PC手册

[I Coding 3](#_Toc11121)

[1 Compile 3](#_Toc19891)

[2 Link 3](#_Toc4513)

[3 Load 3](#_Toc231)

[4 Design Pattern 3](#_Toc4180)

[5 Coding Style 3](#_Toc24324)

[II tools 4](#_Toc32251)

[1 vim 4](#_Toc14893)

[2 git 7](#_Toc19399)

[3 wireshark 9](#_Toc20506)

[4 office 电子表格 10](#_Toc13482)

[5 office-word 10](#_Toc7056)

[6 nmap 10](#_Toc11659)

[III services 11](#_Toc22084)

[1 service control 11](#_Toc26637)

[2 ftp 11](#_Toc22276)

[3 samba 13](#_Toc1174)

[4 net-config 16](#_Toc5677)

[5 ip-tables 17](#_Toc13685)

[IV Commad lines 19](#_Toc12968)

[1 shell command reuse 19](#_Toc25691)

[2 command edit 20](#_Toc25613)

[3 grep 21](#_Toc13446)

[4 windows管理命令 22](#_Toc27291)

[5 man 23](#_Toc20299)

[6 系统信息查看 24](#_Toc21536)

[7 系统时间 25](#_Toc5601)

[8 系统运行级别 25](#_Toc25873)

[9 设置yum源 25](#_Toc27292)

[10 挂载 26](#_Toc12120)

[11 任务控制 26](#_Toc5376)

[12 账户操作 27](#_Toc27814)

[13 文件目录操作 27](#_Toc8009)

[V langs 28](#_Toc8271)

[1 javascript/html 28](#_Toc16736)

[VI protocols 29](#_Toc25842)

[1 HTTP 29](#_Toc19546)

[VII Hardware 35](#_Toc11391)

[1 Scroll Lock 35](#_Toc25318)

# Coding

## Compile

## Link

### Dll export

In MSVC \_\_declspec(dllexport)

## Load

## Design Pattern

## Coding Style

# tools

## vim

### 配置

在vimrc文件中添加配置

设置字体

set guifont=courier\_new:h12

设置窗口显示行数

set lines=40

设置窗口显示列数

set columns=80

设置光标所在行高亮

set cursorline

设置底部状态栏显示信息

set statusline+=%F

自动缩进字符数

set shiftwidth=4

tab键长度

set tabstop=4

空格替换tab

set expandtab

设置显示空白字符

set list

set listchars=tab:>-

去掉窗口工具栏

set guioptions-=T

设置tab页

set guitablabel=%N:%M:%F

是否折行显示

set wrap

set nowrap

配置替换字符

set listchars=eol:↓,tab:→→,trail:·

set list

配置映射

map <M-1> 1gt

map <M-2> 2gt

map <M-3> 3gt

map <M-4> 4gt

map <M-5> 5gt

map <M-6> 6gt

map <M-7> 7gt

map <M-8> 8gt

map <M-9> 9gt

map <M-F1> :tabclose<CR>

map <M-F2> :tabedit<CR>

显示特定字符

可以在vimrc文件中用set listchars来配置用特殊字符显示不可见字符

比如想用$先显示一行的末尾可以配置set listchars=eol:$

然后在命令模式下运行:set list 就可以看到，可以用:set nolist取消显示

对于部分特定字符。

### 代码格式化

自动缩进格式化

VISUAL BLOCK模式下（windows下按ctrl+q），选中后按=可以自动缩进。

左右缩进块代码

VISUAL BLOCK模式下（windows下按ctrl+q），选中后按<或者>

单行缩进

正常模式下，缩进光标所在单行可以按两次<(左缩进)或者>（右缩进）

回到光标上一次位置

反引号

``

或者引号

''

两者有细微区别

### 自动重新加载vimrc文件

:source $myvimrc

### 替换

vim中的替换命令为:s

其中s的取意substitute

语法：

:[range]s/search/replace/[options]

