

Netkiller Shell 手札

目录

自述

1. 写给读者
2. 作者简介
3. 如何获得文档
4. 打赏 (Donations)
5. 联系方式

1. Shell

1. 快捷键
 - 命令行编辑命令
 - 重新执行命令快捷键
 - 终端控制快捷键
 - Bang (!) 命令
2. chsh - change login shell
3. 执行程序返回值

2. Bash Shell

1. bash - GNU Bourne-Again SHell
 - n 检查脚本是否有语法错误
 - x 显示详细运行过程
2. 切换身份
3. I/O 重定向
 - stdout
 - error 重定向
 - 使用块记录日志
 - tee - read from standard input and write to standard output and files
 - 重定向到文件
 - nettee - a network "tee" program
 - 创建文件
 - 快速清空一个文件的内容
4. pipes (FIFOs)

5. mktemp - create a temporary file or directory 临时目录与文件

6. History 命令历史记录

- .bash_history

 - 格式定义

 - 设置忽略命令

- .mysql_history

7. hash - hash database access method

8. prompt

9. 变量 variable

- 系统变量

 - 命令行参数传递

 - \$n \$# \$0 \$?

 - \$? 程序运行返回值

 - shift 移位

- 表达式

- Internal Environment Variables

 - \$RANDOM 随机数

 - 与 history 有关的环境变量

- set 设置变量

 - set -/+e 控制程序是否退出

- unset 变量销毁

- 设置变量默认值

- export 设置全局变量

- declare

- Numerical 数值运算

- Strings 字符串操作

 - ###/

 - %%/

 - 字符串截取

 - #

 - example

 - 计算字符串长度

 - 字符串查找替换

- Array 数组

 - for 与 array

- while 与 array
- array 与 read
- 拆分字符串并转换为数组
- 数组转为字符串
- read 赋值多个变量
- eval
- typeset
- envsubst - substitutes environment variables in shell
- format strings
- 10. conditions if and case
 - if
 - 判断变量是否包含字符串
 - case
- 11. Loops for, while and until
 - for
 - while
 - until
- 12. Functions
 - Local variables
- 13. User interfaces
 - input
- 14. subshell
- 15. Example
 - 有趣的Shell
 - backup
 - CPU 核心数
 - Password
 - processes
 - pid
 - kill
 - pgrep
 - Shell 技巧
 - 行转列, 再批评
 - for vs while
 - 遍历字符串
 - to convert utf-8 from gb2312 code

使用内存的百分比
合并apache被cronlog分割的log文件
Linux 交集 差集 并集

3. 小众 Shell

1. fish shell

安装 fish shell
配置 fish
主题管理

2. Z Shell

installing Z shell
Oh My ZSH!
Starting file
 ~/.zshrc
Prompting
Aliases
History
FAQ
 Home/End key

3. Berkeley UNIX C shell (csh)

4. KornShell

4. Shell 命令

1. Help Commands

man - an interface to the on-line reference manuals
manpath.config
查看man手册位置
指定手册位置

2. getconf - Query system configuration variables

3. test 命令

判断目录

4. 目录和文件

dirname
filename
排除扩展名
取扩展名

test - check file types and compare values
file — determine file type
stat
mkdir - make directories
rename
touch
truncate
ls - list directory contents
 full-time / time-style 定义日期时间格式
cp - copy files and directories
 copy directories recursively
 覆盖已存在的文件
 -a, --archive same as -dR --preserve=all
rm - remove files or directories
 -bash: /bin/rm: Argument list too long
 zsh: sure you want to delete all the files in /tmp
 [yn]?
df - report file system disk space usage
du - estimate file space usage
tac - concatenate and print files in reverse
split - split a file into pieces
 按行分割文件
 按尺寸分割文件
find - search for files in a directory hierarchy
 多目录匹配
 name
 regex
 user
 perm
 type
 分别设置文件与目录的权限
 -delete
 exec
 排除目录
 -mmin n File's data was last modified n minutes
 ago.

- ctime
- mtime / -mmin
- newer
- print / -printf
- size
- path
- 目录深度控制
- maxdepth
- xargs

5. package / compress and decompress

tar — The GNU version of the tar archiving utility

tar examples

gunzip

b2zip

compress

.xz 文件

-t, --list

tar: Removing leading `/' from member names

-C, --directory=DIR

--exclude

-T

日期过滤

保留权限

-r, --append

远程传输

分卷压缩

cpio - copy files to and from archives

gzip

zip, zipcloak, zipnote, zipsplit - package and compress
(archive) files

bzip2, bunzip2 - a block-sorting file compressor

RAR

7-Zip

压缩

浏览压缩包

解压

Creates self extracting archive.

RAR

xz, unxz, xzcat, lzma, unlzma, lzcat - Compress or decompress .xz and .lzma files

6. 日期和时间

日期格式

weekday name

-d --date=

日期偏移量

day

month

year

时间偏移

时间戳

RFC 2822

UTC

字符串转日期

7. 数值与运算

数值运算

seq - print a sequence of numbers

bc - An arbitrary precision calculator language

8. 文本处理

iconv - Convert encoding of given files from one encoding to another

cconv - A iconv based simplified-traditional chinese conversion tool

uconv - convert data from one encoding to another

字符串处理命令expr

cat - concatenate files and print on the standard output

-s, --squeeze-blank suppress repeated empty output lines

-v, --show-nonprinting use ^ and M- notation, except for LFD and TAB

与管道配合使用

nl - number lines of files

tr - translate or delete characters

替换字符

英文大小写转换

[CHAR*] 和 [CHAR*REPEAT]

-s, --squeeze-repeats replace each input sequence of a repeated character that is listed in SET1 with a single occurrence of that character
-d, --delete delete characters in SET1, do not translate

cut - remove sections from each line of files

printf - format and print data

Free `recode' converts files between various character sets and surfaces.

/dev/urandom 随机字符串

col - filter reverse line feeds from input

apg - generates several random passwords

head/tail

彩色输出

跳过 n 行, 输出后面内容

尾部剪掉 n 行

反转字符串或文件内容

TAB符号与空格处理

expand - convert tabs to spaces

unexpand - convert spaces to tabs

grep, egrep, fgrep, rgrep - print lines matching a pattern

删除空行

-v, --invert-match

输出控制 (Output control)

显示行号

-o, --only-matching show only the part of a line matching PATTERN

IP 地址

UUID

行列转换

递归操作

-c, --count print only a count of matching lines
per FILE

binary file matches

Context control

-A, --after-context=NUM print NUM lines of
trailing context

-B, --before-context=NUM print NUM lines of
leading context

-C, --context=NUM print NUM lines of output
context

--color

Regex selection and interpretation

.

2010:(13|14|15|16)

[]与{}

-P, --perl-regexp Perl正则表达式

fgrep

egrep

匹配多个条件

sort - sort lines of text files

对列排序

-s, --stable stabilize sort by disabling last-resort
comparison

uniq

awk

处理列

printf

Pattern(字符匹配)

Pattern, Pattern

Built-in Variables (NR/NF)

NR

NF

练习

使用 ss 命令统计 TCP 状态

TCP/IP Status

用户shell统计

access.log POST与GET统计

Built-in Functions

length

toupper() 转为大写字母

tolower() 转为小写字母

rand() 随机数生成

过滤相同的行

数组演示

sed

查找与替换

正则

aaa="bbb" 提取bbb

"aaa": "bbb" 提取bbb

首字母大写

insert 插入字符

追加字符

修改字符

删除字符

delete

行操作

编辑文件

正则表达式

管道操作

字母大小写转换

perl

案例

HTML 转 文本

9. 表格操作/行列转换

column - columnate lists

paste - merge lines of files

join

10. standard input/output

xargs - build and execute command lines from

standard input

格式化

standard input

-l 替换操作

-n, --max-args=MAX-ARGS use at most MAX-ARGS arguments per command line

-t, --verbose print commands before executing them

-d, --delimiter=CHARACTER items in input stream are separated by CHARACTER, not by whitespace; disables quote and backslash processing and logical EOF processing

-0, --null items are separated by a null, not whitespace; disables quote and backslash processing and logical EOF processing

-r, --no-run-if-empty if there are no arguments, then do not run COMMAND; if this option is not given, COMMAND will be

-p, --interactive prompt before running commands

11. flock - manage locks from shell scripts

12. 进制转换 - 16进制 - 8进制 - 二进制

od - dump files in octal and other formats

16进制

使用 od 随机生成密码

hexdump, hd -- ASCII, decimal, hexadecimal, octal dump

xxd - make a hexdump or do the reverse.

指定每行的列数

跳过字节

binutils

strings - print the strings of printable characters in files.

13. ed, red - text editor

14. vim

查找与替换

删除操作

插入文件

批处理

vi 批处理

line()
set fileformat
空格与TAB转换

15. Wget - The non-interactive network downloader.

Logging and input file

-i, --input-file=FILE download URLs found in local or external FILE.

下载相关参数

-O, --output-document=FILE write documents to FILE 保存到文件

HTTP options (HTTP 选项)

--post-data=STRING use the POST method; send STRING as the data.

header HTTP头定义

Recursive download

-r, --recursive specify recursive download.

-m, --mirror shortcut for -N -r -l inf --no-remove-listing.

--no-passive-ftp disable the "passive" transfer mode.

下载一组连续的文件名

16. CURL - transfer a URL

基本用法

提交表单数据

上传文件

connect-timeout

max-time

compressed

代理服务器

-w, --write-out <format> 输出格式定义

-A/--user-agent <agent string>

referer

-v

-o, --output FILE Write output to <file> instead of stdout

-L, --location

-H/--header <line> Custom header to pass to server

(H)

- Last-Modified / If-Modified-Since
- ETag / If-None-Match
- Accept-Encoding:gzip,defalte
- HOST
- HTTP 认证
- Accept
- Content-Type
- curl-config
- 指定网络接口或者地址
- Cookie 处理
- Restful 应用 JSON 数据处理
 - Curl OAuth2
 - Curl + OAuth2 + Jwt
- 访问自签名证书
- HTTP2
- FAQ
- 17. expect
 - 模拟登录 telnet 获取Cisco配置
 - 模拟登录 ssh
 - SCP
 - openssl 例子
- 18. expect-lite - quick and easy command line automation tool
- 19. sshpass - noninteractive ssh password provider
- 20. Klish - Kommand Line Interface Shell (the fork of clish project)
 - 安装Klish
 - 为用户指定clish作为默认Shell
 - FAQ
 - clish/shell/shell_expat.c:36:19: fatal error: expat.h: No such file or directory
- 21. Limited command Shell (lshell)
- 22. TUI
 - screen - screen manager with VT100/ANSI terminal emulation
 - tmux — terminal multiplexer

byobu - wrapper script for seeding a user's byobu configuration and launching a text based window manager (either screen or tmux)

htop - interactive process viewer

elinks

chat

23. jq - Command-line JSON processor

24. asciinema 终端录屏

25. parallel - build and execute shell command lines from standard input in parallel

26. multital

27. Logging

logger - a shell command interface to the syslog(3) system log module

28. Password

Shadow password suite configuration.

newusers - update and create new users in batch

chpasswd - update passwords in batch mode

sshdpass - noninteractive ssh password provider

29. 信息摘要

cksum, sum -- display file checksums and block counts

md5sum - compute and check MD5 message digest

30. envsubst - substitutes environment variables in shell format strings

6. Shell Terminal

1. terminal

resize - set TERMCAP and terminal settings to current xterm window size

tset, reset - terminal initialization

stty - change and print terminal line settings

2. tput

Change the prompt color using tput

3. dialog

--inputbox

4. whiptail - display dialog boxes from shell scripts

- msgbox
- infobox
- yesno
- inputbox
- passwordbox
- textbox
- checklist
- radiolist
- menu
- gauge

A. 附录

1. 参考文献

表格清单

- 2.1. 文件目录表达式
- 2.2. 字符串表达式
- 2.3. 组合表达式

范例清单

- 2.1. A "Power User" Prompt
- 2.2. A Prompt the Width of Your Term
- 2.3. The Elegant Useless Clock Prompt
- 2.4. Basic conditional example if .. then
- 2.5. Conditionals with variables
- 2.6. case
- 2.7. Functions with parameters sample
- 2.8. Using select to make simple menus
- 2.9. Using the command line
- 2.10. Reading user input with read
- 2.11. read
- 2.12. random password
- 4.1. backup(find + tar)
- 4.2. example for expect

4.3. example for expect

4.4. example 1

4.5. *.exp

4.6. parallel - build and execute shell command lines from
standard input in parallel

6.1. whiptail - yesno

6.2. whiptail - inputbox

6.3. whiptail - passwordbox

6.4. whiptail - passwordbox

6.5. whiptail - example 1

6.6. whiptail - radiolist

Netkiller Shell 手札

Unix, Linux, FreeBSD & Mac Shell

Mr. Neo Chan, 陈景峯(BG7NYT)

中国广东省深圳市望海路半岛城邦三期
518067
+86 13113668890

<netkiller@msn.com>

电子书最近一次更新于 2023-04-20 22:53:31

版权 © 2009-2023 Neo Chan

版权声明

转载请与作者联系，转载时请务必标明文章原始出处和作者信息及本声明。

Netkiller 手札系列电子书 <http://www.netkiller.cn>



Netkiller Linux 手札

陈景峰 著



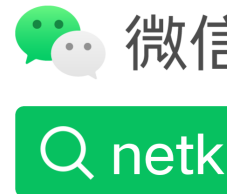
<http://www.netkiller.cn>
<http://netkiller.github.io>
<http://netkiller.sourceforge.net>

微信公众号: netkiller
微信: 13113668890 请注明
“读者”

QQ: 13721218 请注明“读
者”

QQ群: 128659835 请注明
“读者”

[知乎专栏](#) | [多维度架构](#)



打开“微信 / 发现 / 搜一搜”

内容摘要

Netkiller 系列手札 已经被 Github 收录，并备份保存在北极地下250米深的代码库中，备份会保留1000年。



Preserving open source software for future generations

The world is powered by open source software. It is a hidden cornerstone of modern civilization, and the shared heritage of all humanity.

The GitHub Arctic Code Vault is a data repository preserved in the Arctic World Archive (AWA), a very-long-term archival facility 250 meters deep in the permafrost of an Arctic mountain.

We are collaborating with the Bodleian Library in Oxford, the Bibliotheca Alexandrina in Egypt, and Stanford Libraries in California to store copies of 17,000 of GitHub's most popular

and most-depended-upon projects—open source’s “greatest hits”—in their archives, in museum-quality cases, to preserve them for future generations.

我们正在与牛津的博德利安图书馆、埃及的亚历山大图书馆和加利福尼亚州的斯坦福图书馆合作，将 GitHub 最受欢迎和最依赖的项目 17,000 份副本保存在他们的档案中，在博物馆质量的案例中，为子孙后代保存这些作品。

<https://archiveprogram.github.com/arctic-vault/>

我的系列文档:

[Netkiller Linux 手札](#) [Netkiller FreeBSD 手札](#) [Netkiller Shell 手札](#) [Netkiller Security 手札](#)
[Netkiller Web 手札](#) [Netkiller Monitoring 手札](#) [Netkiller Storage 手札](#) [Netkiller Mail 手札](#)
[Netkiller Virtualization 手札](#) [Netkiller Cryptography 手札](#)

以下文档停止更新合并到 《Netkiller Linux 手札》

[Netkiller Debian 手札](#) [Netkiller CentOS 手札](#) [Netkiller Multimedia 手札](#)

致读者

Netkiller 系列手札 已经被 Github 收录，并备份保存在北极地下250米深的代码库中，备份会保留1000年。

Preserving open source software for future generations



The world is powered by open source software. It is a hidden cornerstone of modern civilization, and the shared heritage of all humanity.

The GitHub Arctic Code Vault is a data repository preserved in the Arctic World Archive (AWA), a very-long-term archival facility 250 meters deep in the permafrost of an Arctic mountain.

We are collaborating with the Bodleian Library in Oxford, the Bibliotheca Alexandrina in Egypt, and Stanford Libraries in California to store copies of 17,000 of GitHub’s most popular and most-depended-upon projects—open source’s “greatest hits”—in their archives, in museum-quality cases, to preserve them for future generations.

<https://archiveprogram.github.com/arctic-vault/>

1. 有趣的 Shell 应用

1.1. Ascii 星球大战电影

```
neo@MacBook-Pro-M2 ~> nc towel.blinkenlights.nl 23
```

1.2. 天气预报

neo@MacBook-Pro-M2 ~> curl http://wttr.in/ Weather report: Shenzhen, China			
\ / Partly cloudy _ /"-. 18 °C _(). \ 26 km/h /(__(_) 10 km 0.0 mm			
Mon 09 Jan			
Night	Morning	Noon	Evening
Overcast	Overcast	Overcast	Overcast
---. 17 °C	---. 21 °C	---. 19 °C	
---. 18 °C	---. 21 °C	---. 19 °C	
-(). ↓ 7-9 km/h	-(). ✓ 7-8 km/h	-(). ✓ 7-10 km/h	
-(). ✓ 7-10 km/h	-(). ✓ 7-8 km/h	-(). ✓ 7-10 km/h	
(__.__) 10 km	(__.__) 10 km	(__.__) 10 km	
(__.__) 10 km	(__.__) 10 km	(__.__) 10 km	
0.0 mm 0%	0.0 mm 0%	0.0 mm 0%	
0.0 mm 0%	0.0 mm 0%	0.0 mm 0%	
Tue 10 Jan			
Night	Morning	Noon	Evening
Overcast	Patchy rain po...	Patchy rain po...	Light drizzle
_, _(). 17 °C	_, _(). 18 °C	(). 17 °C	
---. 16 °C	_, _(). 18 °C	(). 17 °C	
/(__(_) ✓ 6-9 km/h	/(__(_) ✓ 8-9 km/h	(__(_) ✓ 9-13 km/h	
-(). ✓ 7-10 km/h	/(__(_) ✓ 8-9 km/h	(__(_) ✓ 9-13 km/h	
(__.__) 10 km	' ' ' ' 10 km	' ' ' ' 2 km	
(__.__) 10 km	' ' ' ' 10 km	' ' ' ' 2 km	
0.1 mm 61%	0.1 mm 81%	0.7 mm 83%	
0.0 mm 0%	0.1 mm 81%	0.7 mm 83%	
0.0 mm 0%	0.1 mm 81%	0.7 mm 83%	
Wed 11 Jan			
Night	Morning	Noon	Evening
Overcast	Patchy rain po...	Patchy rain po...	Partly cloudy
\ / Partly cloudy	_, _(). 16 °C	_, _(). 17 °C	\ / Partly cloudy
_, _(). 16 °C	_, _(). 16 °C	_, _(). 17 °C	_, _(). 17 °C
/"-. 17 °C	/(__(_) ✓ 5-7 km/h	/(__(_) ✓ 5-6 km/h	/"-. 18 °C
/(__(_) ✓ 5-7 km/h	/(__(_) ✓ 5-6 km/h	/(__(_) ✓ 5-6 km/h	/(__(_) ✓ 7-9 km/h
_(). ✓ 7-11 km/h	' ' ' ' 10 km	' ' ' ' 10 km	_(). ✓ 7-9 km/h
' ' ' ' 10 km	' ' ' ' 10 km	' ' ' ' 10 km	' ' ' ' 10 km
/(__(_) 10 km	' ' ' ' 10 km	' ' ' ' 10 km	/(__(_) 10 km
' ' ' ' 0.1 mm 87%	' ' ' ' 0.1 mm 84%	' ' ' ' 0.1 mm 84%	' ' ' ' 0.0 mm 0%
0.0 mm 0%	' ' ' ' 0.1 mm 84%	' ' ' ' 0.1 mm 84%	' ' ' ' 0.0 mm 0%
0.0 mm 0%	' ' ' ' 0.1 mm 84%	' ' ' ' 0.1 mm 84%	' ' ' ' 0.0 mm 0%

Follow @igor_chubin for wttr.in updates

自述

Netkiller 手札系列电子书 <http://www.netkiller.cn>

Netkiller Linux 手札

陈景峰 著



《Netkiller 系列 手札》是一套免费系列电子书，netkiller 是 nickname 从1999 开使用至今，“手札”是札记，手册的含义。

2003年之前我还是以文章形式在BBS上发表各类技术文章，后来发现文章不够系统，便尝试写长篇技术文章加上章节目录等等。随着内容增加，不断修订，开始发布第一版，第二版.....

