实验题目：**Linux** 操作系统基本命令

姓名：王波　　学号：19122557　　实验日期：2021 年 9 月 9 日

Linux 操作系统基本命令

【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。

【实验环境】

虚拟机中 Linux 环境（CentOS7 镜像）

【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。

【实验步骤】

（一）登录命令 Linux 操作系统界面，进入到超级用户模式，查看主机 ip

```
[master@master ~]$ su
密码：
[root@master master]# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.224.138  netmask 255.255.255.0  broadcast 192.168.224.255
        inet6 fe80::a08c:4c11:75ec:bea  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:49:43:a0  txqueuelen 1000  (Ethernet)
        RX packets 215784  bytes 317216554 (302.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18344  bytes 1129865 (1.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

（二）查看命令信息

① 执行 pwd 查看当前目录

② 用 who am i 看看当前用户信息

③ 通过 who 看看有谁在系统中

④ 用 vmstat 显示系统状态

```
[root@master master]# pwd
/home/master
[root@master master]# who am i
master   pts/0        2021-09-11 19:39 (:0)
[root@master master]# who
master   :0           2021-09-11 18:28 (:0)
master   pts/0        2021-09-11 19:39 (:0)
[root@master master]# vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0   1288  81836      0 869448    0    0   332   115  201  257  7  5 88  1  0
[root@master master]# top

top - 19:40:56 up  1:15,  2 users,  load average: 0.43, 0.28, 0.20
Tasks: 209 total,   1 running, 208 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.8 us,  1.0 sy,  0.0 ni, 94.9 id,  0.0 wa,  0.0 hi,  0.3 si,  0.0 st
KiB Mem :  1863004 total,    77744 free,   914136 used,   871124 buff/cache
KiB Swap: 10485756 total, 10484468 free,     1288 used.   752324 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 2352 master    20   0 3035880 205760  69832 S  1.5 11.0   2:40.38 gnome-shell
    9 root      20   0       0      0      0 S  1.2  0.0   0:04.88 rcu_sched
  710 root      20   0  295376   5184   3932 S  1.2  0.3   0:20.07 vmtoolsd
 1310 root      20   0  320592  47784  24328 S  1.2  2.6   0:41.06 X
 5128 master    20   0  688008  28788  17732 S  1.2  1.5   0:02.66 gnome-terminal-
 5290 root      20   0  162100   2332   1580 R  0.6  0.1   0:00.81 top
    4 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
[root@master master]# id
uid=0(root) gid=0(root) 组=0(root) 环境=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@master master]#
```

执行 pwd 查看到当前目录为/home/master;Who am i 查看到当前用户为 master pts/0; who 可以看到系统中有两个用户一个是 master :0,另一个是 master pts/0; vmstat 可以看到当前系统的状态；通过 id 命令还可以查看到当前用户名 stu，及组名。

思考题：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

答：由上图可以知道，用户名 root，用户标识 0，组名 root,组标识 0，当前处在 home/master 目录下。master 和 root 用户一块共享系统文件操作命令。

（三）文件操作命令

①执行 cat > mytext.txt 通过键盘输入一些信息，用 ctrl+c 结束，建立文件 mytext.txt。②执行 cat mytext.txt 显示文件内容。

③执行 ln mytext.txt mytext2.dat cat mytext2.dat

④执行 ls -l mytext?.*

