Table of Contents

- 说明 1.1
- Linux 1.2
  - sed 1.2.1
  - grep 1.2.2
  - find 1.2.3
  - awk 1.2.4
  - ssh 1.2.5
  - dd 1.2.6
  - cat 1.2.7
  - tar 1.2.8
  - unzip 1.2.9
  - netstat 1.2.10
  - vim 1.2.11
  - pandoc 1.2.12
  - tidy 1.2.13
  - tesseract 1.2.14
  - rdesktop 1.2.15
  - git 1.2.16
  - rsync 1.2.17
  - xdotool 1.2.18
  - xrandr 1.2.19
  - iptables 1.2.20
  - docker 1.2.21
  - mplayer 1.2.22
  - 挂载镜像文件 1.2.23
  - Shell编程 1.2.24
  - 备份和恢复系统 1.2.25
  - GNOME 3 1.2.26
- Android 1.3
  - 截图命令 1.3.1

熟练使用Linux，就需要了解Linux下的各种命令行工具的使用方法和技巧。虽然已经有很多书对这些工具集做了介绍，但是通常都是大而全的介绍，而我需要的短而精的经典使用案例。

Linux

这里的记录主题是Linux相关的内容，我以前热衷于新潮的Arch Linux，现在偏爱Debian，所以一部分例子以Arch Linux为运行环境的，另一部分例子是以Debian为运行环境的，Linux发行版本的差别主要是包管理的方式，这些工具集的用法差别并不大。Arch Linux的Wiki确实做得比较好的，涵盖的面比较广，但通常针对的是x86的cpu的，像ARM、MIPS等其他架构的运行环境就要看Debian的Wiki，另外Gentoo的Wiki也是做得不错的。听说是Gentoo是编译安装的，这么多软件，全部都需要编译，需要消耗的时间太漫长了，多版本库的问题已经被Docker技术解决了，Gentoo就没有必要去体验了。

sed

在每行的头添加字符，比如"HEAD"，命令如下：

```
sed 's/^/HEAD&/g' -i test.file
```

在每行的行尾添加字符，比如“TAIL”，命令如下：

```
sed 's/$/&TAIL/g' -i test.file
```

在匹配的开始的第3行，插入指定内容，命令如下：

```
Section "InputClass"
        Identifier "libinput touchpad catchall"
        MatchIsTouchpad "on"
        MatchDevicePath "/dev/input/event*"
        Driver "libinput"
EndSection
```

```
sed -e '/Identifier "libinput touchpad catchall"/{n;n;n;/Driver/a\        Option "Tapping" "on"' -e '}' -i 40-libinp
ut.conf
```

参考文章

1. Sed命令n，N，d，D，p，P，h，H，g，G，x解析
2. linux：sed高级命令之n、N

grep

在grep搜索结果中包括或者排除指定文件：

```
# .表示当前目录。
#只在目录中所有的.php和.html文件中递归搜索字符"main()"
grep "main()" . -r --include *.{php,html}

#在搜索结果中排除所有README文件
grep "main()" . -r --exclude "README"

#在搜索结果中排除filelist文件列表里的文件
grep "main()" . -r --exclude-from filelist
```

当不存在sshd用户时，添加 `sshd` 用户信息到 `/etc/passwd` 中

```
grep -q sshd /etc/passwd && echo "sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/bin/false" >> /etc/passwd
```

find

编译之后需要把当前文件夹中可执行文件复制到指定目录

```
#-type f限定在当前目录，默认会递归搜索子目录，也可以限定为子目录中搜索-type d
find . -perm /+x -type f -exec cp -t /usr/local/redis/ {} +
```

awk

ssh

主要用途：

1. 管理远程主机
2. 加密SSH client 与 SSH server 传输的数据
3. 突破防火墙限制

常用到的命令行参数有：

```
    -C 表示信息压缩传输；
    -f 表示后台认证用户/密码（通常和-N连用），而不登录到远程主机（也就是后台运行）；
    -N 表示不执行shell，仅做端口转发，常常与-f/-g连用；
    -g 表示允许打开的端口让远程主机访问
    -L 表示将远程主机服务器指定端口映射到客户机某端口；
    -D 表示客户机 “动态’’转发；
    -R 表示将远程主机服务器端口转发到客户端计算机指定端口
```

远程操作

```
上传公钥
ssh user@host 'mkdir -p .ssh && cat >> .ssh/authorized_keys' < ~/.ssh/id_rsa.pub
复制本地src目录到远程主机
cd && tar czv src | ssh user@host 'tar xz'
复制远程src目录到本地主机
ssh user@host 'tar cz src' | tar xzv
查看远程主机是否有[h]ttpd进程
ssh user@host 'ps ax | grep [h]ttpd'
```