IT知识变化非常快，而且具有时效性，这样发布非常混乱，经常有读者发现第一版例子已经过时，但他不知道我已经发布第二版。

我便有一种想法，始终维护一个文档，不断更新，使他保持较新的版本不过时。

第一部电子书是《PostgreSQL 实用实例参考》开始我使用 Microsoft Office Word 慢慢随着文档尺寸增加 word 开始表现出力不从心。

我看到PostgreSQL 中文手册使用SGML编写文档，便开始学习 Docbook SGML。使用Docbook写的第一部电子书是《Netkiller Postfix Integrated Solution》这是Netkiller 系列手札的原型。

至于“手札”一词的来历，是因为我爱好摄影，经常去一个台湾摄影网站，名字就叫“摄影家手札”。

由于硬盘损坏数据丢失 《Netkiller Postfix Integrated Solution》的 SGML文件已经不存在；Docbook SGML存在很多缺陷 UTF-8支持不好，转而使用Docbook XML。

目前技术书籍的价格一路飙升，动则¥80，¥100，少则¥50，¥60。技术书籍有时效性，随着技术的革新或淘汰，大批书籍成为废纸垃圾。并且这些书技术内容雷同，相互抄袭，质量越来越差，甚至里面给出的例子错误百出，只能购买影印版，或者翻译的版本。

在这种背景下我便萌生了自己写书的想法，资料主要来源是我的笔记与例子。我并不想出版，只为分享，所以我制作了基于CC License 发行的系列电子书。

本书注重例子，少理论（捞干货），只要你对着例子一步一步操作，就会成功，会让你有成就感并能坚持学下去，因为很多人遇到障碍就会放弃，其实我就是这种人，只要让他看到希望，就能坚持下去。

1. 写给读者

为什么写这篇文章

有很多想法,工作中也用不到所以未能实现,所以想写出来,和大家分享.有一点写一点,写得也不好,只要能看懂就行,就当学习笔记了.

开始零零碎碎写过一些文档,也向维基百科供过稿,但维基经常被ZF封锁,后来发现sf.net可以提供主机存放文档,便做了迁移.并开始了我的写作生涯.

这篇文档是作者20年来对工作的总结,是作者一点一滴的积累起来的,有些笔记已经丢失,所以并不完整.

因为工作太忙整理比较缓慢.目前的工作涉及面比较窄所以新文档比较少.

我现在花在技术上的时间越来越少,兴趣转向摄影,无线电.也想写写摄影方面的心得体会.

写作动力:

曾经在网上看到外国开源界对中国的评价,中国人对开源索取无度,但贡献却微乎其微.这句话一直记在我心中,发誓要为中国开源事业做我仅有的一点微薄贡献

另外写文档也是知识积累,还可以增加在圈内的影响力.

人跟动物的不同,就是人类可以把自己学习的经验教给下一代人.下一代在上一代的基础上再创新,不断积累才有今天.

所以我把自己的经验写出来,可以让经验传承

没有内容的章节:

目前我自己一人维护所有文档,写作时间有限,当我发现一个好主题就会加入到文档中,待我有时间再完善章节,所以你会发现很多章节是空无内容的.

文档目前几乎是流水帐式的写作,维护量很大,先将就着看吧.

我想到哪写到哪,你会发现文章没一个中心,今天这里写点,明天跳过本

章写其它的.

文中例子绝对多,对喜欢复制然后粘贴朋友很有用,不用动手写,也省时间.

理论的东西,网上大把,我这里就不写了,需要可以去网上查.

我爱写错别字,还有一些是打错的,如果发现请指正.

文中大部分试验是在Debian/Ubuntu/Redhat AS上完成.

写给读者

至读者:

我不知道什么时候,我不再更新文档或者退出IT行业去从事其他工作,我必须给这些文档找一个归宿,让他能持续更新下去。

我想捐赠给某些基金会继续运转,或者建立一个团队维护它。

我用了20年时间坚持不停地写作,持续更新,才有今天你看到的《Netkiller 手札》系列文档,在中国能坚持20年,同时没有任何收益的技术类文档,是非常不容易的。

有很多时候想放弃,看到外国读者的支持与国内社区的影响,我坚持了下来。

中国开源事业需要各位参与,不要成为局外人,不要让外国人说:中国对开源索取无度,贡献却微乎其微。

我们参与内核的开发还比较遥远,但是进个人能力,写一些文档还是可能的。

系列文档

下面是我多年积累下来的经验总结,整理成文档供大家参考:

[Netkiller Architect 手札](#)

[Netkiller Developer 手札](#)

[Netkiller PHP 手札](#)

[Netkiller Python 手札](#)

[Netkiller Testing 手札](#)

[Netkiller Cryptography 手札](#)

[Netkiller Linux 手札](#)
[Netkiller FreeBSD 手札](#)
[Netkiller Shell 手札](#)
[Netkiller Security 手札](#)
[Netkiller Web 手札](#)
[Netkiller Monitoring 手札](#)
[Netkiller Storage 手札](#)
[Netkiller Mail 手札](#)
[Netkiller Docbook 手札](#)
[Netkiller Version 手札](#)
[Netkiller Database 手札](#)
[Netkiller PostgreSQL 手札](#)
[Netkiller MySQL 手札](#)
[Netkiller NoSQL 手札](#)
[Netkiller LDAP 手札](#)
[Netkiller Network 手札](#)
[Netkiller Cisco IOS 手札](#)
[Netkiller H3C 手札](#)
[Netkiller Multimedia 手札](#)
[Netkiller Management 手札](#)
[Netkiller Spring 手札](#)
[Netkiller Perl 手札](#)
[Netkiller Amateur Radio 手札](#)

2. 作者简介

陈景峯 ([ネウチン](#))

Nickname: netkiller | English name: Neo chen | Nippon name: ちんけいほう (音訳) | Korean name: 천징봉 | Thailand name: ภูมิภาพภูเขา | Vietnam: Trần Cảnh Phong

Callsign: [BG7NYT](#) | QTH: ZONE CQ24 ITU44 ShenZhen, China

程序猿，攻城狮，挨踢民工，Full Stack Developer, UNIX like Evangelist, 业余无线电爱好者（呼号：BG7NYT），户外运动，山地骑行以及摄影爱好者。

《Netkiller 系列 手札》的作者

成长阶段

1981年1月19日(庚申年腊月十四)出生于黑龙江省青冈县建设乡双富大队第一小队

1989年9岁随父母迁居至黑龙江省伊春市，悲剧的天朝教育，不知道那门子归定，转学必须降一级，我本应该上一年级，但体制让我上学前班，那年多都10岁了

1995年小学毕业，体制规定借读要交3000两银子(我曾想过不升初中)，亲戚单位分楼告别平房，楼里没有地方放东西，把2麻袋书送给我，无意中发现一本电脑书BASIC语言，我竟然看懂了，对于电脑知识追求一发而不可收，后面顶零花钱，压岁钱主要用来买电脑书《MSDOS 6.22》《新编Unix实用大全》《跟我学Foxbase》。。。。。。

1996年第一次接触UNIX操作系统，BSD UNIX, Microsoft Xinux(盖茨亲自写的微软Unix，知道的人不多)

1997年自学Turbo C语言，苦于没有电脑，后来学校建了微机室才第一次使用QBASIC(DOS 6.22 自带命令)，那个年代只能通过软盘拷贝转播，Turbo C编译器始终没有搞到，

1997年第一次上Internet网速只有9600Bps, 当时全国兴起各种信息港域名格式是www.xxxx.info.net, 访问的第一个网站是NASA下载了很多火星探路者拍回的照片，还有“淞沪”sohu的前身

1998~2000年在哈尔滨学习计算机，充足的上机时间，但老师让我们练打字（明伦五笔/WT）打字不超过80个/每分钟还要强化训练，不过这个给我的键盘功夫打了好底。

1999年学校的电脑终于安装了光驱，在一张工具盘上终于找到了Turbo C, Borland C++与Quick Basic编译器，当时对VGA图形编程非常感兴趣，通过INT33中断控制鼠标，使用绘图函数模仿windows界面。还有操作UCDOS中文字库，绘制矢量与点阵字体。

2000年沉迷于Windows NT与Back Office各种技术，神马主域控制器，DHCP，WINS，IIS，域名服务器，Exchange邮件服务器，MS Proxy, NetMeeting...以及ASP+MS SQL开发；用56K猫下载了一张LINUX。ISO镜像，安装后我兴奋的24小时没有睡觉。

职业生涯

2001年来深圳进城打工,成为一名外来务工者. 在一个4人公司做PHP开发，当时PHP的版本是2.0, 开始使用Linux Redhat 6.2.当时很多门户网站都是用FreeBSD,但很难搞到安装盘，在网易社区认识了一个网友,从广州给我寄了一张光盘，FreeBSD 3.2

2002年我发现不能埋头苦干,还要学会"做人".后辗转广州工作了半年，考了一个Cisco CCNA认证。回到深圳重新开始，在车公庙找到一家工作做Java开发

2003年这年最惨,公司拖欠工资16000元,打过两次官司2005才付清.

2004 年开始加入[分布式计算](#)团队,[目前成绩](#), 工作仍然是Java开发并且开始使用PostgreSQL数据库。

2004-10月开始玩户外和摄影

2005-6月成为中国无线电运动协会会员,呼号BG7NYT,进了一部Yaesu FT-60R手台。公司的需要转回PHP与MySQL, 相隔几年发现PHP进步很大。在前台展现方面无人能敌, 于是便前台使用PHP, 后台采用Java开发。

2006 年单身生活了这么多年,终于找到归宿. 工作更多是研究PHP各种框架原理

2007 物价上涨,金融危机, 休息了4个月 (其实是找不到工作), 关外很难上439.460中继, 搞了一台Yaesu FT-7800.

2008 终于找到英文学习方法, 《Netkiller Developer 手札》, 《Netkiller Document 手札》

2008-8-8 08:08:08 结婚,后全家迁居湖南省常德市

2009 《Netkiller Database 手札》,2009-6-13学车, 年底拿到C1驾照

2010 对电子打击乐产生兴趣, 计划学习爵士鼓。由于我对Linux热爱, 我轻松的接管了公司的运维部, 然后开发运维两把抓。我印象最深刻的是公司一次上架10个机柜, 我们用买服务器纸箱的钱改善伙食。我将40多台服务器安装BOINC做压力测试, 获得了中国第二的名次。

2011 平凡的一年, 户外运动停止, 电台很少开, 中继很少上, 摄影主要是拍女儿与家人, 年末买了一辆山地车

2012 对油笔画产生了兴趣, 活动基本是骑行银湖山绿道,

2013 开始学习民谣吉他, 同时对电吉他也极有兴趣; 最终都放弃了。这一年深圳开始推数字中继2013-7-6日入手Motorola

MOTOTRBO XIR P8668, Netkiller 系列手札从Sourceforge向Github迁移; 年底对MYSQL UDF, Engine与PHP扩展开发产生很浓的兴趣, 拾起遗忘10+年的C, 写了几个mysql扩展(图片处理, fifo管道与ZeroMQ), 10月份入Toyota Rezi 2.5V并写了一篇《攻城狮的苦逼选车经历》

2014-9-8 在淘宝上买了一架电钢琴 Casio Privia PX-5S pro 开始陪女儿学习钢琴, 由于这家钢琴是合成器电钢, 里面有打击乐, 我有对键盘鼓产生了兴趣。

2014-10-2号罗浮山两日游, 对中国道教文化与音乐产生了兴趣, 10月5号用了半天时间学会了简谱。10月8号入Canon 5D Mark III + Canon Speedlite 600EX-RT香港过关被查。

2014-12-20号对乐谱制作产生兴趣
(<https://github.com/SheetMusic/Piano>), 给女儿做了几首钢琴伴奏曲, MuseScore制谱然后生成MIDI与WAV文件。

2015-09-01 晚饭后拿起爵士鼓基础教程尝试在Casio Privia PX-5S pro演练, 经过反复琢磨加上之前学钢琴的乐理知识, 终于在02号晚上, 打出了简单的基本节奏, 迈出了第一步。

2016 对弓箭(复合弓)产生兴趣, 无奈天朝法律法规不让玩。每周游泳轻松1500米无压力, 年底入 xbox one s 和 Yaesu FT-2DR, 同时开始关注功放音响这块

2017 7月9号入 Yamaha RX-V581 功放一台, 连接Xbox打游戏爽翻了, 入Kindle电子书, 计划学习蝶泳, 果断放弃运维和开发知识体系转攻区块链。

2018 从溪山美地搬到半岛城邦, 丢弃了多年攒下的家底。11月开始玩 MMDVM, 使用 Yaesu FT-7800 发射, 连接MMDVM中继板, 树莓派, 覆盖深圳湾, 散步骑车通联两不误。

2019 卖了常德的房子, 住了5次院, 哮喘反复发作, 决定停止电子书更新, 兴趣转到知乎, B站

2020 准备找工作

职业生涯路上继续打怪升级

3. 如何获得文档

下载 Netkiller 手札 (epub,kindle,chg,pdf)

EPUB <https://github.com/netkiller/netkiller.github.io/tree/master/download/epub>

MOBI <https://github.com/netkiller/netkiller.github.io/tree/master/download/mobi>

PDF <https://github.com/netkiller/netkiller.github.io/tree/master/download/pdf>

CHM <https://github.com/netkiller/netkiller.github.io/tree/master/download/chm>

通过 GIT 镜像整个网站

<https://github.com/netkiller/netkiller.github.com.git>

```
$ git clone https://github.com/netkiller/netkiller.github.com.git
```

镜像下载

整站下载

```
wget -m http://www.netkiller.cn/index.html
```

指定下载

```
wget -m wget -m http://www.netkiller.cn/linux/index.html
```

Yum 下载文档

获得光盘介质，RPM包，DEB包，如有特别需要，请联系我

YUM 在线安装电子书

<http://netkiller.sourceforge.net/pub/repo/>

```
# cat >> /etc/yum.repos.d/netkiller.repo <<EOF
[netkiller]
name=Netkiller Free Books
```

```
baseurl=http://netkiller.sourceforge.net/pub/repo/  
enabled=1  
gpgcheck=0  
gpgkey=  
EOF
```

查找包

```
# yum search netkiller  
  
netkiller-centos.x86_64 : Netkiller centos Cookbook  
netkiller-cryptography.x86_64 : Netkiller cryptography Cookbook  
netkiller-docbook.x86_64 : Netkiller docbook Cookbook  
netkiller-linux.x86_64 : Netkiller linux Cookbook  
netkiller-mysql.x86_64 : Netkiller mysql Cookbook  
netkiller-php.x86_64 : Netkiller php Cookbook  
netkiller-postgresql.x86_64 : Netkiller postgresql Cookbook  
netkiller-python.x86_64 : Netkiller python Cookbook  
netkiller-version.x86_64 : Netkiller version Cookbook
```

安装包

```
yum install netkiller-docbook
```


4. 打赏 (Donations)

If you like this documents, please make a donation to support the authors' efforts. Thank you!

您可以通过微信，支付宝，贝宝给作者打赏。

银行(Bank)

招商银行(China Merchants Bank)

开户名：陈景峰

账号：9555500000007459

微信 (Wechat)



支付宝 (Alipay)



PayPal Donations

<https://www.paypal.me/netkiller>

5. 联系方式

主站 <http://www.netkiller.cn/>

备用 <http://netkiller.github.io/>

繁体网站 <http://netkiller.sourceforge.net/>

联系作者

Mobile: +86 13113668890

Email: netkiller@msn.com

QQ群: 128659835 请注明“读者”

QQ: 13721218

ICQ: 101888222

注：请不要问我安装问题！

博客 Blogger

知乎专栏 <https://zhuanlan.zhihu.com/netkiller>

LinkedIn: <http://cn.linkedin.com/in/netkiller>

OSChina: <http://my.oschina.net/neochen/>

Facebook: <https://www.facebook.com/bg7nyt>

Flickr: <http://www.flickr.com/photos/bg7nyt/>

Disqus: <http://disqus.com/netkiller/>

solidot: <http://solidot.org/~netkiller/>

SegmentFault: <https://segmentfault.com/u/netkiller>

Reddit: <https://www.reddit.com/user/netkiller/>

Digg: <http://www.digg.com/netkiller>

Twitter: <http://twitter.com/bg7nyt>

weibo: <http://weibo.com/bg7nyt>

Xbox club

我的 xbox 上的ID是 netkiller xbox，我创建了一个俱乐部
netkiller 欢迎加入。

Radio

CQ CQ CQ DE BG7NYT:

如果这篇文章对你有所帮助,请寄给我一张QSL卡片, qrz.cn or
qrz.com or hamcall.net

Personal Amateur Radiostations of P.R.China

ZONE CQ24 ITU44 ShenZhen, China

Best Regards, VY 73! OP. BG7NYT

守听频率 DMR 438.460 -8 Color 12 Slot 2 Group 46001

守听频率 C4FM 439.360 -5 DN/VW

MMDVM Hotspot:

Callsign: BG7NYT QTH: Shenzhen, China

YSF: YSF80337 - CN China 1 - W24166/TG46001

DMR: BM_China_46001 - DMR Radio ID: 4600441

第 1 章 Shell

1. 快捷键

```
Ctrl+p  shell中上一个命令,或者 文本中移动到上一行
Ctrl+n  shell中下一个命令,或者 文本中移动到下一行
Ctrl+r  往后搜索历史命令
Ctrl+s  往前搜索历史命令
Ctrl+f  光标前移
Ctrl+b  光标后退
Ctrl+a  到行首
Ctrl+e  到行尾
Ctrl+d  删除一个字符,删除一个字符, 相当于通常的Delete键
Ctrl+h  退格删除一个字符,相当于通常的Backspace键
Ctrl+u  删除到行首
Ctrl+k  删除到行尾
Ctrl+l  类似 clear 命令效果
Ctrl+y  粘贴
```

命令行编辑命令

```
Ctrl + a : 移到命令行首
Ctrl + e : 移到命令行尾
Ctrl + f : 按字符前移 (右向)
Ctrl + b : 按字符后移 (左向)
Alt + f : 按单词前移 (右向)
Alt + b : 按单词后移 (左向)
Ctrl + xx: 在命令行首和光标之间移动
Ctrl + u : 从光标处删除至命令行首
Ctrl + k : 从光标处删除至命令行尾
Ctrl + w : 从光标处删除至字首
Alt + d : 从光标处删除至字尾
Ctrl + d : 删除光标处的字符
Ctrl + h : 删除光标前的字符
```

Ctrl + y : 粘贴至光标后
Alt + c : 从光标处更改为首字母大写的单词
Alt + u : 从光标处更改为全部大写的单词
Alt + l : 从光标处更改为全部小写的单词
Ctrl + t : 交换光标处和之前的字符
Alt + t : 交换光标处和之前的单词
Alt + Backspace: 与 Ctrl + w 相同类似, 分隔符有些差别

重新执行命令快捷键

Ctrl + r: 逆向搜索命令历史
Ctrl + g: 从历史搜索模式退出
Ctrl + p: 历史中的上一条命令
Ctrl + n: 历史中的下一条命令
Alt + .: 使用上一条命令的最后一个参数

