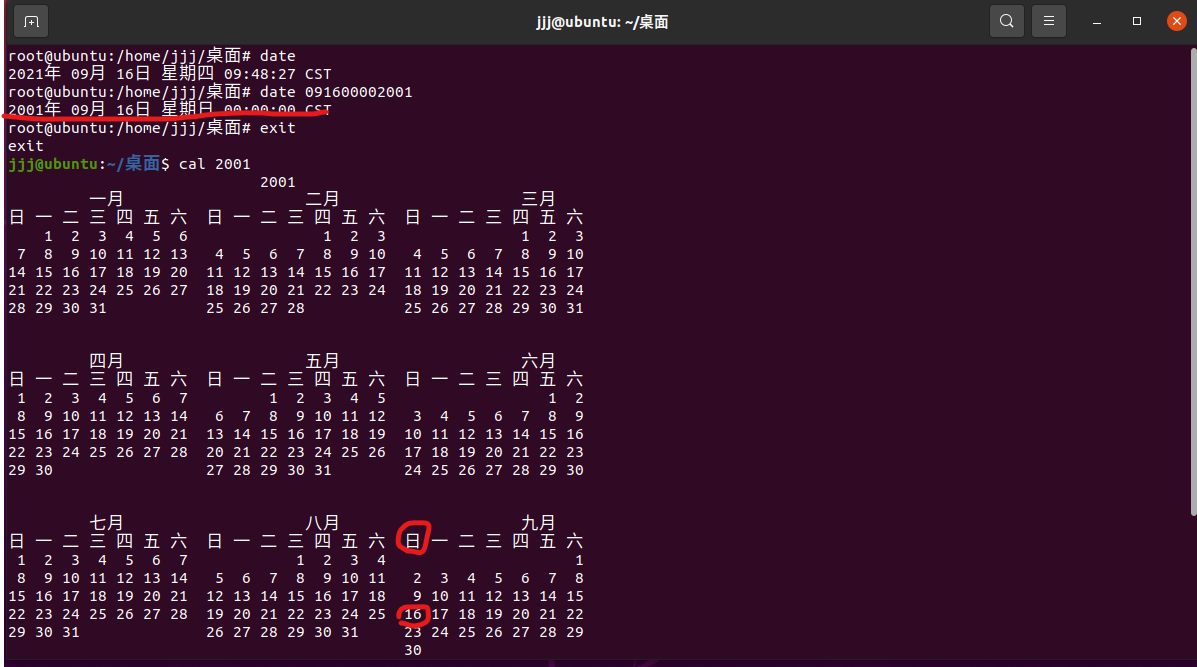
并且类 UNIX 的操作系统在 Shell 命令方面具有高度相似性。熟悉掌握 Shell 命令，不仅有

助于掌握 RHEL Server 5，而且几乎有助于掌握各发行版本的 Linux，甚至 UNIX。

RHEL Server 5 中不仅可在字符界面下使用 Shell 命令，还可以借助于桌面环境下的终端工具使用 Shell 命令。桌面的终端工具中使用 Shell 命令时可显示中文，而字符界面下显示英文。**操作过程：**

1. 在图形化用户界面下的shell命令操作：

1.使用超级用户修改系统时间并查看日历：

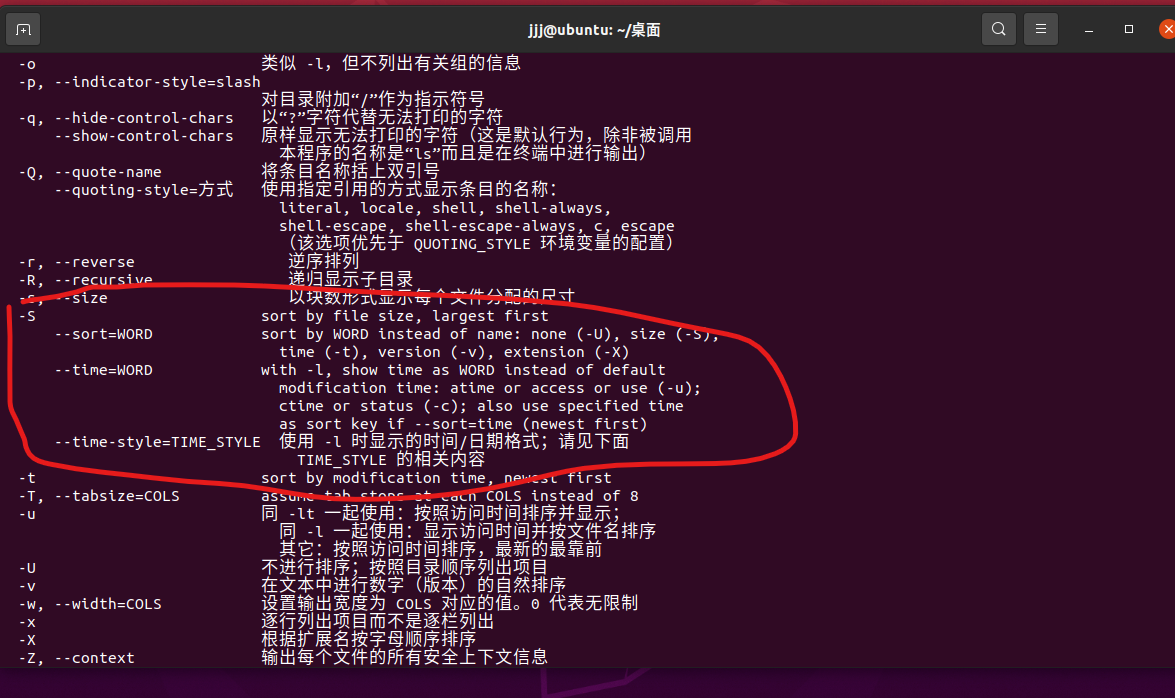


2.查看ls 命令的 –s选项的帮助信息：

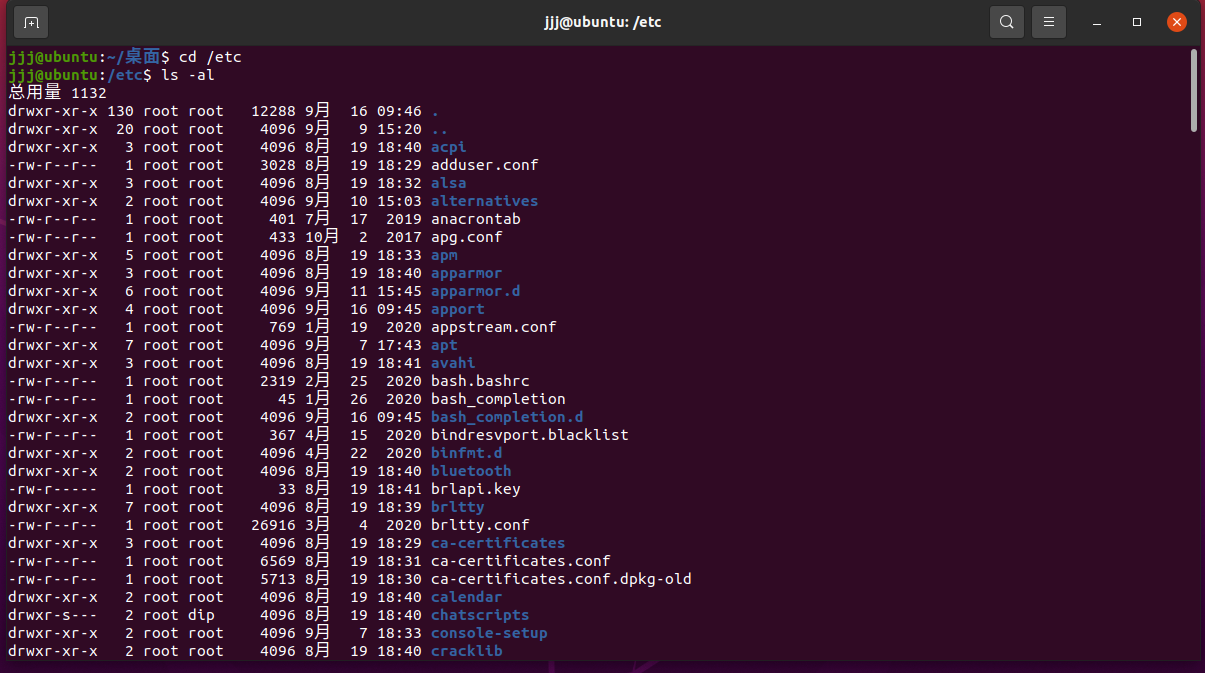
（1）使用man ls命令查看：



（2）使用ls --help查看帮助



3. 查看/etc 目录下所有文件和子目录的详细信息：需要使用cd 命令进入指定目录后使用ls查看文件和子目录的详细信息。

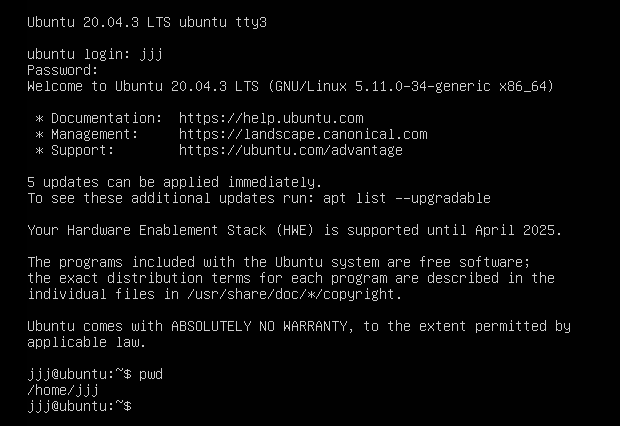


二．字符界面下的shell命令操作：

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

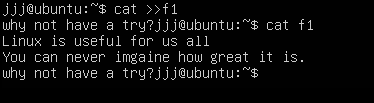
1.从图形化界面登录到字符界面并查看当前目录：



2.用 cat 命令在用户主目录下创建一名为 f1 的文本文件，并且输入相关内容：



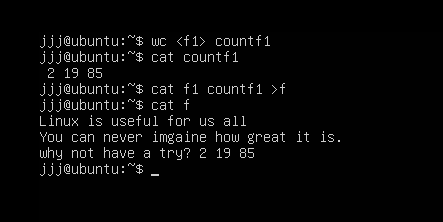
3.向 f1 文件增加以下内容：Why not have a try（涉及到shell命令中关于重定向符号的使用）