建立SSH隧道（端口转发）

```
＃可基于客户机，实现客户机到远程主机（不能被直接访问）特定服务的直接访问，local_host为空或*代表绑定所有接口，也可以指定本机某个IP，以限定可访问范围
ssh -C -f -N  -L local_host:localhost_port:remote_host:remote_host_port username@remote_host_domain
＃可基于远程主机服务器（提供公网访问服务），实现客户机的公网访问，local_host为空或*代表绑定所有接口，也可以指定本机某个IP，以限定可访问范围
ssh -C -f -N -R local_host:localhost_listen_port:remote_host:remote_host_port username@remote_host_domain
#可基于远程主机网络（可访问被屏蔽网络），实现所谓的翻墙 ，local_host为空或*代表绑定所有接口，也指定本机某个IP，以限定可访问范围
ssh -C -f -N -D local_host:localhost_listen_port username@remote_host_domain
```

常用的例子 AB 一个网段 CD一个网段 只有A能SSH连接到C B------A---ssh--->C------D

只要在A主机上执行：如下命令就能打通两个网段。

> 情况一 A想访问D （ 隧道 ） 本地转发(正向（本地）端口转发)
>
> ```
> ssh -L 8000:D:8888 c@c
> ```
>
> 情况二 C想访问B （ 内网穿透 ） 远程转发(反向（远程）端口转发)
>
> ```
> ssh -R 8888:B:22 c@c
> ```
>
> 情况三 A想无障碍访问墙外的内容，C位于GFW外
>
> ```
> ssh -D 1080 c@c
> ```

如何保持持久的SSH链接

用ssh链接服务端，一段时间不操作或屏幕没输出（比如复制文件）的时候，会自动断开 解决：（2种办法）

1、在客户端配置

vi /etc/ssh/ssh_config（注意不是/etc/ssh/sshd_config文件）,后面添加

```
Host *
ServerAliveInterval 30
ServerAliveCountMax 99
```

每隔30秒，向服务器发出一次心跳。若超过99次请求，都没有发送成功，则会主动断开与服务器端的连接。 2、在服务端

编辑服务器 /etc/ssh/sshd_config，最后增加

```
ClientAliveInterval 60
ClientAliveCountMax 5
```

ClientAliveInterval表示每隔多少秒，服务器端向客户端发送心跳，是的，你没看错。 下面的ClientAliveInterval表示上述多少次心跳无响应之后，会认为Client已经断开。 所以，总共允许无响应的时间是60*5=300秒。

参考文章

1. 老调重弹: SSH与隧道和端口转发
2. SSH原理与应用
3. 让终端走代理的几种方法
4. 解决SSH自动断线，无响应的问题。

Linux下显示dd命令的进度：

```
dd if=/dev/zero of=/tmp/zero.img bs=10M count=100000
```

想要查看上面的dd命令的执行进度，可以使用下面几种方法：

比如：每5秒输出dd的进度

方法一：

```
watch -n 5 pkill -USR1 ^dd$
```

方法二：

```
watch -n 5 killall -USR1 dd
```

方法三：

```
while killall -USR1 dd; do sleep 5; done
```

方法四：

```
while (ps auxww |grep " dd " |grep -v grep |awk '{print $2}' |while read pid; do kill -USR1 $pid; done) ; do sleep 5
; done
```

上述四种方法中使用三个命令：pkill、killall、kill向dd命令发送SIGUSR1信息，dd命令进程接收到信号之后就打印出自己当前的进度。

在某些场合，可能我们需要在脚本中生成一个临时文件，然后把该文件作为最终文件放入目录中。（可参考ntop.spec文件）这样有几个好处，其中之一就是临时文件不是唯一的，可以通过变量赋值，也可根据不同的判断生成不同的最终文件等等。

一、cat和EOF cat命令是linux下的一个文本输出命令，通常是用于观看某个文件的内容的； EOF是“end of file”，表示文本结束符。 结合这两个标识，即可避免使用多行echo命令的方式，并实现多行输出的结果。

二、使用 看例子是最快的熟悉方法：

```
cat << EOF > test.sh
#!/bin/bash
#you Shell script writes here.
EOF
```

结果：

```
# cat test.sh
#!/bin/bash
#you Shell script writes here.
```

可以看到，test.sh的内容就是cat生成的内容。

三、其他写法 1、追加文件

```
# cat << EOF >> test.sh
```

2、换一种写法

```
# cat > test.sh << EOF
```

3、EOF只是标识，不是固定的

```
cat << HHH > iii.txt
sdlkfjksl
sdkjflk
asdlfj
HHH
```

这里的“HHH”就代替了“EOF”的功能。结果是相同的。 引用

```
# cat iii.txt
sdlkfjksl
sdkjflk
asdlfj
```

4、非脚本中 如果不是在脚本中，我们可以用Ctrl-D输出EOF的标识

```
# cat > iii.txt
skldjfklj
sdkfjkl
kljkljklj
kljlk
Ctrl-D
```

结果： 引用 cat iii.txt skldjfklj sdkfjkl kljkljklj kljlk

※关于“>”、“>>”、“<”、“<<”等的意思，请自行查看bash的介绍 Shell脚本——cat/EOF输出多行

解压的时候重命名

1 Use -C and --strip-components (See man tar). -C, --directory=DIR Change to DIR before performing any operations. This option is order-sensitive, i.e. it affects all options that follow. --strip-components=1 Strip NUMBER leading components from file names on extraction.