```
[root@master master]# cat>mytext.txt
my text
^C
[root@master master]# cat mytext.txt
my text
[root@master master]# ln mytext.txt mytext2.dat
ln: 无法创建硬链接"mytext2.dat": 文件已存在
[root@master master]# ln mytext.txt mytext3.dat
[root@master master]# cat mytext3.dat
my text
[root@master master]# cat mytext2.dat
my text
[root@master master]# ls -l mytext?.*
-rwxr-x--x. 3 master master 8 9月  11 19:44 mytext2.dat
-rwxr-x--x. 3 master master 8 9月  11 19:44 mytext3.dat
[root@master master]# ls -1 mytext?.*
mytext2.dat
mytext3.dat
[root@master master]#
```

cat>命令新建一个文件夹，键盘输入其内容，然后用 cat 命令显示文件内容，ln 命令建立链接，且链接文件的内容与原来的文件的内容是一致的。

思考：文件链接是什么意思？有什么作用？

答 ：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，实际上是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的问题。如果链接指向目录，用户就可以利用该链接直接进入被链接的目录而不用打一大堆的路径名。而且，即使我们删除这个链接，也不会破坏原来的目录。

（四）目录操作

①执行 ls -l 看看当前目录的内容

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

```
[root@master master]# ls -l
总用量 5192
-rw-r--r--.  1 root   root       745 7月   1 09:45 ]
-rw-r--r--.  1 root   root      2164 7月   1 16:27 4-1.c
-rw-r--r--.  1 root   root      2778 7月   1 16:32 4.c
drwxr-xr-x.  5 master master     138 3月   4 2021 Desktop
-rw-r--r--.  1 root   root        43 7月   2 08:48 mima.txt
-rw-rw-r--.  1 master master       5 9月  11 19:15 mytext
-rwxr-x--x.  3 master master       8 9月  11 19:44 mytext2.dat
-rwxr-x--x.  3 master master       8 9月  11 19:44 mytext3.dat
-rwxr-x--x.  3 master master       8 9月  11 19:44 mytext.txt
drwxr-xr-x.  2 master master      60 1月  19 2021 OpenMP
drwxr-xr-x. 20 root   root      4096 7月   1 15:14 openssl-1.0.2f
-rw-r--r--.  1 root   root   5258384 1月  28 2016 openssl-1.0.2f.tar.gz
-rwxr-xr-x.  1 root   root      8760 7月   2 08:49 passwd
-rw-r--r--.  1 root   root      1236 7月   1 13:46 passwd.c
drwxr-xr-x.  2 master master      53 1月  14 2021 Pictures
drwxr-xr-x   2 master master       6 1月  19 2021 公共
```

```
[root@master master]# cd /lib
[root@master lib]# ls -l|more
总用量 52
drwxr-xr-x.  3 root root    18 1月  13 2021 alsa
drwxr-xr-x.  2 root root     6 10月  2 2020 binfmt.d
lrwxrwxrwx.  1 root root    10 1月  13 2021 cpp -> ../bin/cpp
drwxr-xr-x.  3 root root    43 1月  13 2021 crda
drwxr-xr-x.  9 root root   109 1月  13 2021 cups
drwxr-xr-x.  3 root root    64 4月  11 2018 debug
drwxr-xr-x.  4 root root   236 1月  13 2021 dracut
drwxr-xr-x.  7 root root    81 1月  13 2021 firewalld
drwxr-xr-x. 91 root root 12288 1月  13 2021 firmware
drwxr-xr-x.  3 root root    19 1月  13 2021 fontconfig
dr-xr-xr-x.  2 root root     6 4月  11 2018 games
drwxr-xr-x.  3 root root    33 9月  30 2020 gcc
drwxr-xr-x.  3 root root    21 1月  13 2021 grub
drwxr-xr-x.  2 root root     6 11月 21 2015 java
drwxr-xr-x.  2 root root     6 11月 21 2015 java-1.5.0
drwxr-xr-x.  2 root root     6 11月 21 2015 java-1.6.0
drwxr-xr-x.  2 root root     6 11月 21 2015 java-1.7.0
drwxr-xr-x.  2 root root     6 11月 21 2015 java-1.8.0
drwxr-xr-x.  2 root root     6 11月 21 2015 java-ext
drwxr-xr-x.  4 root root  4096 1月  13 2021 jvm
```

