

实验八：设备管理观察

实验目的：

- 1. 掌握设备管理的基本命令
- 2. 可以获取和设置系统当前设备相关的主要信息

预备知识：

1. 基本命令

命令名	功能说明
mknod	建立块/字符特殊文件
dislocate	使进程和终端断开连接或重新连接
getty	设置终端工作方式
stty	改变/查询终端行设置
setterm	设置终端属性
tset	终端初始化
tput	初始化终端或查询 terminfo 数据库
resizecons	改变控制台尺寸的核心数据
unicode_start	使控制台在 Unicode 方式下工作
unicode_stop	使控制台不在 Unicode 方式下工作
kbd_mode	报告或设置键盘工作方式
kbdrate	重置键盘重复率和延迟时间
loadkeys	装入键盘转换表
dumpkeys	转储键盘转换表
setmetamode	定义键盘元键处理
showkey	检查键盘送来的扫描码和键码
chvt	改变前台虚拟终端
fgconsole	显示虚拟活动终端数
deallocvt	释放空闲的虚拟终端数
openvt,open	在一个新的虚拟终端上启动一个程序
switchto	切换至新的虚拟终端
vlock	锁住虚拟终端
screen	VT100/ANSI 终端仿真的屏幕管理器
mev	报告鼠标事件

2. /proc 文件系统

文件(目录)名	内容说明
/proc/devices	主要的字符和块设备编号及分配给这些编号的驱动程序名字
/proc/ioports	各种设备驱动程序注册的 I/O 端口范围
/proc/dma	被驱动程序留作专用的 DMA 通道以及驱动程序赋予的名字
/proc/scsi	scsi 设备及其相关信息
/proc/pci	PCI 设备信息
/proc/rtc	硬件实时时钟的相关信息
/proc/misc	被内核函数 misc_register 注册的驱动程序

3. 几类典型设备

设备文件名	设备(说明)
-------	--------

/dev/null	用于不需存储的输出（虚拟字符设备）
/dev/zero	用于二进制“0”的无限提供（虚拟字符设备）
/dev/random	随机数池（虚拟字符设备）
/dev/urandom	伪随机数池（虚拟字符设备）
/dev/ttyS0	COM1
/dev/ttyS1	COM2
/dev/lp0	LPT1
/dev/lp1	LPT2
/dev/psaux	PS/2 端口
/dev/fd0~/dev/fd7	软驱
/dev/hda~/dev/hdh	IDE 设备
/dev/sda~/dev/sddx	SCSI 设备

实验内容:

1. 利用手册页，学习设备相关主要命令（以上所列全部）的用法，并列出的系统当前的信息。

(1) mknod-建立块/字符特殊文件

该命令的标准形式为:

```
mknod DEVNAME {b | c} MAJOR MINOR
```

① DEVNAME 是要创建的设备文件名，如果想将设备文件放在一个特定的文件夹下，就需要先用 `mkdir` 在 `dev` 目录下新建一个目录:

② `b` 和 `c` 分别表示块设备和字符设备:

- `b` 表示系统从块设备中读取数据的时候，直接从内存的 `buffer` 中读取数据，而不经磁盘;
- `c` 表示字符设备文件与设备传送数据的时候是以字符的形式传送，一次传送一个字符，比如打印机、终端都是以字符的形式传送数据;

③ MAJOR 和 MINOR 分别表示主设备号和次设备号:

为了管理设备，系统为每个设备分配一个编号，一个设备号由主设备号和次设备号组成。主设备号标示某一种类的设备，次设备号用来区分同一类型的设备。Linux 操作系统中为设备文件编号分配了 32 位无符号整数，其中前 12 位是主设备号，后 20 位为次设备号，所以在向系统申请设备文件时主设备号不要超过 4095，次设备号不要超过 $2^{20}-1$ 。

- 使用 `mknod` 命令创建主设备号为 128，次设备号为 512 的字符特殊文件 `mydev1`:

```
→ 操作系统实验八 sudo mknod mydev1 c 128 512
→ 操作系统实验八 ls -la mydev1
crwxrwxrwx 1 myself myself 128, 512 5月 27 15:05 mydev1
```