Example:

mkdir FOLDER

for remote tar file

curl -L ’remote_tar_file' | tar -xz - -C FOLDER --strip-components=1

for local tar file

tar -xz FILENAME -C FOLDER --strip-components=1

2 Use --transform

You can also use the –transform option for a bit more flexibility. It accepts any sed replacement (s) operation.

For example, this is how I extract a Linux tarball to a new directory so I can apply a patch:

tar -xjf linux-2.6.38.tar.bz2 --transform 's/linux-2.6.38/linux-2.6.38.1/'

unzip

zip文件解压乱码如何解决？

以下内容来自知乎答案，主要内容整理如下： Arch Linux 下可以直接安装 AUR 里面的 unzip-natspec 。接下来只要像普通文件一样解压缩就可以了。关键是 libnatspec 能自动识别编码。unzip test.zip Gentoo 的话也可以在重新编译 unzip 添加 natspec 支持。 Debian 的话有人给unzip打了个补丁https://github.com/ikohara/dpkg-unzip-iconv 也可以直接使用7zip来解压， 安装完之后，就可以用7za和convmv两个命令完成解压缩任务。

```
sudo apt-get install p7zip convmv
LANG=C 7za x your-zip-file.zip
convmv -f GBK -t utf8 --notest -r .
```

第一条命令用于解压缩，而LANG=C表示以US-ASCII这样的编码输出文件名，如果没有这个语言设置，它同样会输出乱码，只不过是UTF8格式的乱码(convmv会忽略这样的乱码)。第二条命令是将GBK编码的文件名转化为UTF8编码，-r表示递归访问目录，即对当前目录中所有文件进行转换。

netstat

```
#root权限才能查看所有的进程
sudo netstat -tunpl | grep 22
```

1、替换（substitute）:[range]s/pattern/string/[c,e,g,i]5.1

range 指的是范围，1,7 指从第一行至第七行，1,$ 指从第一行至最后一行，也就是整篇文章，也可以 % 代表。还记得吗？ % 是目前编辑的文章，# 是前一次编辑的文章。

pattern 就是要被替换掉的字符串，可以用 regexp 来表示。

string 将 pattern 由 string 所取代。

c confirm，每次替换前会询问。

e 不显示 error。

g globe，不询问，整行替换。

i ignore 不分大小写。

g 大概都是要加的，否则只会替换每一行的第一个符合字符串。可以合起来用，如 cgi，表示不分大小写，整行替换，替换前要询问是否替换。

[实例] :%s/Edwin/Edward/g

这样整篇文章的 Edwin 就会替换成 Edward。 更进阶的搜寻、替换的例子在说明 regexp 的时候还会再详述。目前只知道最基本的用法就可以了！其实光这样就非常好用了。:-)

2、替换说明（示例 ）

VIM中常用的替换模式总结。

1，简单替换表达式

替换命令可以在全文中用一个单词替换另一个单词：

:%s/four/4/g

“%” 范围前缀表示在所有行中执行替换。最后的 “g” 标记表示替换行中的所有匹配点。如果仅仅对当前行进行操作，那么只要去掉%即可

如果你有一个象 “thirtyfour” 这样的单词，上面的命令会出错。这种情况下，这个单词会被替换成”thirty4″。要解决这个问题，用 “\<” 来指定匹配单词开头：

:%s/\

显然，这样在处理 “fourty” 的时候还是会出错。用 “>” 来解决这个问题：

:%s/\/4/g

如果你在编码，你可能只想替换注释中的 “four”，而保留代码中的。由于这很难指定，可以在替换命令中加一个 “c” 标记，这样，Vim 会在每次替换前提示你：

:%s/\/4/gc

2，删除多余的空格

要删除这些每行后面多余的空格，可以执行如下命令：

:%s/\s+$//

命令前面指明范围是 “%”，所以这会作用于整个文件。”substitute” 命令的匹配模式是

“\s+$”。这表示行末（$）前的一个或者多个（+）空格（\s）。替换命令的 “to” 部分是空的：”//”。这样就会删除那些匹配的空白字符。

3，匹配重复性模式

星号项 “*” 规定在它前面的项可以重复任意次。因此:

/a*

匹 配 “a”，”aa”，”aaa”，等等。但也匹配 “” (空字符串)，因为零次也包含在内。星号 “ ” 仅仅应用于那个紧邻在它前面的项。因此 “*ab*” 匹配 “a”，”ab”，”abb”,”abbb”，等等。如要多次重复整个字符串，那么该字符串必须被组成一个项。组成一项的方法就是在它前面加 “(”，后面加 “)”。因此这个命令:

/(ab)*

匹配: “ab”，”abab”，”ababab”，等等。而且也匹配 “”。

要避免匹配空字符串，使用 “+”。这表示前面一项可以被匹配一次或多次。

/ab+

匹配 “ab”，”abb”，”abbb”，等等。它不匹配 后面没有跟随 “b” 的 “a”。

要匹配一个可选项，用 “\=”。 例如:

/folders\=

匹配 “folder” 和 “folders”。

4，指定重复次数

要匹配某一项的特定次数重复，使用 “{n,m}” 这样的形式。其中 “n” 和 “m” 都是数字。在它前面的那个项将被重复 “n” 到 “m” 次 (|inclusive| 包含 “n” 和 “m”)。例如:

/ab{3,5}

匹配 “abbb”，”abbbb” 以及 “abbbbb”。

当 “n” 省略时，被默认为零。当 “m” 省略时，被默认为无限大。当 “,m” 省略时，就表示重复正好 “n” 次。例如:

模式 匹配次数

{,4} 0，1，2，3 或 4

{3,} 3，4，5，等等

{0,1} 0 或 1，同 \=

{0,} 0 或 更多，同 *

{1,} 1 或 更多，同 +

{3} 3

5，多选一匹配

在一个查找模式中，”或” 运算符是 “|”。例如:

/foo|bar

这个命令匹配了 “foo” 或 “bar”。更多的抉择可以连在后面:

/one|two|three

匹配 “one”，”two” 或 “three”。

如要匹配其多次重复，那么整个抉择结构须置于 “(” 和 “)” 之间:

/(foo|bar)+

这个命令匹配 “foo”，”foobar”，”foofoo”，”barfoobar”，等等。

再举个例子:

/end(if|while|for)

这个命令匹配 “endif”，”endwhile” 和 “endfor”。

用word随手记录的笔记，很方便就可以转换成markdown格式

pandoc -s from.docx -o to.md --extract-media=. -t markdown_github

参考文章

Pandoc用户指南

tidy

清理word文档导出的html中无用的标签。

```
#-i indent output file
tidy -config tidy-config.txt -o clean.html -i dirty.html
# 一些常见的多余的属性
#\s+class=[^ >]*
#\s+align=[^ >]*
#\s+width=[^ >]*
#\s+valign=[^ >]*
#\s+style='+[^']*'
#\s+style="+[^"]*"
#</?span+\s+[^>]*>
#\s+border=[^ >]*
#\s+cellpadding=[^ >]*
#\s+cellspacing=[^ >]*
# wps导出的HTML文档是采用cp936编码的
# tidy是处理标签的，标签都是英文字母，处理过程使用latin1编码就可以，所以并不需要支持cp936编码的文本
tidy -config tidy-config.txt -i dirty.html | sed -r -e 's/\s+valign=".*"//g' > clean.html
# 可以用iconv转换为utf-8编码
tidy -config tidy-config.txt -i dirty.html |
sed -r -e 's/\s+valign=".*"//g;/<meta/{n;a\<meta charset="utf-8">' -e '}' |
iconv -f cp936 -t utf-8 > clean.html
```

tidy-config.txt

```
clean: yes
doctype: html5
word-2000: yes
char-encoding: latin1
bare: yes
drop-empty-paras: yes
drop-proprietary-attributes: yes
enclose-block-text: yes
drop-font-tags: yes
coerce-endtags: yes
join-styles: yes
output-html: yes
error-file: error.txt
```

经常遇到同事要求排查故障，但却提供截图而不出示文本信息。为此，我考虑用 OCR（Optical Character Recognition，光学字符识别）技术从截图中将文本提取出来。通过试用和比较，我感觉 Tesseract 还不错，故在此略作推荐。Tesseract 原由 HP 实验室开发，后来开源，它不仅支持许多语言，而且识别效果也不错。

安装 Tesseract

在 Arch Linux 上，可通过如下命令安装 Tesseract，其他 Linux 发行版可通过自身的包管理器安装：

```
sudo pacman -S tesseract tesseract-data-eng tesseract-data-chi_sim
```

除了安装 Tesseract OCR 引擎之外，此处也安装了对简体中文语言的支持。

使用 Tesseract

Tesseract 需在命令行下使用，假如我想要从 wb.jpeg 这张图片中提取文本，那么可以执行：

```
tesseract wb.jpeg stdout -l chi_sim
```

stdout 是将提取的文本打印到标准输出，如果文本较多则不妨将其放到文件中；-l 指定所用的语言 chi_sim（简体中文）。

参考文章

- Tesseract

rdesktop www.520hx.com.cn:8080 -u administrator -p $password -r clipboard:PRIMARYCLIPBOARD -f 全屏 -r 粘贴板

rdesktop: Connect to Windows 7 and Vista with ClearType font smoothing enabled Posted on 2008/03/10 by Andre Beckedorf

So Windows Vista finally allows to enable ClearType font smoothing for Remote Desktop / Terminal Services sessions. Update: Windows XP SP3 does too! If you try to connect to a machine running Windows XP SP 3 or later using rdesktop, you won't get smoothed font typing since at the time of this writing rdesktop does not officially offer an option to control this feature. However, here is a workaround:

rdesktop allows to specify the RDP5 experience via the -x experience switch. One can either define one of three default experiences (modem, broadband, lan) or one can specify a raw hex value that is send to the server.

NOTE: You can skip over this rather technical part, if you're not interested in the details. You'll find the workaround below.