终端控制快捷键

Ctrl + l: 清屏
Ctrl + o: 执行当前命令, 并选择上一条命令
Ctrl + s: 阻止屏幕输出
Ctrl + q: 允许屏幕输出
Ctrl + c: 终止命令
Ctrl + z: 挂起命令

Bang (!) 命令

!!: 执行上一条命令
!blah: 执行最近的以 blah 开头的命令, 如 !ls
!blah:p: 仅打印输出, 而不执行

2. chsh - change login shell

```
# chsh --list
/bin/sh
/bin/bash
/sbin/nologin
/bin/tcsh
/bin/csh
/bin/ksh

# chsh --list-shells
/bin/sh
/bin/bash
/sbin/nologin
/bin/dash
/bin/zsh
```

```
$ chsh -s /bin/zsh
or
$ usermod -s /bin/zsh
```

show me current shell

```
neo@netkiller:~$ echo $SHELL
/bin/zsh

neo@netkiller:~$ cat /etc/passwd|grep neo
neo:x:1000:1000:Neo Chen,,,:/home/neo:/bin/zsh
```


3. 执行程序返回值

```
"OS error code 1: Operation not permitted"
"OS error code 2: No such file or directory"
"OS error code 3: No such process"
"OS error code 4: Interrupted system call"
"OS error code 5: Input/output error"
"OS error code 6: No such device or address"
"OS error code 7: Argument list too long"
"OS error code 8: Exec format error"
"OS error code 9: Bad file descriptor"
"OS error code 10: No child processes"
"OS error code 11: Resource temporarily unavailable"
"OS error code 12: Cannot allocate memory"
"OS error code 13: Permission denied"
"OS error code 14: Bad address"
"OS error code 15: Block device required"
"OS error code 16: Device or resource busy"
"OS error code 17: File exists"
"OS error code 18: Invalid cross-device link"
"OS error code 19: No such device"
"OS error code 20: Not a directory"
"OS error code 21: Is a directory"
"OS error code 22: Invalid argument"
"OS error code 23: Too many open files in system"
"OS error code 24: Too many open files"
"OS error code 25: Inappropriate ioctl for device"
"OS error code 26: Text file busy"
"OS error code 27: File too large"
"OS error code 28: No space left on device"
"OS error code 29: Illegal seek"
"OS error code 30: Read-only file system"
"OS error code 31: Too many links"
"OS error code 32: Broken pipe"
"OS error code 33: Numerical argument out of domain"
"OS error code 34: Numerical result out of range"
"OS error code 35: Resource deadlock avoided"
"OS error code 36: File name too long"
"OS error code 37: No locks available"
"OS error code 38: Function not implemented"
"OS error code 39: Directory not empty"
```

"OS error code 40: Too many levels of symbolic links"
"OS error code 42: No message of desired type"
"OS error code 43: Identifier removed"
"OS error code 44: Channel number out of range"
"OS error code 45: Level 2 not synchronized"
"OS error code 46: Level 3 halted"
"OS error code 47: Level 3 reset"
"OS error code 48: Link number out of range"
"OS error code 49: Protocol driver not attached"
"OS error code 50: No CSI structure available"
"OS error code 51: Level 2 halted"
"OS error code 52: Invalid exchange"
"OS error code 53: Invalid request descriptor"
"OS error code 54: Exchange full"
"OS error code 55: No anode"
"OS error code 56: Invalid request code"
"OS error code 57: Invalid slot"
"OS error code 59: Bad font file format"
"OS error code 60: Device not a stream"
"OS error code 61: No data available"
"OS error code 62: Timer expired"
"OS error code 63: Out of streams resources"
"OS error code 64: Machine is not on the network"
"OS error code 65: Package not installed"
"OS error code 66: Object is remote"
"OS error code 67: Link has been severed"
"OS error code 68: Advertise error"
"OS error code 69: Srmount error"
"OS error code 70: Communication error on send"
"OS error code 71: Protocol error"
"OS error code 72: Multihop attempted"
"OS error code 73: RFS specific error"
"OS error code 74: Bad message"
"OS error code 75: Value too large for defined data type"
"OS error code 76: Name not unique on network"
"OS error code 77: File descriptor in bad state"
"OS error code 78: Remote address changed"
"OS error code 79: Can not access a needed shared library"
"OS error code 80: Accessing a corrupted shared library"
"OS error code 81: .lib section in a.out corrupted"
"OS error code 82: Attempting to link in too many shared
libraries"
"OS error code 83: Cannot exec a shared library directly"
"OS error code 84: Invalid or incomplete multibyte or wide
character"

"OS error code 85: Interrupted system call should be restarted"
"OS error code 86: Streams pipe error"
"OS error code 87: Too many users"
"OS error code 88: Socket operation on non-socket"
"OS error code 89: Destination address required"
"OS error code 90: Message too long"
"OS error code 91: Protocol wrong type for socket"
"OS error code 92: Protocol not available"
"OS error code 93: Protocol not supported"
"OS error code 94: Socket type not supported"
"OS error code 95: Operation not supported"
"OS error code 96: Protocol family not supported"
"OS error code 97: Address family not supported by protocol"
"OS error code 98: Address already in use"
"OS error code 99: Cannot assign requested address"
"OS error code 100: Network is down"
"OS error code 101: Network is unreachable"
"OS error code 102: Network dropped connection on reset"
"OS error code 103: Software caused connection abort"
"OS error code 104: Connection reset by peer"
"OS error code 105: No buffer space available"
"OS error code 106: Transport endpoint is already connected"
"OS error code 107: Transport endpoint is not connected"
"OS error code 108: Cannot send after transport endpoint shutdown"
"OS error code 109: Too many references: cannot splice"
"OS error code 110: Connection timed out"
"OS error code 111: Connection refused"
"OS error code 112: Host is down"
"OS error code 113: No route to host"
"OS error code 114: Operation already in progress"
"OS error code 115: Operation now in progress"
"OS error code 116: Stale NFS file handle"
"OS error code 117: Structure needs cleaning"
"OS error code 118: Not a XENIX named type file"
"OS error code 119: No XENIX semaphores available"
"OS error code 120: Is a named type file"
"OS error code 121: Remote I/O error"
"OS error code 122: Disk quota exceeded"
"OS error code 123: No medium found"
"OS error code 124: Wrong medium type"
"OS error code 125: Operation canceled"
"OS error code 126: Required key not available"
"OS error code 127: Key has expired"
"OS error code 128: Key has been revoked"

```
"OS error code 129: Key was rejected by service"  
"OS error code 130: Owner died"  
"OS error code 131: State not recoverable"
```

第 2 章 Bash Shell

1. bash - GNU Bourne-Again SHell

-n 检查脚本是否有语法错误



-x 显示详细运行过程



2. 切换身份

判断当前用户是否为root

```
#!/bin/bash
if [[ $EUID -ne 0 ]]; then
    echo "This script must be run as root"
    exit 1
fi
```

使用 `#!/bin/su` 可以切换当前shell的所有者，全局切换

```
# cat test.sh

#!/bin/su www
ls
```

局部切换，运行\$PROG后将pid（进程ID）写入\$PIDFILE文件

```
su - $USER -c "$PROG & echo \${!} > $PIDFILE"
```

3. I/O 重定向

```
cat <<End-of-message
  8 -----
  9 This is line 1 of the message.
10 This is line 2 of the message.
11 This is line 3 of the message.
12 This is line 4 of the message.
13 This is the last line of the message.
14 -----
End-of-message
```

```
MYSQL=mysql
MYSQLOPTS="-h $zs_host -u $zs_user -p$zs_pass $zs_db"

$MYSQL $MYSQLOPTS <<SQL
SELECT
    category.cat_id AS cat_id ,
    category.cat_name AS cat_name ,
    category.cat_desc AS cat_desc ,
    category.parent_id AS parent_id ,
    category.sort_order AS sort_order ,
    category.measure_unit AS measure_unit ,
    category.style AS style ,
    category.is_show AS is_show ,
    category.grade AS grade
FROM category
SQL
```

<<-LimitString可以抑制输出时前边的tab(不是空格). 这可以增加一个脚本的可读性.

```
cat <<-ENDOFMESSAGE
    This is line 1 of the message.
    This is line 2 of the message.
    This is line 3 of the message.
    This is line 4 of the message.
    This is the last line of the message.
ENDOFMESSAGE
```

关闭参数替换

```
NAME="John Doe"
RESPONDENT="the author of this fine script"

cat <<'Endofmessage'

Hello, there, $NAME.
Greetings to you, $NAME, from $RESPONDENT.

Endofmessage
```

```
NAME="John Doe"
RESPONDENT="the author of this fine script"

cat <<\Endofmessage

Hello, there, $NAME.
Greetings to you, $NAME, from $RESPONDENT.

Endofmessage
```

stdout


```
$ ln -s /dev/stdout test
$ cat file > test
```

error 重定向

```
your_shell 2>&1
```

错误输出演示

```
[root@localhost ~]# id ethereum
id: ethereum: no such user

# 这里可以看到错误输出 id: ethereum: no such user
```

```
[root@localhost ~]# id ethereum > test
id: ethereum: no such user
```

我们尝试将他重定向到文件 test，但是结果仍是输出 id: ethereum: no such user

```
[root@localhost ~]# cat test
[root@localhost ~]#
```

查看 test 文件，内容空。

继续做实验

```
[root@localhost ~]# id ethereum > test 2>&1
[root@localhost ~]# cat test
id: ethereum: no such user
```

测试实验结果成功了，将错误输出转到标准输出，然后写入文件。

使用块记录日志

```
{  
    ...  
    ...  
} > $LOGFILE 2>&1
```

tee - read from standard input and write to standard output and files

```
echo 1 > /proc/sys/net/ipv4/ip_forward  
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward;
```

重定向到文件

```
sudo mkdir -p /etc/docker  
sudo tee /etc/docker/daemon.json <<-'EOF'  
{  
    "registry-mirrors": ["https://du8clin9.mirror.aliyuncs.com"]  
}  
EOF  
sudo systemctl daemon-reload  
sudo systemctl restart docker
```

nettee - a network "tee" program

创建文件

```
cat << EOF > foo.sh
    printf "%s was here" "$name"
EOF

cat >> foo.sh <<EOF
    printf "%s was here" "$name"
EOF
```

快速清空一个文件的内容

```
$ > /www/access.log
```

4. pipes (FIFOs)

create a pipes

```
$ mkfifo /tmp/pipe
$ mkfifo -m 0644 /tmp/pipe
$ mknod /tmp/pipe p
```

let's see it

```
$ ls -l /tmp/piple
prw-r--r-- 1 neo neo 0 2009-03-13 14:40 /tmp/piple
```

remove a pipes

```
rm /tmp/pipe
```

using it

standing by pipe

```
$ cat /tmp/pipe
```

push string to pipe

```
$ echo hello world > /tmp/pipe
```

fetch string from /tmp/pipe

```
$ cat /tmp/piple  
hello world
```

5. mktemp - create a temporary file or directory 临时目录与文件

```
# mktemp  
/tmp/tmp.p8p0v5YzPf  
  
# mktemp /tmp/test.XXX  
/tmp/test.d8J  
  
# mktemp /tmp/test.XXXXXXX  
/tmp/test.cFebDX  
  
# mktemp /tmp/test.XXXXXXXX  
/tmp/test.CnyLr7C
```

创建临时目录

```
# mktemp -d  
/tmp/tmp.xg5gFj0w8D  
  
# mktemp -d --suffix=.tmp /tmp/test.XXXXXX  
/tmp/test.TDpz8.tmp  
  
$ mktemp -d --suffix=.tmp -p /tmp deploy.XXXXXXX  
/tmp/deploy.FwebCc.tmp
```

6. History 命令历史记录

.bash_history

从安全角度考虑禁止记录history

```
ln -s /dev/null .bash_history
```

格式定义

定制.bash_history格式

```
export HISTSIZE=1000
export HISTFILESIZE=2000
export HISTTIMEFORMAT="%Y-%m-%d-%H:%M:%S "
export HISTFILE=~/.bash_history"
```

看看实际效果

```
$ history | head
 1 2012-02-27-09:10:45 do-release-upgrade
 2 2012-02-27-09:10:45 vim /etc/network/interfaces
 3 2012-02-27-09:10:45 vi /etc/network/interfaces
 4 2012-02-27-09:10:45 ping www.163.com
```

提示

CentOS 可以添加到 /etc/bashrc 这样可以对所有用户起作用

```
echo 'export HISTTIMEFORMAT="%Y-%m-%d-%H:%M:%S "' >>
/etc/bashrc
```

设置忽略命令

HISTIGNORE 可以设置那些命令不记入history列表。

```
HISTIGNORE="ls:ll:la:cd:exit:clear:logout"  
HISTTIMEFORMAT "[%Y-%m-%d - %H:%M:%S] "  
HISTFILE=~/.history  
HISTSIZE=50000  
SAVEHIST=50000
```

.mysql_history

```
ln -s /dev/null .mysql_history
```

插入时间点，在~/.bashrc中加入下面命令

```
$ tail ~/.bashrc  
echo `date` >> ~/.mysql_history
```

```
$ tail ~/.mysql_history  
EXPLAIN SELECT * FROM stuff where id=3 \G  
EXPLAIN SELECT * FROM stuff where id='3' \G  
EXPLAIN SELECT * FROM stuff where id='2' \G  
Mon Feb 27 09:15:18 CST 2012  
EXPLAIN SELECT * FROM stuff where id='2' and created = '2012-02-01' \G  
EXPLAIN SELECT * FROM stuff where id='1' and created = '2012-02-01' \G  
EXPLAIN SELECT * FROM stuff where id='3' and created = '2012-02-01' \G  
EXPLAIN SELECT * FROM stuff where id='2' and created = '2012-02-01' \G  
EXPLAIN SELECT * FROM stuff where id='2' or created = '2012-02-01' \G  
EXPLAIN SELECT * FROM stuff where id='2' and created = '2012-02-01' \G
```


Mon Feb 27 11:48:37 CST 2012

7. hash - hash database access method

hase 命令：用来显示和清除哈希表，执行命令的时候，系统将先查询哈希表。

当你输入命令，首先在hash表中寻找，如果不存在，才会利用\$PATH环境变量指定的路径寻找命令，然后加以执行。同时也会将其放入到hash table 中，当下一次执行同样的命令时就不会再通过\$PATH寻找。以此提高命令的执行效率。

显示哈希表中命令使用频率

```
$ hash
hits    command
  6      /usr/bin/svn
  1      /bin/chown
  3      /bin/bash
  4      /usr/bin/git
 12      /usr/bin/php
  1      /bin/rm
  1      /bin/chmod
  1      /usr/bin/nmap
  5      /bin/cat
 13      /usr/bin/vim
  3      /usr/bin/sudo
  4      /bin/sed
  2      /bin/ps
  2      /usr/bin/man
 23      /bin/ls
```

显示哈希表

```
$ hash -l
builtin hash -p /usr/bin/svn svn
builtin hash -p /bin/chown chown
builtin hash -p /bin/bash bash
builtin hash -p /usr/bin/git git
builtin hash -p /usr/bin/php php
```

```
builtin hash -p /bin/rm rm
builtin hash -p /bin/chmod chmod
builtin hash -p /usr/bin/nmap nmap
builtin hash -p /bin/cat cat
builtin hash -p /usr/bin/vim vim
builtin hash -p /usr/bin/sudo sudo
builtin hash -p /bin/sed sed
builtin hash -p /bin/ps ps
builtin hash -p /usr/bin/man man
builtin hash -p /bin/ls ls
```

显示命令的完整路径

```
$ hash -t git
/usr/bin/git
```

向哈希表中增加内容

```
$ hash -p /home/www/deployment/run run

$ run
Usage: /home/www/deployment/run [OPTION] <server-id>
<directory/timepoint>

OPTION:
    development <domain> <host>
    testing <domain> <host>
    production <domain> <host>

    branch {development|testing|production} <domain> <host>
<branchname>
    revert {development|testing|production} <domain> <host>
<revision>
    backup <domain> <host> <directory>
    release <domain> <host> <tags> <message>

    list
    list <domain> <host>
```

```
clean {development|testing|production} <domain> <host>  
log <project> <line>  
  
conf list  
cron show  
cron setup  
cron edit
```

命令等同于

```
PATH=$PATH:$HOME/www/deployment  
  
export PATH
```

删除哈希表内容

```
$ hash -r  
  
$ hash -l  
hash: hash table empty
```

8. prompt

.bashrc

```
# Prompt definitions
if [ -f ~/.bash_prompt ]; then
    . ~/.bash_prompt
fi
```

.bash_prompt

```
#!/bin/bash

function tonka2 {
local GRAY="\[\033[1;30m\]"
local LIGHT_GRAY="\[\033[0;37m\]"
local WHITE="\[\033[1;37m\]"

local LIGHT_BLUE="\[\033[1;34m\]"
local LIGHT_RED="\[\033[1;31m\]"
local YELLOW="\[\033[1;33m\]"

case $TERM in
    xterm*)
        TITLEBAR='\[\033]0;\u@\h:\w\007\]'
        ;;
    *)
        TITLEBAR=""
        ;;
esac

PS1="$TITLEBAR\
$YELLOW-$LIGHT_BLUE-(\
$YELLOW\u$LIGHT_BLUE@$YELLOW\h\
$LIGHT_BLUE)-(\
$YELLOW\$PWD\
$LIGHT_BLUE)-$YELLOW-\
$LIGHT_GRAY\n\
$YELLOW-$LIGHT_BLUE-(\
$YELLOW\$(date +%F)$LIGHT_BLUE:$YELLOW\$(date +%I:%M:%S)\\"
```

```
$LIGHT_BLUE:$WHITE\$$LIGHT_BLUE)-$YELLOW-$LIGHT_GRAY "
```

```
PS2="$LIGHT_BLUE-$YELLOW-$YELLOW-$LIGHT_GRAY "  
}
```

```
function proml {  
local BLUE="\[\033[0;34m\  
local RED="\[\033[0;31m\  
local LIGHT_RED="\[\033[1;31m\  
local WHITE="\[\033[1;37m\  
local NO_COLOUR="\[\033[0m\  
case $TERM in  
    xterm*|rxvt*)  
        TITLEBAR='\[\033]0;\u@\h:\w\007\  
        ;;  
    *)  
        TITLEBAR=""  
        ;;  
esac
```

```
PS1="${TITLEBAR}\  
$BLUE[$RED\$(date +%H%M)$BLUE]\  
$BLUE[$LIGHT_RED\u@\h:\w$BLUE]\  
$WHITE\$$NO_COLOUR "  
PS2='> '  
PS4='+ '  
}
```

```
function neo_prompt {  
local GRAY="\[\033[1;30m\  
local LIGHT_GRAY="\[\033[0;37m\  
local WHITE="\[\033[1;37m\  
  
local LIGHT_BLUE="\[\033[1;34m\  
local LIGHT_RED="\[\033[1;31m\  
local YELLOW="\[\033[1;33m\  