先设options为空则

如果range为空，则替换光标当前行的第一个匹配项。

:8,10s

则替换8-10行中，每一行的提一个匹配项。

现在option选择为g

:s/search/replace/g #替换每一行中出现的所有匹配项。

:8,10s/search/replace/g #替换8-10行中每一行出现的所有匹配项。

更多参见这里：

http://vim.wikia.com/wiki/Search\_and\_replace

### 匹配

匹配括号移动，包括大括号，小括号，中括号，注意先把光标移到一个括号上

%

匹配光标当前所在单词，移动光标到上一个或者写一个匹配单词

# 上一个单词

\* 下一个单词

如果你想匹配边界可以考虑用

\< or \>

如果要匹配空格用斜杠转义

\空格

### 删除

d$ //删除光标到行末

d^ //删除光标到行头

dgg //删除当前行到第一行的所有行

dG //删除当前行到最后一行的所有行

### 拷贝

y //拷贝当前行

ygg //拷贝从当前行到第一行的所有行

yG //拷贝从当前行到最后一行的所有行

y0 //拷贝从光标到行首

y^ //拷贝从光标到非空行首

y$ //拷贝从光标到行尾

yg\_ //拷贝从光标到非空行尾

ye //拷贝从当前字符到单词结束的所有字符

yw //拷贝从当前字符到下一个单词开始

### 查看帮助

查看普通模式命令 无前缀

:help x

查看可视模式命令 v\_

:help v\_u

查看命令行模式命令 :

:help :quit

找到标签后按

CTRL-]

进入选项的详细信并可以按

CTRL-T或者CTRL-O

返回，进入帮助之后可以用/进行特定内容查找。

### 输入特殊字符

在插入模式下按CTRL-K，此时会出现一个问号

然后直接输入表示特殊字符的符号码

如何查看某个特殊字符的符号码？

在命令模式下输入:digraphs就会弹出字符映射表

然后找到你需要的特殊字符

其中的前两个符号就是特殊字符的符号码

中间是特殊字符的样子

最后是对应的十进制的unicode值

比如输入向下箭头

就可以先按CTRL-K然后再输入-v就可以

## git

### 创建项目仓库

方法1：创建崭新的git项目仓库：

git init

方法2：克隆某个仓库自动初始化：

git clone

### 查看

查看状态

git status

status = tracked + untracked

tracked = work + index + history + remote

查看日志

git log

git log -p

git log -2

查看本地clone的代码库

git remote

git remote -v

查看本地clone版本origin库的所有分支

git remote show origin

### diff

work和index之间

git diff

work和history之间

git diff HEAD

work和remote之间

git diff HEAD~

index和history之间

git diff --cached

git diff --staged

index和remote之间

git diff --cached HEAD~

### 提交修改

从untracked到tracked

git add

从work到index

git add

git add .

git add -u .

git add -p <file>

从index到history

from index to history

git commit -m <some message>

from work to history

git commit -a -m <some message>

从history到remote

git push https://github.com/shaojwa/leetcode.git master

从remote到history

获取remote版本并和本地版本合并

git pull <remote>

### 回退

从tracked到untracked

git rm --cached <file>

从index到work

git checkout --files

从history到index

git reset --files

### 更新

从remote更新到本地history

git remote update

### 分支

远程添加新的分支

git remote add doc https://github.com/shaojwa/doc.git

可以使用反引用动态获取数据

git commit -m "`date`"

## wireshark

### 过滤

http方法过滤

http.request.method == "POST"

注意点1加引号2大写POST

### Host过滤

http.host == “www.zhihu.com”

http.host contains “www.zhihu.com”

为什么不像method一样http.request.host是因为host不适于rfc协议里的定义的Request-Line部分，而是request-header部分。而http.request仅仅是指Request中的Request-Line部分。

## office 电子表格

单元格内换行

openoffice:

ctrl+enter

ms-office:

alt+enter

## office-word

显示所有空白字符

wps设置如下：

显示/隐藏段落标记

去除拼写检查

去掉[工具]->[选项]->[输入时拼写检查]的勾选

## nmap

### 网段主机发现

nmap -sP 192.168.24.1/24

# services

## service control

从CentOS7开始服务的操作方式发生变化

### service 改为 systemctl

服务操作

# service <servcie> status|stop|start|restart

# systemctl status|start|stop|restart <service>

### chkconfig改为systemctl

系统自启动列表中删除或者添加

# chkconfig <service> off|on

# systemctl enable|disable <service>

# chkconfig <service>

# systemctl is-enabled <service>

# chkconfig -list

# systemctl list-unit-files --type=service

## ftp

### 主动模式

主动就是传输数据时

客户端开启某端口并PORT命令告诉服务器

服务器连接客户端

### 被动模式

被动就是传输数据时

服务器开启某端口并通过PASV命令通知客户端

客户端连接服务器

### 配置文件

/usr/sbin/vsftpd #vsftpd的主程序

/etc/vsftpd/vsftpd.conf #主配置文件

/etc/vsftpd.ftpusers #禁止使用vsftpd的用户列表文件

/etc/vsftpd.user\_list #禁止或允许使用vsftpd的用户列表文件

/var/ftp #匿名用户主目录

/var/ftp/pub #匿名用户的下载目录

### CentOS下ftp服务器参数含义

是否允许匿名访问

anonymous\_enable=YES

是否允许本地用户登入

local\_enable=YES

允许任何形式的写操作

write\_enable=YES

umask屏蔽权限022表示所有者群组和其他用户不能写

local\_umask=022

是否允许匿名用户上传文件

#anon\_upload\_enable=YES

是否允许你您不过用户创建文件

#anon\_mkdir\_write\_enable=YES

是否在进入目录是显示欢迎信息

dirmessage\_enable=YES

上传下载文件时记录日志

xferlog\_enable=YES

是否使用20端口传输数据即是否采用主动模式

connect\_from\_port\_20=YES

是否允许匿名上传的文件改变其所有者

#chown\_uploads=YES

#chown\_username=whoever

设置日志文件路

#xferlog\_file=/var/log/xferlog

是否采用标准的日志文件格式

xferlog\_std\_formats=YES

空闲会话超时时间

#idle\_session\_timeout=600

数据链接超时时间

#date\_connection\_timeout=120

是否采用非特权用户

#nopriv\_user=ftpsecure

是否识别匿名用户的ABOR请求

不建议开启但不开启可能使老的ftp客户端混乱

#async\_abor\_enable=YES

ASCII模式下的上传和下载

#ascii\_upload\_enable=YES

#ascii\_download\_enable=YES

欢迎提示语句

#ftpd\_banner=Welcome to blah FTP service

拒绝登入邮件列表

#deny\_email\_enables=YES

#banned\_email\_file=/etc/vsftpd/banned\_emails

chroot用户使能

#chroot\_local\_user=YES

#chroot\_list\_enable=YES

#chroot\_list\_file=/etc/vsftpd/chroot\_list

是否开启递归

#ls\_recurse\_enable=YES

是否监听ipv4端口

#listen=YES

是否监听ipv6端口，默认情况下开启该选项能同时监听v4和v6端口

listen\_ipv6=YES

服务名

pam\_service\_name=vsftpd

用户列表使能

`userlist\_enable=NO

tcp

tcp\_wrappers=YES

### 防火墙设置

需要关闭防火墙或放行ftp服务，然后重新加载

firewall-cmd --permanent --add-service=ftp

firewall-cmd --reload

所谓的允许匿名登入不是不使用用户名，而是可以用ftp或anonymous和空密码登入，更多选项参见这里：

https://www.centos.org/docs/5/html/Deployment\_Guide-en-US/s1-ftp-vsftpd-conf.html

### 指定命令文件执行ftp操作

ftp -s:script.txt ftp.server.com

script.txt 内容：

your\_username

your\_password

bin

cd /files

put file.zip

bye

## samba

基于CentOS-7，参考文档:

http://www.unixmen.com/install-configure-samba-server-centos-7/

### 安装

检查samba是否已经安装

安装包检查

rpm -qa |grep samba

或者

yum list installed | grep samba

配置一个允许所有权限的匿名共享

首先创建一个目录

mkdir -p /home/shaojwa/work

chmod -R 777 /home/shaojwa/work

配置samba

samba配置文件路径，配置前先备份

/etc/samba/smb.conf

mv /etc/samba/smb.conf /etc/samba/smb.conf.bak

在[global]下配置

workgroup = WORKGROUP

添加共享信息

[work]