- 使用 `mknod` 命令创建主设备号为 128，次设备号为 512 的块特殊文件 `mydev2`:

```

→ 操作系统实验八 sudo mknod mydev2 b 128 512
→ 操作系统实验八 ls -la mydev2
brwxrwxrwx 1 myself myself 128, 512 5月 27 15:09 mydev2

```

(2) dislocate-使进程和终端断开连接或重新连接

dislocate 是一个简洁的工具，可以让人们通过伪终端把程序分离出来，然后，当需要这些程序的时候，在重新挂载它们。这个工具是专门为那些缓慢或不稳定的终端会话（它们很容易中断）而提供的。

```

→ 操作系统实验八 dislocate
no connectable processes
→ 操作系统实验八 ls
mydev1 mydev2 操作系统实验八.docx 实验八-杨添宝.docx
→ 操作系统实验八 dislocate ls
Escape sequence is ^]
mydev1 mydev2 操作系统实验八.docx 实验八-杨添宝.docx
→ 操作系统实验八 dislocate ls -l
Escape sequence is ^]
总用量 61
crwxrwxrwx 1 myself myself 128, 512 5月 27 15:05 mydev1
brwxrwxrwx 1 myself myself 128, 512 5月 27 15:09 mydev2
-rwxrwxrwx 1 myself myself 15833 5月 27 14:40 操作系统实验八.docx
-rwxrwxrwx 1 myself myself 44173 5月 27 15:12 实验八-杨添宝.docx

```

(3) getty-设置终端工作方式

是 Unix 类操作系统启动时必须的三个步骤之一，用来开启终端，进行终端的初始化，设置终端。

用法：

getty [-h][-d<组态配置文件>][-r<延迟秒数>][-t<超时秒数>][-w<等待字符串>][终端机编号][连线速率<终端机类型><管制线路>]

或

getty [-c<定义配置文件>]

getty 命令设置和管理终端线路和端口。getty 命令由 init 命令来运行。getty 命令与终端状态管理员程序相链接。终端状态管理员程序提供了终端控制和登录的复合功能。 注意：getty 命令不在命令行输入。

(4) stty-改变/查询终端行设置

stty 命令修改终端命令行的相关设置。

参数：

-a: 以容易阅读的方式打印当前的所有配置；

-g: 以 stty 可读方式打印当前的所有配置。

tput lines 报告终端行数

tput sc 保存当前光标位置

tput sgr0 恢复默认值

```
chy@chy-VirtualBox:~$ tput setb 3
chy@chy-VirtualBox:~$ tput setf 0
chy@chy-VirtualBox:~$ tput setb 2
chy@chy-VirtualBox:~$ tput setf 5
chy@chy-VirtualBox:~$
```

(8) resizecons-改变控制台尺寸的核心数据

调用方法为:

resizecons COLSxROWS

或

resizecons COLS ROWS

或

resizecons -lines ROWS

其中 ROWS 为 25, 28, 30, 34, 36, 40, 44, 50, 60 中的一个

```
→ ~ resizecons 25 25
resizecons: resizecons: 无法找到 videomode 文件 25x25
→ ~ resizecons 25x25
resizecons: resizecons: 无法找到 videomode 文件 25x25
→ ~ resizecons -lines 40
resizecons: 无法获取 I/O 权限。
→ ~ sudo resizecons -lines 40
旧的模式: 240x40 新的模式: 240x40
旧的扫描行个数为: 480 新的扫描行个数为: 480 字符高度为: 12
resizecons: 不要忘记修改 TERM (也许修改为 con240x40 或 linux-240x40)
```

(9) unicode_start-使控制台在 Unicode 方式下工作

unicode_stop-使控制台不在 Unicode 方式下工作

```
→ ~ unicode_start
unicode_start skipped on /dev/pts/0
→ ~ unicode_stop
unicode_stop skipped on /dev/pts/0
```

(10) kbd_mode-报告或设置键盘工作方式

如果没有参数, kbd_mode 会显示当前键盘的模式, 如果有参数, 它会把键盘设置成相应的模式。

参数:

-s: 键盘扫描码模式 (原始)

-k: 键值 (keycode) 模式 (半原始)

-a : ASCII 模式(XLATE)

-u : UTF-8 模式(UNICODE)

XLATE 模式是传统模式, 所用的代码可以是任何 8-bit(8 位)的字符集。一般这个字符集同后面用到的字符集是匹配的, 在它们被传给屏幕后, 它们会根据 `consolechars -m` 选择的字符集在内部转换为 Unicode。

在 UNICODE 模式, 键盘会产生 16 位的字符, 这些字符会以 UTF-8 编码形式传给内核。UTF-8 在这后两种模式中要用到 `loadkeys` 定义的键盘映射表。

注意: 如果不是把键盘模式改为 ASCII 或者 Unicode, 很可能会使键盘不可用。

这个命令也可以在有些程序使你的键盘处于错误状态时用来把键盘改回 XLATE 或者 UNICODE 模式 (比如通过远程登录)。在有些过时的版本的程序中 -u 和 -s 是一样的。

```
→ ~ kbd_mode  
键盘处于某种未知模式
```

(11) `kbdrate`-重置键盘重复率和延迟时间

按键延时是只长按一个按键多少时间才会开始重复这个按键。开始重复过程后, 字符会以一定频率出现(cps), 也就是重复频率。终端中, 这些值可以通过 `kbdrate` 设置。

用法:

`kbdrate [-d delay] [-r rate]`

延迟 200 ms 重复频率是 30 cps:

```
→ ~ sudo kbdrate -d 200 -r 30  
字节输入速率设置为 30.0 cps (延迟 = 250 ms)
```

不加任何参数会还原到默认值 250 ms 和 10.9 cps:

```
→ ~ sudo kbdrate  
字节输入速率设置为 10.9 cps (延迟 = 250 ms)
```

(12) `loadkeys`-装入键盘转换表

`loadkeys` 命令可以根据一个键盘定义表改变 Linux 键盘驱动程序转译键盘输入过程。

参数:

-v --verbose: 印出详细的资料, 你可以重复以增加详细度。

-q --quiet: 不要显示任何讯息。

-c --clearcompose: 清除所有 `compose` 定义。

-s --clearstrings: 将定串定义表清除。

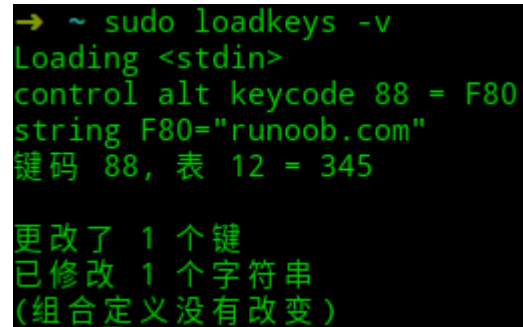
下面演示了定义按键组合, 在执行完 `loadkeys` 命令后输入如下两行语句:

`control alt keycode 88 = F80` //现确定键代码

`string F80="runoob.com"` //给变变量设定值

然后按下 `Ctrl+D` 键，确定输入。

执行后的效果是按下 `Ctrl+Alt+F12` 输出 `runoob.com`



```
→ ~ sudo loadkeys -v
Loading <stdin>
control alt keycode 88 = F80
string F80="runoob.com"
键码 88, 表 12 = 345

更改了 1 个键
已修改 1 个字符串
(组合定义没有改变)
```

(13) dumpkeys-转储键盘转换表

`dumpkeys` 命令用于显示键盘映射表，输出的内容可以被 `loadkeys` 命令识别，改变映射关系。

参数：

`-i` 驱动信息(键码范围、数量、状态键)

`-l` 详细驱动信息

`-n` 十六进制显示

`-f` 显示全部信息

`-l` 分行显示按键组合

`-S` 设定输出格式(0：预设 1：完整 2：分行 3 简单)