```

```
case $TERM in  
    xterm*)  
        TITLEBAR='\[\033]0;\u@\h:\w\007\  
        ;;  
    *)  
        TITLEBAR=""  
        ;;  
esac
```



```

#-----
#
#   Created August 98, Last Modified 9 November 98 by Giles
#
#   Problem: when load is going down, it says "1.35down-.08",
get rid
#   of the negative

function prompt_command
{
#   Create TotalMeg variable: sum of visible file sizes in
current directory
local TotalBytes=0
for Bytes in $(ls -l | grep "^-" | awk '{print $5}')
do
    let TotalBytes=$TotalBytes+$Bytes
done
TotalMeg=$(echo -e "scale=3 \nx=$TotalBytes/1048576\n if (x<1)
{print \"0\"} \n print x \nquit" | bc)

#   This is used to calculate the differential in load values
#   provided by the "uptime" command. "uptime" gives load
#   averages at 1, 5, and 15 minute marks.
#
local one=$(uptime | sed -e "s/.*load average: \(.*\...\), \
(.*\...\), \(.*\...\)/\1/" -e "s/ //g")
local five=$(uptime | sed -e "s/.*load average: \(.*\...\), \
(.*\...\), \(.*\...\).*/\2/" -e "s/ //g")
local diff1_5=$(echo -e "scale = scale ($one) \nx=$one - $five\n
if (x>0) {print \"up\"} else {print \"down\"}\n print x \nquit
\n" | bc)
loaddiff=$(echo -n "${one}${diff1_5}")

#   Count visible files:
let files=$(ls -l | grep "^-" | wc -l | tr -d " ")
let hiddenfiles=$(ls -l -d .* | grep "^-" | wc -l | tr -d " ")
let executables=$(ls -l | grep ^-..x | wc -l | tr -d " ")
let directories=$(ls -l | grep "^d" | wc -l | tr -d " ")
let hiddendirectories=$(ls -l -d .* | grep "^d" | wc -l | tr -d
" ") -2
let linktemp=$(ls -l | grep "^l" | wc -l | tr -d " ")
if [ "$linktemp" -eq "0" ]
then
    links=""
else

```



```

    links=" ${linktemp}l"
fi
unset linktemp
let devicetemp=$(ls -l | grep "^[bc]" | wc -l | tr -d " ")
if [ "$devicetemp" -eq "0" ]
then
    devices=""
else
    devices=" ${devicetemp}bc"
fi
unset devicetemp
}

PROMPT_COMMAND=prompt_command

function pprom2 {

local          BLUE="\[\033[0;34m\"
local  LIGHT_GRAY="\[\033[0;37m\"
local  LIGHT_GREEN="\[\033[1;32m\"
local  LIGHT_BLUE="\[\033[1;34m\"
local  LIGHT_CYAN="\[\033[1;36m\"
local          YELLOW="\[\033[1;33m\"
local          WHITE="\[\033[1;37m\"
local          RED="\[\033[0;31m\"
local  NO_COLOUR="\[\033[0m\"

case $TERM in
    xterm*)
        TITLEBAR='\[\033]0;\u@\h:\w\007\]'
        ;;
    *)
        TITLEBAR=""
        ;;
esac

PS1="$TITLEBAR\
$BLUE[$RED\$(date +%H%M)$BLUE]\
$BLUE[$RED\u@\h$BLUE]\
$BLUE[\
$LIGHT_GRAY\${files}.\${hiddenfiles}-\
$LIGHT_GREEN\${executables}x \
$LIGHT_GRAY(\${TotalMeg}Mb) \
$LIGHT_BLUE\${directories}.\
\${hiddendirectories}d\

```

```

$LIGHT_CYAN\${links}\
$YELLOW\${devices}\
$BLUE]\
$BLUE[\${WHITE}\${loaddiff}$BLUE]\
$BLUE[\
$WHITE\$(ps ax | wc -l | sed -e \"s: ::g\")proc\
$BLUE]\
\n\
$BLUE[$RED\${PWD}$BLUE]\
$WHITE\$\
\
$NO_COLOUR "
PS2='> '
PS4='+ '
}

```

例 2.2. A Prompt the Width of Your Term

```

#!/bin/bash
#   termwide prompt with tty number
#       by Giles - created 2 November 98, last tweaked 31 July
2001
#
#       This is a variant on "termwide" that incorporates the tty
number.
#
hostnam=$(hostname -s)
usernam=$(whoami)
temp="$(tty)"
#   Chop off the first five chars of tty (ie /dev/):
cur_tty="${temp:5}"
unset temp

function prompt_command {

#   Find the width of the prompt:
TERMWIDTH=${COLUMNS}

#   Add all the accessories below ...
local temp="--(${usernam}@${hostnam}):${cur_tty})---(${PWD})--"

```

```

let fillsize=${TERMWIDTH}-${#temp}
if [ "$fillsize" -gt "0" ]
then
    fill="-----"
    -----
    -----"
    #   It's theoretically possible someone could need more
    #   dashes than above, but very unlikely!  HOWTO users,
    #   the above should be ONE LINE, it may not cut and
    #   paste properly
    fill="${fill:0:${fillsize}}"
    newPWD="${PWD}"
fi

if [ "$fillsize" -lt "0" ]
then
    fill=""
    let cut=3-${fillsize}
    newPWD="...${PWD:${cut}}"
fi
}

PROMPT_COMMAND=prompt_command

function twtty {

local WHITE="\[\033[1;37m\"
local NO_COLOUR="\[\033[0m\"

local LIGHT_BLUE="\[\033[1;34m\"
local YELLOW="\[\033[1;33m\"

case $TERM in
    xterm*|rxvt*)
        TITLEBAR='\[\033]0;\u@\h:\w\007\]'
        ;;
    *)
        TITLEBAR=""
        ;;
esac

PS1="$TITLEBAR\
$YELLOW-$LIGHT_BLUE-(\
$YELLOW\$usernam$LIGHT_BLUE@$YELLOW\$hostnam$LIGHT_BLUE:$WHITE\$
cur_tty\
${LIGHT_BLUE})-${YELLOW}-\${fill}${LIGHT_BLUE}-(\

```

```

$YELLOW\${newPWD}\
$LIGHT_BLUE)-$YELLOW-\
\n\
$YELLOW-$LIGHT_BLUE-(\
$YELLOW\$(date +%H%M)$LIGHT_BLUE:$YELLOW\$(date \"+%a,%d %b
%Y\")\
$LIGHT_BLUE:$WHITE\$$LIGHT_BLUE)-\
$YELLOW-\
$NO_COLOUR "

PS2="$LIGHT_BLUE-$YELLOW-$YELLOW-$NO_COLOUR "

}

```

例 2.3. The Elegant Useless Clock Prompt

```

#!/bin/bash

# This prompt requires a VGA font. The prompt is anchored at
# the bottom
# of the terminal, fills the width of the terminal, and draws
# a line up
# the right side of the terminal to attach itself to a clock
# in the upper
# right corner of the terminal.

function prompt_command {
# Calculate the width of the prompt:
hostnam=$(echo -n $HOSTNAME | sed -e "s/[\.].*//")
# "whoami" and "pwd" include a trailing newline
usernam=$(whoami)
newPWD="${PWD}"
# Add all the accessories below ...
let promptsize=$(echo -n "--(${usernam}@${hostnam})---(${PWD})---" \
| wc -c | tr -d " ")
# Figure out how much to add between user@host and PWD (or how
# much to
# remove from PWD)
let fillsize=${COLUMNS}-${promptsize}
fill=""
# Make the filler if prompt isn't as wide as the terminal:

```

```

while [ "$fillsize" -gt "0" ]
do
    fill="${fill}Ä"
    # The A with the umlaut over it (it will appear as a long
dash if
    # you're using a VGA font) is \304, but I cut and pasted it
in
    # because Bash will only do one substitution - which in this
case is
    # putting $fill in the prompt.
    let fillsize=${fillsize}-1
done
# Right-truncate PWD if the prompt is going to be wider than
the terminal:
if [ "$fillsize" -lt "0" ]
then
    let cutt=3-${fillsize}
    newPWD="...$(echo -n $PWD | sed -e "s/\(^.\{$cutt\}\)\(
(*.*)/\2/")"
fi
#
# Create the clock and the bar that runs up the right side of
the term
#
local LIGHT_BLUE="\033[1;34m"
local YELLOW="\033[1;33m"
# Position the cursor to print the clock:
echo -en "\033[2;${COLUMNS}-9)H"
echo -en "$LIGHT_BLUE($YELLOW$(date
+%H%M)$LIGHT_BLUE)\304$YELLOW\304\304\277"
local i=${LINES}
echo -en "\033[2;${COLUMNS}H"
# Print vertical dashes down the side of the terminal:
while [ $i -ge 4 ]
do
    echo -en "\033[${i-1});${COLUMNS}H\263"
    let i=i-1
done

let prompt_line=${LINES}-1
# This is needed because doing \${LINES} inside a Bash
mathematical
# expression (ie. $(( ))) doesn't seem to work.
}

PROMPT_COMMAND=prompt_command

```

```

function clock3 {
local LIGHT_BLUE="\[\033[1;34m\"
local      YELLOW="\[\033[1;33m\"
local      WHITE="\[\033[1;37m\"
local LIGHT_GRAY="\[\033[0;37m\"
local  NO_COLOUR="\[\033[0m\"

case $TERM in
    xterm*)
        TITLEBAR='\[\033]0;\u@\h:\w\007\]'
        ;;
    *)
        TITLEBAR=""
        ;;
esac

PS1="$TITLEBAR\
\[\033[\${prompt_line};0H\
$YELLOW\332$LIGHT_BLUE\304(\
$YELLOW\${username}$LIGHT_BLUE@$YELLOW\${hostname}\
${LIGHT_BLUE})\304${YELLOW}\304\${fill}${LIGHT_BLUE}\304(\
$YELLOW\${newPWD}\
$LIGHT_BLUE)\304$YELLOW\304\304\304\331\
\n\
$YELLOW\300$LIGHT_BLUE\304(\
$YELLOW\$(date \"+%a,%d %b %y\")\
$LIGHT_BLUE:$WHITE\$$LIGHT_BLUE)\304\
$YELLOW\304\
$LIGHT_GRAY  "

PS2="$LIGHT_BLUE\304$YELLOW\304$YELLOW\304$NO_COLOUR  "

}

```

9. 变量 variable

系统变量

系统变量,Shell常用的系统变量并不多，但却十分有用，特别是在做一些参数检测的时候。下面是Shell常用的系统变量

表示方法	描述
<code>\$n</code>	<code>\$1</code> 表示第一个参数, <code>\$2</code> 表示第二个参数 ...
<code>\$#</code>	命令行参数的个数
<code>\$0</code>	当前程序的名称
<code>\$?</code>	前一个命令或函数的返回码
<code>\$*</code>	以"参数1 参数2 ... " 形式保存所有参数
<code>@</code>	以"参数1" "参数2" ... 形式保存所有参数
<code>\$\$</code>	本程序的(进程ID号)PID
<code>!</code>	上一个命令的PID

命令行参数传递

```
[root@cc tmp]# cat test.sh
echo $#
echo $@

[root@cc tmp]# ./test.sh helloworld
1
helloworld
```

`$n` `$#` `$0` `$?`

其中使用得比较多得是 `$n` `$#` `$0` `$?` ,看看下面的例子:

```
#!/bin/sh
if [ $# -ne 2 ] ; then
echo "Usage: $0 string file";
exit 1;
fi
grep $1 $2 ;
if [ $? -ne 0 ] ; then
echo "Not Found \"$1\" in $2";
exit 1;
fi
echo "Found \"$1\" in $2";
上面的例子中使用了$0 $1 $2 $# $? 等变量
下面运行的例子:
./chapter2.2.sh usage chapter2.2.sh
```

```
Not Found "usage" in chapter2.2.sh
-bash-2.05b$ ./chapter2.2.sh Usage chapter2.2.sh
echo "Usage: $0 string file";
Found "Usage" in chapter2.2.sh
```

\$? 程序运行返回值

0 表示正常结束运行，1 表示异常退出

```
[root@iz621r6pk9aZ nginx]# ping -W 2 -c 2 www.google.com
PING www.google.com (172.217.24.196) 56(84) bytes of data.
64 bytes from hkg12s13-in-f4.1e100.net (172.217.24.196): icmp_seq=1 ttl=57
time=1.51 ms
64 bytes from hkg12s13-in-f4.1e100.net (172.217.24.196): icmp_seq=2 ttl=57
time=1.44 ms

--- www.google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.447/1.479/1.512/0.050 ms

[root@iz621r6pk9aZ nginx]# echo $?
0
```

我们ping 一个不存在的IP地址，然后 Ctrl+C 推出程序，返回值是 1.

```
[root@iz621r6pk9aZ nginx]# ping -W 2 -c 2 172.16.1.100
PING 172.16.1.100 (172.16.1.100) 56(84) bytes of data.
^C
--- 172.16.1.100 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 999ms

[root@iz621r6pk9aZ nginx]# echo $?
1
```

如果 redis 用户不存，就创建一个名为 redis 的用户。

```
id redis
if [ $? -eq 1 ]
then
    adduser -s /bin/false -d /var/lib/redis redis
fi
```


shift 移位

shift 移位传递过来的参数

```
$ cat test.sh
echo $@
shift
echo $@

$ ./test.sh aaa bbb ccc ddd
aaa bbb ccc ddd
bbb ccc ddd
```

```
$ cat test.sh
echo $@
shift
echo $@

shift 2
echo $@
$ ./test.sh aaa bbb ccc ddd eee
aaa bbb ccc ddd eee
bbb ccc ddd eee
ddd eee
```

表达式

```
!!: 再次执行上一条命令
!$: 上一条命令的最后一个单词
{a..b}: 按照从a到b顺序的一个数字列表
{a,b,c}: 三个词a,b,c. 可以这样使用 touch /tmp/{a,b,c}
${1-$9}: 执行shell脚本时的命令行参数
$0: 正在执行的命令名称
$#: 当前启动的命令中传入的参数个数
$?: 上一条命令的执行返回值。
$$: 该shell的进程号。
```

\$*: 从\$1开始, 启动该shell脚本的所有参数。

```
$ mkdir -p {a..z}
$ ls
a b c d e f g h i j k l m n o p q r s t u v w x y z

$ mkdir -p {a..z}{0..9}
$ ls
a0 b0 c0 d0 e0 f0 g0 h0 i0 j0 k0 l0 m0 n0 o0 p0 q0 r0 s0 t0
u0 v0 w0 x0 y0 z0
a1 b1 c1 d1 e1 f1 g1 h1 i1 j1 k1 l1 m1 n1 o1 p1 q1 r1 s1 t1
u1 v1 w1 x1 y1 z1
a2 b2 c2 d2 e2 f2 g2 h2 i2 j2 k2 l2 m2 n2 o2 p2 q2 r2 s2 t2
u2 v2 w2 x2 y2 z2
a3 b3 c3 d3 e3 f3 g3 h3 i3 j3 k3 l3 m3 n3 o3 p3 q3 r3 s3 t3
u3 v3 w3 x3 y3 z3
a4 b4 c4 d4 e4 f4 g4 h4 i4 j4 k4 l4 m4 n4 o4 p4 q4 r4 s4 t4
u4 v4 w4 x4 y4 z4
a5 b5 c5 d5 e5 f5 g5 h5 i5 j5 k5 l5 m5 n5 o5 p5 q5 r5 s5 t5
u5 v5 w5 x5 y5 z5
a6 b6 c6 d6 e6 f6 g6 h6 i6 j6 k6 l6 m6 n6 o6 p6 q6 r6 s6 t6
u6 v6 w6 x6 y6 z6
a7 b7 c7 d7 e7 f7 g7 h7 i7 j7 k7 l7 m7 n7 o7 p7 q7 r7 s7 t7
u7 v7 w7 x7 y7 z7
a8 b8 c8 d8 e8 f8 g8 h8 i8 j8 k8 l8 m8 n8 o8 p8 q8 r8 s8 t8
u8 v8 w8 x8 y8 z8
a9 b9 c9 d9 e9 f9 g9 h9 i9 j9 k9 l9 m9 n9 o9 p9 q9 r9 s9 t9
u9 v9 w9 x9 y9 z9

$ touch {a..z}{0..9}/{a..z}{0..9}
$ ls
a0 b0 c0 d0 e0 f0 g0 h0 i0 j0 k0 l0 m0 n0 o0 p0 q0 r0 s0 t0
u0 v0 w0 x0 y0 z0
a1 b1 c1 d1 e1 f1 g1 h1 i1 j1 k1 l1 m1 n1 o1 p1 q1 r1 s1 t1
u1 v1 w1 x1 y1 z1
a2 b2 c2 d2 e2 f2 g2 h2 i2 j2 k2 l2 m2 n2 o2 p2 q2 r2 s2 t2
u2 v2 w2 x2 y2 z2
a3 b3 c3 d3 e3 f3 g3 h3 i3 j3 k3 l3 m3 n3 o3 p3 q3 r3 s3 t3
u3 v3 w3 x3 y3 z3
a4 b4 c4 d4 e4 f4 g4 h4 i4 j4 k4 l4 m4 n4 o4 p4 q4 r4 s4 t4
u4 v4 w4 x4 y4 z4
a5 b5 c5 d5 e5 f5 g5 h5 i5 j5 k5 l5 m5 n5 o5 p5 q5 r5 s5 t5
u5 v5 w5 x5 y5 z5
a6 b6 c6 d6 e6 f6 g6 h6 i6 j6 k6 l6 m6 n6 o6 p6 q6 r6 s6 t6
u6 v6 w6 x6 y6 z6
a7 b7 c7 d7 e7 f7 g7 h7 i7 j7 k7 l7 m7 n7 o7 p7 q7 r7 s7 t7
u7 v7 w7 x7 y7 z7
a8 b8 c8 d8 e8 f8 g8 h8 i8 j8 k8 l8 m8 n8 o8 p8 q8 r8 s8 t8
u8 v8 w8 x8 y8 z8
a9 b9 c9 d9 e9 f9 g9 h9 i9 j9 k9 l9 m9 n9 o9 p9 q9 r9 s9 t9
u9 v9 w9 x9 y9 z9
$ ls a0
a0 b0 c0 d0 e0 f0 g0 h0 i0 j0 k0 l0 m0 n0 o0 p0 q0 r0 s0 t0
u0 v0 w0 x0 y0 z0
a1 b1 c1 d1 e1 f1 g1 h1 i1 j1 k1 l1 m1 n1 o1 p1 q1 r1 s1 t1
```

u1	v1	w1	x1	y1	z1														
a2	b2	c2	d2	e2	f2	g2	h2	i2	j2	k2	l2	m2	n2	o2	p2	q2	r2	s2	t2
u2	v2	w2	x2	y2	z2														
a3	b3	c3	d3	e3	f3	g3	h3	i3	j3	k3	l3	m3	n3	o3	p3	q3	r3	s3	t3
u3	v3	w3	x3	y3	z3														
a4	b4	c4	d4	e4	f4	g4	h4	i4	j4	k4	l4	m4	n4	o4	p4	q4	r4	s4	t4
u4	v4	w4	x4	y4	z4														
a5	b5	c5	d5	e5	f5	g5	h5	i5	j5	k5	l5	m5	n5	o5	p5	q5	r5	s5	t5
u5	v5	w5	x5	y5	z5														
a6	b6	c6	d6	e6	f6	g6	h6	i6	j6	k6	l6	m6	n6	o6	p6	q6	r6	s6	t6
u6	v6	w6	x6	y6	z6														
a7	b7	c7	d7	e7	f7	g7	h7	i7	j7	k7	l7	m7	n7	o7	p7	q7	r7	s7	t7
u7	v7	w7	x7	y7	z7														
a8	b8	c8	d8	e8	f8	g8	h8	i8	j8	k8	l8	m8	n8	o8	p8	q8	r8	s8	t8
u8	v8	w8	x8	y8	z8														
a9	b9	c9	d9	e9	f9	g9	h9	i9	j9	k9	l9	m9	n9	o9	p9	q9	r9	s9	t9
u9	v9	w9	x9	y9	z9														

Internal Environment Variables

<http://tldp.org/LDP/abs/html/internalvariables.html>

\$RANDOM 随机数

```
neo@MacBook-Pro ~ % echo $RANDOM
15254
```

\$RANDOM 的范围是 0 ~ 32767

```
for i in {1..10};
do
    echo -e "$i \t $RANDOM"
done
```