```
[root@master lib]# cd /etc
[root@master etc]# ls -l|more
总用量 1416
drwxr-xr-x.  3 root root      101 1月  13 2021 abrt
-rw-r--r--.  1 root root       16 1月  13 2021 adjtime
-rw-r--r--.  1 root root     1529 4月   1 2020 aliases
-rw-r--r--.  1 root root    12288 1月  14 2021 aliases.db
drwxr-xr-x.  3 root root       65 1月  13 2021 alsa
drwxr-xr-x.  2 root root     4096 1月  13 2021 alternatives
-rw-------.  1 root root      541 8月   9 2019 anacrontab
-rw-r--r--.  1 root root       55 8月   8 2019 asound.conf
-rw-r--r--.  1 root root        1 10月 31 2018 at.deny
drwxr-x---.  3 root root       43 1月  13 2021 audisp
drwxr-x---.  3 root root       83 1月  14 2021 audit
-rw-r--r--.  1 root root    15137 9月  30 2020 autofs.conf
-rw-------.  1 root root      232 9月  30 2020 autofs_ldap_auth.conf
-rw-r--r--.  1 root root      795 9月  30 2020 auto.master
drwxr-xr-x.  2 root root        6 9月  30 2020 auto.master.d
-rw-r--r--.  1 root root      524 9月  30 2020 auto.misc
-rwxr-xr-x.  1 root root     1260 9月  30 2020 auto.net
-rwxr-xr-x.  1 root root      687 9月  30 2020 auto.smb
drwxr-xr-x.  4 root root       71 1月  13 2021 avahi
drwxr-xr-x.  2 root root     4096 1月  13 2021 bash_completion.d
```

```
[root@master etc]# cd /bin
[root@master bin]# ls -l|more
总用量 181808
-rwxr-xr-x.   1 root root      41488 8月   20 2019 [
-rwxr-xr-x.   1 root root     107848 10月   2 2020 a2p
-rwxr-xr-x.   1 root root      11248 10月   2 2020 abrt-action-analyze-backtrace
-rwxr-xr-x.   1 root root      11240 10月   2 2020 abrt-action-analyze-c
-rwxr-xr-x.   1 root root       1345 10月   2 2020 abrt-action-analyze-ccpp-local
-rwxr-xr-x.   1 root root       6821 10月   2 2020 abrt-action-analyze-core
-rwxr-xr-x.   1 root root      11224 10月   2 2020 abrt-action-analyze-oops
-rwxr-xr-x.   1 root root      11232 10月   2 2020 abrt-action-analyze-python
-rwxr-xr-x.   1 root root       2814 10月   2 2020 abrt-action-analyze-vmcore
-rwxr-xr-x.   1 root root       1348 10月   2 2020 abrt-action-analyze-vulnerability
-rwxr-xr-x.   1 root root      11264 10月   2 2020 abrt-action-analyze-xorg
-rwxr-xr-x.   1 root root       5002 10月   2 2020 abrt-action-check-oops-for-hw-error
-rwxr-xr-x.   1 root root      11256 10月   2 2020 abrt-action-generate-backtrace
-rwxr-xr-x.   1 root root      11240 10月   2 2020 abrt-action-generate-core-backtrace
-rwxr-xr-x.   1 root root       8341 10月   2 2020 abrt-action-install-debuginfo
-rwxr-xr-x.   1 root root       3207 10月   2 2020 abrt-action-list-dsos
-rwxr-xr-x.   1 root root       8958 10月   2 2020 abrt-action-notify
-rwxr-xr-x.   1 root root       3535 10月   2 2020 abrt-action-perform-ccpp-analysis
-rwxr-xr-x.   1 root root       1292 10月   2 2020 abrt-action-save-kernel-data
-rwxr-xr-x.   1 root root      23712 10月   2 2020 abrt-action-save-package-data
```