`--funcs-only` 功能键信息

`--keys-only` 键组合信息

`--compose-only` 普通键信息

执行的 `dumpkeys | less` 后的部分结果如下：

```
sudo dumpkeys | less
keymaps 0-2,4-5,8,12
keycode 1 = Escape          Escape
      alt keycode 1 = Meta_Escape
keycode 2 = one             exclam
      alt keycode 2 = Meta_one
keycode 3 = two             at
      control keycode 3 = nul
      shift control keycode 3 = nul
      alt keycode 3 = Meta_two
keycode 4 = three          numbersign
      control keycode 4 = Escape
      alt keycode 4 = Meta_three
keycode 5 = four           dollar
      control keycode 5 = Control_backslash
      alt keycode 5 = Meta_four
keycode 6 = five           percent
      control keycode 6 = Control_bracketright
      alt keycode 6 = Meta_five
keycode 7 = six            asciicircum
      control keycode 7 = Control_asciicircum
      alt keycode 7 = Meta_six
keycode 8 = seven          ampersand braceleft
      control keycode 8 = Control_underscore
      alt keycode 8 = Meta_seven
keycode 9 = eight          asterisk bracketleft
      control keycode 9 = Delete
      alt keycode 9 = Meta_eight
keycode 10 = nine          parenleft bracketright
      alt keycode 10 = Meta_nine
keycode 11 = zero          parenright braceright
      alt keycode 11 = Meta_zero
keycode 12 = minus         underscore backslash
      control keycode 12 = Control_underscore
      shift control keycode 12 = Control_underscore
      alt keycode 12 = Meta_minus
keycode 13 = equal         plus
:~
```

显示驱动信息:

```
→ ~ dumpkeys -i
内核支持的键码范围: 1 - 255
可绑定到一个键上的动作的最大个数: 256
实际使用的键映射个数为: 7
其中 0 个为动态分配
内核支持的行为码范围:
内核支持的功能键的个数: 256
组合定义的最大个数: 256
实际使用的组合定义的个数: 68
```

(14) setmetamode-定义键盘元键处理

用法:

setmetamode [meta|bit|metabit | esc|prefix|escprefix]

没有参数时, setmetamode 将打印当前 Meta 键模式; 有参数时, 设置所指出的 Meta 键模式。

在图形界面下执行的结果:


```
→ ~ setmetamode
setmetamode: 读取当前设置出错。也许标准输入不是一个 VT? : ioctl KDGBMETA: 对设备不适当的 ioctl 操作
```

在字符界面下执行的结果：

```
chy@chy-VirtualBox:~$ setmetamode
Meta key gives Esc prefix
```

(15) showkey-检查键盘送来的扫描码和键码

```
chy@chy-VirtualBox:~$ showkey
kb mode was UNICODE
[ if you are trying this under X, it might not work
since the X server is also reading /dev/console ]

press any key (program terminates 10s after last keypress)...
keycode 28 release
keycode 30 press
keycode 30 release
keycode 48 press
keycode 48 release
keycode 46 press
keycode 46 release
keycode 32 press
keycode 32 release
keycode 18 press
keycode 18 release
keycode 33 press
keycode 33 release
keycode 29 press
keycode 56 press
keycode 30 press
```

(16) chvt-改变前台虚拟终端

chvt N 命令用来生成/dev/ttyN 的前台终端。如果它本来不存在，即创建相应的屏幕。为了删除掉不用的 VT(虚拟终端)，可使用 deallocvt。

```
chy@chy-VirtualBox:~$ sudo chvt 1
```

```
Ubuntu 12.04.5 LTS chy-VirtualBox tty1
chy-VirtualBox login: _
```

(17) fgconsole-显示虚拟活动终端数

```
→ ~ fgconsole
1
```

(18) deallocvt-释放空闲的虚拟终端数

用法:

deallocvt [N1 N2 ...]