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。

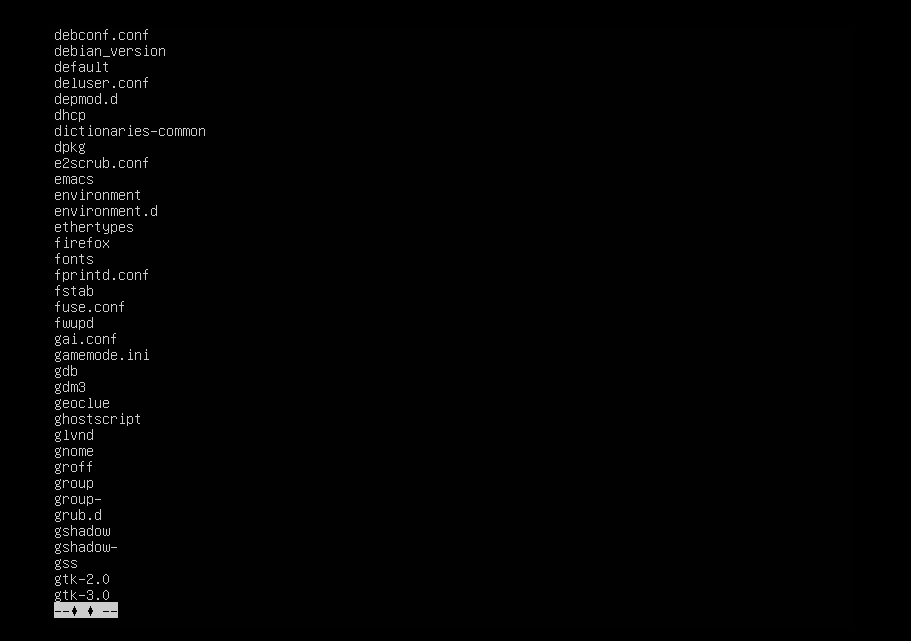


5. 将 f1 和 countf1 文件的合并为 f 文件：

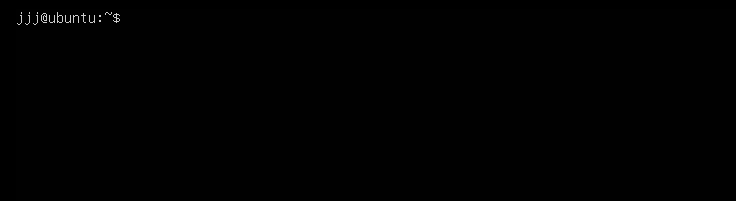


6. 分页显示/etc 目录中所有文件和子目录的信息：输入命令:ls /etc|more

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。



7.使用clear清楚屏幕内容，命令提示符在屏幕左上角：



三．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。

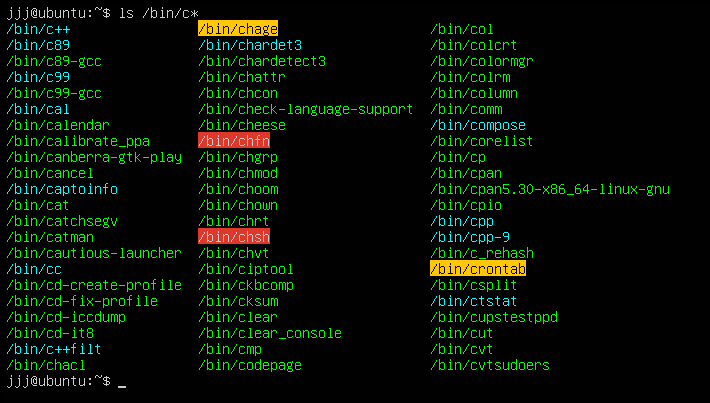
[ ] ：表示范围。

- ：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。

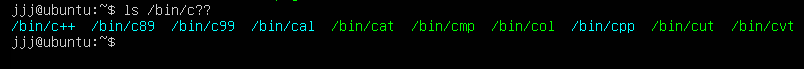
1. 显示/bin/目录中所有以 c 为首字母的文件和目录：

\* ：匹配任意长度的任何字符。



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。

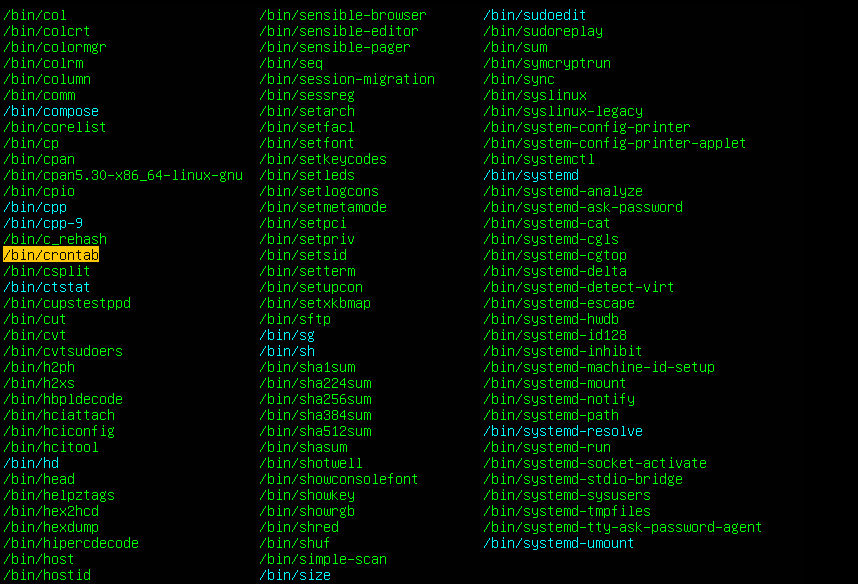
? ：匹配一个字符。因此只需要在操作1的基础上输入ls /bin/c??即可。



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。

[ ] ：表示范围。

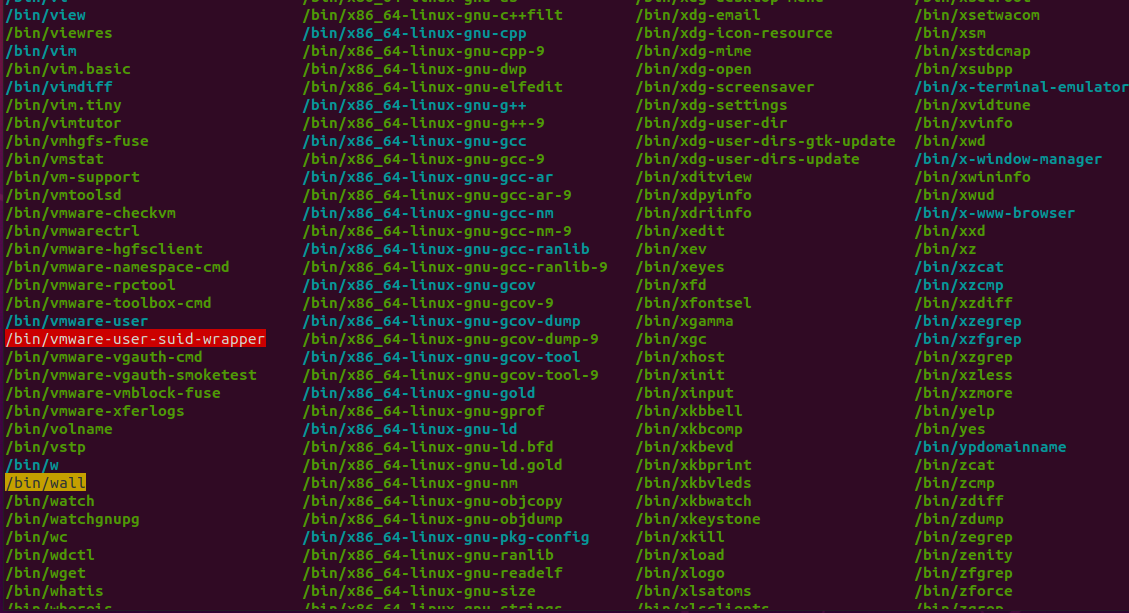
输入：ls /bin/[csh]\*



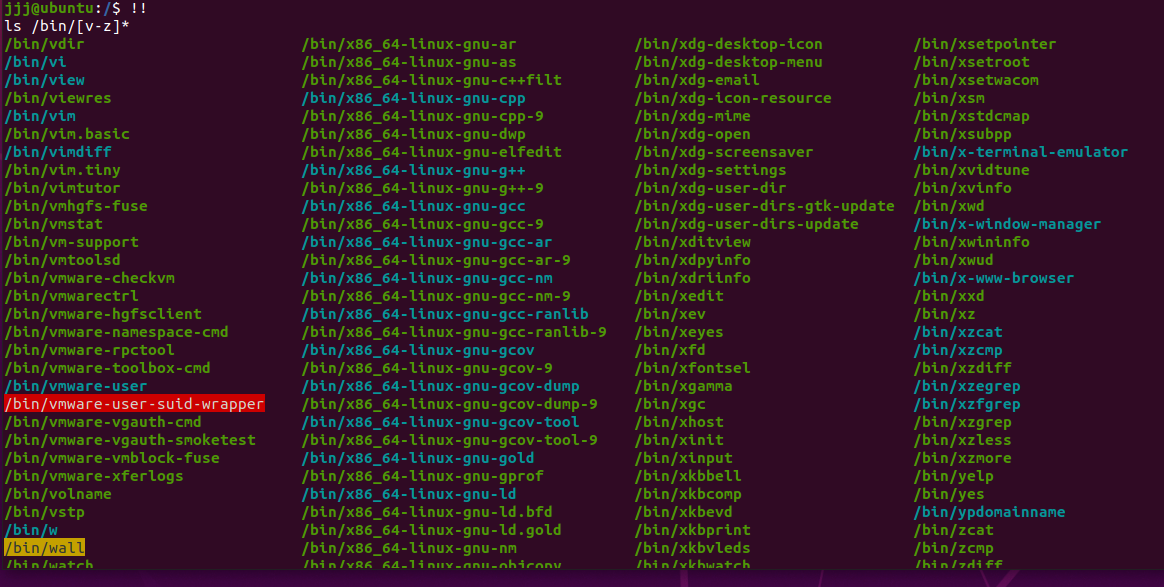
4. 显示/bin/目录中所有的首字母是 v、w、x、y、z 的文件和目录。

！ ：表示不在范围，通常也与[ ]配合使用。

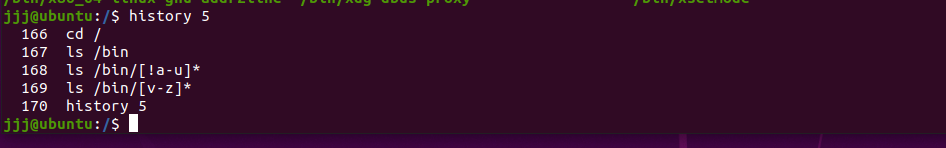
输入： ls/bin/[!a-u]\*



5.使用！！重复上一步命令：



6.使用history命令查看最近执行过的命令：



四．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。

1.设置开机不启动图形化用户界面：

打开 /etc 目录中的 inittab 文件，将“id：5：initdefault：”所在行的“5”修改为“3”后重新启动虚拟机。但是ubuntu更新后已经不采用init而是使用upstart代替。

设置开机后为字符界面：$**echo “false” | sudo tee /etc/X11/default-display-manager**

设置开机后图形化界面：$**echo “/usr/sbin/gdm” | sudo tee /etc/X11/default-display-manager**

有的系统不是默认为gdm,可以改为lightdm.