```
-rwxr-xr-x.   1 root root      19464 10月   2 2020 abrt-dump-oops
[root@master bin]# cd /home
[root@master home]# ls -l|more
总用量 4
drwx------. 19 master master 4096 9月   11 19:45 master
[root@master home]#
```

思考：Linux 文件类型有哪几种？文件的存取控制模式如何描述？

答：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)。

（五）修改文件属性

① 执行 chmod 751 mytext.txt 、ls -l mytext.txt

② 执行 chown stud090 mytext.txt

```
[master@master ~]$ chmod 751 mytext.txt
[master@master ~]$ ls -l mytext.txt
-rwxr-x--x. 3 master master 8 9月   11 19:44 mytext.txt
[master@master ~]$ chown stud090 mytext.txt
chown: 无效的用户: "stud090"
[master@master ~]$
```

通过 chomd 751 mytext.txt 命令设置 mytext.txt 文件的访问模式为，文件主可读、可写、可执行，同组人可读、 可执行，其他人只可执行；通过 chown 更改文件所有者为 090

思考题：执行了上述操作后，若想再修改该文件，看能不能执行。为什么？

答：不能再修改文件，因为文件的所属用户不再是当前登录的用户。

（六）进程管理

① 执行 ps -ef

② 执行 wait 和 sleep 命令

```
chown: 无效的用户: 'stu091'
[master@master ~]$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 18:25 ?        00:00:11 /usr/lib/systemd/systemd --switched-root --system --
root           2       0  0 18:25 ?        00:00:00 [kthreadd]
root           4       2  0 18:25 ?        00:00:00 [kworker/0:0H]
root           6       2  0 18:25 ?        00:00:03 [ksoftirqd/0]
root           7       2  0 18:25 ?        00:00:00 [migration/0]
root           8       2  0 18:25 ?        00:00:00 [rcu_bh]
root           9       2  0 18:25 ?        00:00:05 [rcu_sched]
root          10       2  0 18:25 ?        00:00:00 [lru-add-drain]
root          11       2  0 18:25 ?        00:00:00 [watchdog/0]
root          13       2  0 18:25 ?        00:00:00 [kdevtmpfs]
root          14       2  0 18:25 ?        00:00:00 [netns]
root          15       2  0 18:25 ?        00:00:00 [khungtaskd]
root          16       2  0 18:25 ?        00:00:00 [writeback]
root          17       2  0 18:25 ?        00:00:00 [kintegrityd]
root          18       2  0 18:25 ?        00:00:00 [bioset]
root          19       2  0 18:25 ?        00:00:00 [bioset]
root          20       2  0 18:25 ?        00:00:00 [bioset]
root          21       2  0 18:25 ?        00:00:00 [kblockd]
root          22       2  0 18:25 ?        00:00:00 [md]
root          23       2  0 18:25 ?        00:00:00 [edac-poller]
root          24       2  0 18:25 ?        00:00:00 [watchdogd]
```