与 **history** 有关的环境变量

HISTSIZE 将最后多少条历史记录保存到文件中

HISTFILESIZE 定义 ~/.bash_history 的行数

HISTTIMEFORMAT 历史记录格式

```
export HISTSIZE=10000
export HISTFILESIZE=10000
export HISTTIMEFORMAT="%Y-%m-%d %H:%M:%S "
export TIME_STYLE=long-iso
```

格式如下

```
903 2019-06-03 00:48:46 docker ps
904 2019-06-03 00:48:49 docker images
905 2019-06-03 00:48:53 docker rmi -f $(docker images -q)
906 2019-06-03 00:48:56 docker stop $(docker ps -a -q)
907 2019-06-03 00:48:57 docker rm -f $(docker ps -a -q)
908 2019-06-03 00:48:57 docker rmi -f $(docker images -q)
909 2019-06-03 00:48:57 docker volume rm $(docker volume ls -q)
910 2019-06-03 00:49:00 docker
```

set 设置变量

```
$ set -- `echo aa bb cc`
$ echo $1
aa
$ echo $2
bb
$ echo $3
cc

$ set -- aa bb cc
```

set -/+e 控制程序是否退出

set -e: 执行的时候如果出现了返回值为非零，整个脚本 就会立即退出("Exit immediately if a simple command exits with a non-zero status.")

set +e: 执行的时候如果出现了返回值为非零将会继续执行下面的脚本

演示脚本，使用 set -e 运行 aaa 找不到这个命令出错，脚本此时会退出。

```
[root@gitlab ~]# cat test.sh
set -e
```

```
echo "A"
aaa
echo "B"

[root@gitlab ~]# bash test.sh
A
test.sh: line 3: aaa: command not found
```

将 `set -e` 改为 `set +e` 后，`aaa` 虽然执行失败，程序不会退出，并且继续运行，我们可以看到输出 `B`

```
[root@gitlab ~]# cat test.sh
set +e
echo "A"
aaa
echo "B"
[root@gitlab ~]# bash test.sh
A
test.sh: line 3: aaa: command not found
B
```

unset 变量销毁

```
$ unset logfile
```

设置变量默认值

如果 `CHANNEL_NAME` 没有被赋值，那么他的默认值是 `"mychannel"`

```
CHANNEL_NAME=$1
: ${CHANNEL_NAME:="mychannel"}
echo $CHANNEL_NAME
```

如果 `logfile` 值已经存在则不会覆盖

```
$ logfile=/var/log/test.log

$ echo $logfile
/var/log/test.log
```

```
$ logfile=${logfile:-/tmp/test.log}

$ echo $logfile
/var/log/test.log
```

如果变量为空才能设置

```
$ unset logfile
$ logfile=${logfile:-/tmp/test.log}
$ echo $logfile
/tmp/test.log
```

export 设置全局变量

```
export CATALINA_OUT=/www/logs/tomcat/catalina.out
```

unset 销毁变量

```
unset CATALINA_OUT
```

declare

功能说明：声明 shell 变量。

语 法：declare [+/-][rxi][变量名称=设置值] 或 declare -f

补充说明：declare为shell指令，在第一种语法中可用来声明变量并设置变量的属性([rix]即为变量的属性)，在第二种语法中可用来显示shell函数。若不加上任何参数，则会显示全部的shell变量与函数(与执行set指令的效果相同)。

参 数：

- +/- "-"可用来指定变量的属性，"+"则是取消变量所设的属性。
- f 仅显示函数。
- r 将变量设置为只读。
- x 指定的变量会成为环境变量，可供shell以外的程序来使用。
- i [设置值]可以是数值，字符串或运算式。

Numerical 数值运算

数值运算表达式

```
$((EXPR))
```

```
neo@netkiller ~ % echo $((1+1))
neo@netkiller ~ % echo $((5*5))

neo@netkiller ~ % echo $(( (1 + 1) * 2 ))
4
```

```
num=$(awk "BEGIN {print $num1+$num2; exit}")
num=$(python -c "print $num1+$num2")
num=$(perl -e "print $num1+$num2")
num=$(echo $num1 + $num2 | bc)
```

巧用linux服务器下的/dev/shm,实现斐波拉切数列

```
[neo@netkiller ~]# cat mblq.sh

TEMP_FILE=/dev/shm/mblq
echo 0 > $TEMP_FILE
echo 1 >> $TEMP_FILE
count=$1
for i in `seq $count`
do
    first=$(tail -2 $TEMP_FILE | head -1)
    two=$(tail -1 $TEMP_FILE)
    echo $((first+two)) >> $TEMP_FILE
done
cat $TEMP_FILE
[neo@netkiller ~]# bash mblq.sh 15
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
```

Strings 字符串操作

```
[neo@netkiller ~]# cat abcde.sh
#!/bin/bash
str="abcde";
for ((m=1;m<=${#str};m++));do
    for ((n=0;n<${#str};n++));do
        [[ ${str}-${n} -lt $m ]] && continue || echo -n ${str:$n:$m}' '
    done
done
[neo@netkiller ~]# bash abcde.sh
a b c d e ab bc cd de abc bcd cde abcd bcde abcde
```

###

```
$ MYVAR=foodforthought.jpg
$ echo ${MYVAR##*fo}
rthought.jpg
$ echo ${MYVAR#*fo}
odforthought.jpg
```

一个简单的脚本例子

```
mytar.sh

#!/bin/bash

if [ "${1##*.}" = "tar" ]
then
    echo This appears to be a tarball.
else
    echo At first glance, this does not appear to be a tarball.
fi

$ ./mytar.sh thisfile.tar
This appears to be a tarball.
$ ./mytar.sh thatfile.gz
At first glance, this does not appear to be a tarball.
```

%%/%%


```
$ MYFOO="chickensoup.tar.gz"
$ echo ${MYFOO%%.*}
chickensoup
$ echo ${MYFOO%.*}
chickensoup.tar

MYFOOD="chickensoup"
$ echo ${MYFOOD%%soup}
chicken
```

```
$ test="aaa bbb ccc ddd"

$ echo ${test% *}
aaa bbb ccc

$ echo ${test%% *}
aaa
```

字符串截取

:n1:n2

左侧截取

```
neo@MacBook-Pro-Neo ~/git/Lisa % STR=Netkiller; echo ${STR:3}
killer
```

右侧截取

```
file=netkiller.rpm
$echo ${file: -3}
```

范围截取： `${variable:n1:n2}`:截取变量variable从n1到n2之间的字符串。

```
$ EXCLAIM=cowabunga

$ echo ${EXCLAIM:0:3}
```

```
cow
```

```
$ echo ${EXCLAIM:3:7}  
abunga
```

```
neo@MacBook-Pro-Neo ~ % str="123456789"  
neo@MacBook-Pro-Neo ~ % str="123456789"; echo ${str:3:(6-3)}
```

#

: \${variable:n1:n2}:截取变量variable从n1到n2之间的字符串。

example

```
$cat name.sh  
#!/bin/bash  
while read line ; do  
    firstname=${line% *}  
    lastname=${line#* }  
    echo $firstname $lastname  
done <<EOF  
neo chen  
jam zheng  
EOF  
  
$ bash name.sh  
neo chen  
jam zheng
```

计算字符串长度

计算字符串长度

```
echo ${#PATH}
```

```
$ VAR="This string is stored in a variable VAR"  
$ echo ${#VAR}
```

39

字符串查找替换

```
# str="1 2 3 4";echo ${str// /}  
1234  
  
# str="1 2 3 4";echo ${str// /,}  
1,2,3,4  
  
# str="1 2 3 4";echo ${str// /+}  
1+2+3+4  
  
# str="neo netkiller";echo ${str//neo/hello}  
hello netkiller
```

Array 数组

定义数组

```
arr=(Hello World)  
  
arr[0]=Hello  
arr[1]=World
```

访问数组

```
echo ${arr[0]} ${arr[1]}  
  
${arr[*]}           # All of the items in the array  
${!arr[*]}          # All of the indexes in the array  
${#arr[*]}          # Number of items in the array  
${#arr[0]}          # Length of item zero
```

追加操作

```
ARRAY=(  
ARRAY+=( 'foo' )  
ARRAY+=( 'bar' )
```

for 与 array

```
#!/bin/bash

array=(one two three four [5]=five)

echo "Array size: ${#array[*]}"

echo "Array items:"
for item in ${array[*]}
do
    printf "    %s\n" $item
done

echo "Array indexes:"
for index in ${!array[*]}
do
    printf "    %d\n" $index
done

echo "Array items and indexes:"
for index in ${!array[*]}
do
    printf "%4d: %s\n" $index ${array[$index]}
done
```

```
#!/bin/bash

array=("first item" "second item" "third" "item")

echo "Number of items in original array: ${#array[*]}"
for ix in ${!array[*]}
do
    printf "    %s\n" "${array[$ix]}"
done
echo

arr=(${array[*]})
echo "After unquoted expansion: ${#arr[*]}"
for ix in ${!arr[*]}
do
    printf "    %s\n" "${arr[$ix]}"
done
echo

arr=("${array[@]}")
echo "After * quoted expansion: ${#arr[*]}"
for ix in ${!arr[*]}
do
    printf "    %s\n" "${arr[$ix]}"
done
echo

arr=("${array[@]}")
echo "After @ quoted expansion: ${#arr[*]}"
for ix in ${!arr[*]}
```

```
do
    printf "    %s\n" "${arr[$ix]}"
done
```

```
array=({23..32} {49,50} {81..92})

echo "Array size: ${#array[*]}"

echo "Array items:"
for item in ${array[*]}
do
    printf "    %s\n" $item
done
```

while 与 array

while 与 array

```
declare -a array=('1:one' '2:two' '3:three');
len=${#array[@]}
i=0
while [ $i -lt $len ]; do
    echo "${array[$i]}"
    let i++
done
```

array 与 read

array 与 read

```
declare -a array=('1:one' '2:two' '3:three');

while read -e item ; do
    echo "$item \n"
done <<< ${array[@]}

mapfile CONFIG <<END
192.168.0.1 80
192.168.0.1 8080
192.168.0.2 8000
192.168.0.2 80
192.168.0.1 88
END

printf %s "${CONFIG[@]}"
```

```
for line in "${CONFIG[@]}"
do
    read ipaddr port <<< $(echo ${line})
    echo "$ipaddr : $port"
done
```

拆分字符串并转换为数组

Split string into an array in Bash

字符串

```
QUEUES="example|sss"
```

类似列表的数据结构

```
for caption in $(echo $QUEUES | tr '|' ' '); do
    echo $caption
done
```

拆分为数组形式

```
captions=$(echo $QUEUES | tr '|' ' ')
for element in "${captions[@]}"
do
    echo "$element"
done

for key in ${!captions[@]}; do
    echo ${key} ${captions[${key}]}
done
```

数组转为字符串

```
ids=(1 2 3 4);echo ${ids[*]} // ||}
ids=(1 2 3 4); lst=$( IFS='|'; echo "${ids[*]}" ); echo $lst

array=(1 2 3 4); echo ${array[*]} // ||}
array=(1 2 3 4);string="${ids[@]}";echo ${string// ||}
array=(1 2 3 4);string="${ids[@]}";echo ${string// /,}
```

```
IFS='|';echo "${[*]// /|}";
```

read 赋值多个变量

```
[net@netkiller tmp]# cat test.sh
read ipaddr port <<< $(echo www.netkiller.cn 80)

echo $ipaddr
echo $port

[net@netkiller tmp]# bash test.sh
www.netkiller.cn
80
```

eval

```
$ i=5
$ a_5=250
$ eval echo $"a_$i"
```

```
# neo="Neo Chen"
# name=neo
# eval "echo \$$name"

Neo Chen
```

typeset

有两个选项 -l 代表小写 -u 代表大写。

```
typeset -u name
name='neo'
echo $name

typeset -l nickname
nickname='netkiller'
echo $nickname

typeset -l nickname
nickname='NETKILLER'
echo $nickname
```

操作演示

```
[root@localhost ~]# typeset -u name
[root@localhost ~]# name='neo'
[root@localhost ~]# echo $name
NEO
[root@localhost ~]#
[root@localhost ~]# typeset -l nickname
[root@localhost ~]# nickname='netkiller'
[root@localhost ~]# echo $nickname
netkiller
[root@localhost ~]#
[root@localhost ~]# typeset -l nickname
[root@localhost ~]# nickname='NETKILLER'
[root@localhost ~]# echo $nickname
netkiller
```

envsubst - substitutes environment variables in shell format strings

替换 **shell** 中的环境变量字符串

envsubst 的功能非常类似模版引擎，我这么一说，做开发的小伙伴瞬间秒懂。现在做一个实验。

添加环境变量到预设文件 source.sh

```
export NAME=Neo
export NICKNAME=Netkiller
```

模版文件 template.tpl

```
NAME=${NAME}
NICKNAME=${NICKNAME}
```

生成目标文件


```
[root@localhost tmp]# source source.sh && envsubst < template.tpl > my.conf
[root@localhost tmp]# cat my.conf
NAME=Neo
NICKNAME=Netkiller
```

设置默认值

```
cat <<'EOF'> template.tpl
#!/bin/bash
echo ${NAME}
echo ${NICKNAME}
echo ${AGE}
echo ${HOST}
EOF

export NAME=${NAME:-'NONE'}
export NICKNAME=${NICKNAME:-'NONE'}
export AGE=${AGE:-'26'}
export HOST=${HOST:-`hostname -I|awk '{print $1}'`}
envsubst < template.tpl > my.sh

cat my.sh
bash my.sh
```

运行结果

```
[root@localhost tmp]# cat <<'EOF'> template.tpl
> #!/bin/bash
> echo ${NAME}
> echo ${NICKNAME}
> echo ${AGE}
> echo ${HOST}
> EOF
[root@localhost tmp]#
[root@localhost tmp]# export NAME=${NAME:-'NONE'}
[root@localhost tmp]# export NICKNAME=${NICKNAME:-'NONE'}
[root@localhost tmp]# export AGE=${AGE:-'26'}
[root@localhost tmp]# export HOST=${HOST:-`hostname -I|awk '{print $1}'`}
[root@localhost tmp]# envsubst < template.tpl > my.sh
[root@localhost tmp]#
[root@localhost tmp]# cat my.sh
#!/bin/bash
echo NONE
echo Netkiller
echo 26
echo 192.168.30.12
[root@localhost tmp]# bash my.sh
NONE
```

Netkiller

26

192.168.30.12

10. conditions if and case

表 2.1. 文件目录表达式

Primary	意义
[-a FILE]	如果 FILE 存在则为真。
[-b FILE]	如果 FILE 存在且是一个块特殊文件则为真。
[-c FILE]	如果 FILE 存在且是一个字特殊文件则为真。
[-d FILE]	如果 FILE 存在且是一个目录则为真。
[-e FILE]	如果 FILE 存在则为真。
[-f FILE]	如果 FILE 存在且是一个普通文件则为真。
[-g FILE]	如果 FILE 存在且已经设置了SGID则为真。
[-h FILE]	如果 FILE 存在且是一个符号连接则为真。
[-k FILE]	如果 FILE 存在且已经设置了粘制位则为真。
[-p FILE]	如果 FILE 存在且是一个名字管道(FIFO)则为真。
[-r FILE]	如果 FILE 存在且是可读的则为真。
[-s FILE]	如果 FILE 存在且大小不为0则为真。
[-t FD]	如果文件描述符 FD 打开且指向一个终端则为真。
[-u FILE]	如果 FILE 存在且设置了SUID (set user ID)则为真。
[-w FILE]	如果 FILE 存在且是可写的则为真。
[-x FILE]	如果 FILE 存在且是可执行的则为真。
[-O FILE]	如果 FILE 存在且属有效用户ID则为真。
[-G FILE]	如果 FILE 存在且属有效用户组则为真。
[-L FILE]	如果 FILE 存在且是一个符号连接则为真。
[-N FILE]	如果 FILE 存在 and has been modified since it was last read则为真。
[-S FILE]	如果 FILE 存在且是一个套
[FILE1 -nt FILE2]	如果 FILE1 has been changed more recently than FILE2, or 如果 FILE1 exists and FILE2 does not则为真。
[FILE1 -ot FILE2]	如果 FILE1 比 FILE2 要老, 或者 FILE2 存在且 FILE1 不存

ot FILE2]	在则为真。
[FILE1 - ef FILE2]	如果 FILE1 和 FILE2 指向相同的设备和节点号则为真。

表 2.2. 字符串表达式

Primary	意义
[-o OPTIONNAME]	如果 shell选项 “OPTIONNAME” 开启则为真。
[-z STRING]	“STRING” 的长度为零则为真。
[-n STRING] or [STRING]	“STRING” 的长度为非零 non-zero则为真。
[STRING1 == STRING2]	如果2个字符串相同则为真。
[STRING1 != STRING2]	如果字符串不相等则为真。
[STRING1 < STRING2]	如果 “STRING1” sorts before “STRING2” lexicographically in the current locale则为真。
[STRING1 > STRING2]	如果 “STRING1” sorts after “STRING2” lexicographically in the current locale则为真。
[ARG1 OP ARG2]	“OP” 为 -eq, -ne, -lt, -le, -gt or -ge.

```
[ -z "$VAR" ] && VAR="some default"
[ ! "$VAR" ] && VAR="some default"
[ "$VAR" ] || VAR="some default"
```

Arithmetic relational operators

1. -lt (<)
2. -gt (>)
3. -le (<=)

- 4. -ge (>=)
- 5. -eq (==)
- 6. -ne (!=)

表 2.3. 组合表达式

操作	效果
[! EXPR]	如果 EXPR 是false则为真。
[(EXPR)]	返回 EXPR的值。这样可以用来忽略正常的操作符优先级。
[EXPR1 -a EXPR2]	如果 EXPR1 and EXPR2 全真则为真。
[EXPR1 -o EXPR2]	如果 EXPR1 或者 EXPR2 为真则为真。

if

```
if [ ! -d /directory/to/check ]; then
    mkdir -p /directory/toc/check
fi

if [ -z "$VAR" ]; then
    VAR="some default"
fi
```

例 2.4. Basic conditional example if .. then

```
#!/bin/bash
if [ "foo" = "foo" ]; then
    echo expression evaluated as true
fi
```

例 2.5. Conditionals with variables

```
#!/bin/bash
T1="foo"
T2="bar"
if [ "$T1" = "$T2" ]; then
    echo expression evaluated as true
else
    echo expression evaluated as false
fi
```

```
(( $# != 1 )) && bool=0 || bool=${1}
[[ -f /tmp/test ]] && echo "True" || echo "False"
```

判断变量是否包含字符串

str中是够包括特定的字符串

```
str="www.netkiller.cn"
```

方法一：

```
if [[ "${str}" =~ "net" ]]; then
    echo "true"
fi
```

方法二：

```
if [[ ${str} = *killer* ]]; then
    echo "true"
```

```
fi
```

方法三:

```
if echo ${str} |grep -q "netkiller"; then
    echo "true"
fi
```

方法四:

```
echo ${str} |grep -q "kill" && echo "true" || echo "false"
```

case

case 高级用法, 匹配 Yes,YES,YeS,yES,yEs ...

```
read -p "Do you want to continue [Y/n]?" BOOLEAN

case $BOOLEAN in
    [yY][eE][sS])
        echo 'Thanks' $BOOLEAN
        ;;
    [yY]|[nN])
        echo 'Thanks' $BOOLEAN
        ;;
    'T' | 'F')
        echo 'Thanks' $BOOLEAN
        ;;
    [Tt]ure|[Ff]alse)
        echo 'Thanks' $BOOLEAN
        ;;
    *)
        exit 1
        ;;
esac
```