2.手工启动图形化界面。

在上述操作完成的情况下，开机后默认启动字符界面，使用startx，启动图形化界面。

**实验三 进程管理及进程通信**

19121525 蒋俊杰 2021年9月30日

**实验环境：**

Vmware worksatation下的Linux虚拟机-abuntu-20.04.3。

**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。

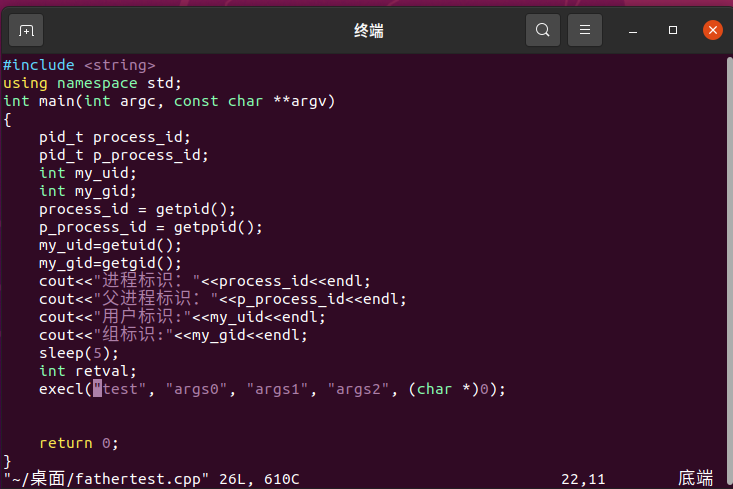
**实验内容：**

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）。

根据《实验指导》第五部分的内容指导，用vi编写c++程序进行进程控制。

**操作过程：**

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。



根据以上的要求，我编写了两个.c的程序，在父程序中显示进程的有关标识，并且用sleep函数在等待5秒后用execl函数调用另一个程序。下面是子程序：



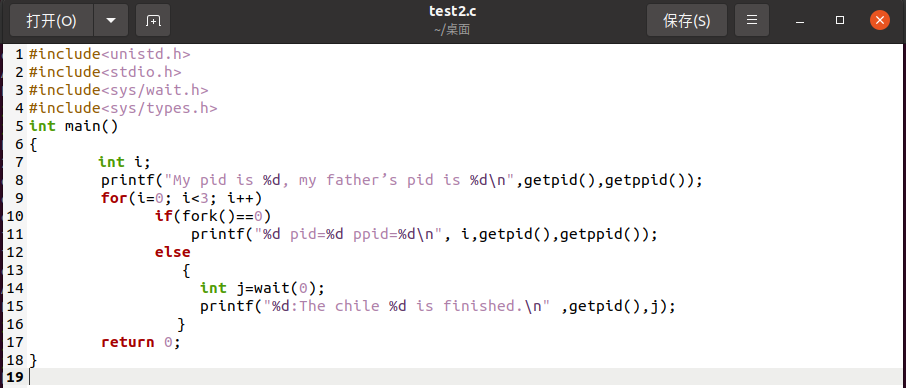
1. 编写程序。实现父进程创建一个子进程。体会子进程与父进程分别获得不同返回值，进而执行不同的程序段的方法。



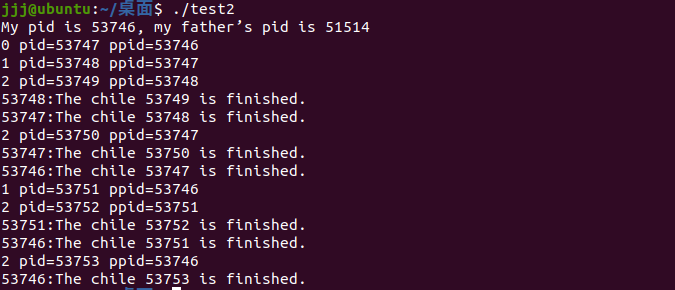
在实验指导书给出的例1当中有着几处错误较为明显：（1）头文件没有补全；（2）wait（）函数使用时参数没有补全；（3）exit（）函数使用时参数没有补全。这些问题在之后实验手册当中的例题中也有体现，不再赘述。

本例当中，使用fork()函数创建进程后会在父进程和子进程中分别返回子进程id和0，但由于进程的先后进行没有确定的顺序，谁先抢到了资源谁就先使用，因此不能确定父进程和子进程在if语句之后是谁先进行。因此，这里使用wait()函数，使父进程必须等待子进程结束之后再打印“It is parent process”,“The child process,ID number \* ,is finished”。由这个例子可以体会到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。

1. 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。（循环调用fork创建多个进程）



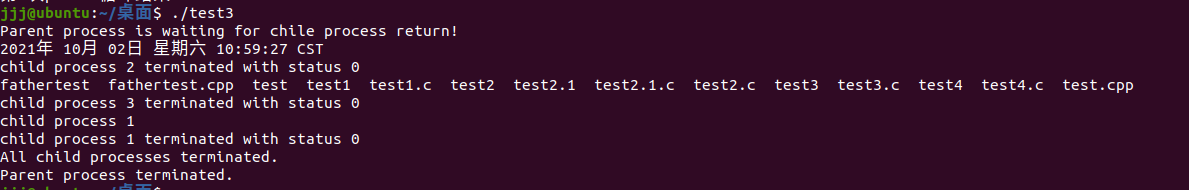
在本程序当中，在每个循环下使用fork()函数创建子进程并获得不同的返回值之后，由于在父进程中使用了wait()函数，因此父进程必须等待相应的子进程完成后才能从就绪转为执行，运行结果如下：



1. 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<stdio.h>  #include<unistd.h>  #include<sys/wait.h>  #include<sys/types.h>  #include<stdlib.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);  } |

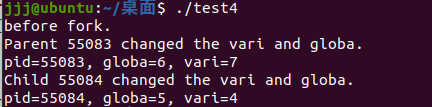
上述程序中将所有的父进程放在了一个while循环里面，等待相应子进程后就会紧接着执行。而三个子进程又启动了不同的子程序，可以通过不同程序的不同功能直观的看出是哪一个子程序先执行，并且也可以反复执行来观察子进程执行的先后顺序（这一部分的内容在思考题中有所体现）。结果如下：



1. 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。



通过上述程序可以验证子进程继承了父进程的程序、数据等资源，并且父、子进程都可以对公共变量和私有变量进行处理；子进程也继承了父进程的整个内存空间，包括栈、堆、数据段、代码段、标准I/O缓冲区等。结果如下：

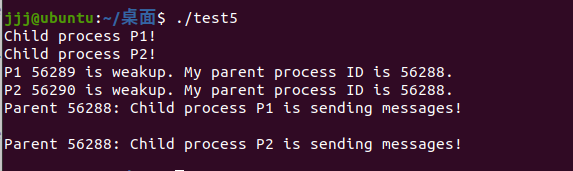


并且多次执行也可以发现，尽管pid会改变，但是对变量的修改结果并不会改变。

1. 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。

|  |
| --- |
| #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);  }  }  } |

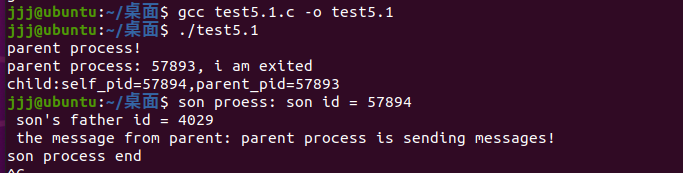
要成功实现父、子进程之间的通信，首先要了解有关管道操作的系统调用有关的一些函数；本程序中使用while与if,else嵌套实现一个父进程创建子进程，并且通过wait()函数的使用使得父进程等待子进程执行结束后再执行；通过pipe()创建管道，通过write()、read()函数来进行管道通信；通过lockf()函数来对这个管道文件进行加锁解锁，从而解决临界资源的共享问题。结果如下：



1. 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。

|  |
| --- |
| #include <stdio.h>  #include <sys/types.h>  #include <unistd.h>  #include <stdlib.h>  #include <sys/wait.h>  int main()  {  pid\_t pid;//创建进程  int fd[2],r;  char buf[50],s[50];  pipe(fd);  pid=fork();  if (pid < 0){ // 出错  perror("fork error:");  exit(0);  }  else if (pid == 0){//子进程  printf("child:self\_pid=%d,parent\_pid=%d\n",getpid(),getppid());  sleep(5); // 保证父进程先结束  if(r=read(fd[0],s,50)==-1)  printf("Can't read pipe. \n");  else  printf("son proess: son id = %d\n son's father id = %d\n the message from parent: %s", getpid(), getppid(),s);  printf("son process end\n");  exit(0);    }  else if(pid > 0){ // 父进程  lockf(fd[1],1,0); /\*管道写入端加锁\*/  sprintf(buf,"parent process is sending messages! \n");  printf("parent process! \n");  write(fd[1],buf,50);  lockf(fd[1],0,0);  /\*信息写入管道\*/  /\*管道写入端解锁\*/  printf("parent process: %d, i am exited\n",getpid());  exit(0);  }    return 0;  } |

在上述程序中我设计让父进程创建一个子进程，并且让父进程向子进程发送消息，在发送完消息后父进程就撤销，子进程段的代码中用sleep函数调控来等待父进程结束。在sleep前后用getppid()来对比父进程的变化。结果如下：



通过对比可以发现，子进程的PID为57894，并且在父进程未结束时，子进程的父进程号为57893，在父进程退出后由linux环境中的4029进程来接管孤儿进程，因此孤儿进程对系统是无害的。（pid=4029的进程是操作系统该用户下的第一号进程，它的ppid=1）.

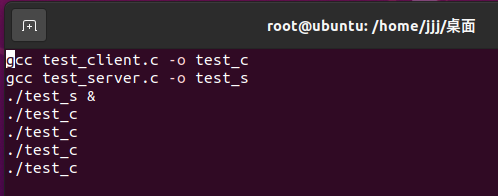
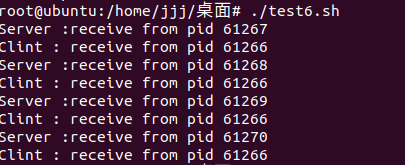


1. 编写两个程序一个是服务者程序，一个是客户程序。执行两个进程之间通过消息机制通信。消息标识MSGKEY 可用常量定义，以便双方都可以利用。客户将自己的进程标识（pid）通过消息机制发送给服务者进程。服务者进程收到消息后，将自己的进程号和父进程号发送给客户，然后返回。客户收到后显示服务者的pid 和ppid，结束。

实验可以在后台运行服务器进程（命令行后加& ），前台运行客户进程或用不同终端运行客户进程。并可通过ps 命令察看后台进程。

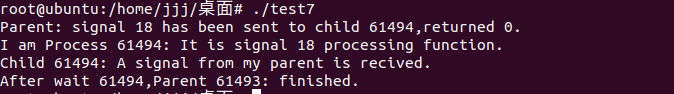
用Shell编写一个脚本程序。先在后台启动服务程序，再在前台反复执行多个客户程序，观察系统反映。体会客户/服务器体系结构。实验结束向服务器发出终止服务的信号。

编写脚本程序运行后如下：

1. 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。

程序编译运行结果：

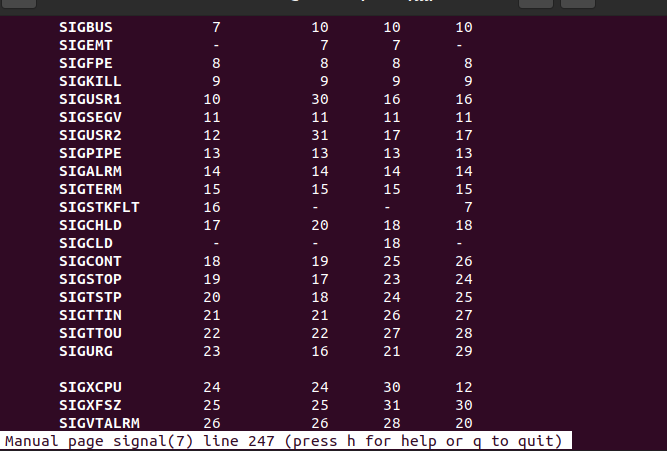
所用到了两个函数：signal()&kill()

signal()：

函数调用包含文件： #include<signal.h>

函数原型：void (\*signal(int sig, void (\*func)(int)))(int)设置一个函数来处理信号，即带有sig参数的信号处理程序。本实验使用的sig为18，缩写成不带SIG前缀，可以看到第18个是CONT，即SIGCONT

输入man 7 signal查看更详细的说明



可以看到全面的标准信号常量，同时不同架构的电脑信号对应数字不同，我这里是x86架构，所以18对应SIGCONT，原程序所想表示的也是这个。

其描述为：Continue executing, if stopped.

signal(18)作用即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。

kill()：

kill()系统调用可以把某个信号发送给一个或一组进程。接到该信号的进程依照事先设定的程序来处置。

函数调用包含文件： #include<sys/types.h>

#include<signal.h>

函数调用原形：int kill(pid\_tpid,int sig);