path = /home/shaojwa/work

public = yes

browsable = yes

writable = yes

guest ok = yes

read only = no

启动samba

systemctl start smb

systemctl start nmb

systemctl enable smb

systemctl enable nmb

用testparm 检查smb配置

tesparm

配置selinux

setsebool -P samba\_enable\_home\_dirs on

chcon -t samba\_share\_t /samba/anonymous\_share/

关闭SELinux

查看selinux状态

sestatus

如果状态是enabled则尝试关闭selinux

编辑/etc/selinux/config文件

将其中的SELINUX=enforcing改为SELINUX=disabled

重启

关闭防火墙

systemctl stop firewalld #注意是firewalld

或者配置防火墙

firewall-cmd --permanent --add-port=137/tcp

firewall-cmd --permanent --add-port=138/tcp

firewall-cmd --permanent --add-port=139/tcp

firewall-cmd --permanent --add-port=445/tcp

firewall-cmd --permanent --add-port=901/tcp

firewall-cmd --reload

此时可以看到存在word共享，但是无法打开文件夹

配置权限打开文件夹

此时从windows下访问linux的贡献时提示输入用户名密码

匿名访问

需要在conf文件中的[global]部分添加如下两行

security = user

map to guest = bad user

此时从windows下访问linux已经能出现共享目录

但是没有权限访问

### 创建安全共享

创建用户

useradd -s /sbin/nologin smbusr

创建组

groupadd smbgrp

将用户加入组

usermod -a -G gmbgroup smbgrp

设置密码

smbpasswd -a smbusr

创建共享目录后设置权限

mkdir /samba/share

chmod -R 0755 /samba/share

chown -R smbusr:smbgrp /samba/share

配置smb.conf

[share]

path = /samba/share

writable = yes

browsable = yes

guest ok = no

valid users = @smbgrp

如果不关闭SELinux就需要添加如下配置

chcon -t samba\_share\_t /samba/share/

重启smb

systemctl restart smb

systemctl restart nmb

最后提醒，共享目录最好不要放在home下，因为home是其中一个特殊共享

有对应的配置。

## net-config

从CentOS-7以及RHEL-7开始开始使用systemd。

sudo systemctl status network.service

sudo systemctl status network

sudo systemctl start network.service

sudo systemctl start network

sudo systemctl stop network.service

sudo systemctl stop network

sudo systemctl restart network.service

sudo systemctl restart network

在此之前的版本可以用：

service network status

service network stop

service network start

service network restart

等价于使用初始化脚本：

/etc/init.d/network status

/etc/init.d/network restart

/etc/init.d/network stop

/etc/init.d/network start

### VMware相关的虚拟网络设备

参考这里：

http://blog.csdn.net/lwbeyond/article/details/7648509

虚拟交换机：

VMWare安装完成之后有三个虚拟交换机

VMnet0：桥接模式下的虚拟交换机

VMnet1：在host-only模式下的虚拟交换机

VMnet8：在NAT模式下的虚拟交换机

虚拟网卡：

VMnet Network Adapter VMnet1：在host-only模式下，Host于Host-Only虚拟网络进行通信达呃虚拟网卡，这是在物理机的虚拟网卡，和虚拟机上的虚拟网卡不是一个概念。

VMnet Network Adapter VMnet8：在NAT模式下，Host与NAT虚拟网络进行通信的虚拟网卡。

虚拟机里配置CentOS为NAT模式

虚拟机安装好CentOS之后默认采用NAT模式，所以只需要在MVware中设置使用NAT模式，并在CentOS中打开开启启动开关 ONBOOT=yes就可以，默认情况下得到的IP地址是192.168.59.0/24，这个地址是VMWare配置的DHCP地址范围，可以在VMWare上修改。

虚拟机里配置CentOS为bridged模式

虚拟桥接网卡

在虚拟机里安装好CentOS之运行ifconfig会友现在一张虚拟网卡virbr0，要了解这方面内容，详细的可以参考：

http://wiki.libvirt.org/page/VirtualNetworking

虚拟网络交换机

在宿主机里，虚拟网络交换机以网卡的形式显示。默认的一个交换机，在

libvirt守护进程安装并启动时创建，显示为virbr0。

可以通过通过查看libvirtd来查看该守护进程是否存在：

$ ps -aux | grep libvirtd

你可以通过ifconfig virbr0 来查看相关的信息。

$ ifconfig virbr0

或者你使用ip命令

$ ip addr show virbr0

默认模式

在默认情况下，虚拟网络交换机工作在NAT模式下。DHCP给每个虚拟网络交换机分配一个地址段。Libvirt通过dnsmasq来作为DNS和DHCP服务器。可以通过查看dnsmasq查看该守护进程：