如果不指定参数, deallocvt 程序会释放所有未使用的虚拟终端的核心内存和数据结构。如果给定了参数 Ni 那么就只释放 TTY /dev/ttyNi.

```
→ ~ deallocvt 1
deallocvt: VT 1 是控制台并且无法被回收
→ ~ deallocvt 2
deallocvt: 无法回收控制台 23: ioctl1 VT_DISALLOCATE: 设备或资源忙
```

(19) openvt, open-在一个新的虚拟终端上启动一个程序

在 Debian 系统内, open 命令是 openvt 命令的符号链接, 用于在一个新的虚拟终端中运行应用程序。

用法:

openvt [-c vtnumber] [-s] [-u] [-l] [-v] [--] command command_options

`openvt` 将会找到第一个可用的虚拟终端（VT），在上面以给定的参数 `command_options` 运行给定的命令 `command`，标准输入、标准输出和标准错误设备将被引导到那个终端，当前搜索路径（`$PATH`）被用来寻找请求的命令。如果没有找到命令将会使用环境变量 `$SHELL`。

```
→ ~ openvt bash
→ ~ openvt -l bash
→ ~ openvt -- ls -l
→ ~ openvt -s -- ls -l
→ ~
```

```
→ ~ openvt xeyes
→ ~
```



(20) `switchto`-切换至新的虚拟终端

`switchto` 将切换当前终端到指定终端。当系统热键不可用或在脚本中使用该命令十分有用。

(21) `vlock`-锁住虚拟终端

用法：

`vlock [-achv]`

参数：

`-a` 或 `--all` 锁住所有的终端阶段作业，如果您在全屏幕的终端中使用本参数，则会禁用键盘

切换终端机的功能一并关闭。

`-c` 或 `--current` 锁住目前的终端阶段作业，此为预设值。

`-h` 或 `--help` 在线帮助。

`-v` 或 `--version` 显示版本信息。

```
→ ~ vlock
```

```
This TTY is now locked.
Please press [ENTER] to unlock.
myself's Password:
→ ~
```

(22) `screen`-VT100/ANSI 终端仿真的屏幕管理器

GNU Screen 是一款由 GNU 计划开发的用于命令行终端切换的自由软件。用户可以通过该软件同时连接多个本地或远程的命令行会话，并在其间自由切换。

GNU Screen 可以看作是窗口管理器的命令行界面版本。它提供了统一的管理多个会话的界面和相应的功能。

- 会话恢复

只要 Screen 本身没有终止，在其内部运行的会话都可以恢复。这一点对于远程登录的用户特别有用——即使网络连接中断，用户也不会失去对已经打开的命令行会话的控制。只要再次登录到主机上执行 `screen -r` 就可以恢复会话的运行。同样在暂时离开的时候，也可以执行分离命令 `detach`，在保证里面的程序正常运行的情况下让 Screen 挂起（切换到后台）。这一点和图形界面下的 VNC 很相似。

- 多窗口

在 Screen 环境下，所有的会话都独立的运行，并拥有各自的编号、输入、输出和窗口缓存。用户可以通过快捷键在不同的窗口下切换，并可以自由的重定向各个窗口的输入和输出。Screen 实现了基本的文本操作，如复制粘贴等；还提供了类似滚动条的功能，可以查看窗口状况的历史记录。窗口还可以被分区和命名，还可以监视后台窗口的活动。

- 会话共享

Screen 可以让一个或多个用户从不同终端多次登录一个会话，并共享会话的所有特性（比如可以看到完全相同的输出）。它同时提供了窗口访问权限的机制，可以对窗口进行密码保护。用法：

```
screen [-AmRvx -ls -wipe][[-d <作业名称>][[-h <行数>]][-r <作业名称>][[-s ][-S <作业名称>]
```