其中：pid的取值有 4 种

（1）pid大于 0 时，是要送往的进程标识符。

（2）pid等于 0 时，信号送往所有与调用进程同一用户组的进程。

（3）pid等于-1 时，若调用进程是超级用户的，则信号送往发送进程本身以及所有正在执行的中的程序。

（4）pid等于-1 时，若调用进程不是超级用户的，则信号送往所有有效

用户标识符与发送进程的真实用户标识符相同的进程，但发送者自己除外，即用来发送信号给某一用户下的所有进程。

1. 用信号量机制编写一个解决生产者—消费者问题的程序。

|  |
| --- |
|  |

**思考题：**

1. 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？

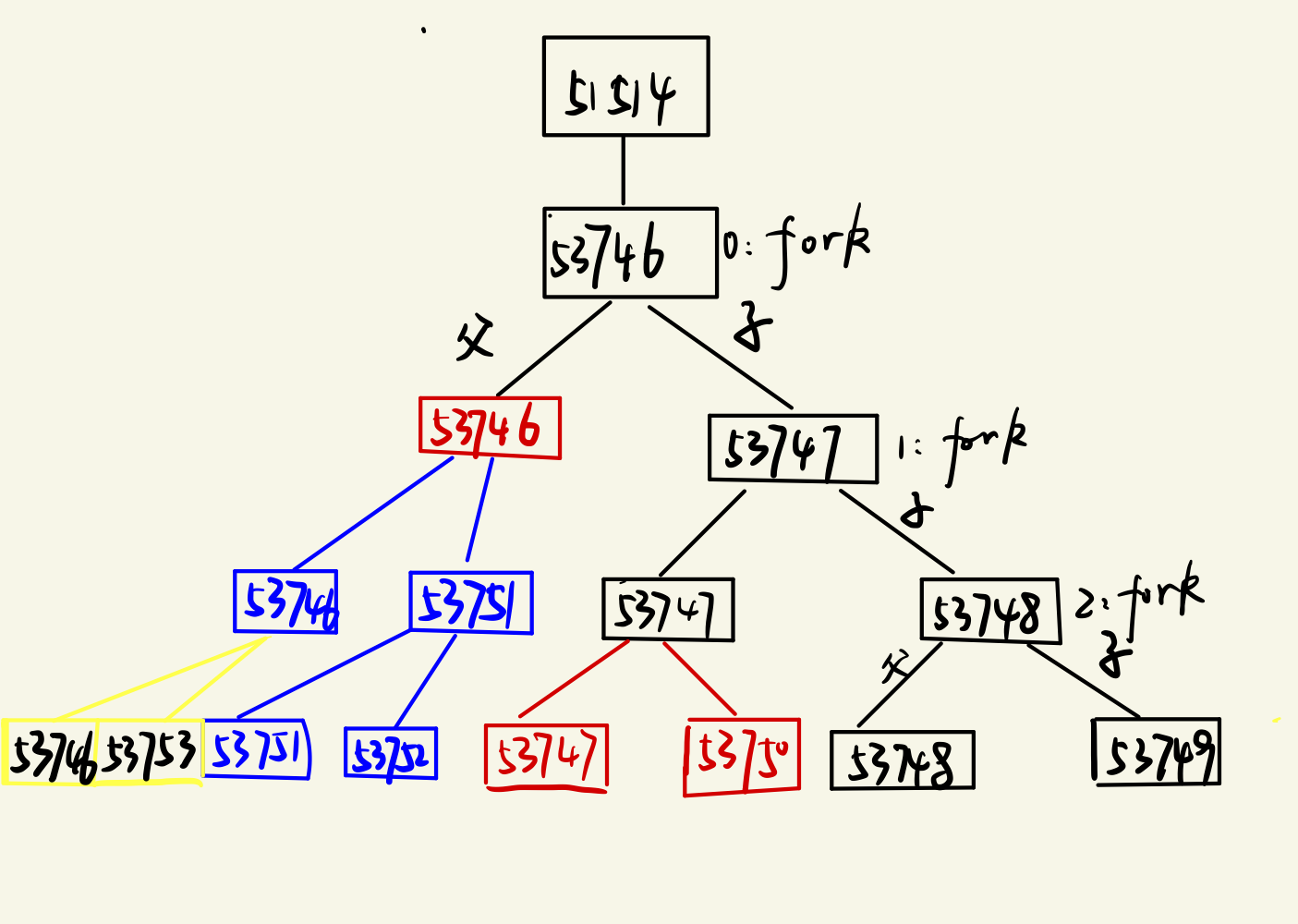
答：Linux创建进程不是从0直接创建的，而是通过一个父进程调用fork()复制一个相同的进程的形式来创建子进程的。

当一个进程由于某种原因停止时，内核并不是直接将为它保存的状态信息从内核移除，相反，进程会一直被保持在一种已经终止的状态，直到被它的父进程回收，当父进程回收已经终止的子进程时，内核会将子进程的退出状态传递给父进程，然后抛弃已经终止的进程，从此刻开始，这个进程才会消失。

子进程会继承父进程PCB里的所有资源，相当于父进程的拷贝，除了PID不一样，在调用fork函数创建进程后其余都一样。

2. 在上述操作编号3中

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。



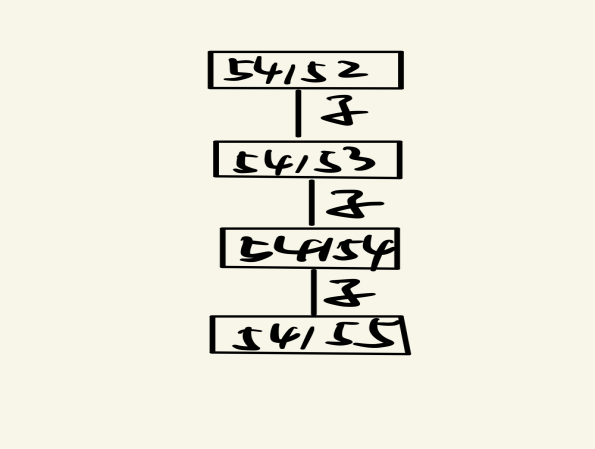
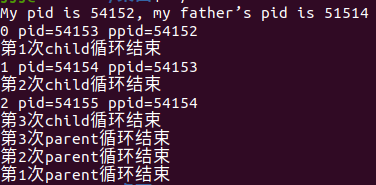
子进程的运行环境是通过拷贝父进程建立的，反复编译运行此程序可以发现，在结果上每个printf()函数输出的顺序并没有发生改变，但是进程的pid发生了变化，在不断增大。因为在两次运行程序之间系统还有别的进程进行了创建。

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。

答：修改后的程序如下所示：



在循环使用fork创建子进程后的父进程中在除了使用wait函数等待子进程执行结束后再执行父进程之外，直接使用exit（）退出父进程，使得父进程只创建一次子进程。



3.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。

一个进程调用 exec 函数执行另一个程序后，这个进程就完全被新程序代替。由于并没有产生新进程所以进程标识号不改变，除此之外旧进程的其它信息，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。反复运行此程序可以发现三个子进程执行的先后顺序不同，因为进程直接没有固定的执行顺序，谁先从就绪状态抢占到了cpu资源谁就先执行。

4. 子进程被创建后，对父进程的运行环境有影响吗？解释一下。

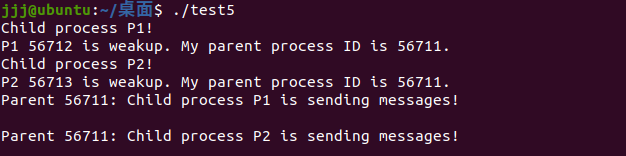
子进程创建后对父进程的运行环境没有影响，经过fork()以后，父进程和子进程拥有相同内容的代码段、数据段和用户堆栈，就像父进程把自己克隆了一遍。事实上，父进程只复制了自己的PCB块。而代码段，数据段和用户堆栈内存空间并没有复制一份，而是与子进程共享。只有当子进程在运行中出现写操作时，才会产生中断，并为子进程分配内存空间。由于父进程的PCB和子进程的一样，所以在PCB中断中所记录的父进程占有的资源，也是与子进程共享使用的。

5. 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。

管道是Unix中最古老的进程间的通信方式。我们把从一个进程连接到另一个进程的一个数据流称为一个“管道”， Linux 中两个进程可以通过管道来传递消息。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，看看会有什么变化。请解释。

如果将睡眠时间改为0，我们就可以清楚的看到程序没有人为调控时进程的执行过程：



我们对比结果不难发现：在有sleep(5)调控时child process P2是紧接着child process P1输出的，因为在fork创建的子进程中等待了5秒，不会紧跟着执行下面的代码，从而使得父进程中有fork出了子进程2；子进程2中的sleep也是同理。而去掉了sleep调控之后，父进程创建子进程2是在子进程1执行完之后的，再通过调控甚至可以找到子进程执行的大致时间。

③加锁、解锁起什么作用？不用它行吗？

当多个用户共同使用、操作一个文件的情况时，Linux采用的方法就是给文件上锁，来避免共享的资源产生竞争的状态。这样可以防止多端输入发生冲突，保证写入数据的稳定性和准确性，实现资源的互斥与同步访问。其实，管道本身具有同步互斥的机制，即在进行管道读写操作时不会被打断，所以即使没有加锁也是可以正常执行的。当某一个子进程被调度，向管道中写入数据后，等一段时间，等待管道中数据被读，读完管道之后，才允许下一次写入。

6.关于上述操作的编号7对此作何感想？

父进程如果不等待子进程退出，在子进程之前就结束了自己的 “生命” ，此时的子进程便称为孤儿进程。Linux 系统为避免存在过多孤儿进程，设置了 init 进程收留孤儿进程，使其变为孤儿进程的父进程。

7. 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？解释一下你的设想，有兴趣试一试吗。

还可以使用管道通信方式，通过一个管道文件暂存通信发送的消息，多个客户程序以争抢的形式与服务器程序通信。

8. 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？

硬中断是由硬件产生的，如外部设备鼠标、键盘，通知系统外部设备发生变化，让CPU处理。硬中断可以直接中断CPU，它会引起内核中相关的代码被触发。软中断则是由运行着的一些进程产生的，如运行一个要求输入的程序，产生I/O请求，这些请求会调用内核中可以调度I/O发生的程序，即软中断。软中断并不会直接中断CPU，这种中断是一种需要内核为正在运行的进程去做一些事情（通常为I/O）的请求。

**讨论题**：

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？

Linux系统进程是程序的一次执行，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。进程由程序段、相关数据段和PCB三部分组成。系统利用PCB来描述进程的基本情况和活动过程，进而控制和管理进程。进程除了程序所没有的PCB结构外，还具有动态性、并发性、独立性和异步性四个特征，同时具有就绪、执行和阻塞三个基本状态，在此三种基本状态间转换。

1. 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。

操作系统与用户间接口之一的程序接口是为用户程序在执行中访问系统资源而设置的，是用户程序取得操作系统服务的唯一途径。它是由一组系统调用组成的，每一个系统调用都是一个能完成特定功能的子程序。每当应用程序要求OS提供某种服务（功能）时，便调用具有相应功能的系统调用（子程序）。C语言中，提供了与各系统调用一一对应的库函数，这样，应用程序便可通过调用对应的库函数来使用系统调用。Linux其系统调用本身已经采用C语言编写，并以函数形式提供，故在用C语言编制的程序中，可直接使用系统调用。

如：在C语言程序中使用系统调用ls

#include <unistd.h>

int main(void)

{

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

return 0;

}



1. 什么是进程？如何产生的？举例说明。

进程是程序的一次执行，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。为了使参与并发执行的每个程序（含数据）都能独立的运行，在操作系统中必须为之配置一个专门的数据结构，称为进程控制块（ProcessControlBlock, PCB）。系统利用PCB来描述进程的基本情况和活动过程，进而控制和管理进程。这样，由程序段、相关数据段和PCB三部分便构成了进程，这就是进程的产生。