This hex value is actually a combination of defined bit flags. After some tinkering I found that the hex value 0x80 will enable font smoothing for the connection. The file constants.h of the rdesktop sources contains these flags:

```
    #define RDP5_DISABLE_NOTHING    0x00
    #define RDP5_NO_WALLPAPER   0x01
    #define RDP5_NO_FULLWINDOWDRAG  0x02
    #define RDP5_NO_MENUANIMATIONS  0x04
    #define RDP5_NO_THEMING     0x08
    #define RDP5_NO_CURSOR_SHADOW   0x20
    #define RDP5_NO_CURSORSETTINGS  0x40    /* disables cursor blinking */
```

So, naturally an additional flag constant can be defined like this:

```
    #define RDP5_ENABLE_FONT_SMOOTHING 0x80
```

The file rdesktop.c would have to be extended preferably with an additional argument that controls the font smoothing. If you want to use font smoothing with rdesktop now you have to combine the flags (bitwise OR, addition will do too) and specify the result via the -x switch.

Here is the workaround for the three defaults mentioned above:

```
rdesktop -x 0x8F mywinserver   # equals the modem default + font smoothing
rdesktop -x 0x81 mywinserver   # equals the broadband default + font smoothing
rdesktop -x 0x80 mywinserver   # equals the LAN default + font smoothing
```

seamlessrdp

rdesktop -A "c:\seamlessrdp\seamlessrdpshell.exe C:\Program Files\Microsoft Office\Office11\winword.exe" -a 24 -x l 192.168.1.4 -u wing -p password

rdesktop怎么使用吧，开个终端吧

```
main rdesktop   //看一下帮助信息吧
rdesktop 192.168.1.1 //打开了一个8位色彩的，
rdesktop -a 16 192.168.1.1 //这个是16位色彩的了，看起来好多了
rdesktop -u administrator -p ****** -a 16 192.168.1.1 //都直接登陆了，呵,还差点什么呢
还有就是 -f 全屏操作，-g 指定使用屏幕大小 -g 800*600+0+0 这个+0啊就是，就是你
这个窗口的在你Linux上出现的位置，
其它没什么了吧!加上-r sound:local可以把声音也搞过来了
$rdesktop -u administrator -p ****** -a 16 -r sound:local 192.168.1.1
其它吧,-r 的作用挺多的可以重定向许多东西
-r comport:COM1=/dev/ttyS0    // 将串口 /dev/ttyS0 重定向为 COM1
-r comport:COM1=/dev/ttyS0,COM2=/dev/ttyS1    // 多个串口重定向
-r disk:floppy=/mnt/floppy    // 将 /mnt/floppy 重定向为远程共享磁盘 'floppy'
-r disk:floppy=/mnt/floppy,cdrom=/mnt/cdrom,root=/,c=/mnt/c    // 多个磁盘重定向
-r clientname=<client name>    // 为重定向的磁盘设置显示的客户端名称
-r lptport:LPT1=/dev/lp0    // 将并口 /dev/lp0 重定向为 LPT1
-r lptport:LPT1=/dev/lp0,LPT2=/dev/lp1    // 多个并口重定向
-r printer:mydeskjet    // 打印机重定向
```

```
-r printer:mydeskjet="HP LaserJet IIIP"    // 打印机重定向
-r sound:[local|off|remote]    // 声音重定向
```

rdesktop是linux下一个好用的用来连接Windows远程桌面（当然不仅仅在于此，只要是基于RDP协议的好像都行吧），传说中的3389嘛。
不过他是一个基于命令行的，对某些人来说可能有一些困难，这里就集合了一些很不错的命令，共享一下。
最简单的：

```
rdesktop ip
```

这个最实用，如果其他的你还没有掌握那就用这个吧，这个都是默认参数。
如果你想全屏：

```
rdesktop -f ip
```

Ctrl + Alt + Enter 开关全屏模式
这样已经很不错了，可以满足很多人了。这里一个最重要的东西，退出全屏，是什么呢？（很多初学者都对这个进去就不能会Linux的家伙很郁闷，）是Ctrl+Alt+Enter。

```
rdesktop -f sound:local ip
```

这个是把远程主机的声音带到本机（用过windows远程桌面的都知道）

```
rdesktop -f -r clipboard:PRIMARYCLIPBOARD sound:local ip
```

-r clipboard:PRIMARYCLIPBOARD是允许在远程主机和本机之间共享剪切板，就是可以复制粘贴。

```
rdesktop -f -r disk:MyDisk=/home/comet/temp ip
```

-r disk:MyDisk=/home/comet/temp就是把你的Linux下某个文件夹挂载到远程主机上

git

设置git用户名 / 邮箱

```
git config --global user.name [username]
git config --global user.email [email]
```

查看已设配置

```
git config --list
```

但是这个仅仅是设置用户名密码，如果你的Git 源每次操作需要你输入用户名/密码验证，你依然需要每次设置，那么该如何办呢？

git保存用户名密码

这里主要是配置一个config项 有两个方法，基本上原理都是一样，都是修改.git/config文件

1. 使用如下命令，修改config文件即可保存

```
echo -e "[credential]\n    helper = store" >> .git/config
```