参数：

-A 将所有的视窗都调整为目前终端机的大小。

-d <作业名称> 将指定的 screen 作业离线。

-h <行数> 指定视窗的缓冲区行数。

-m 即使目前已在作业中的 screen 作业，仍强制建立新的 screen 作业。

-r <作业名称> 恢复离线的 screen 作业。

-R 先试图恢复离线的作业。若找不到离线的作业，即建立新的 screen 作业。

-s 指定建立新视窗时，所要执行的 shell。

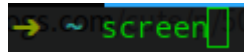
-S <作业名称> 指定 screen 作业的名称。

-v 显示版本信息。

-x 恢复之前离线的 screen 作业。

-ls 或--list 显示目前所有的 screen 作业。

-wipe 检查目前所有的 screen 作业，并删除已经无法使用的 screen 作业。



```
GNU Screen version 4.06.02 (GNU) 23-Oct-17

Copyright (c) 2015-2017 Juergen Weigert, Alexander Naumov, Amadeusz Slawinski
Copyright (c) 2010-2014 Juergen Weigert, Sadrul Habib Chowdhury
Copyright (c) 2008-2009 Juergen Weigert, Michael Schroeder, Micah Cowan, Sadrul Habib Chowdhury
Copyright (c) 1993-2007 Juergen Weigert, Michael Schroeder
Copyright (c) 1987 Oliver Laumann

Please press [ENTER] to unlock.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public
License as published by the Free Software Foundation; either version 3, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for
more details.

You should have received a copy of the GNU General Public License along with this program (see the file COPYING);
if not, see http://www.gnu.org/licenses/, or contact Free Software Foundation, Inc., 51 Franklin Street, Fifth
Floor, Boston, MA 02111-1301, USA.
Copyright (c) 1987 Oliver Laumann
Send bugreports, fixes, enhancements, t-shirts, money, beer & pizza to screen-devel@gnu.org

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public
License as published by the Free Software Foundation; either version 3, or (at your option) any later version.

Capabilities: the GNU General Public License as published by the Free Software Foundation; either version 3, or
(at your option) any later version.
+copy +remote-detach +power-detach +multi-attach +multi-user +font +color-256 +utf8 +rxvt +builtin-telnet

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program (see the file COPYING); if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02111-1301, USA.

[Press Space or Return to end.]
[?] bugreports, fixes, enhancements, t-shirts, money, beer & pizza to screen-devel@gnu.org
```

(23) mev-报告鼠标事件

使用 Ubuntu 12.04，输入 mev 命令后点击鼠标，将出现类似如下的结果：

```
chy@chy-VirtualBox:~$ mev
chy@chy-VirtualBox:~$ P*#P* G1#G1 Q3#Q3 Q7#Q7 Z4#Z4 H*#H* <2#<2
```

2. 利用/proc 文件系统，列出你的系统当前的信息，并解释相关内容。

(1) /proc/devices-主要的字符和块设备编号及分配给这些编号的驱动程序名字

```
→ /proc cat devices
Character devices:
```

```
1 mem
4 /dev/vc/0
4 tty
4 ttyS
5 /dev/tty
5 /dev/console
5 /dev/ptmx
5 ttyprintk
6 lp
7 vcs
10 misc
13 input
21 sg
29 fb
81 video4linux
89 i2c
99 ppdev
108 ppp
116 alsa
128 ptm
136 pts
180 usb
189 usb_device
195 nvidia-frontend
204 ttyMAX
216 rfcomm
226 drm
239 media
240 nvidia-nvlink
241 mei
242 ipmidev
243 aux
244 hidraw
245 ttyDBC
246 bsg
247 hmm_device
248 watchdog
249 rtc
250 dax
251 dimmctl
252 ndctl
253 tpm
254 gpiochip
```

```
Block devices:
```

```
7 loop
8 sd
9 md
11 sr
65 sd
66 sd
67 sd
68 sd
69 sd
70 sd
71 sd
128 sd
129 sd
130 sd
131 sd
132 sd
133 sd
134 sd
135 sd
253 device-mapper
254 mdp
259 blkext
```