举例：使用fork()函数很直观的创建一个子进程，这样就产生了一个进程。

1. 进程控制如何实现的？举例说明。

进程控制是通过原语实现的。原语的特点是执行期间不允许中断，一个操作中的所有动作要么全做，要么全不做。这种不可被中断的操作也称为原子操作。与进程控制相关的有创建、中止、阻塞与唤醒、挂起与激活这些原语。

举例：使用fork()函数创建一个子进程，这就是父进程通过OS调用进程创建原语Creat产生了一个进程。

1. 进程通信方式各有什么特点？用程序及运行结果举例说明。

通信方式有：

1. 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。
2. 管道通信系统：能够连接一个写进程和一个读进程，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，又名pipe文件。由于写进程从管道的入端将数据写入管道，而读进程则从管道的出端读出数据。
3. 消息传递系统：源程序可以直接或间接地将消息传送给目标进程。
4. 客户机-服务器系统：通过套接字、远程过程调用和远程方法调用通信。

举例：上面的实验7就是一个使用管道通信的例子。

1. 管道通信如何实现？该通信方式可以用在何处？
2. 互斥，即当一个进程正在对pipe执行读/写操作时其它进程必须等待；
3. 同步，指当写（输入）进程把一定数量的数据写入pipe时，便去睡眠等待，知道读（输出）进程取走数据后，再把它唤醒。当读进程读一空pipe时，也应睡眠等待，直至写进程将数据写入管道后，才将之唤醒。
4. 确定对方是否存在，只有确定了对方已存在时，才能进行通信。

用处：由于当前操作系统的不惟一性，各个系统都有其独自的通信协议，导致了不同系统间通信的困难。尽管TCP/IP协议目前已发展成为Internet的标准，但仍不能保证C/S应用程序的顺利进行。命名管道作为一种通信方法，有其独特的优越性，这主要表现在它不完全依赖于某一种协议，而是适用于任何协议——只要能够实现通信。

1. 什么是软中断？软中断信号通信如何实现？

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。通过signal()和kill()函数实现，kill()函数发送信号给进程，signal()函数通过调用自己设置方式处理对应信号。每个信号都对应一个正整数常量，代表同一用户的进程之间传送事先约定的信息的类型。每个进程在运行时，都要通过信号量机制来检查是否有信号到达。若有，便中断正在执行的程序，转向与该信号对应的处理程序，以完成对该事件的处理，处理结束后再返回到原来的中断处继续执行。实质上，信号机制是对中断机制的一种模拟。

**体会：**

本次实验基本全是使用日常很少用的Linux操作系统完成的，学到了许多linux的指令，也学到了在linux上编译C语言程序。最主要的还是对进程的学习，通过自己新手实践，加深了对进程的理解，更深刻的体会了进程的控制与通信，也体会了Linux这袭承unix操作系统的与windows截然不同的设计思想。具体有如对进程PID的理解，知道了父子进程相关的孤儿进程和僵尸进程（wait()、exit()、fork()函数），学会了使用C语言调用系统调用（exec函数簇），了解了进程间的管道通信方式——互斥使用临界资源的思想（pipe(fd)、lock(fd[1],0,0)、write(fd[1],buf,50)）和消息机制通信方式，还有对软中断和硬中断的认识（signal()、kill()）。收获颇丰。

**实验三 LINUX进程调度与系统监视**

19121525 蒋俊杰 2021年10月9日

**实验环境：**

Vmware worksatation下的Linux虚拟机-abuntu-20.04.3。

**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：**

熟悉并掌握在linux环境下的作业和进程调度的基本管理、at进程调度和cron调度的相关操作、进行系统性能监视。

**操作过程：**

一、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。

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

（2） 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。



（3） 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。



【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。



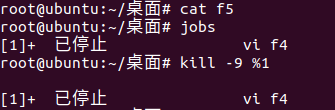
1. 输入命令“fg 2”，将“find / -name inittab > f5 &”作业切换到前台。屏幕显 示出“find / -name inittab > f5”命令，并执行此命令。稍等片刻，作业完成后屏幕 再次出现命令提示符。



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

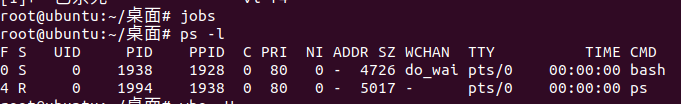
（4） 再次输入命令“jobs”，可发现当前系统中的只有一个已停止的作业“vi f4”。

（5） 输入命令“kill -9 %1”，终止“vi f4”作业。

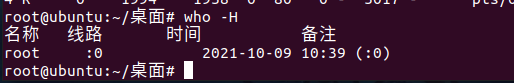


（6） 稍等片刻，输入命令“jobs”，查看到当前没有任何作业。

（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。



二、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。

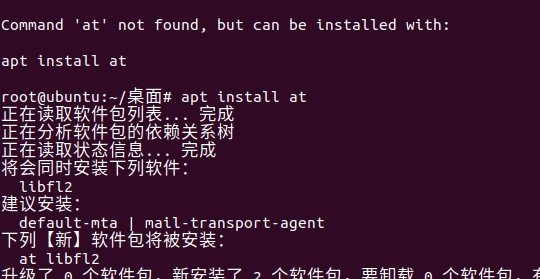
at 调度：在指定的时间执行一次特定的作业。

batch 调度：在系统空闲时执行一次特定的作业。

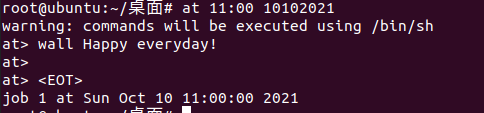
cron 调度：每到指定的时间就执行特定的作业，可执行多次。

【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。

1. 超级用户输入命令“at 11:00 10102021”，设置 2021 年 10 月 10 日 11 时执行的 at 调度的内容。（at 命令可能需要使用apt install at安装）



1. 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。
2. 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。

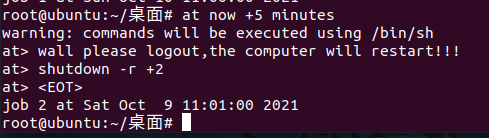


【操作要求 2】设置一个调度，要求 5 分钟后向所有用户发送系统即将重启的消息，并在 2 分钟后重新启动计算机。

（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。

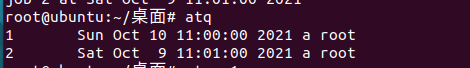
（2） 屏幕出现 at 调度的命令提示符“at>”，输入“wall please logout; the computer will restart.”，向所有用户发送消息。

（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。

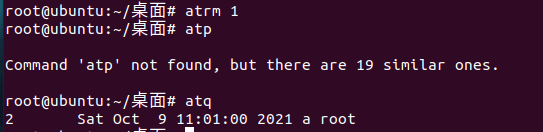


【操作要求 3】查看所有的 at 调度，并删除 21 年 10 月 10 日执行的调度任务。

（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。



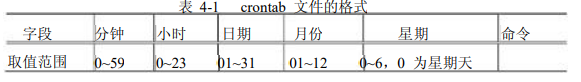
（2）输入“atrm 1”命令删除作业号为 1 的 at 调度，并再次输入“atq”命令查看剩余的 所有 at 调度内容。



（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。

三、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， 如图所示。



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，并将查看结果保存于 ps.log 文件。

（1） 以普通用户登录，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。最后退回到普通用户。

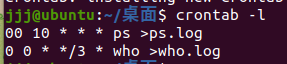
（6） 等待 1 分钟后，查看 ps.log 文件的内容，如果显示出正确的内容，那么说明 crontab 调度设置成功。的



【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。

（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 ”。

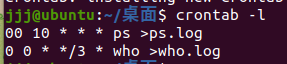


（3） 最后保存并退出 vi 编辑器。

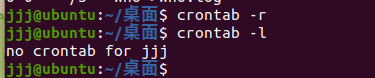
（4） 为立即查看到 crontab 调度的结果，切换为超级用户，并适当修改系统时间，如修改 为 3 月 31 日 23 点 59 分。最后退回到 helen 用户。

【操作要求 3】查看 cron 调度内容，最后删除此调度。

1. 输入命令“crontab -l”，查看 cron 调度内容。



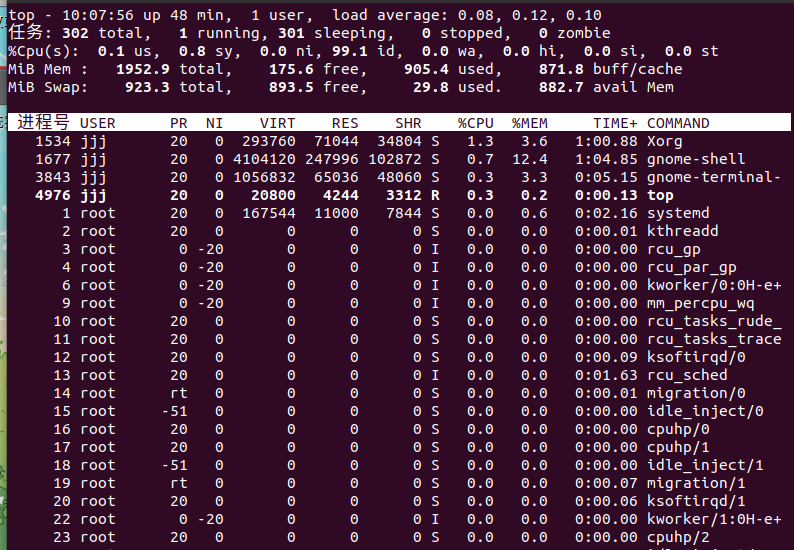
1. 输入命令“crontab -r”，删除 cron 调度内容。
2. 再次输入命令“crontab -l”，此时无 cron 调度内容。



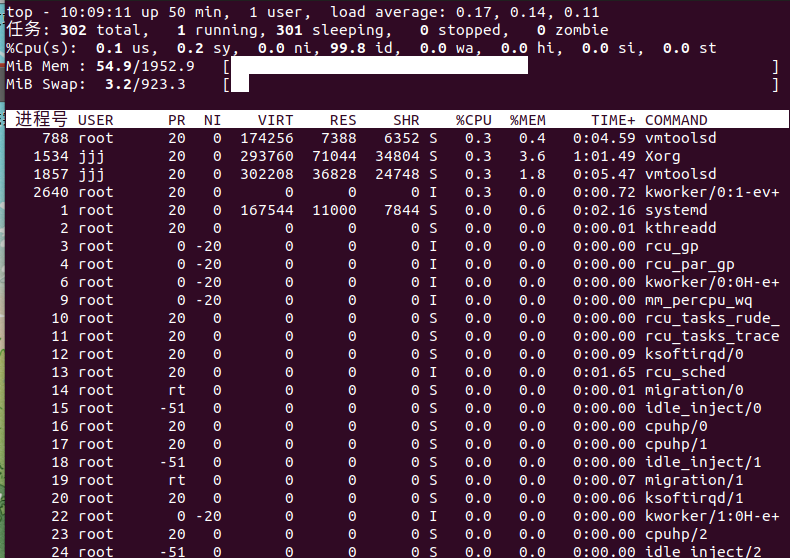
四、系统性能监视

【操作要求 1】利用 Shell 命令监视系统性能

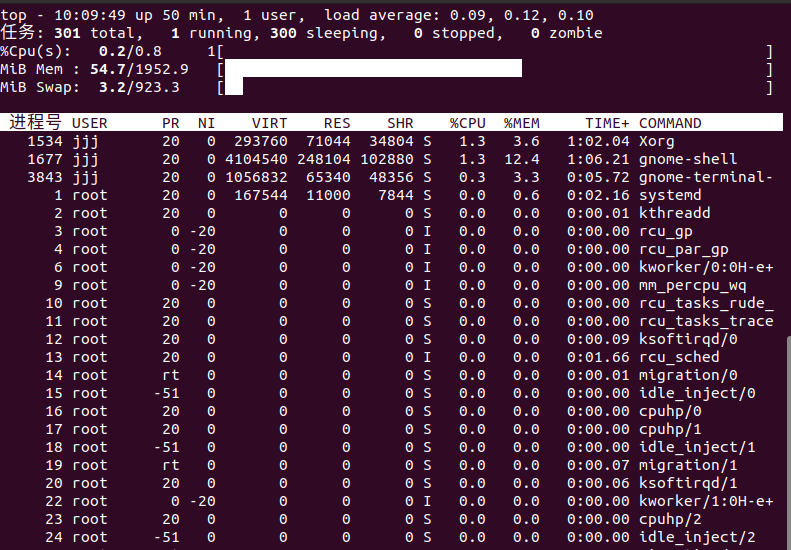
1. 输入命令“top”，屏幕动态显示 CPU 利用率、内存利用率和进程状态等相关信息。