1. 直接修改.git/config文件 vim .git/config

```
##修改成如下

[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
[remote "origin"]
    url = https://github.com/Miss-you/kernel-netfilter-sample-code.git
    fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
    remote = origin
    merge = refs/heads/master
##主要是后面这两行，如果不想保存，则删除即可
[credential]
    helper = store
##保存
```

```
git init
#添加远程仓库
git remote add origin git@github.com:wing-ho/learngit.git
#push之前需要pull下来
git pull #已经把远程仓库的内容下载下来
git pull origin master #就不需要在重新下载了
#内容不一致，需要忽略历史
git merge origin/master --allow-unrelated-histories
```

修改远程仓库

项目一开始克隆了别人的项目，做了一些改动之后，觉得有必要pull request给项目作者，于是就fork一份到自己的仓库中，需要修改远程仓库地址到自己仓库，才能提交pull request。

```
git remote set-url origin https://github.com/wing-ho/tcrevenge.git
git push
```

同步更新github上fork的repo

方法一

1.打开你的github fork repo; 2.点击Pull request; 3.点击new pull request.默认情况下，github会比较original/your fork，这时应该不会有任何输出，因为你并没有做过任何变更； 4.点击switching the base.这时github将反过来比较yourfork/original，这时你将看到original相对你fork时的所有commit; 5.点击create a pull request，这时将会反过来向你的repo提交一个pull request; 6.这时你作为你自己fork的repo的owner，提交comment，点击confirm the merge，就合并所有的改动！

方法二：

```
git remote add upstream <pathtooriginalrepo>
git fetch upstream
git merge upstream/master master
git push origin master
```

参考文章

1. git设置用户名密码
2. 如何直接在github网站上更新你fork的repo?

rsync命令

xdotool

```
  firefox --new-window -foreground --new-tab index.html
xdotool search --onlyvisible --classname Navigator windowactivate --sync key F5
 # xdotool search --onlyvisible --classname Navigator windowactivate
 # sleep 0.1
 # 打开web console
 # xdotool key --clearmodifiers ctrl+shift+k
```

xrandr

使用两个显示器，笔记本的显示器临时关闭又打开的时候，有时候用Display Setting没有办法设置正确的显示模式。

首先直接运行xrandr命令，查看设备的相关信息。

运行之后会显示当前连接设备的屏幕信息，如下图，LVDS-1和HDMI-1屏幕在用，VGA-1、DP-1和DP-2屏幕为disconnect，意为没有连接：

```
Screen 0: minimum 320 x 200, current 2646 x 1024, maximum 8192 x 8192
LVDS-1 connected primary 1366x768+0+0 (normal left inverted right x axis y axis) 293mm x 164mm
   1366x768      60.02*+
   1360x768      59.80    59.96
   1024x768      60.04    60.00
   960x720       60.00
   928x696       60.05
   896x672       60.01
   960x600       60.00
   960x540       59.99
   800x600       60.00    60.32    56.25
   840x525       60.01    59.88
   800x512       60.17
   700x525       59.98
   640x512       60.02
   720x450       59.89
   640x480       60.00    59.94
   680x384       59.80    59.96
   576x432       60.06
   512x384       60.00
   400x300       60.32    56.34
   320x240       60.05
VGA-1 disconnected (normal left inverted right x axis y axis)
HDMI-1 connected 1280x1024+1366+0 (normal left inverted right x axis y axis) 376mm x 301mm
   1280x1024     60.02*+  75.02    72.05
   1152x864      75.00
   1024x768      75.03    70.07    60.00
   832x624       74.55
   800x600       72.19    75.00    60.32
   640x480       75.00    72.81    66.67    59.94
   720x400       70.08
   640x350       70.07
DP-1 disconnected (normal left inverted right x axis y axis)
DP-2 disconnected (normal left inverted right x axis y axis)
```

这时候只要关闭内置显示器就可以了。

```
xrandr --output LVDS-1 --off
```

二.设置双屏幕显示相关命令：

(1)打开外接显示器，双屏幕显示相同的内容--克隆，（auto为最高分辨率）

```
xrandr --output LVDS-1 --same-as HDMI-1 --auto
```

(2）若要指定外接显示器的分辨率可以使用下面的命令（1280*1024）：

```
xrandr --output LVDS-1 --same-as HDMI-1 --mode 1280x1024
```

(3）打开外接显示器，设置为右侧扩展

```
xrandr --output LVDS-1 --right-of HDMI-1 --auto
```

（4）关闭内置显示器

```
xrandr --output LVDS-1 --off
```

（5）打开LVDS-1接口显示器，关闭HDMI-1接口显示器

```
xrandr --output LVDS-1 --auto --output HDMI-1 --off
```

参考文章

1. ubuntu 使用xrandr 双屏显示

Docker

因为自己对Qemu比较熟悉，Docker技术出来并没有引起我过多的关注，但是使用下来，发现docker确实有流行的理由，以前配置环境就得花费一个下午的时间，现在直接拉一个镜像下来就可以开始写代码，节省了很多不必要的麻烦。熟悉了macOS下docker之后，发现macOS平台下docker也是参考Qemu实现的。

```
docker images
```