$ps -aux | grep dnsmasq

同时可以看到该守护进程的配置文件。

其他模式

还有两种模式：第一是routed模式，就是我们常说的bridged模式。第二种是Isolated模式。

查看

$ ifconfig

或者

$ virsh net-list

结果一般为：

Name State Autostart Persistent

-----------------------------------------

default active yes yes

禁用

$ virsh net-destroy default

启用

$ virsh net-start default

删除

$ virsh net-undefine default

编辑

$ virsh net-edit default

nat改为bridge

待补充

## ip-tables

Iptables是绝大多数linux发行版内建的防火墙，在CentOS<7的版本中可以使用service命令：

查看Iptables的状态

# service iptables status

# service ipv6tables status

关闭iptables

# service iptables stop

# service ipv6tables stop

启动iptables

# service iptables start

# service ipv6tables start

从系统自启动列表中删除iptables

# chkconfig iptables off

# chkconfig ipv6tables off

将iptables添加进系统自启动列表

# chkconfig iptables on

# chkconfig ipv6tables on

在CentOS7之后的版本统一用systemctl命令:

# systemctl [status|start|stop|restart] iptables

# systemctl [status|start|stop|restart] ipv6tables

# Commad lines

其实也只是网上的资料汇总出来的东西。只是很多没有单独拿出来介绍，总是和其他的一些技巧一并介绍。而我写的这篇只关注怎么用最简单的命令复用之前输入过的命令。

## shell command reuse

### ！系列

网上的说法是，这些以！开始的标记（暂且就用标记来称呼吧）都是一些列特殊的环境变量。

!! #上一条命令的全部内容\*\*

示例

$ cd test

$ echo !!

cd test

!^ #上条命令的第一个参数

示例

$ cd test

$ echo !^

test

注意最后的输出不是cd

!$ //上条命令的最后一个参数\*\*

示例

$ cd first second last

$ echo !$

last

!\* #上条命令的最后一个参数\*\*

示例：

$ cd first second last

$ echo !$

first second last

!prefix #最近一条以prefix开始的命令

注意这里的prefix可以是一个或者多个字母表，看了以上几条命令我只有感慨正则表达无处不在

!prefix:p #打印但不执行最近一条以prefix开始的命令

### ^系列

^foo #上一条命令中去掉foo之后的命令

^foo bar #上一条命令中去掉foo和bar之后的命令

^foo^bar #上一条命令中把foo换成bar之后的命令

### 其他常用的：

cd - #//回到上一次的目录

cd ~ #//回到登入用户的home目录

## command edit

linux下命令操作是我最喜欢的方式。但是编辑命令本身也有一些小技巧值得学习，因为这样可以让你更快得编辑命令，进而节省时间。

依次显示之前命令的最后一个参数

alt+.

这个还是很有用的，比如本来打算输入cat xxx，结果输入 cd xxx 那么你只需要输入cat 之后按 alt + . 就可以自动补全，如果最后一个参数比较长就很有用。作为对比：