```
[master@master ~]$ wait 12
bash: wait: 进程号 12 不是当前 shell 的子进程
[master@master ~]$ sleep 60
```

wait 12 命令表示 Shell 等待进程号为 12 的后台进程终止，并报告终止报告，若缺省 12，则等待所有后台进程终止；sleep 60 表示 60s 后执行命令。

思考题：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：Linux 系统为了管理进程，用 task_struct 数据结构表示每个进程，任务向量是一个指针数组，里面的指针指向系统中的每一个 task_struct 数据结构。进程的家族关系是进程家族树体现的，可以通过这种继承体系从系统的任何一个进程发现胡查找到任意指定的其他进程。但大多数时候，只需要通过简单的重复方式就可以遍历系统中的所有进程。

【实验心得】

本次实验我学会了部署虚拟环境、安装 Linux 系统，了解了 Linux 中常见操作命令的使用方法，通过在虚拟机上实操，让我对 Linux 系统有了更深入的认识，但对于部分命令的具体使用方法和参数含义并没有理解透彻，必须在实验手册的指导下才能完成，希望自己可以多利用课下的时间，加强练习，熟练使用 Linux 中各命令。

【讨论题】

1. Liunx 系统命令很多，在手头资料不全时，如何查看命令格式？

答：ls --help 或者 man + 命令，实在不行，可以通过网络查询所需知识。

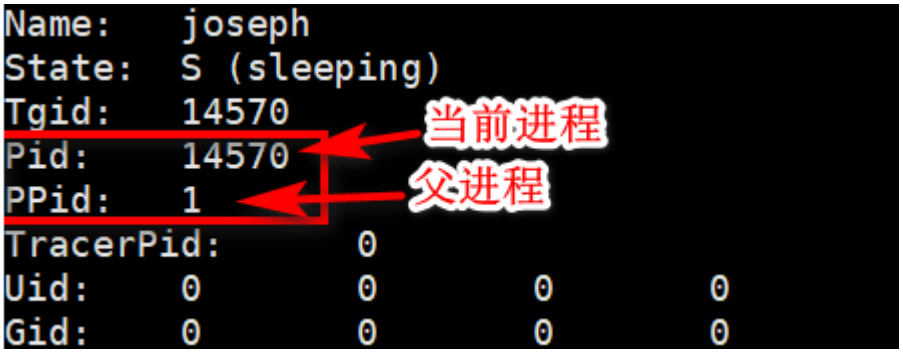
2. Liunx 系统用什么方式管理多个用户操作？如何管理用户文件，隔离用户空间？用命令及结果举例说明。

答：通过用户和用户组方式；chmod 修改文件权限，chgrp 修改文件所属组，chrow 修改文件的所有者和所属组。结果展示如上述操作五；

3. 用什么方式查看你的进程的管理参数？这些参数怎么体现父子关系？当结束一个父进

程后其子程序如何处理？用命令及结果举例说明。

答：Linux 查看进程 ps 命令，ps -H 显示树状结构，表示程序间的相互关系；pid 表示当前进程，ppid 表示父进程；若父进程比子进程先终止，则该父进程的所有子进程的父进程都变为 init 进程。其执行顺序大致如下：在一个进程终止时，内核逐个检查所有活动进程，以判断它是否是正要终止的进程的子进程，如果是，则该进程的父进程 ID 就更改为 1（init 进程的 ID);



4. Liunx 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：Linux 中所有内容都是以文件的形式保存和管理，所以一切皆是文件；Linux 文件类型有 6 种：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]；每种文件都有自己的标识符；

5. Liunx 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。

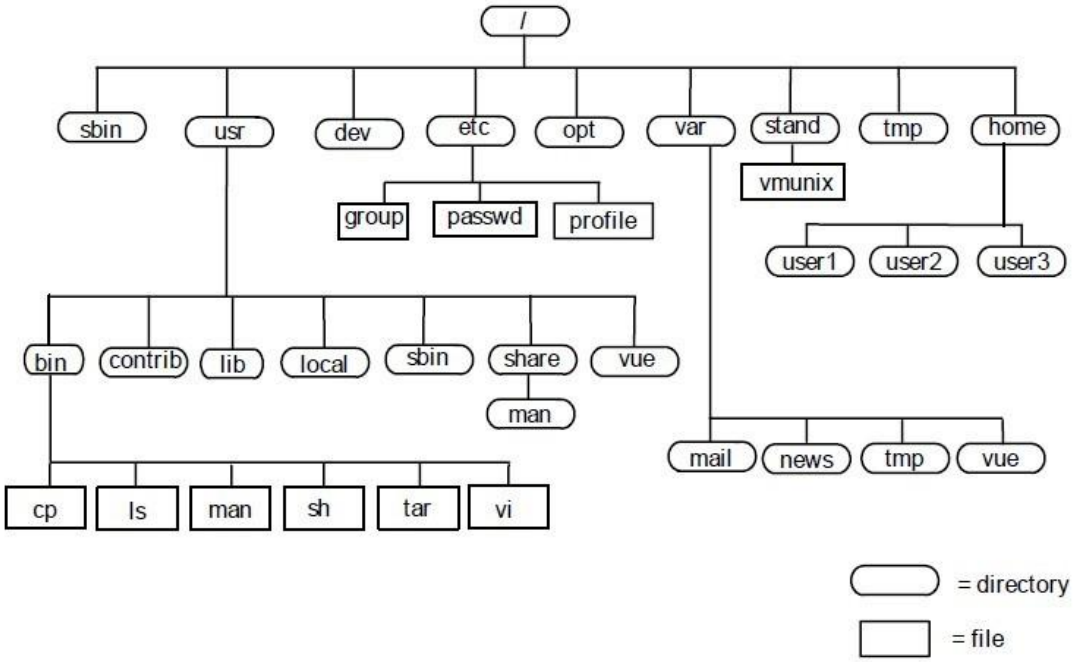
答：bin /usr/bin；

6. Liunx 系统得设备是如何管理的？在什么地方可以找到描述设备的信息？