(2) /proc/ioports-各种设备驱动程序注册的 I/O 端口范围

```

→ /proc cat ioports
0000-0000 : PCI Bus 0000:00
0000-0000 : dma1
0000-0000 : pic1
0000-0000 : timer0
0000-0000 : timer1
0000-0000 : keyboard
0000-0000 : keyboard
0000-0000 : rtc0
0000-0000 : dma page reg
0000-0000 : pic2
0000-0000 : dma2
0000-0000 : fpu
0000-0000 : PNPOC04:00
0000-0000 : iTCO_wdt
0000-0000 : pnp 00:00
0000-0000 : pnp 00:01
0000-0000 : PNPOC09:00
0000-0000 : EC data
0000-0000 : PNPOC09:00
0000-0000 : EC cmd
0000-0000 : PCI conf1
0000-0000 : PCI Bus 0000:00
0000-0000 : pnp 00:00
0000-0000 : pnp 00:00
0000-0000 : ACPI PM1a_EVT_BLK
0000-0000 : ACPI PM1a_CNT_BLK
0000-0000 : ACPI PM_TMR
0000-0000 : iTCO_wdt
0000-0000 : ACPI PM2_CNT_BLK
0000-0000 : pnp 00:03
0000-0000 : ACPI GPE0_BLK
0000-0000 : PCI Bus 0000:02
0000-0000 : 0000:02:00.0
0000-0000 : r8169
0000-0000 : PCI Bus 0000:01
0000-0000 : 0000:01:00.0
0000-0000 : 0000:00:02.0
0000-0000 : 0000:00:1f.4
0000-0000 : i801_smbus
0000-0000 : 0000:00:17.0
0000-0000 : ahci
0000-0000 : 0000:00:17.0
0000-0000 : ahci
0000-0000 : 0000:00:17.0
0000-0000 : ahci
0000-0000 : pnp 00:08
0000-0000 : pnp 00:00
0000-0000 : pnp 00:00
0000-0000 : pnp 00:00

```



```

→ /proc cat misc
51 vboxnetctl
52 vboxdrv
53 vboxdrv
232 kvm
54 wmi/dell-smbios
55 acpi_thermal_rel
235 autofs
234 btrfs-control
56 memory_bandwidth
57 network_throughput
58 network_latency
59 cpu_dma_latency
184 microcode
227 mcelog
236 device-mapper
223 uinput
1 psaux
200 tun
237 loop-control
60 lightnvm
183 hw_random
228 hpet
229 fuse
61 ecryptfs
231 snapshot
62 rfkill
63 vga_arbiter

```

3. 利用/proc 文件系统，列出你的系统当前的信息，并解释相关内容观察/dev 目录中的文件，使用 ls -l 命令，解释各项信息的含义。

```

crw-rw-rw- 1 root root 1, 3 6月 2 23:36 null
crw-rw-rw- 1 root root 1, 5 6月 2 23:36 zero
crw-rw-rw- 1 root root 1, 8 6月 2 23:36 random
crw-rw-rw- 1 root root 1, 9 6月 2 23:36 urandom
crw-rw-rw- 1 root dialout 4, 64 6月 2 23:36 ttyS0
crw-rw-rw- 1 root dialout 4, 65 6月 2 23:36 ttyS1
crw----- 1 root root 10, 1 6月 2 23:36 psaux
lrwxrwxrwx 1 root root 13 6月 2 23:36 fd -> /proc/self/fd

→ /dev ll /proc/self/fd
总用量 0
lrwx----- 1 myself myself 64 6月 3 00:05 0 -> /dev/pts/0
lrwx----- 1 myself myself 64 6月 3 00:05 1 -> /dev/pts/0
lrwx----- 1 myself myself 64 6月 3 00:05 2 -> /dev/pts/0
lr-x----- 1 myself myself 64 6月 3 00:05 3 -> /proc/9617/fd

```

/dev/hd[a-t]: IDE 设备
/dev/sd[a-z]: SCSI 设备
/dev/fd[0-7]: 标准软驱
/dev/md[0-31]: 软 raid 设备
/dev/loop[0-7]: 本地回环设备
/dev/ram[0-15]: 内存
/dev/null: 无限数据接收设备,相当于黑洞
/dev/zero: 无限零资源
/dev/tty[0-63]: 虚拟终端
/dev/ttyS[0-3]: 串口
/dev/lp[0-3]: 并口
/dev/console: 控制台
/dev/fb[0-31]: framebuffer
/dev/cdrom => /dev/hdc
/dev/modem => /dev/ttyS[0-9]
/dev/pilot => /dev/ttyS[0-9]
/dev/random: 随机数设备
/dev/urandom: 随机数设备