ctrl+alt+y #只会记住上一条命令的第一个参数

查询

ctrl+r //进入查询模式

ctrl+g //撤销查询

撤销修改

ctrl+/ //撤销对命令的修改

互换

ctrl+xx //光标所在位置和行首之间互换

只移动光标不更改内容

光标的移动分三种，行移动，单词移动，字母移动。

ctrl+a //移到行首

ctrl+e //移到行尾

alt+f //以单词为单位向前移动

atl+b //以单词为单位向后移动

ctrl+f //以字母为单位向前移动

ctrl+b //以字母为单位向后移动

删除命令

ctrl+u //删除光标后的所有内容

ctrl+k //删除光标前的所有内容

ctrl+w //删除光标后面的一个单词

alt+d //删除光标前面的一个单词

ctrl+h //删除光标后面的一个字母

ctrl+d //删除光标前面的一个字母

字母互换

ctrl+t //互换光标左右字母后光标移到后一个位置

单词互换

alt+t //互换光标左右两边的单词后移到下个单词位置

大小写转换

alt+c //把光标所在出的字符改为大写然后跳到下一个单词词首

alt+l //把光标所在处的字符到单词结尾都改为小写

alt+u //把光标所在处的字符到单词结尾都改为大写

查看前后命令

ctrl+p //查看之前命令

ctrl+n //查看之后命令

## grep

### 文件内容查找

grep <options> <pattern> <files>

示例：

$ grep -rn "loosgood" \*

\* 表示当前目录所有文件，也可以是某个文件名

-r 是递归查找

-n 是显示行号

-R 查找所有文件包含子目录

-i 忽略大小写

-l 只列出匹配的文件名

-L 列出不匹配的文件名，

-w 只匹配整个单词，而不是字符串的一部分

### 查找某个目录信息

ll root | grep roor$ #使用$来定位

### 查找man文档中以连字符开始的选项

$ man chmod | grep [-]R # -R, --recursive

加上-n显示行号：

$ man chmod | grep [-]R

111: -R, --recursive

用more显示从某行开始：

$ man chmod | more +111

## windows管理命令

### 获取卷(分区)的序列号

vol [drive:]

### 获取字符映射表

charmap

### 控制面板类

sysdm.cpl

WinNT 5.0和6.0以上都可用，打开系统属性对话框。

在6.0以后还有另一个系统配置对话框可以通过control system打开。

ncpa.cpl #打开系统网络链接控制面板

appwiz.cpl 添加删除程序

timedate.cpl 时间日期面板

desk.cpl 桌面配置面板

access.cp 辅助功能配置（鼠标，鼠标，声音等）

inetcpl.cp Internet配置

firewall.cp 防火墙配置

Netcpl.cpl

Intl.cpl

### 区域和语言设置

control admintools

control mouse

control keyboard

## man

linux下命令巨多，有的是所谓的内建（built-in command），比如read，有的不是比如grep。平时查命令用法用man，但是内建命令怎么查？一直不清楚，最近专门了解了下，做点记录。

### 查内建命令用法

Help <cmd>

help也是一个内建命令，查起来很方便，不需要依赖网络查用法，这是重点。

### help和man区别

只能网上找，按照自己的理解，help是bash内建命令，使用bash内部的数据结构获取保存信息。而man不是。help只用来查看bash命令。所谓的内建，在我看来就是在bash进程里执行，终端是一个进程。写到这里，想到另外一个问题。如何查看当前shell的pid。

### 查看一个命令是否为内建命令

$ type <cmd>

示例：

# type cd

cd is a shell buildin

# type grep

grep is hashed (/bin/grep)

### 一个命令能否同时是内建也是外部命令

可以比如pwd，至于为什么还不清楚， 默认情况下是先使用内建命令。

### 如何查询是否存在相关功能的命令

直接输入

apropos <keyword>

即可查看与keyword相关功能的命令，很强大。

如何关闭内建命令

enable -n

查看所有激活的内建命令

enable -a

## 系统信息查看

### 查看所有用户

$ cat /etc/passwd

$ awk -F':' '{ print $1}' /etc/passwd

### 所有组

$ cat /etc/group

### 组内所有成员

lid group\_name|user\_name

### 查看系统完整信息

uname -a

### 查看系统架构

arch # deprecated since release 2.13

uname -m # identical with arch

### 查看快捷键绑定

bind -p

### 查看当前正在使用的bash的pid

其实很简单$$命令就能显示当前进程的pid：

# echo $$

bash: 2085: command not found

# ps axu | grep 2085

root 2085 0.0 0.1 5232 1664 pts/0 S 09:38 0:00 bash

### 查看系统当前登入用户

$ who

说明，TTY字段下显示tty表示直连终端，pts表示虚拟终端。

查看系统最后一次重启的时间

who -b

查看当前终端

who -m

查看对应的shell 进程id：

who -u （自然可以和-m配合）

查看run-level：

who -r

### 查看系统用户以及执行的命令

w

它告诉你当前系统有谁登入，从哪里登入，都在执行什么命令。所有linux发行版都包含该命令。它差不多是who和uptime功能的组合。

## 系统时间

### 同步系统时间

clock

同事管理linux服务器，服务器时间一直显示有问题。今天专门去看了下。系统设置时区之后，修改当前时间，重启系统又有问题。后来发现，是没有写入到硬件时钟。linux系统的两个时钟，硬件使用和系统时钟需要同步。系统启动时，系统时钟需要从硬件时钟中读取时间。查看硬件时间用clock 或者hwclock命令。和系统时间的同步用：

clock --systohc #把系统时间同步到硬件

clock --hctosys #把硬件时间同步到系统

## 系统运行级别

### 调整运行级别

$ init

### 查看运行级别

$ runlevel

或者

$ who -r

linux下的run-leveld的含义就好像windows下的正常模式，安全模式，命令行模式等。以图形界面登入linux，runlevel一般是5，而字符界面登入linux，runlevel一般是3。linux一般有8个level：