例 2.6. case

```
case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    status)
        status
        ;;
    restart)
        stop
        start
        ;;
    condrestart)
        condrestart
        ;;
    *)
        echo $"Usage: $0
{start|stop|restart|condrestart|status}"
        exit 1
esac
```


11. Loops for, while and until

for

```
#!/bin/bash
for i in 1 2 3 4 5
do
    echo "Welcome $i times"
done

for i in $( ls ); do
    echo item: $i
done

for i in `seq 1 10`;
do
    echo $i
done

for i in {1..5}
do
    echo "Welcome $i times"
done

for (( c=1; c<=5; c++ ))
do
    echo "Welcome $c times..."
done

for ((i=1; $i<=9;i++)); do echo $i; done
```

```
for i in {0..10..2}
do
    echo "Welcome $i times"
done
```

```
for i in $(seq 1 2 20)
do
    echo "Welcome $i times"
done
```

单行实例

```
for ip in {1..10};do echo $ip; done

for i in `seq 1 10`;do echo $i;done

for ip in {81..92}; do ssh root@172.16.3.$ip date; done

for n in {23..32} {49,50} {81..92}; do mkdir /tmp/$n; echo $n; done
```

```
for keyword in bash cmd ls
do
    echo $keyword
done

string="aaa bbb ccc ddd" && for word in $string; do echo "$word"; done

files=( "/etc/passwd" "/etc/group" "/etc/hosts" )
for file in "${files[@]}"
do
    echo $file
done
```

infinite loops

```
#!/bin/bash
for (( ; ; ))
do
    echo "infinite loops [ hit CTRL+C to stop]"
done
```

find file

```
#!/bin/bash
for file in /etc/*
do
    if [ "${file}" == "/etc/resolv.conf" ]
    then
        countNameservers=$(grep -c nameserver
/etc/resolv.conf)
        echo "Total  ${countNameservers} nameservers
defined in ${file}"
        break
    fi
done
```

backup file

```
#!/bin/bash
FILES="$@"
for f in $FILES
do
    # if .bak backup file exists, read next file
    if [ -f ${f}.bak ]
    then
        echo "Skipping $f file..."
        continue # read next file and skip cp
    fi
    # we are hear means no backup file exists, just use
command
```

```
cp command to copy file
    /bin/cp $f $f.bak
done
```

```
for n in {23..32} {49,50} {81..92}; do mkdir /tmp/$n; echo $n;
done
```

while

```
#!/bin/bash
COUNTER=0
while [ $COUNTER -lt 10 ]; do
    echo The counter is $COUNTER
    let COUNTER=COUNTER+1
done
```

```
while read name age
do
    echo $name $age
done << EOF
neo 30
jam 31
john 29
EOF
```

```
while read name age
do
    [[ $age -gt 30 ]] && echo $name
done << EOF
neo 30
jam 31
john 29
```

EOF

```
$ cat mount.sh
#!/bin/sh
while read LINE
do

    s=`echo $LINE|awk '{print $1}'`
    d=`echo $LINE|awk '{print $2}'`

    umount -f $d
    mount -t nfs4 $s $d

done < mount.conf

$ cat mount.conf
172.16.0.1:/      /www/logs/1
172.16.0.2:/      /www/logs/2
172.16.0.3:/      /www/logs/3
172.16.0.4:/      /www/logs/4
172.16.0.5:/      /www/logs/5
```

读一行

```
while IFS='' read -r line || [[ -n "$line" ]]; do
    echo "Text read from file: $line"
done < "$1"
```

until

```
#!/bin/bash
COUNTER=20
until [ $COUNTER -lt 10 ]; do
    echo COUNTER $COUNTER
    let COUNTER-=1
done
```

12. Functions

例 2.7. Functions with parameters sample

```
#!/bin/bash
function quit {
    exit
}
function e {
    echo $1
}

e Hello
e World
quit
echo foo
```

Local variables

```
#!/bin/bash
HELLO=Hello
function hello {
    local HELLO=World
    echo $HELLO
}
echo $HELLO
hello
echo $HELLO
```

13. User interfaces

例 2.8. Using select to make simple menus

```
#!/bin/bash
OPTIONS="Hello Quit"
select opt in $OPTIONS; do
    if [ "$opt" = "Quit" ]; then
        echo done
        exit
    elif [ "$opt" = "Hello" ]; then
        echo Hello World
    else
        clear
        echo bad option
    fi
done
```

例 2.9. Using the command line

```
#!/bin/bash
if [ -z "$1" ]; then
    echo usage: $0 directory
    exit
fi
SRCD=$1
TGTD="/var/backups/"
OF=home-$(date +%Y%m%d).tgz
tar -czf $TGTD$OF $SRCD
```

例 2.10. Reading user input with read

In many ocations you may want to prompt the user for some input, and there are several ways to achive this. This is one of those ways:

```
#!/bin/bash
echo Please, enter your name
read NAME
echo "Hi $NAME!"
```

As a variant, you can get multiple values with read, this example may clarify this.

```
#!/bin/bash
echo Please, enter your firstname and lastname
read FN LN
echo "Hi! $LN, $FN !"
```

input

例 2.11. read

限时30秒内，输入你的名字

```
$ read -p "Please input your name: " -t 30 named
Please input your name: neo

$ echo $named
```

```
READ_TIMEOUT=60
```

```
read -t "$READ_TIMEOUT" input

# if you do not want quotes, then escape it
input=$(sed "s/[\;\`\\"\$\' ]//g" <<< $input)

# For reading number, then you can escape other characters
input=$(sed 's/[^0-9]*//g' <<< $input)
```

14. subshell

echo \$\$ \$BASHPID ; (echo \$\$ \$BASHPID)

su 运行 shell 并获取 PID 并存入文件

```
su - $USER -c "$PROG & echo \#! > $PIDFILE"
```

15. Example

有趣的Shell

运行后会不停的fork新的进程，直到你的资源消耗尽。

```
:() { :|:& }; :  
.() { .|. & }; .
```

backup

```
#!/bin/sh  
umount /mnt/backup  
mount /dev/sdb1 /mnt/backup  
  
if [ `date +%d` = '01' ] #每月1号进行完全备份  
then  
    bakdir="/mnt/bak/daybak/month/"`date +%m%d`  
    z1="" #进行完全备份  
else  
    backup_dir="/mnt/backup/"`date +%d`  
    z1="-N "`date +%Y-%m-01 00:00:01`"; #差异备份  
    #z1="-N "`date -d '-1 day' +%Y-%m-%d 00:00:01`" #日增量  
备份  
fi  
  
tar "${z1}" -czf ${backup_dir}/www.tgz /var/www  
umount /mnt/backup
```

CPU 核心数

```
cat /proc/cpuinfo | grep processor | wc -l
```

Password

例 2.12. random password

```
cat /dev/urandom | head -1 | md5sum | head -c 8  
od -N 4 -t x4 /dev/random | head -1 | awk '{print $2}'
```

processes

pid

```
neo@debian:~/html/temp$ pidof lighttpd  
2775  
  
neo@debian:~/html/temp$ pgrep lighttpd  
2775  
  
neo@debian:~/html/temp$ pid=`pidof lighttpd`  
neo@debian:~/html/temp$ echo $pid  
2775
```

```
# user=`whoami`  
# pgrep -u $user -f cassandra | xargs kill -9
```

kill

kill 占用7800端口的进程

```
kill -9 `netstat -nlp | grep '192.168.0.5:7800' | awk -F ' ' '{print $7}'`
```

```
'{print $7}' | awk -F '/' '{print $1}'`
```

pgrep

```
#!/bin/bash
ntpdate 172.16.10.10

pid=$(pgrep rsync)

if [ -z "$pid" ]; then

rsync -auzP --delete -e ssh --exclude=example/images --
exclude=project/product --exclude=project/templates/caches
root@172.16.10.10:/www/project /www

fi
```

Shell 技巧

行转列，再批评

```
echo "abc def gfh ijk" | sed "s:\ :\\n:g" | grep -w gfh
```

for vs while

```
echo "aaa bbb ccc" > test.txt
echo "ddd eee fff" >> test.txt
```

```
for line in $(cat test.txt)
do
    echo $line
done
```

```
cat test.txt | while read line
do
    echo $line
done
```

遍历字符串

```
# find . -name "*.html" -o -name "*.php" -o -name '*.dwt' -
printf "[%p] " -exec grep -c 'head' {} \; | grep -v "0$" | more
```

to convert utf-8 from gb2312 code

```
perl -MEncode -pi -e '
$_=encode_utf8(decode(gb2312=>$_)) ' filename
for f in `find .`; do [ -f $f ] && perl -MEncode -pi -e
'$_=encode_utf8(decode(gb2312=>$_))' $f; done;
```

使用内存的百分比

```
$ free | sed -n 2p | awk '{print
"used="$3/$2*100"%", "free="$4/$2*100"%"}'
used=53.9682% free=46.0318%
```

合并apache被cronlog分割的log文件

```
$ find 2009 -type f -name access.log -exec cat {} >> access.log
\;
```

Linux 交集 差集 并集

测试文件如下：

```
[root@test23 ~]# cat a.txt
```

```
1.1.1.1
```

```
2.2.2.2
```

```
3.3.3.3
```

```
1.2.3.4
```

```
[root@test23 ~]# cat b.txt
```

```
4.4.4.4
```

```
1.2.3.4
```

```
2.2.2.2
```

```
a.b.c.d
```

```
```
```

```
grep
```

```
```
```

1) 差集

// 使用 `grep -v` 和 `-f` 参数方式 是最容易想到的

```
[root@test23 ~]# grep -v -f a.txt b.txt
```

```
4.4.4.4
```

```
a.b.c.d
```

```
[root@test23 ~]# grep -v -f b.txt a.txt
```

```
1.1.1.1
```

```
3.3.3.3
```

```
```
```

```
uniq
```

```
```
```

1) 差集

// `-u`表示的是输出出现次数为1的内容

```
[root@test23 ~]# sort a.txt b.txt | uniq -u
```

```
1.1.1.1
```

```
3.3.3.3
```

```
4.4.4.4
```

```
a.b.c.d
```

2) 并集

```
[root@test23 ~]# sort a.txt b.txt | uniq
```

```
1.1.1.1
```

```
1.2.3.4
```

```
2.2.2.2
```


3.3.3.3

4.4.4.4

a.b.c.d

3) 交集

// -d 表示的是输出出现次数大于1的内容

```
[root@test23 ~]# sort a.txt b.txt | uniq -d
```

1.2.3.4

2.2.2.2

第 3 章 小众 Shell

1. fish shell

安装 fish shell

Linux 安装

```
dnf install -y fish
```

配置 fish

主题管理

```
fish_config theme show
```

2. Z Shell

<http://www.zsh.org/>

installing Z shell

```
$ sudo apt install zsh
```

Oh My ZSH!

<http://ohmyz.sh/>

Oh My ZSH 是z shell命令主题

```
$ sh -c "$(curl -fsSL https://raw.githubusercontent.com/robbyrussell/oh-my-zsh/master/tools/install.sh)"
```

Starting file

`~/.zshrc`

```
neo@netkiller:~$ cat .zshrc
# Created by newuser for 4.3.9
PROMPT='%n%M:%~$ '

# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    eval "`dircolors -b`"
    alias ls='ls --color=auto'
```

```
alias dir='dir --color=auto'
alias vdir='vdir --color=auto'

alias grep='grep --color=auto'
alias fgrep='fgrep --color=auto'
alias egrep='egrep --color=auto'
fi

# some more ls aliases
alias ll='ls -l'
alias la='ls -A'
alias l='ls -CF'

# Home/End/Del key
bindkey '\e[1~' beginning-of-line
bindkey '\e[4~' end-of-line
bindkey "\e[3~" delete-char
```

Prompting

```
$ PROMPT='%n%M:%~$ '
neo@netkiller:~$
```

```
autoload colors; colors
export PS1="%B[%{$fg[red]%}%n%{$reset_color%}%b@%B%
{$fg[cyan]%}%m%b%{$reset_color%}:%~%B]%b "
```

```
[neo@netkiller:~/ .oh-my-zsh/themes]
```

Aliases

```
# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    eval "`dircolors -b`"
    alias ls='ls --color=auto'
    alias dir='dir --color=auto'
    alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# some more ls aliases
alias ll='ls -l'
alias la='ls -A'
alias l='ls -CF'
```

History

```
$ !
```

```
$ history
 18  cd workspace/Document
 19  ls
 20  ls

$ !20
ls
Docbook  makedoc  Tex
```

FAQ

Home/End key

```
bindkey '\e[1~' beginning-of-line  
bindkey '\e[4~' end-of-line
```

3. Berkeley UNIX C shell (csh)

```
$ sudo apt install csh
```

4. KornShell

```
$ sudo apt install ksh
```


第 4 章 Shell 命令

1. Help Commands

man - an interface to the on-line reference manuals

manpath.config

```
cat /etc/manpath.config
```

查看**man**手册位置

```
$ man -aw ls  
/usr/share/man/man1/ls.1.gz
```

指定手册位置

```
man -M /home/mysql/man mysql
```

2. getconf - Query system configuration variables

```
$ getconf LONG_BIT
32
$ getconf WORD_BIT
32
```

LINK_MAX	65000
_POSIX_LINK_MAX	65000
MAX_CANON	255
_POSIX_MAX_CANON	255
MAX_INPUT	255
_POSIX_MAX_INPUT	255
NAME_MAX	255
_POSIX_NAME_MAX	255
PATH_MAX	4096
_POSIX_PATH_MAX	4096
PIPE_BUF	4096
_POSIX_PIPE_BUF	4096
SOCK_MAXBUF	
_POSIX_ASYNC_IO	
_POSIX_CHOWN_RESTRICTED	1
_POSIX_NO_TRUNC	1
_POSIX_PRIO_IO	
_POSIX_SYNC_IO	
_POSIX_VDISABLE	0
ARG_MAX	2097152
ATEXIT_MAX	2147483647
CHAR_BIT	8
CHAR_MAX	127
CHAR_MIN	-128
CHILD_MAX	63918
CLK_TCK	100
INT_MAX	2147483647
INT_MIN	-2147483648
IOV_MAX	1024
LOGNAME_MAX	256

LONG_BIT	64
MB_LEN_MAX	16
NGROUPS_MAX	65536
NL_ARGMAX	4096
NL_LANGMAX	2048
NL_MSGMAX	2147483647
NL_NMAX	2147483647
NL_SETMAX	2147483647
NL_TEXTMAX	2147483647
NSS_BUFLLEN_GROUP	1024
NSS_BUFLLEN_PASSWD	1024
NZERO	20
OPEN_MAX	1024
PAGESIZE	4096
PAGE_SIZE	4096
PASS_MAX	8192
PTHREAD_DESTRUCTOR_ITERATIONS	4
PTHREAD_KEYS_MAX	1024
PTHREAD_STACK_MIN	16384
PTHREAD_THREADS_MAX	
SCHAR_MAX	127
SCHAR_MIN	-128
SHRT_MAX	32767
SHRT_MIN	-32768
SSIZE_MAX	32767
TTY_NAME_MAX	32
TZNAME_MAX	
UCHAR_MAX	255
UINT_MAX	4294967295
UIO_MAXIOV	1024
ULONG_MAX	18446744073709551615
USHRT_MAX	65535
WORD_BIT	32
_AVPHYS_PAGES	972844
_NPROCESSORS_CONF	8
_NPROCESSORS_ONLN	8
_PHYS_PAGES	4106156
_POSIX_ARG_MAX	2097152
_POSIX_ASYNCIO	200809
_POSIX_CHILD_MAX	63918
_POSIX_FSYNC	200809
_POSIX_JOB_CONTROL	1
_POSIX_MAPPED_FILES	200809
_POSIX_MEMLOCK	200809
_POSIX_MEMLOCK_RANGE	200809

_POSIX_MEMORY_PROTECTION	200809
_POSIX_MESSAGE_PASSING	200809
_POSIX_NGROUPS_MAX	65536
_POSIX_OPEN_MAX	1024
_POSIX_PII	
_POSIX_PII_INTERNET	
_POSIX_PII_INTERNET_DGRAM	
_POSIX_PII_INTERNET_STREAM	
_POSIX_PII_OSI	
_POSIX_PII_OSI_CLTS	
_POSIX_PII_OSI_COTS	
_POSIX_PII_OSI_M	
_POSIX_PII_SOCKET	
_POSIX_PII_XTI	
_POSIX_POLL	
_POSIX_PRIORITIZED_IO	200809
_POSIX_PRIORITY_SCHEDULING	200809
_POSIX_REALTIME_SIGNALS	200809
_POSIX_SAVED_IDS	1
_POSIX_SELECT	
_POSIX_SEMAPHORES	200809
_POSIX_SHARED_MEMORY_OBJECTS	200809
_POSIX_SSIZE_MAX	32767
_POSIX_STREAM_MAX	16
_POSIX_SYNCHRONIZED_IO	200809
_POSIX_THREADS	200809
_POSIX_THREAD_ATTR_STACKADDR	200809
_POSIX_THREAD_ATTR_STACKSIZE	200809
_POSIX_THREAD_PRIORITY_SCHEDULING	200809
_POSIX_THREAD_PRIO_INHERIT	200809
_POSIX_THREAD_PRIO_PROTECT	200809
_POSIX_THREAD_ROBUST_PRIO_INHERIT	
_POSIX_THREAD_ROBUST_PRIO_PROTECT	
_POSIX_THREAD_PROCESS_SHARED	200809
_POSIX_THREAD_SAFE_FUNCTIONS	200809
_POSIX_TIMERS	200809
TIMER_MAX	
_POSIX_TZNAME_MAX	
_POSIX_VERSION	200809
T_IOV_MAX	
XOPEN_CRYPT	
XOPEN_ENH_I18N	1
XOPEN_LEGACY	1
XOPEN_REALTIME	1
XOPEN_REALTIME_THREADS	1

_XOPEN_SHM	1
_XOPEN_UNIX	1
_XOPEN_VERSION	700
_XOPEN_XCU_VERSION	4
_XOPEN_XPG2	1
_XOPEN_XPG3	1
_XOPEN_XPG4	1
BC_BASE_MAX	99
BC_DIM_MAX	2048
BC_SCALE_MAX	99
BC_STRING_MAX	1000
CHARCLASS_NAME_MAX	2048
COLL_WEIGHTS_MAX	255
EQUIV_CLASS_MAX	
EXPR_NEST_MAX	32
LINE_MAX	2048
POSIX2_BC_BASE_MAX	99
POSIX2_BC_DIM_MAX	2048
POSIX2_BC_SCALE_MAX	99
POSIX2_BC_STRING_MAX	1000
POSIX2_CHAR_TERM	200809
POSIX2_COLL_WEIGHTS_MAX	255
POSIX2_C_BIND	200809
POSIX2_C_DEV	200809
POSIX2_C_VERSION	200809
POSIX2_EXPR_NEST_MAX	32
POSIX2_FORT_DEV	
POSIX2_FORT_RUN	
_POSIX2_LINE_MAX	2048
POSIX2_LINE_MAX	2048
POSIX2_LOCALEDEF	200809
POSIX2_RE_DUP_MAX	32767
POSIX2_SW_DEV	200809
POSIX2_UPE	
POSIX2_VERSION	200809
RE_DUP_MAX	32767
PATH	/bin:/usr/bin
CS_PATH	/bin:/usr/bin
LFS_CFLAGS	
LFS_LDFLAGS	
LFS_LIBS	
LFS_LINTFLAGS	
LFS64_CFLAGS	-D_LARGEFILE64_SOURCE
LFS64_LDFLAGS	
LFS64_LIBS	