远程控制mplayer

```
export DISPLAY=:0
mkfifo ~/mplayer-control
nohup mplayer -slave -input file=~/mplayer-control <moviefile>
```

控制指令说明，请参考https://www.mplayerhq.hu/DOCS/tech/slave.txt

```
#比如停止和播放
echo pause > ~/mplayer-control
#全屏
echo vo_fullscreen > ~/mplayer-control
```

`

linux挂载img镜像文件

1. 先查看第一个空闲loop设备

```
sudo losetup -f
/dev/loop0
```

1. 使用上一步得到的设备名，第一次创建loop设备

```
sudo losetup /dev/loop0 archlinux-2008.06-core-i686.img
```

1. 查看信息

```
sudo fdisk -lu /dev/loop0

Disk /dev/loop0: 322 MB, 322469376 bytes
53 heads, 12 sectors/track, 990 cylinders, total 629823 sectors
Units = sectors of 1 * 512 = 512 bytes
Disk identifier: 0x00000000

      Device Boot      Start         End      Blocks   Id  System
/dev/loop0p1   *          63      629822      314880   83  Linux
Partition 1 has different physical/logical beginnings (non-Linux?):
     phys=(0, 1, 1) logical=(0, 5, 4)
Partition 1 has different physical/logical endings:
     phys=(39, 52, 12) logical=(990, 15, 3)
```

我们可以看到，该镜像只有一个分区(loop0p1)，从第63扇区开始(Start列)，每扇区512字节(Units = sectors of 1 * 512 = 512 bytes)，我们算出offset，下面mout命令会用到：

```
63*512=32256
```

1. mout

```
sudo losetup -o 32256 /dev/loop1 archlinux-2008.06-core-i686.img
sudo mount -o loop /dev/loop1 /mnt/
ls /mnt/
addons  archlive.sqfs  boot  lost+found
```

事实上，fdisk可以直接查看img文件(虽然功能不全，下面会说到)，mount可以自动创建loop设备，所以上面步骤可以简化为：
I. 查看信息

```
sudo fdisk -lu archlinux-2008.06-core-i686.img
You must set cylinders.
You can do this from the extra functions menu.

Disk archlinux-2008.06-core-i686.img: 0 MB, 0 bytes
53 heads, 12 sectors/track, 0 cylinders, total 0 sectors
Units = sectors of 1 * 512 = 512 bytes
Disk identifier: 0x00000000

                Device Boot      Start         End      Blocks   Id  System
archlinux-2008.06-core-i686.img1   *          63      629822      314880   83  Linux
Partition 1 has different physical/logical beginnings (non-Linux?):
     phys=(0, 1, 1) logical=(0, 5, 4)
Partition 1 has different physical/logical endings:
     phys=(39, 52, 12) logical=(990, 15, 3)
```

第一行抱怨不能得到cylinders，原因是普通文件上没有实现ioctl操作，我们可以看到0 cylinders，但这对我们不重要，关键是我们依然可以得到第一个分区（archlinux-2008.06-core-i686.img1）的偏移值

II. 直接mount

```
sudo mount -o loop,offset=32256 archlinux-2008.06-core-i686.img /mnt/
ls /mnt/
addons  archlive.sqfs  boot  lost+found
```

高级版

查看分区|过滤前面是三个空格的行|格式化成单个空格分割|去掉行开始的空格|选择第一行|第2列数据

```
StartSector=`gdisk -l usb.img | grep "^   " | fmt -u -s | sed -e 's/^[ \t]*//' | head -1 | cut -d " " -f 2`
sudo mount -o loop,offset=$((StartSector*512)) usb.img /mnt/boot
```

Mount single logical partition from image of entire disk (device)

利用内核模块加载参数加载

重新加载内核模块可以自动加载分区，一般情况下，nbd模块不是内建，可以自由加载和卸载，loop模块如果是内建的时候，可以使用nbd模块来加载

```
rmmod loop
modprobe loop max_part=16
losetup -f debian.img
rmmod nbd
modprobe nbd max_part=16
qemu-nbd -c /dev/nbd0 win10.vhd
# 如果仍然无法正确挂载分区
# 用这个命令删除加载的内核模块后，重新加载
modprobe nbd -r
modprobe nbd max_part=16
# 添加verbose
qemu-nbd --verbose -c /dev/nbd0 win10.vhd
```

利用kpartx挂载虚拟文件系统

对于内建的loop模块，没有办法自动加载分区信息，用内核模块参数加载的方法算是最好的

```
sudo apt-get install kpartx
# 添加分区
sudo kpartx -va /dev/nbd0
# 删除分区
sudo kpartx -vd /dev/nbd0
```