0 Halt the system

1 Single-user mode

2 Multi-user mode

3 Multi-user mode with networking

4 Not used/user-deinable

5 Multi-user mode with GUI

6 Reboot the system

## 设置yum源

创建cdrom.repo文件包含如下内容：

[cdrom]

name=cdrom-repo

baseurl=file:///mnt/cdrom

enable=1

gpgcheck=0

## **挂**载

mdir /mnt/cdrom

mount -t auto /dev/cdrom /mnt/cdrom

## 任务控制

当前进程挂起

ctrl+z # suspend the current job

在一个tty下操作时，如果想去做别的事而挂起当前的操作，可以用该快捷键。当我们想恢复之前挂起的工作时用fg（move job to the forground）命令。

$ fg [job\_spec]

其中的job\_spec一般翻译为任务号，如果挂起的只有一个任务，则可以省略该参数，如果多个，如何指定？使用jobs命令

$ jobs

[1]- Stopped vim hello

[2]+ Stopped vim world

不带参数时，该命令将输出所有的活动任务（active jobs），其他参数用请help

后台运行进程

于fg相对应的是bg，bg是让被挂起的某个进程在后台运行。

$ bg [job\_spec ...]

或者启动时就在后台运行，在命令后加&符号（ampersand）

<cmd> &

后台进程的终止

kill %job\_spec

kill pid

以上两种方式都可以，job\_spec就是通过jobs命令获得的任务号。

前台进程中止

ctrk+c

kill的说明

kill是否有效得分情况，kill不带参数时，其实是向进程发送SIGTERM的信号，进程度过不监听该信号，那发送这个命令是无效的。如果要强制中止进程，就需要用如下命令：

kill -SIGKILL pid

也就是我们常常说的

kill -9 pid

因为通过以下命令知道SIGKILL 就是编号为9的信号

kill -l

当然kill和jobs一起用很方便

kill -9 `jobs -p`

## 账户操作

### 添加用户

useradd shaojwa

## 文件目录操作

### 拷贝文件夹

cp -r /root/test /root/user/test #注意-r

# langs

## javascript/html

### 语法

script在chrome下与需要闭合标签

<script src="ut.js"></script>

### showModalDialog卡死JS引擎

尽管Chrome等已经陆续废弃对该API的支持：

https://blog.chromium.org/2014/07/disabling-showmodaldialog.html

FireFox：

<https://developer.mozilla.org/en-US/docs/Web/API/Window/showModalDialog>

### 异步机制

http://ejohn.org/blog/how-javascript-timers-work/

# protocols

## HTTP

### 缓存相关

缓存的类型

数据库数据缓存，常见方案memcached。代理服务器缓存，常见方案Squid。CDN缓存，或者叫反向代理缓存。浏览器缓存以及Web应用层缓存。

浏览器端缓存的机制

从新鲜度和交验值两个维度进行。

头部属性

If-Modified-Since

Last-Modified

标准的请求头标签，和请求一并发送给服务器。



可以看到，尽管返回的是304，但是Last-Modified字段还是作为响应头返回。同时请求的If-Modified-Since和Last-Modified的时间是一致的。因为客户端会用200的响应头中的Last-Modified头来更新文件的最后一次更新时间。

If-None-Match

ETag

这两个成对使用和If-Modified-Since与Last-Modified类似。首先，是ETag是服务器先构造的。之后，浏览器将该几号传给服务器，服务器进行验证页面是否被修改。那么已经有Last-Modified为什么还需要ETag：

1. 有的页面时间被周期性修改但内容没有发生改变。
2. 有写文件修改非常频繁，1秒内修改N次，If-Modified-Since能检测到的粒度是秒级，无法应对这种高频修改。
3. 有些服务器无法精确得到文件的最后修改时间。