LFS64_LINTFLAGS	-D_LARGEFILE64_SOURCE
_XBS5_WIDTH_RESTRICTED_ENVS	XBS5_LP64_OFF64
XBS5_WIDTH_RESTRICTED_ENVS	XBS5_LP64_OFF64
_XBS5_ILP32_OFF32	
XBS5_ILP32_OFF32_CFLAGS	
XBS5_ILP32_OFF32_LDFLAGS	
XBS5_ILP32_OFF32_LIBS	
XBS5_ILP32_OFF32_LINTFLAGS	
_XBS5_ILP32_OFFBIG	
XBS5_ILP32_OFFBIG_CFLAGS	
XBS5_ILP32_OFFBIG_LDFLAGS	
XBS5_ILP32_OFFBIG_LIBS	
XBS5_ILP32_OFFBIG_LINTFLAGS	
_XBS5_LP64_OFF64	1
XBS5_LP64_OFF64_CFLAGS	-m64
XBS5_LP64_OFF64_LDFLAGS	-m64
XBS5_LP64_OFF64_LIBS	
XBS5_LP64_OFF64_LINTFLAGS	
_XBS5_LPBIG_OFFBIG	
XBS5_LPBIG_OFFBIG_CFLAGS	
XBS5_LPBIG_OFFBIG_LDFLAGS	
XBS5_LPBIG_OFFBIG_LIBS	
XBS5_LPBIG_OFFBIG_LINTFLAGS	
_POSIX_V6_ILP32_OFF32	
POSIX_V6_ILP32_OFF32_CFLAGS	
POSIX_V6_ILP32_OFF32_LDFLAGS	
POSIX_V6_ILP32_OFF32_LIBS	
POSIX_V6_ILP32_OFF32_LINTFLAGS	
_POSIX_V6_WIDTH_RESTRICTED_ENVS	POSIX_V6_LP64_OFF64
POSIX_V6_WIDTH_RESTRICTED_ENVS	POSIX_V6_LP64_OFF64
_POSIX_V6_ILP32_OFFBIG	
POSIX_V6_ILP32_OFFBIG_CFLAGS	
POSIX_V6_ILP32_OFFBIG_LDFLAGS	
POSIX_V6_ILP32_OFFBIG_LIBS	
POSIX_V6_ILP32_OFFBIG_LINTFLAGS	
_POSIX_V6_LP64_OFF64	1
POSIX_V6_LP64_OFF64_CFLAGS	-m64
POSIX_V6_LP64_OFF64_LDFLAGS	-m64
POSIX_V6_LP64_OFF64_LIBS	
POSIX_V6_LP64_OFF64_LINTFLAGS	
_POSIX_V6_LPBIG_OFFBIG	
POSIX_V6_LPBIG_OFFBIG_CFLAGS	
POSIX_V6_LPBIG_OFFBIG_LDFLAGS	
POSIX_V6_LPBIG_OFFBIG_LIBS	
POSIX_V6_LPBIG_OFFBIG_LINTFLAGS	

_POSIX_V7_ILP32_OFF32	
_POSIX_V7_ILP32_OFF32_CFLAGS	
_POSIX_V7_ILP32_OFF32_LDFLAGS	
_POSIX_V7_ILP32_OFF32_LIBS	
_POSIX_V7_ILP32_OFF32_LINTFLAGS	
_POSIX_V7_WIDTH_RESTRICTED_ENVS	_POSIX_V7_LP64_OFF64
_POSIX_V7_WIDTH_RESTRICTED_ENVS	_POSIX_V7_LP64_OFF64
_POSIX_V7_ILP32_OFFBIG	
_POSIX_V7_ILP32_OFFBIG_CFLAGS	
_POSIX_V7_ILP32_OFFBIG_LDFLAGS	
_POSIX_V7_ILP32_OFFBIG_LIBS	
_POSIX_V7_ILP32_OFFBIG_LINTFLAGS	
_POSIX_V7_LP64_OFF64	1
_POSIX_V7_LP64_OFF64_CFLAGS	-m64
_POSIX_V7_LP64_OFF64_LDFLAGS	-m64
_POSIX_V7_LP64_OFF64_LIBS	
_POSIX_V7_LP64_OFF64_LINTFLAGS	
_POSIX_V7_LPBIG_OFFBIG	
_POSIX_V7_LPBIG_OFFBIG_CFLAGS	
_POSIX_V7_LPBIG_OFFBIG_LDFLAGS	
_POSIX_V7_LPBIG_OFFBIG_LIBS	
_POSIX_V7_LPBIG_OFFBIG_LINTFLAGS	
_POSIX_ADVISORY_INFO	200809
_POSIX_BARRIERS	200809
_POSIX_BASE	
_POSIX_C_LANG_SUPPORT	
_POSIX_C_LANG_SUPPORT_R	
_POSIX_CLOCK_SELECTION	200809
_POSIX_CPUTIME	200809
_POSIX_THREAD_CPUTIME	200809
_POSIX_DEVICE_SPECIFIC	
_POSIX_DEVICE_SPECIFIC_R	
_POSIX_FD_MGMT	
_POSIX_FIFO	
_POSIX_PIPE	
_POSIX_FILE_ATTRIBUTES	
_POSIX_FILE_LOCKING	
_POSIX_FILE_SYSTEM	
_POSIX_MONOTONIC_CLOCK	200809
_POSIX_MULTI_PROCESS	
_POSIX_SINGLE_PROCESS	
_POSIX_NETWORKING	
_POSIX_READER_WRITER_LOCKS	200809
_POSIX_SPIN_LOCKS	200809
_POSIX_REGEX	1

_REGEX_VERSION	
_POSIX_SHELL	1
_POSIX_SIGNALS	
_POSIX_SPAWN	200809
_POSIX_SPORADIC_SERVER	
_POSIX_THREAD_SPORADIC_SERVER	
_POSIX_SYSTEM_DATABASE	
_POSIX_SYSTEM_DATABASE_R	
_POSIX_TIMEOUTS	200809
_POSIX_TYPED_MEMORY_OBJECTS	
_POSIX_USER_GROUPS	
_POSIX_USER_GROUPS_R	
POSIX2_PBS	
POSIX2_PBS_ACCOUNTING	
POSIX2_PBS_LOCATE	
POSIX2_PBS_TRACK	
POSIX2_PBS_MESSAGE	
SYMLOOP_MAX	
STREAM_MAX	16
AIO_LISTIO_MAX	
AIO_MAX	
AIO_PRIO_DELTA_MAX	20
DELAYTIMER_MAX	2147483647
HOST_NAME_MAX	64
LOGIN_NAME_MAX	256
MQ_OPEN_MAX	
MQ_PRIO_MAX	32768
_POSIX_DEVICE_IO	
_POSIX_TRACE	
_POSIX_TRACE_EVENT_FILTER	
_POSIX_TRACE_INHERIT	
_POSIX_TRACE_LOG	
RTSIG_MAX	32
SEM_NSEMS_MAX	
SEM_VALUE_MAX	2147483647
SIGQUEUE_MAX	63918
FILESIZEBITS	64
POSIX_ALLOC_SIZE_MIN	4096
POSIX_REC_INCR_XFER_SIZE	
POSIX_REC_MAX_XFER_SIZE	
POSIX_REC_MIN_XFER_SIZE	4096
POSIX_REC_XFER_ALIGN	4096
SYMLINK_MAX	
GNU_LIBC_VERSION	glibc 2.28
GNU_LIBPTHREAD_VERSION	NPTL 2.28

POSIX2_SYMLINKS	1
LEVEL1_ICACHE_SIZE	65536
LEVEL1_ICACHE_ASSOC	2
LEVEL1_ICACHE_LINESIZE	64
LEVEL1_DCACHE_SIZE	65536
LEVEL1_DCACHE_ASSOC	2
LEVEL1_DCACHE_LINESIZE	64
LEVEL2_CACHE_SIZE	524288
LEVEL2_CACHE_ASSOC	16
LEVEL2_CACHE_LINESIZE	64
LEVEL3_CACHE_SIZE	16777216
LEVEL3_CACHE_ASSOC	16
LEVEL3_CACHE_LINESIZE	64
LEVEL4_CACHE_SIZE	0
LEVEL4_CACHE_ASSOC	0
LEVEL4_CACHE_LINESIZE	0
IPV6	200809
RAW_SOCKETS	200809
_POSIX_IPV6	200809
_POSIX_RAW_SOCKETS	200809

3. test 命令

```
test -x $HAPROXY || exit 0  
test -f "$CONFIG" || exit 0
```

判断目录

```
test -d /path/to/directory || mkdir -p /path/to/directory
```

4. 目录和文件

dirname

```
$ dirname /usr/bin/find
/usr/bin
```

filename

```
$ basename /usr/bin/find
find
```

排除扩展名

```
file=test.txt
b=${file%.*}
echo $b
```

```
$ for file in *.JPG;do mv $file ${file%.*}.jpg;done
```

取扩展名

```
file=test.txt
b=${file##*.}
echo $b
```

test - check file types and compare values

```
test -x /usr/bin/munin-cron && /usr/bin/munin-cron
```

file — determine file type

```
$ file mis.netkiller.cn-0.0.1.war
mis.netkiller.cn-0.0.1.war: Zip archive data, at least v2.0 to extract

$ file dian_icon.png
dian_icon.png: PNG image data, 8 x 24, 8-bit/color RGBA, non-interlaced

$ file sms-s3.jpg
sms-s3.jpg: JPEG image data, JFIF standard 1.01

$ file -i favicon.ico
favicon.ico: image/x-icon; charset=binary

$ file netkiller.wmv
netkiller.wmv: Microsoft ASF

$ file netkiller.flv
netkiller.flv: Macromedia Flash Video

$ file neo.swf
neo.swf: Macromedia Flash data (compressed), version 10

$ file cs800.css
cs800.css: ISO-8859 text, with CRLF line terminators
```

查看mime

```
$ file -i sms.jpg
sms.jpg: image/jpeg; charset=binary

$ file -i call.png
call.png: image/png; charset=binary

$ file -i cs800.css
cs800.css: text/plain; charset=iso-8859-1

$ file -i neo.swf
neo.swf: application/x-shockwave-flash; charset=binary

$ file -i neo.wmv
neo.wmv: video/x-ms-asf; charset=binary

$ file -i neo.flv
neo.flv: video/x-flv; charset=binary
```

stat

modification time (mtime, 修改时间)：当该文件的“内容数据”更改时，会更新这个时间。内容数据指的是文件的内容，而不是文件的属性。

status time (ctime, 状态时间)：当该文件的“状态 (status)”改变时，就会更新这个时间，举例，更改了权限与属性，就会更新这个时间。

access time (atime, 存取时间)：当“取用文件内容”时，就会更新这个读取时间。举例来使用cat去读取该文件，就会更新atime了。

```
[root@apache www]# stat index.html
  File: `index.html'
  Size: 145355          Blocks: 296          IO Block: 4096   regular file
Device: fd0lh/64769d   Inode: 15861815      Links: 1
Access: (0755/-rwxr-xr-x)  Uid: ( 502/  upuser)   Gid: ( 502/  upuser)
Access: 2010-10-28 11:09:52.000000000 +0800
Modify: 2010-10-28 10:23:13.000000000 +0800
Change: 2010-10-28 10:23:13.000000000 +0800
```

mkdir - make directories

```
mkdir -p /tmp/test/{aaa,bbb,ccc,ddd}

mkdir -p /tmp/test/{aaa,bbb,ccc,ddd}/{eee,fff}

mkdir -p
/tmp/test/{2008,2009,2010,2011}/{01,02,03,04,05,06,07,08,09,10,11,12}/{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30}
```

rename

批量更改扩展名

```
rename 's/\.png/\.PNG/' *.png

rename 's/\.mp3/\.MP3/' *.mp3
rename .mp3 .MP3 *.mp3

rename GIF gif *.GIF
```

```
for file in *.GIF
do
    mv $file ${file%.*}.gif
done
```

```
$ mkdir chapter.command.xxx.xml
$ mkdir chapter.command.bbb.xml
$ mkdir chapter.command.ccc.xml
$ mkdir chapter.command.ddd.xml

$ rename 's/command/cmd/' *.command.*.xml
```

touch

创建空文件，修改文件日期时间

```
touch [-acdm] 文件
参数：
-a : 仅修改access time。
-c : 仅修改时间，而不建立文件。
-d : 后面可以接日期，也可以使用 --date="日期或时间"
-m : 仅修改mtime。
-t : 后面可以接时间，格式为 [YYMMDDhhmm]

# touch filename
# touch -d 20050809 filename
# touch -t 0507150202 bashrc
# touch -d "2 days ago" bashrc
# touch --date "2011-06-03" filename
```

truncate

truncate - shrink or extend the size of a file to the specified size

创建指定大小的文件

```
truncate -s 1k /tmp/test.txt
truncate -s 100m /tmp/test100.txt
```

ls - list directory contents

```
$ ls
$ ls ~
$ ls -l
$ ls -a
$ ls -l
$ ls -F
bg7nyt.txt* Desktop/ Firefox_wallpaper.png Music/ public_html@
Videos/
bg7nyt.wav* Documents/ Mail/ nat.txt* script/
workspace/
BOINC/ Examples@ mbox Pictures/ Templates/
```

{ }通配符

```
ls {*.py,*.php,*.sh,shell}}
```

take a look at below

```
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -l'
alias ls='ls --color=auto'
```

full-time / time-style 定义日期时间格式

默认风格

```
[www@www.netkiller.cn ~]$ ls -l /var/log/message*
-rw----- 1 root root 302533 Jun 18 09:50 /var/log/messages
-rw----- 1 root root 392028 May 23 03:30 /var/log/messages-20160523
-rw----- 1 root root 334328 May 29 03:09 /var/log/messages-20160529
-rw----- 1 root root 395792 Jun 5 03:44 /var/log/messages-20160605
-rw----- 1 root root 308984 Jun 13 03:33 /var/log/messages-20160613
```

修改后

--full-time = --time-style=full-iso

```
[www@www.netkiller.cn ~]$ ls -l --full-time /var/log/messages*
-rw----- 1 root root 308659 2016-06-18 10:24:49.186979051 +0800
/var/log/messages
-rw----- 1 root root 392028 2016-05-23 03:30:01.869219181 +0800
/var/log/messages-20160523
-rw----- 1 root root 334328 2016-05-29 03:09:02.158442470 +0800
/var/log/messages-20160529
-rw----- 1 root root 395792 2016-06-05 03:44:02.424073354 +0800
/var/log/messages-20160605
-rw----- 1 root root 308984 2016-06-13 03:33:02.004785063 +0800
/var/log/messages-20160613

[www@www.netkiller.cn ~]$ ls -l --time-style=full-iso /var/log/messages*
-rw----- 1 root root 308659 2016-06-18 10:24:49.186979051 +0800
/var/log/messages
-rw----- 1 root root 392028 2016-05-23 03:30:01.869219181 +0800
/var/log/messages-20160523
-rw----- 1 root root 334328 2016-05-29 03:09:02.158442470 +0800
/var/log/messages-20160529
-rw----- 1 root root 395792 2016-06-05 03:44:02.424073354 +0800
/var/log/messages-20160605
-rw----- 1 root root 308984 2016-06-13 03:33:02.004785063 +0800
/var/log/messages-20160613
```

long-iso

```
[www@www.netkiller.cn ~]$ ls -lh --time-style long-iso /var/log/message*
-rw----- 1 root root 296K 2016-06-18 10:00 /var/log/messages
-rw----- 1 root root 383K 2016-05-23 03:30 /var/log/messages-20160523
-rw----- 1 root root 327K 2016-05-29 03:09 /var/log/messages-20160529
-rw----- 1 root root 387K 2016-06-05 03:44 /var/log/messages-20160605
-rw----- 1 root root 302K 2016-06-13 03:33 /var/log/messages-20160613
```

通过配置 TIME_STYLE 环境变量，改变日期格式

```
[www@www.netkiller.cn ~]$ export TIME_STYLE=long-iso

[www@www.netkiller.cn ~]$ ls -l /var/log/message*
-rw----- 1 root root 302533 2016-06-18 09:50 /var/log/messages
-rw----- 1 root root 392028 2016-05-23 03:30 /var/log/messages-
20160523
-rw----- 1 root root 334328 2016-05-29 03:09 /var/log/messages-
20160529
-rw----- 1 root root 395792 2016-06-05 03:44 /var/log/messages-
20160605
```



```

-rw----- 1 root root 308984 2016-06-13 03:33 /var/log/messages-20160613

[www@www.netkiller.cn ~]$ export TIME_STYLE=iso
[www@www.netkiller.cn ~]$ ls -l /var/log/message*
-rw----- 1 root root 302533 06-18 09:50 /var/log/messages
-rw----- 1 root root 392028 05-23 03:30 /var/log/messages-20160523
-rw----- 1 root root 334328 05-29 03:09 /var/log/messages-20160529
-rw----- 1 root root 395792 06-05 03:44 /var/log/messages-20160605
-rw----- 1 root root 308984 06-13 03:33 /var/log/messages-20160613

```

自定义格式

```

[www@www.netkiller.cn ~]$ ls -l --time-style="+%Y-%m-%d"
/var/log/message*
-rw----- 1 root root 302533 2016-06-18 /var/log/messages
-rw----- 1 root root 392028 2016-05-23 /var/log/messages-20160523
-rw----- 1 root root 334328 2016-05-29 /var/log/messages-20160529
-rw----- 1 root root 395792 2016-06-05 /var/log/messages-20160605
-rw----- 1 root root 308984 2016-06-13 /var/log/messages-20160613

[root@www.netkiller.cn ~]# export TIME_STYLE='+%Y/%m/%d %H:%M:%S'
[root@www.netkiller.cn ~]# ls -l /var/log/messages*
-rw----- 1 root root 189352 2016/06/18 10:20:01 /var/log/messages
-rw----- 1 root root 322453 2016/05/22 03:48:02 /var/log/messages-20160522
-rw----- 1 root root 247398 2016/05/30 03:37:01 /var/log/messages-20160530
-rw----- 1 root root 174633 2016/06/05 03:14:02 /var/log/messages-20160605
-rw----- 1 root root 196728 2016/06/12 03:17:01 /var/log/messages-20160612

```

cp - copy files and directories

copy directories recursively

```
cp -r /etc/* ~/myetc
```

覆盖已存在的文件

overwrite an existing file

-f, --force 覆盖已经存在的目标文件而不给出提示。 if an existing destination file cannot be opened, remove it and try again (this option is ignored when the -n option is also used)

当使用 -f 参数时仍然会提示询问覆盖

```
cp -f file1 file2  
cp: overwrite 'file2'?
```

使用 alias 命令查看，可以看到 cp 命令 增加 -i 参数，使用 unalias cp 可以删除 cp 别名。

```
# alias cp  
alias cp='cp -i'  
  
# unalias cp  
# alias cp  
-bash: alias: cp: not found
```

另一种解决方案是在 cp 前增加斜杠禁止别名

```
\cp -f file1 file2
```

-a, --archive same as **-dR --preserve=all**

```
# cp -a file file2
```

-a 参数可以保留原文件的日期与权限等等信息。

```
# ll  
-rw-r--r--. 1 root root      2559 Aug 27 05:00 yum.sh
```

```
# cp -a yum.sh yum1.sh
# cp yum.sh yum2.sh

# ll yum*
-rw-r--r--. 1 root root 2559 Aug 27 05:00 yum1.sh
-rw-r--r--. 1 root root 2559 Aug 27 05:58 yum2.sh
-rw-r--r--. 1 root root 2559 Aug 27 05:00 yum.sh
```