1. 按下 M 键，所有进程按照内存使用率排列。



1. 按下 T 键，所有进程按照执行时间排列

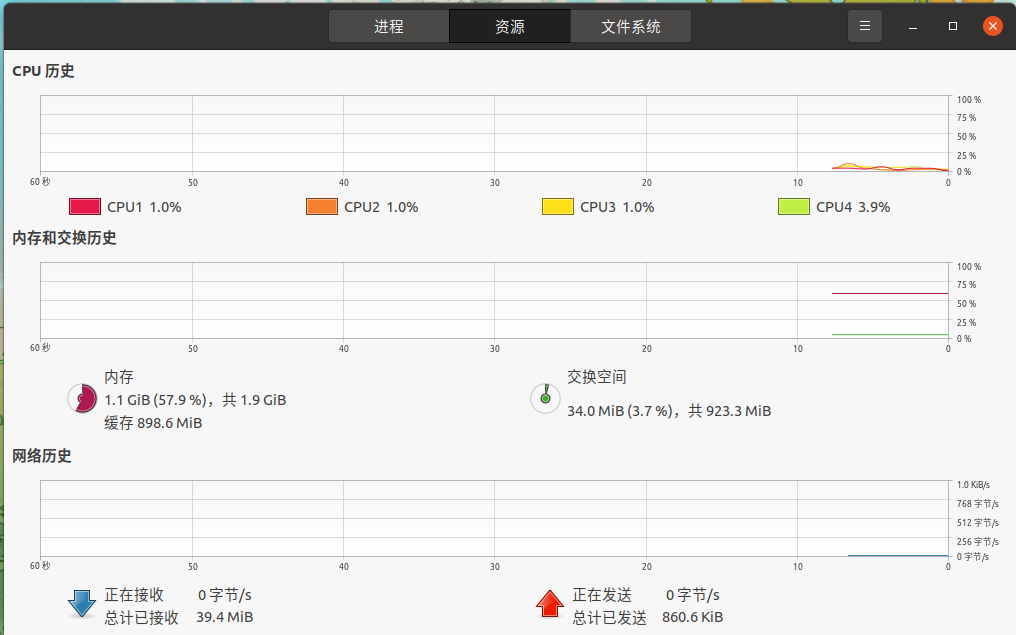


1. 最后按下 P 键，恢复按照 CPU 使用率排列所有进程。
2. 按下 CTRL+C 组合键结束 top 命令。

【操作要求 1】利用「系统监视器」工具监视 CPU 使用情况。

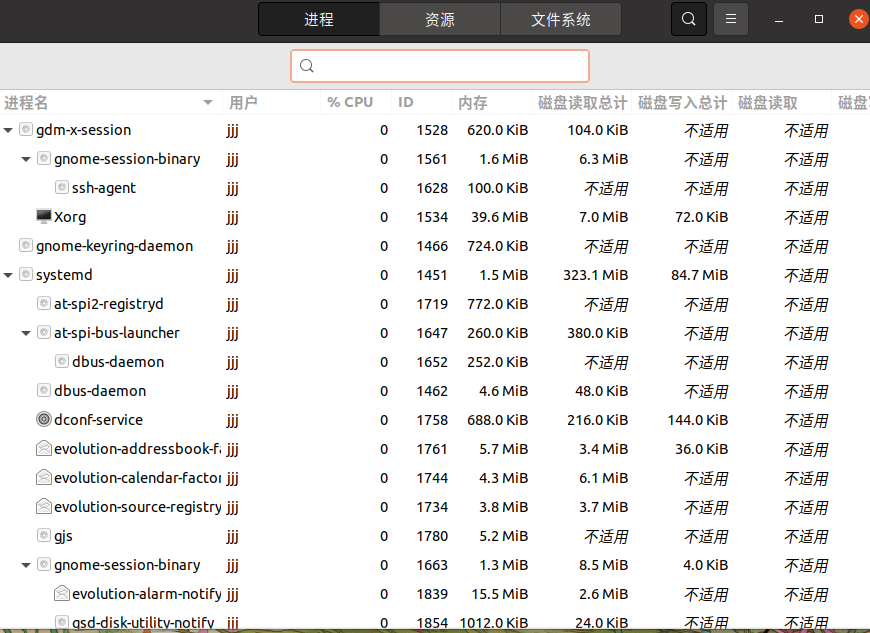
（1） 启动 GNOME 桌面环境，依次单击「系统」菜单=>「管理」=>「系统监视器」，打 开「系统监视器」窗口。

（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，

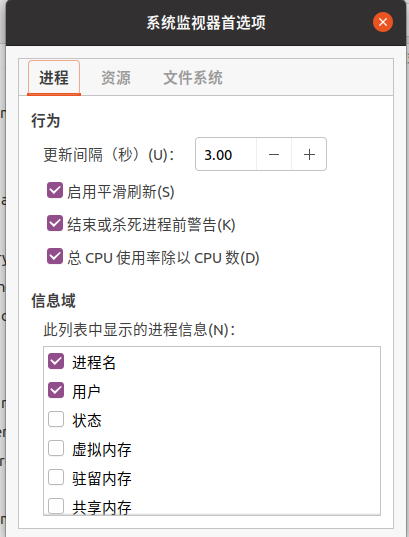


【操作要求 2】利用「系统监视器」查看当前所有的进程，要求显示出启动进程的用户。

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。



1. 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。

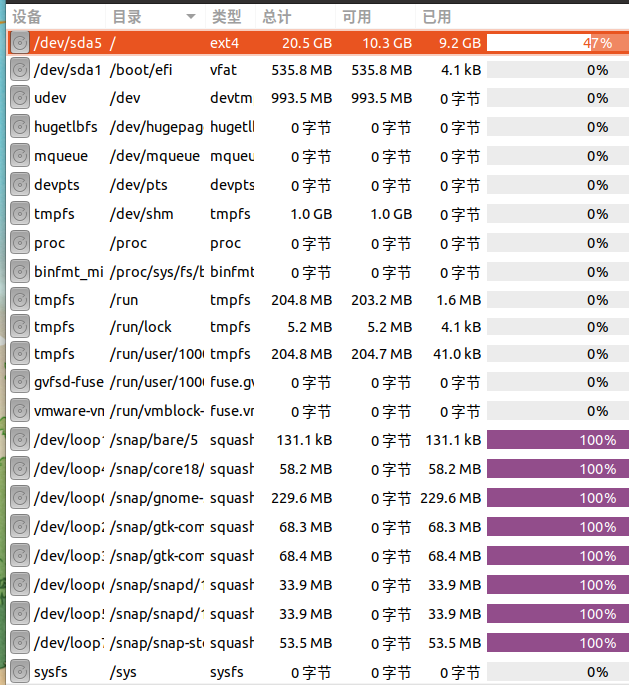


【操作要求 3】利用「系统监视器」查看所有的文件系统

1. 在「系统监视器」窗口单击「文件系统」选项卡，显示当前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。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：