HTTP协议没有规定ETag的内容是什么或者要怎么实现，只是要求ETag要放在双引号内。关于ETag的实现可以参考：

<http://bitworking.org/news/150/REST-Tip-Deep-etags-give-you-more-benefits>

Cache-Control

Expires

Cache-Control的优先级高于Expires。Cache-Control/Expires的优先级要高于Last-Modified/ETag。一般情况下，使用Cache-Control/Expires会配合Last-Modified/ETag一起使用，因为即使服务器设置缓存时间，当用户点击“刷新”按钮时，浏览器会忽略缓存继续向服务器发送请求，这时Last-Modified/ETag将能够很好利用304，从而减少响应开销。

强制刷新

ctrl+F5

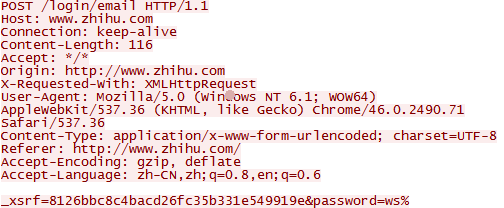
缓存参考

<http://www.alloyteam.com/2012/03/web-cache-1-web-cache-overview/>

### Cookie和Session

cookie 被设计用来保存用户的状态信息，因为HTTP是无状态的，需要用额外的手段来记住用户的某些信息。比如登入状态，同时服务器也可以根据cookie来识别不同的用户。session是在cookie之后产生的，因为cookie保存在浏览器端，容易被修改或者伪造。session一般通过session\_id来关联，并在cookie中传输。

清除所有cookie之后尝试登入zhihu，故意输错校验码时，抓包：



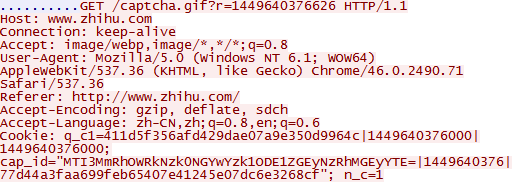
发现请求头中没有包含cookie信息。

响应头如下：

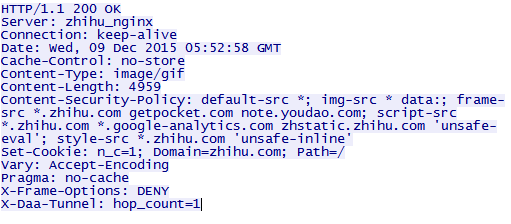


我们可以看到有三个Set-Cookie字段。

紧接着浏览器发出第二个请求：



此时我们就发现，请求中已经携带Cookie信息。而携带的信息就是之前三个Set-Cookie中的q\_c1，cap\_id，以及n\_c，响应就是请求的图片数据：



当我们再次尝试用错误的验证码登入。

发现，和最初的一次唯一的区别就是多了包含三对数据的Cookie。如下：



但响应中重新设置Cookie：



在整个流程中，没有见到session，那session体现在哪，当我们正常登入，抓包：



对比发现，在多次请求中，发现q\_c1的值一直没变，而cap\_id的值一直在变。

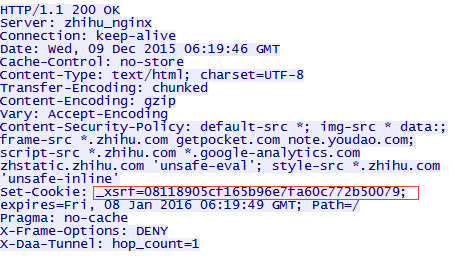
这两个值最初都是server给的，但是cap\_id会被重置，看起来有点像标记一个会话。登入成功后：



发现有新的cookie：z\_c0，unlock\_ticket， n\_c也同时被设置为空。于是接下去的请求就建立在四个cookie基础上：q\_c1，cap\_id，z\_c0以及unlock\_ticket。



尝试登入后，重新设置\_xsrf：



之后的请求就是在之前四个cookie的基础上加上\_xsrf：



XSS拿到cookie

# Hardware

## Scroll Lock

在虚拟机里的CentOS上运行命令，发现无法输入，看了下发现ScrollLockz指示亮着，于是按ScrollLock键后可以输入。在物理机上不会有这个问题，在虚拟机的shell里存在。在shell里试了下发现按下改键之后，之后输入的所有的命令都会缓存起来，等你再次按ScrollLock随时，之前的所有输入会一次性得显示出来。