答：Linux 系统采用设备文件统一管理硬件设备，从而将硬件设备的特性及管理细节对用户隐藏起来，实现用户程序与设备无关性。在 media 目录中；

7. 画出 Liunx 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？

答：



/：根目录

/bin：linux 的常用命令

/sbin：linux 的常用命令

/lib：库文件（so、elf）

/etc：系统配置文件和脚本文件

/sys：驱动相关的信息

/dev：设备节点目录

/proc：系统运行时，进程信息和内核信息存放在此

/root：root 用户目录

/home：普通用户目录，新创建的用户会在此目录下，包括个人配置文件和环境变量等

/usr：包含系统用户工具和程序

/usr/bin：非必须的普通用户可执行命令

/usr/sbin：非必须的可执行文件

/usr/share：共享文件目录

/usr/include：头文件目录

/usr/local：安装本地程序的默认路径

/usr/src：内核源码目录

/mnt：临时挂载文件目录

/tmp：系统临时文件，系统重启后消失

/opt：可选的应用程序包

/var：此目录经常变动，比如存放系统日志等

8. Liunx 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。

实际上 Shell 是一个命令解释器，它解释由用户输入的命令并且把它们送到内核。不仅如此，Shell 有自己的编程语言用于对命令的编辑，它允许用户编写由 shell 命令组成的程序。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。

bash

大多数 Linux 系统默认使用的 shell，bash shell 是 Bourne shell 的一个免费版本，它是最早的 Unix shell,bash 还有一个特点，可以通过 help 命令
来查看帮助。包含的功能几乎可以涵盖 shell 所具有的功能，所以一般的 shell 脚本都会指定它为执行路径。

csh

C shell 使用的是“类 C”语法，csh 是具有 C 语言风格的一种 shell，其内部命令有 52 个，较为庞大。目前使用的并不多，
已经被/bin/tcsh 所取代。

ksh

Korn shell 的语法与 Bourne shell 相同，同时具备了 C shell 的易用特点。许多安装脚本都使用 ksh,ksh 有 42 条内部命令，与 bash 相比有一定的限制性。

tcsh

tcsh 是 csh 的增强版，与 C shell 完全兼容。

sh

是一个快捷方式，已经被/bin/bash 所取代。

nologin

指用户不能登录

zsh

目前 Linux 里最庞大的一种 shell：zsh。它有 84 个内部命令，使用起来也比较复杂。一般情况下，不会使用
该 shell。

9.下面每一项说明的是哪类文件。

（1）-rwxrw-r--　　（2）/bin　　(3)ttyx3　　(4)brw-rw-rw-

(5) /etc/passwd　　(6) crw-rw-rw　　(7)/usr/lib　　(8) Liunx

答：（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