【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

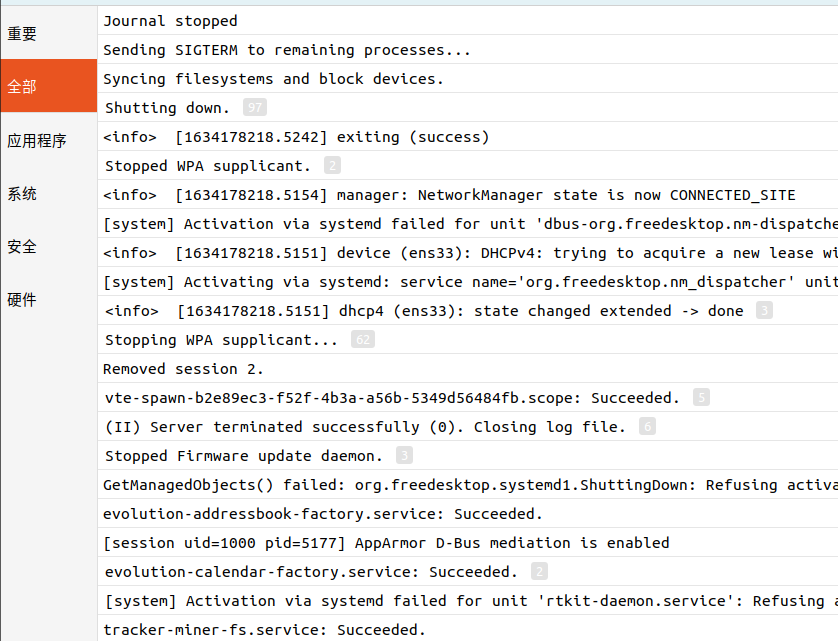
messages 记录系统运行过程的相关信息，包括 I/O、网络等

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

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：



**实验五 用户与组群管理**

19121525 蒋俊杰 2021年10月21日

**实验环境：**

Vmware worksatation下的Linux虚拟机-abuntu-20.04.3。

**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：**

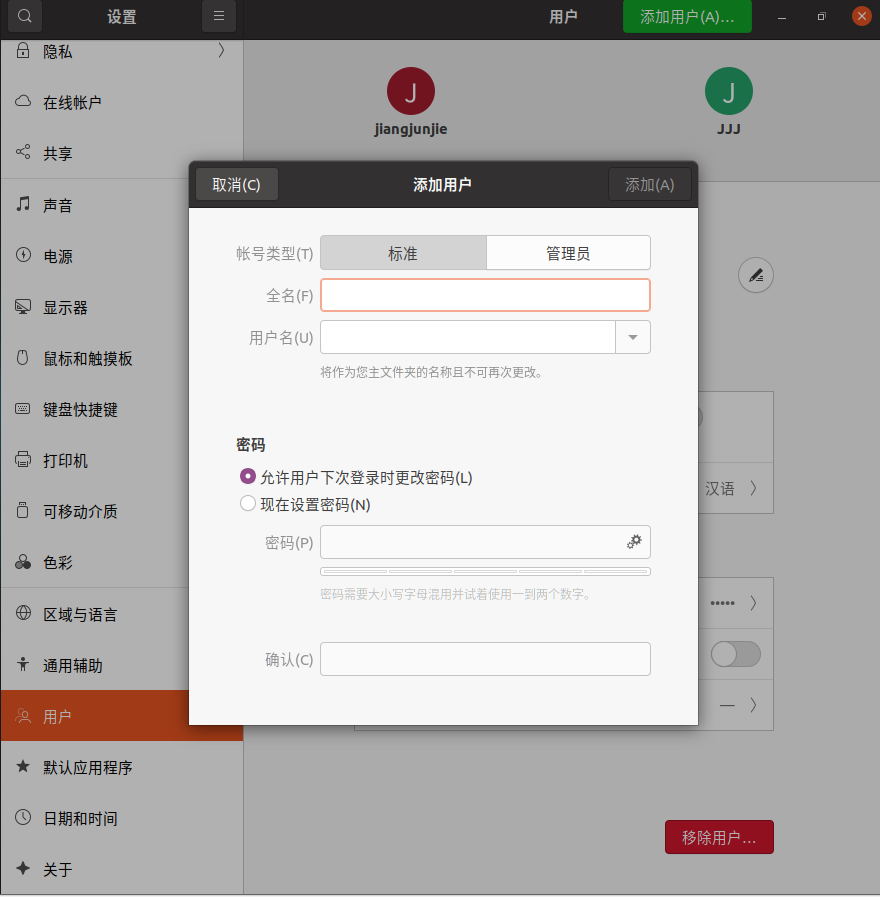
熟悉并掌握在linux环境下的不同方式创建用户与组群。

**操作过程：**

一、桌面环境下管理用户与组群

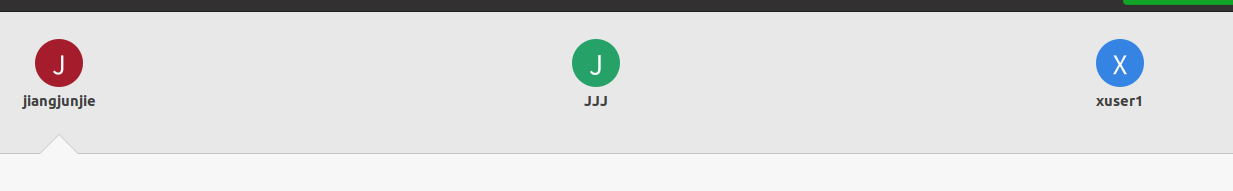
【操作要求 1】新建两个用户帐号，其用户名为 xuser1 和 xuser2，口令为“e12ut59er”和 “wfu1t28er”。

1. 以超级用户身份登录，依次单击「设置」菜单=>「用户」，启动「用户管理者」窗口。

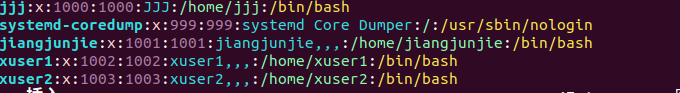


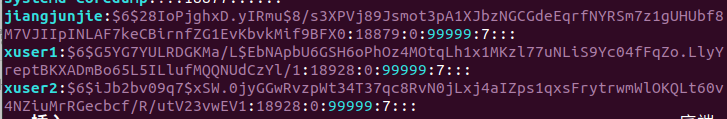
（2）单击工具栏上的「添加用户」按钮，出现「添加用户」窗口。在「用户名」文本 框中输入用户名“xuser1”，在「口令」文本框中输入口令“e12ut59er”，在「确认口令」 文本框中再次输入口令“e12ut59er”，然后单击「确定」按钮，返回「用 户管理者」窗口。

（3）用同样的方法新建用户 xuser2，完成后「用户管理者」窗口如图所示。（有一些账户是之前添加的）



（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。

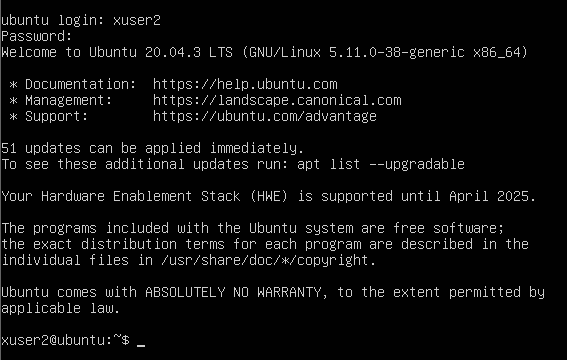








（5） 按下 CTRL+ALT+F2 组合键切换到第 2 个虚拟终端，输入用户名 xuser2 和相应的口 令可登录 Linux 系统，说明新建用户操作已成功。



（6） 输入“pwd”命令，屏幕显示用户登录后进入用户主目录“/home/xuser2”。

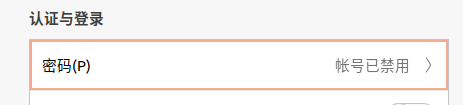


（7） 输入“exit”命令，xuser2 用户退出登录。

（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

1. 打开终端输入命令：passwd –l xuser2(如果想解除锁定则使用-u)
2. 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。



1. 再按下 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】

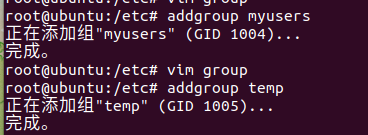
1. 以超级用户身份登录，依次单击「设置」菜单=>「用户」，启动「用户管理者」窗口,点击右下角的删除用户按键即可删除。



【操作要求 4】新建两个组群，分别是 myusers 和 temp。（在我的虚拟机上的图形界面没有相关功能，只能通过命令的形式进行）

【操作步骤】

1. 使用命令addgroup myusers创建用户组myusers。



（2） 用相同的方法新建 temp 组群，在etc/group中可以看到相关信息

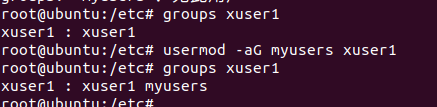


【操作要求 5】修改 myusers 组群属性，将 xuser1用户加入 myusers 组群。

【操作步骤】

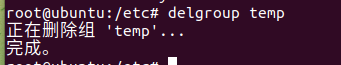
（1） 先使用groups xuser1查看用户xuser1所在组群。

（2） 使用usermod –a myusers xuser1 将xuser1添加到用户组myusers中去，其中参数-a是为了不离开原组群，又同时加入新组群。



【操作要求 6】删除 temp 组群

【操作步骤】 使用命令delgroup temp:



二、编辑用户配置文件（在我本人的虚拟机上因为图形界面没有想过功能。下面的操作并没有实践，只是进行了相关了解）

【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。 「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。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，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，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。

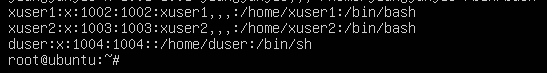
（4） 以 xuser1 用户登录，并启动 GNOME 桌面环境，查看应用程序的字体和桌面环境。 GNOME 的用户配置文件编辑功能允许超级用户创建多种用户配置方案，分配给不同类型 的普通用户。

三、利用 Shell 命令管理用户与组群

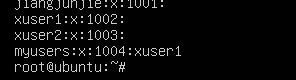
【操作要求 1】新建一名为 duser 的用户，其口令是“tdd63u2”，主要组群为 myusers。

【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。



1. 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。

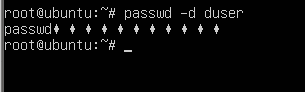
（7） 输入“exit”命令，duser 用户退出登录。

【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。



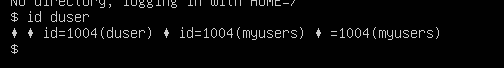
（3）按下 ALT+F3 组合键，再次切换到第 3 个虚拟终端，在“Login：”后输入用户名“duser”， 按下 Enter 键就直接出现 Shell 命令提示符，说明 duser 用户不需口令即可登录，如下所示。



【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。



【操作要求 4】从普通用户 duser 切换为超级用户

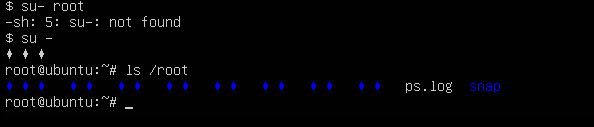
【操作步骤】

（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 第37页 共 103 页 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。



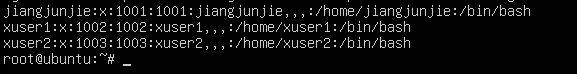
【操作要求 5】一次性删除 duser 用户及其工作目录

【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。



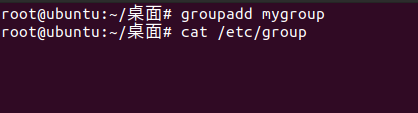
1. 输入命令“ls /home”，发现 duser 的主目录/home/duser 也不复存在。



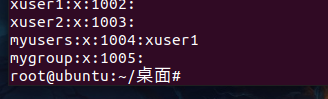
【操作要求 6】新建组群 mygroup

【操作步骤】

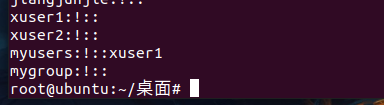
1. 在超级用户的 Shell 提示符后输入命令“groupadd mygroup”，建 立 mygroup.



（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。（/etc/group中newgroup的信息被删除了）



四、批量新建多个用户帐号

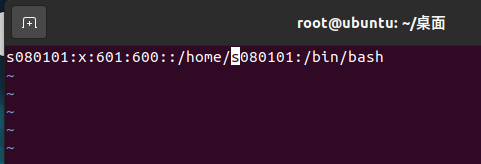
【操作要求】为全班同学 20 位同学创建用户帐号，用户名为“s”+学号的组合，其中班级 名册中第一位同学的学号为 080101。所有同学都属于 class0801 组群。所有同学的初始口令 为 111111。

【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。



1. 输入命令“vi student”，新建用户信息文件。
2. 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”。

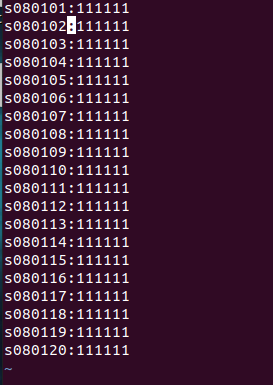


1. 按下 ESC 键，切换到命令行模式，拖动鼠标，将整行选中，然后 按下字母键 y 两次。也就是将当前选中的行放到 vi 的暂存区域（类似于 Windows 的剪贴板）。
2. 然后按下字母键 p，就复制一行信息，重复此操作 19 次，然后部分 修改每位同学用户信息不同的地方。
3. 最后编辑完成的文件，如图所示。最后保存并退出 vi。
4. 输入命令“vi stu-passwd”，新建用户口令文件。



（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。

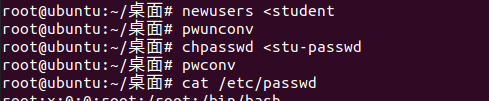


（10） 输入命令“newusers < students”，批量新建用户帐号。

（11） 输入命令“pwunconv”，暂时取消 shadow 加密。

（12） 输入命令“chpasswd

（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。

**实验六 SHELL 编程**

19121525 蒋俊杰 2021年11月4日

**实验环境：**

Vmware worksatation下的Linux虚拟机-abuntu-20.04.3。

**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《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。

vi 编辑程序中有三种不同的操作方式。当用户正在编辑一个文件时，vi 可能处在插入 方式、转义命令方式或末行命令方式中的一种方式下：

命令模式：打开文件后首先进入的就是命令模式，是使用vi的入口；可以输入命令来执行多种功能如：移动、删除、替换等。

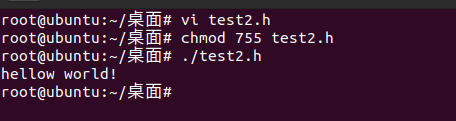
插入方式：在需要输入文本内容时进入插入方式；通常通过三个按键来进入插入方式：

i-光标前插入、a-光标后插入、o-下一行插入。在插入方式中通过[Esc]键退出插入方式进入转义命令方式。

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。可以通过在末行命令方式下输入wq或者x来进行存盘并退出。需要注意的是，x与wq都可以写入文件并退出，但wq是强制文件写入并退出。即使文件没有被修改也强制写入，并且更新文件的修改时间。X写入文件并退出，仅仅当文件被修改时才写入，并更新文件的修改时间，否则不会更新文件修改时间。

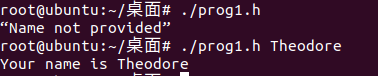
2. 创建和执行Shell 程序：用前面介绍的Vi 或其他文本编辑器编写Shell 程序，并将文件以文本文件方式保存在相应的目录中，同时使用chmod 命令将文件的权限设置为可执行模式。

Shell程序中的内容为：echo “hellow world!”