Shell编程

权限数字含义

系统只能识别二进制数字，我们rwx三们分别用二进制代码0和1表示，其中0表示无权限，1表示有权限，那用rwx组成的二进制串正好对应0-7八个数字，第一位r为1时对应4，同样的w对2，x对应1，当全为0时用-表示，这正好说明这文章最开始的数字来源

test命令参数

方括号的写法只是test命令的简写而已。

```
shell判断文件,目录是否存在或者具有权限
#!/bin/sh

myPath="/var/log/httpd/"
myFile="/var /log/httpd/access.log"

# 这里的-x 参数判断$myPath是否存在并且是否具有可执行权限
if [ ! -x "$myPath"]; then
mkdir "$myPath"
fi

# 这里的-d 参数判断$myPath是否存在
if [ ! -d "$myPath"]; then
mkdir "$myPath"
fi

# 这里的-f参数判断$myFile是否存在
if [ ! -f "$myFile" ]; then
touch "$myFile"
fi

# 其他参数还有-n,-n是判断一个变量是否是否有值
if [ ! -n "$myVar" ]; then
echo "$myVar is empty"
exit 0
fi

# 两个变量判断是否相等
if [ "$var1" = "$var2" ]; then
echo '$var1 eq $var2'
else
echo '$var1 not eq $var2'
fi
```

> -a file exists. -b file exists and is a block special file. -c file exists and is a character special file. -d file exists and is a directory. -e file exists (just the same as -a). -f file exists and is a regular file. -g file exists and has its setgid(2) bit set. -G file exists and has the same group ID as this process. -k file exists and has its sticky bit set. -L file exists and is a symbolic link. -n string length is not zero. -o Named option is set on. -O file exists and is owned by the user ID of this process. -p file exists and is a first in, first out (FIFO) special file or named pipe. -r file exists and is readable by the current process. -s file exists and has a size greater than zero. -S file exists and is a socket. -t file descriptor number fildes is open and associated with a terminal device. -u file exists and has its setuid(2) bit set. -w file exists and is writable by the current process. -x file exists and is executable by the current process. -z string length is zero.

进行数学运算

位运算

```
# 开关散热风扇
cur_state=/sys/devices/virtual/thermal/cooling_device4/cur_state
state=$(cat $cur_state)
echo $((~state&0x01)) | sudo tee $cur_state
```

四则运算

Linux备份和恢复

备份

tar参数“–exclude=”后面添加需要排除的目录，需按照各发行版进行相应的修改，以arch为例：

```
sudo tar -cvzpf root-backup.tar.gz --exclude=/home --exclude=/proc --exclude=/lost+found --exclude=/mnt --exclude=/s
ys --exclude=/run/media --exclude=/var/cache --exclude=/root-backup.tar.gz /
```

因Gzip是单线程运行的，不能利用到所有的系统资源进行压缩，我们可以调用pigz来实现多线程压缩，更节省时间：

```
sudo pacman -S pigz
sudo tar --use-compress-program=pigz -cvpf root-backup.tar.gz --exclude=/home --exclude=/proc --exclude=/lost+found
--exclude=/mnt --exclude=/sys --exclude=/run/media --exclude=/var/cache --exclude=/root-backup.tar.gz /
```

如果你跟我一样Linux安装在一个分区中，可以去掉 `--exclude=/home` ，完整备份整个系统。

恢复

使用Arch的启动盘引导，挂载储存备份文件的移动硬盘到 `/usb` ，挂载安装系统的分区到 `/mnt` ，解压备份文件。

```
mkdir /usb
#挂载移动硬盘
mount /dev/sda /usb
#挂载安装系统的分区
mount /dev/nvme0n1p5 /mnt
tar -zxvf /usb/root-backup.tar.gz -C /mnt
#挂载EFI分区
mount /dev/nvme0n1p1 /mnt/boot/EFI
#生成fstab
genfstab -U > /mnt/etc/fstab
arch-chroot /mnt /bin/bash
#生成ramdisk环境
mkinitcpio -p linux
#安装efi启动器
grub-install --target=x86_64-efi --efi-directory=/boot/efi --bootloader-id=arch_grub --recheck
grub-mkconfig -o /boot/grub/grub.cfg
```

技巧

当修改了GNOME的设置，需要重启 GNOME shell，（ Alt + F2 再输入 r 再 Enter ）你可能会把窗口最大化并且隐藏了标题栏, 用合适的按键组合， Alt + F5, Alt + F10 或 Alt + Space 解决这个问题。

插件

这个视频介绍了一些非常cool的插件。 Video recorded on Arch Linux with GNOME 3.22

Dash To Dock Extension https://github.com/micheleg/dash-to-dock

Dash To Panel https://github.com/jderose9/dash-to-panel

I also have used these extensions:

Workspaces to Dock https://github.com/passingthru67/workspaces-to-dock

Topicons Plus https://github.com/phocean/TopIcons-plus

Themes Icons - La capitaine https://github.com/keeferrourke/la-capitaine-icon-theme

Cursors - Breeze Snow | KDE Plasma default GTK - Adwaita (dark variant) | GNOME's default

Shell - Adapta https://github.com/adapta-project/adapta-gtk-theme

shell - Numix https://github.com/numixproject/numix-gtk-theme

Fonts Product Sans

ZSH Theme* https://github.com/bhilburn/powerlevel9k

Wallpapers https://goo.gl/y8xhIy

截图命令 该截图方法适用于Boox M96，也就Android 4.04平台

```
adb connect hostip
adb shell screencap -p /extsd/screeenshot.png
adb pull /sdcard/screeenshot.png
```