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。

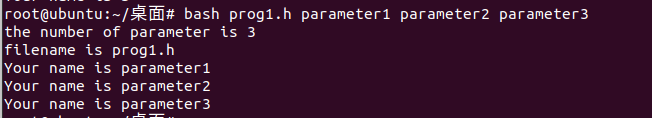




当没有指定参数时输出Name not provided;当指定参数时可以根据参数个数的变化来输出对应的参数。

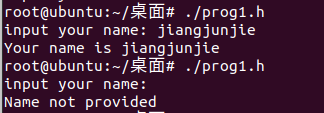
|  |
| --- |
| #Name display program  if [ $# == 0 ]  then  echo “Name not provided”  else  echo "Your name is" $1  fi |

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。



|  |
| --- |
| #Name display program  if [ $# == 0 ]  then  echo "Name not provided"  else  echo "the number of parameter is" $#  echo "filename is" $0  while [ -n "$1" ] # 通过循环，每次读取$1位置参数  do  echo "Your name is" $1  shift # 把$2位置参数向前移动到$1，原$1位置参数不可用  done  fi |

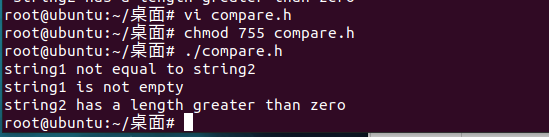
5. 修改例1程序（即上面的 prog1.h），用read命令接受键盘输入。若没有输入显示第一种 提示，否则第二种提示。



|  |
| --- |
| #Name display program  read -p "input your name: " NAME  if [ ! -n "$NAME" ]  then  echo "Name not provided"  else  echo "Your name is" $NAME  fi |

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。

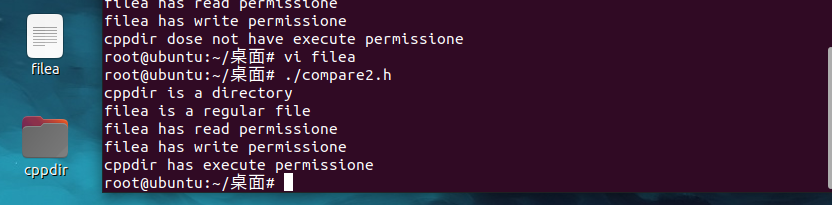
【例2】：



|  |
| --- |
| string1="The First one"  string2="The second one"  if [ string1 = string2 ]  then  echo "string1 equal to string2"  else  echo "string1 not equal to string2"  fi  if [ string1 ]  then  echo "string1 is not empty"  else  echo "string1 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 |

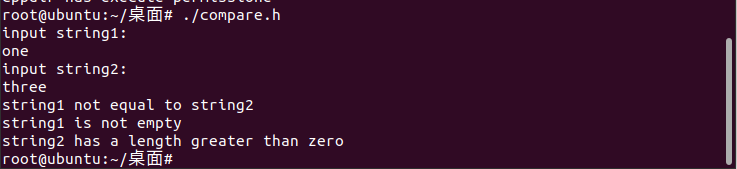
【例3】：

在该例中如果想达到预期的输出结果需要在compare2.h所在目录下创建一个子目录cppdir和一个文件filea。



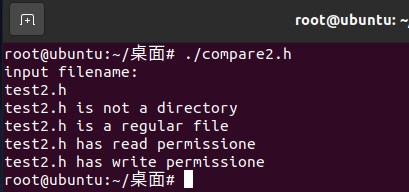
|  |
| --- |
| if [ -d cppdir ]  then  echo "cppdir is a directory"  else  echo "cppdir is not a directory"  fi  if [ -f filea ]  then  echo "filea is a regular file"  else  echo "filea is not a regular file"  fi  if [ -r filea ]  then  echo "filea has read permissione"  else  echo "filea dose not have 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 dose not have execute permissione"  fi |

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。



|  |
| --- |
| echo "input string1:"  read string1  echo "input string2:"  read string2  if [ string1 = string2 ]  then  echo "string1 equal to string2"  else  echo "string1 not equal to string2"  fi  if [ string1 ]  then  echo "string1 is not empty"  else  echo "string1 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 |

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。

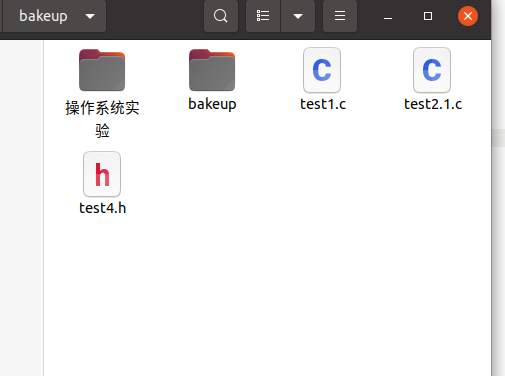


|  |
| --- |
| echo "input filename:"  read filename  if [ -d "$filename" ]  then  echo "$filename is a directory"  else  echo "$filename is not a directory"  fi  if [ -f "$filename" ]  then  echo "$filename is a regular file"  else  echo "$filename is not a regular file"  fi  if [ -r "$filename" ]  then  echo "$filename has read permissione"  else  echo "$filename dose not have read permissione"  fi  if [ -w "$filename" ]  then  echo "$filename has write permissione"  else  echo "$filename dose not have write permissione"  fi |

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。

【例4】：将当前目录中的文件(用命令 ls 列出)，复制到 backup 子目录中建立。

打开bakeup后发现成功复制。





但是cp不能将文件夹复制到此文件夹的任何子目录之内。

|  |
| --- |
| for filename in `ls`  do    cp -rf $filename bakeup/$filename  if [ $? != 0 ]  then  echo "copy $filename failed"  fi  done |

【例5】：进行十个偶数的相加



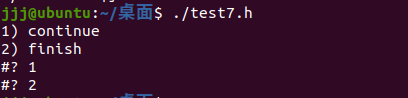
|  |
| --- |
| loopcount=0  result=0  while [ $loopcount -lt 10 ]  do  let loopcount+=1  let result+=($loopcount \* 2)  done  echo "result is $result" |

【例6】：用until语句实现改写例5



|  |
| --- |
| loopcount=0  result=0  until [ $loopcount -ge 10 ]  do  let loopcount+=1  let result+=($loopcount \* 2)  done  echo "result is $result" |

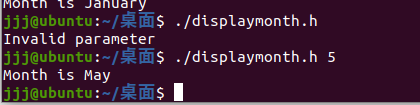
【例7】：执行以下语句时系统生成一个显示数字的菜单，其中 1 表示 continue，2 表示 finish。 用户选择 1，则 item 中含有 continue；选择 2，则 item 中含有 finish。



|  |
| --- |
| select item in continue finish  do  if [ $item = "finish" ]  then  break  fi  done |

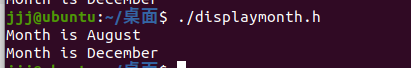
10. 用vi 编写《实验指导》“第四部分 Shell 程序设计”中的例8 及例9掌握条件语句的用法，函数的用法，观察运行结果。

【例8】：假如给出月份数字作为参数编写程序 displaymonth.h



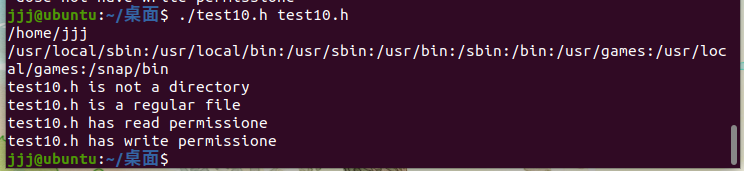
|  |
| --- |
| case $1 in  01|1) echo "Month is January";;  02|2) echo "Month is February";;  03|3) echo "Month is March";;  04|4) echo "Month is April";;  05|5) echo "Month is May";;  06|6) echo "Month is June";;  07|7) echo "Month is July";;  08|8) echo "Month is August";;  09|9) echo "Month is September";;  010|10) echo "Month is October";;  011|11) echo "Month is November";;  012|12) echo "Month is December";;  \*) echo "Invalid parameter";;  esac |

【例9】：上面显示月份的程序改成函数如下



|  |
| --- |
| displaymonth ( ) { #定义函数  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";;  \*) echo "Invalid parameter";;  esac  }  displaymonth 8 #调用函数  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。



|  |
| --- |
| echo $HOME  echo $PATH  filename="$1"  if [ -d "$filename" ]  then  echo "$filename is a directory"  else  echo "$filename is not a directory"  fi  if [ -f "$filename" ]  then  echo "$filename is a regular file"  else  echo "$filename is not a regular file"  fi  if [ -r "$filename" ]  then  echo "$filename has read permissione"  else  echo "$filename dose not have read permissione"  fi  if [ -w "$filename" ]  then  echo "$filename has write permissione"  else  echo "$filename dose not have write permissione"  fi |

思考：到此为止你对Shell 有所认识了吧？怎么样？自己再编两个程序：

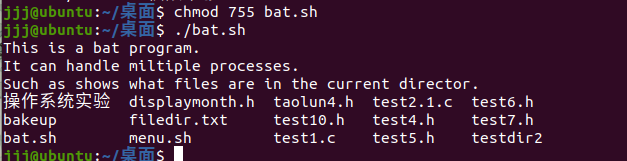
1. 做个批处理程序，体会一下批处理概念。
2. 做个菜单，显示系统环境参数。将此程序设置为人人可用。

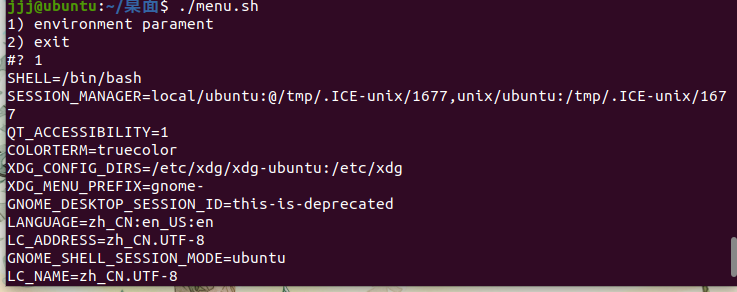
1.批处理程序

|  |
| --- |
| echo "This is a bat program."  echo "It can handle miltiple processes."  echo "Such as shows what files are in the current director."  ls |

2.菜单程序

|  |
| --- |
| select var in "environment parament" "exit"  do  case $var in  "exit") break;;  "environment parament") env;;  \*)echo "Invalid parameter";;  esac  done |





**研讨：**

1. Linux 的 Shell 有什么特点？

Shell 编程语言具有普通编程语言的很多特点，比如它也有循环结构和分支控 制结构等，用这种编程语言编写的 Shell 程序与其他应用程序具有同样的效 果。

1. 怎样进行 Shell 编程？如何运行？有什么条件？

$ 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）

1. 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（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。

4.编程：

|  |
| --- |
| ls >> filedir.txt  mkdir testdir2  cp \*.c testdir2  cd testdir2  for filename in `ls`  do  chmod -r $filename  done  ls -l  cd ..  ls >>filedir.txt  cat /etc/group >>filedir.txt |