现在可以看到 yum1.sh 与 yum.sh 日期是相同的，而没有实现-a参数的 yum2.sh 日期为当前日期。

rm - remove files or directories

-bash: /bin/rm: Argument list too long

```
ls -l | xargs rm -f
find . -name 'spam-*' | xargs rm
find . -exec rm {} \;

ls | xargs -n 10 rm -fr # 10个为一组
```

zsh: sure you want to delete all the files in /tmp [yn]?

```
yes | rm -i file
```

df - report file system disk space usage

```
neo@netkiller:~$ df -lh
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        19G   3.1G   15G  17% /
none             996M   224K   996M   1% /dev
none            1000M     0 1000M   0% /dev/shm
none            1000M   520K 1000M   1% /var/run
none            1000M     0 1000M   0% /var/lock
none            1000M     0 1000M   0% /lib/init/rw
/dev/sda6        19G    13G   4.5G  75% /home
/dev/sda10       556M   178M   351M  34% /boot
/dev/sda7        46G   4.4G   40G  10% /var
/dev/sda8       367G    60G  289G  18% /opt
/dev/sda9        6.5G   143M   6.0G   3% /tmp
```

du - estimate file space usage

```
neo@netkiller:~$ sudo du -sh /usr/local
63M      /usr/local
```

tac - concatenate and print files in reverse

```
$ tac /etc/issue
Kernel \r on an \m
CentOS release 5.4 (Final)
```

split - split a file into pieces

按行分割文件

-l, --lines=NUMBER put NUMBER lines per output file

每10000行产生一个新文件

```
# split -l 10000 book.txt myfile
```

按尺寸分割文件

-b, --bytes=SIZE put SIZE bytes per output file

下面的例子是每10兆分割为一个新文件

```
split -b 10m large.bin new_file_prefix
```

find - search for files in a directory hierarchy

多目录匹配

`{/System,}/Library/Fonts`

匹配 `/System/Library/Fonts` 和 `/Library/Fonts` 两个目录

```
find {/System,}/Library/Fonts -name \*ttf
```

name

Find every file under directory /usr ending in "stat".

```
$ find /usr -name *stat
/usr/src/linux-headers-2.6.24-22-generic/include/config/cpu/freq/stat
/usr/bin/lnstat
/usr/bin/sar.sysstat
/usr/bin/mpstat
/usr/bin/rtstat
/usr/bin/nstat
/usr/bin/lpstat
/usr/bin/ctstat
/usr/bin/stat
/usr/bin/kpsestat
/usr/bin/pidstat
/usr/bin/iostat
/usr/bin/vmstat
/usr/lib/sysstat
/usr/share/doc/sysstat
/usr/share/gnome/help/battstat
/usr/share/omf/battstat
/usr/share/zsh/help/stat
/usr/share/zsh/4.3.4/functions/Completion/Unix/_diffstat
/usr/share/zsh/4.3.4/functions/Completion/Zsh/_stat
/usr/share/zsh/4.3.4/functions/Zftp/zfstat
```

```
find \( -iname '*.jpg' -o -iname '*.png' -o -iname '*.gif' \)

find /www/images -type f \( -iname '*.js' -o -iname '*.css' -o -iname
'*.html' \) | xargs tar -czf ~/images.tgz
```

使用通配符

```
find . -name "*.jsp" -delete
find . -name "*.xml" -delete
```

regex

```
find . -regex ".*\.(jpg|png)"
```

下面regex与name作用相同

```
find . -regex ".*\.(txt|sh)"  
find . -name "*.sh" -o -name "*.txt"
```

-regex参数, 使用正则表达式来匹配. 查找当前目录以及子目录中以 ".sh", 并改为以 ".shell" 结尾.

```
[neo@netkiller test]# tree a  
a  
├── a.py  
├── a.sh  
└── b  
    ├── b.py  
    ├── b.sh  
    ├── c  
    │   └── c.sh  
    └── d  
        └── d.sh  
  
[neo@netkiller test]# find ./a -type f -regex ".*\.sh$" | sed -r -n 's#  
(.*\.)sh$#mv & \1shell#e'  
[neo@netkiller test]# tree a  
a  
├── a.py  
├── a.shell  
└── b  
    ├── b.py  
    ├── b.shell  
    ├── c  
    │   └── c.shell  
    └── d  
        └── d.shell  
  
// 注意 sed s->e 使用方式, 官方文档是这样解释的.
```

This command allows one to pipe input from a shell command into pattern space. If a substitution was made, the command that is found in pattern space is executed and pattern space is replaced with its output. A trailing newline is suppressed; results are undefined if the command to be executed contains a NUL character. This is a GNU sed extension.

user

Find every file under /home and /var/www owned by the user neo.

```
$ find /home -user neo
$ find /var/www -user neo
$ find . -user nobody -iname '*.php'
```

perm

```
find ./ -perm -7 -print | xargs chmod o-w
find . -perm -o=w
```

查找当前目录下权限为777的文件并显示到标准输出

```
find ./ -type f -perm 777 -print
```

type

分别设置文件与目录的权限

```
find /usr/www/phpmyadmin -type d -exec chmod 755 {} \;
find /usr/www/phpmyadmin -type f -exec chmod 644 {} \;
```

-delete

```
# find /var/spool/clientmqueue/ -type f -delete
```

保留最近7天的问题，其他全部删除

```
find . -type f -mtime +7 -delete
```

exec

替换文本

```
# find ./ -exec grep str1 '{}' \; -exec sed -i.bak s/str1/str2/g '{}' \;
```

```
find -exec ls -l {} \; | grep '2011-01-18'
```

查找*.html文件中aaa替换为bbb

```
find . -name "*.html" -type f -exec sed -i "s/aaa/bbb/" {} \;
```

查找文件中含有openWindow字符串的文件

```
# find -type f -name "*.js" -exec grep -H -A2 openWindow {} \;

./javascript/commonjs.js:function openWindow(url){
./javascript/commonjs.js-         window.open(url + "?rand=" +
getRandom(), 'gamebinary');
./javascript/commonjs.js-}
```

```
find -type f -regex ".*\.(css|js)" -exec yuicompressor {} -o {} \;
find -type f -name "*.js" -exec yuicompressor --type js {} -o {} \;
find -type f -name "*.css" -exec yuicompressor --type css {} -o {} \;
```

排除目录

```
find /usr/local -path "/usr/local/share" -prune -o -print

find /usr/local \( -path /usr/local/bin -o -path /usr/local/sbin \) -
prune -o -print

find /usr/local \( -path /usr/local/dir1 -o -path /usr/local/file1 \) -
prune -o -name "temp" -print
```

查找当前目录下的php文件,排除子目录templates_c,caches


```
find . \( -path ./templates_c -o -path ./cache \) -prune -o -name  
"*.php" -print
```

-mmin n File's data was last modified n minutes ago.

```
# find . -mmin +5 -mmin -10
```

```
find /www -type f -mtime +60s
```

-ctime

查找当前目录下超过6天且是空文件并删除

```
find ./ -type d -empty -ctime +6 -exec rm -f {} \;
```

查找7天前的文件并删除

```
find /backup/svn/day -type f -ctime +7 -exec rm -f {} \;  
find /backup/svn/day -type f -ctime +7 -delete  
find /backup/svn/day -type f -ctime +7 | xargs rm -f
```

-mtime / -mmin

查询最近3天前内修改的文件

```
find . -type f -mtime -3
```

3天前

```
find . -type f -mtime +3
```

例 4.1. backup(find + tar)

```
find / -type f -mtime -7 | xargs tar -rf weekly_incremental.tar  
gzip weekly_incremental.tar
```

保留7天，删除7天的日志文件

```
COPIES=7  
find /var/log -type f -mtime +$COPIES -delete
```

--newer

```
tar --newer="2011-07-04" -zcvf backup.tar.gz /var/www/  
tar cvzf foo.tgz /bak -N "2004-03-03 16:49:17"
```

-print / -printf

```
[root@scientific ~]# find / -maxdepth 1 -name '[!..]*' -printf 'Name:  
%16f Size: %6s\n'  
Name:          / Size:    4096  
Name:          misc Size:    0  
Name:          media Size:  4096  
Name:          home Size:  4096  
Name:          dev Size:   3840  
Name:          net Size:    0  
Name:          proc Size:   0  
Name:          sbin Size:  12288  
Name:          root Size:  4096  
Name:          lib Size:   4096  
Name:          cgroup Size: 4096  
Name:          srv Size:   4096  
Name:          mnt Size:   4096  
Name:          etc Size:  12288  
Name:          usr Size:   4096  
Name:          lib64 Size: 12288  
Name:          boot Size:  1024  
Name:          var Size:   4096  
Name:          selinux Size: 0  
Name:          opt Size:   4096  
Name:          tmp Size:   4096  
Name:          lost+found Size: 16384  
Name:          sys Size:    0  
Name:          bin Size:   4096
```

```
# find /etc/ -type f -printf "%CY-%Cm-%Cd %Cr %8s %f\n"
```

-size

查找0字节文件

```
find /www -type f -size 0
```

查找根目录下大于1G的文件

```
find / -type f -size +1000M
```

-path

搜索当前目录下除了keys目录下所以子目录中的文件

```
find ./ -path "./keys" -prune -o -type f -print
```

find排除多个目录

```
find ./ \( -path ./conf -o -path ./logs \) -prune -o -print  
  
find /data/ \( -path /data/data_backup -o -path /data/mysql \) -prune -o  
-name "core.*" -type f  
/data/mysql  
/data/data_backup
```

ps 要么都是绝对路径 要么都是相对路径 /data/ 必须有"/" path 后面的路径必须没有"/"

包含 */target/* 目录

```
[gitlab-runner@localhost cloud.netkiller.cn]$ find . -type f -name  
"*.jar" -path "*/target/*"
```

排除 lib 目录

```
[gitlab-runner@localhost cloud.netkiller.cn]$ find . -type f -name "*.jar" ! -path "lib"
```

```
[gitlab-runner@localhost cloud.netkiller.cn]$ find . \( ! -path "*/zito-common/*" -a ! -path "./lib/*" \) -type f -name "*.jar"
```

目录深度控制

```
neo@MacBook-Pro ~/workspace/Linux % find */images -type d -d 0 -exec echo {} \;  
Cryptography/images  
Monitoring/images  
OpenLDAP/images  
Project/images  
Web/images
```

```
find */images -type d -d 0 -exec rsync -au {}/*  
$(PUBLIC_HTML)/linux/images \;
```

-maxdepth

-maxdepth和-mindepth，最大深度，最小深度搜索，搜索当前目录下最大深度为1的所以文件

```
find . -maxdepth 1 -type f
```

xargs

```
find /etc -type f|xargs md5sum
```

shasum

```
find /etc -type f|xargs shasum
```

```
find ./ -name "*.html" | xargs -n 1 sed -i -e 's/aaa/bbb/g'
```

```
find /tmp -name core -type f -print | xargs /bin/rm -f  
find . -type f -exec file '{}' \;
```

find后执行xargs提示xargs: argument line too long解决方法:

```
find . -type f -name "*.log" -print0 | xargs -0 rm -f
```

-i 参数可以使用 {}

```
[gitlab-runner@localhost cloud.sfzito.com]$ find . \( ! -path "*/zito-  
common/*" -a ! -path "./lib/*" -a ! -path "./dist/*" \) -type f -name  
"*.jar" | xargs -i cp {} dist/
```

5. package / compress and decompress

tar — The GNU version of the tar archiving utility

tar examples

tar

```
tar -cvf foo.tar foo/  
    tar contents of folder foo in foo.tar  
  
tar -xvf foo.tar  
    extract foo.tar
```

gunzip

```
tar -zcvf foo.tar foo/  
    tar contents of folder foo in foo.tar.gz
```

```
tar -xvzf foo.tar.gz  
    extract gzipped foo.tar.gz
```

b2zip

b2zip

```
tar -jcvf foo.tar.bz2 foo/  
    tar contents of folder foo in foo.tar.bz2  
  
tar -jxvf foo.tar.bz2  
    extract b2zip foo.tar.bz2
```

compress

compress/uncompress

```
tar -Zcvf foo.tar.Z foo/
    tar contents of folder foo in foo.tar.Z

tar -Zxvf foo.tar.Z
    extract compress foo.tar.Z
```

.xz 文件

```
tar -Jxf file.pkg.tar.xz
```

-t, --list

-t, --list list the contents of an archive

列出tar包中的文件

```
tar tvf neo.tar.gz
```

```
# mkdir -p /www/test.com/www.test.com/
# echo helloworld > /www/test.com/www.test.com/test.txt
# tar zcvf www.test.com.tar.gz /www/test.com/www.test.com/

# tar ztvf www.test.com.tar.gz
drwxr-xr-x root/root          0 2013-08-08 15:24
www/test.com/www.test.com/
-rw-r--r-- root/root        11 2013-08-08 15:24
www/test.com/www.test.com/test.txt

# tar zxvf www.test.com.tar.gz
www/test.com/www.test.com/
```

```
www/test.com/www.test.com/test.txt  
  
# find www  
www  
www/test.com  
www/test.com/www.test.com  
www/test.com/www.test.com/test.txt
```

tar: Removing leading `/' from member names

-P, --absolute-names don't strip leading `/'s from file names

```
$ tar -czvPf neo.tar.gz /home/neo/  
$ tar -xzvPf neo.tar.gz
```

```
tar zcvfP www.test.com.tar.gz /www/test.com/www.test.com/  
tar zxvfP www.test.com.tar.gz
```

-C, --directory=DIR

-C, --directory=DIR change to directory DIR

解压到目标目录

```
tar -xzvf neo.tar.gz -C /tmp
```

```
# tar zxvf www.test.com.tar.gz -C /tmp  
www/test.com/www.test.com/  
www/test.com/www.test.com/test.txt  
  
# find /tmp/www/  
/tmp/www/  
/tmp/www/test.com  
/tmp/www/test.com/www.test.com  
/tmp/www/test.com/www.test.com/test.txt
```



```
# rm -rf /www/test.com/*

# tar zxvf www.test.com.tar.gz -C /
www/test.com/www.test.com/
www/test.com/www.test.com/test.txt

# find /www/test.com/
/www/test.com/
/www/test.com/www.test.com
/www/test.com/www.test.com/test.txt
```

--exclude

排除neo目录

```
tar --exclude /home/neo -zcvf myfile.tar.gz /home/* /etc

tar zcvf rpmbuild/SOURCES/netkiller-1.0.tar.gz
~/workspace/public_html/* --exclude .git --exclude .svn
```

-T

```
find . -name "*.jpg" -print >list
tar -T list -czvf picture.tar.gz

find /etc/ | tar czvf xxx1.tar.gz -T -
```

日期过滤

打包 2010/08/01 之后的文件和目录

```
tar -N '2010/08/01' -zcvf home.tar.gz /home
```

保留权限

```
tar -zxvpf /tmp/etc.tar.gz /etc
```

-r, --append

追加最近7天更改过的文件

```
find / -type f -mtime -7 | xargs tar -rf weekly_incremental.tar
```

远程传输

tar -jcpvf - file | ssh remote "tar -jxpvf -"

```
tar -jcpvf - file.php | ssh root@172.16.3.1 "tar -jxpvf -"
```

分卷压缩

分卷压缩一个目录：如doc

在doc目录的上次目录

```
#tar cvf doc | split -b 2m (已2M大小分卷压缩)  
#cat x* > doc.tar (合成分卷压缩包)
```

或者

```
#tar czvf doc.tar.gz doc/  
#tar czvfp - doc.tar.gz | split -b 5m  
#cat x* > doc.tar.gz
```

查看压缩包里面的内容：

```
#tar -tf doc.tar
#tar -tzvf doc.tar.gz
```

cpio - copy files to and from archives

```
find /opt -print | cpio -o > opt.cpio

find . -type f -name '*.sh' -print | cpio -o | gzip >sh.cpio.gz

cpio -i < opt.cpio
```

gzip

gzip/gunzip

```
# ls access.2010-{10,11}-???.log
access.2010-10-01.log  access.2010-10-17.log  access.2010-11-
02.log  access.2010-11-18.log
access.2010-10-02.log  access.2010-10-18.log  access.2010-11-
03.log  access.2010-11-19.log
access.2010-10-03.log  access.2010-10-19.log  access.2010-11-
04.log  access.2010-11-20.log
access.2010-10-04.log  access.2010-10-20.log  access.2010-11-
05.log  access.2010-11-21.log
access.2010-10-05.log  access.2010-10-21.log  access.2010-11-
06.log  access.2010-11-22.log
access.2010-10-06.log  access.2010-10-22.log  access.2010-11-
07.log  access.2010-11-23.log
access.2010-10-07.log  access.2010-10-23.log  access.2010-11-
08.log  access.2010-11-24.log
access.2010-10-08.log  access.2010-10-24.log  access.2010-11-
09.log  access.2010-11-25.log
access.2010-10-09.log  access.2010-10-25.log  access.2010-11-
10.log  access.2010-11-26.log
access.2010-10-10.log  access.2010-10-26.log  access.2010-11-
11.log  access.2010-11-27.log
access.2010-10-11.log  access.2010-10-27.log  access.2010-11-
```

```
12.log  access.2010-11-28.log
access.2010-10-12.log  access.2010-10-28.log  access.2010-11-
13.log  access.2010-11-29.log
access.2010-10-13.log  access.2010-10-29.log  access.2010-11-
14.log  access.2010-11-30.log
access.2010-10-14.log  access.2010-10-30.log  access.2010-11-
15.log
access.2010-10-15.log  access.2010-10-31.log  access.2010-11-
16.log
access.2010-10-16.log  access.2010-11-01.log  access.2010-11-
17.log
# gzip access.2010-{10,11}-???.log
```

```
# ls access.2010-{0?,10,11}-???.log
access.2010-08-28.log  access.2010-10-02.log  access.2010-10-
13.log  access.2010-10-27.log  access.2010-11-06.log
access.2010-11-17.log  access.2010-11-26.log
access.2010-08-31.log  access.2010-10-03.log  access.2010-10-
14.log  access.2010-10-28.log  access.2010-11-08.log
access.2010-11-18.log  access.2010-11-27.log
access.2010-09-24.log  access.2010-10-04.log  access.2010-10-
15.log  access.2010-10-29.log  access.2010-11-09.log
access.2010-11-19.log  access.2010-11-28.log
access.2010-09-25.log  access.2010-10-06.log  access.2010-10-
17.log  access.2010-10-30.log  access.2010-11-10.log
access.2010-11-20.log  access.2010-11-29.log
access.2010-09-26.log  access.2010-10-07.log  access.2010-10-
19.log  access.2010-10-31.log  access.2010-11-11.log
access.2010-11-21.log  access.2010-11-30.log
access.2010-09-27.log  access.2010-10-08.log  access.2010-10-
20.log  access.2010-11-02.log  access.2010-11-12.log
access.2010-11-22.log
access.2010-09-29.log  access.2010-10-09.log  access.2010-10-
22.log  access.2010-11-03.log  access.2010-11-14.log
access.2010-11-23.log
access.2010-09-30.log  access.2010-10-10.log  access.2010-10-
23.log  access.2010-11-04.log  access.2010-11-15.log
access.2010-11-24.log
access.2010-10-01.log  access.2010-10-12.log  access.2010-10-
25.log  access.2010-11-05.log  access.2010-11-16.log
access.2010-11-25.log
# gzip access.2010-{0?,10,11}-???.log &
```

zip, zipcloak, zipnote, zipsplit - package and compress (archive) files

*.zip

zip/unzip file[.zip]

压缩文件

```
zip -r dist.zip dist
```

解压到指定目录

```
unzip dist.zip -d /var/www/html
```

bzip2, bunzip2 - a block-sorting file compressor

```
[root@localhost src]# yum install bzip2
```

查看RPM包所含文件

```
[root@localhost src]# rpm -ql bzip2-1.0.6-13.el7
/usr/bin/bunzip2
/usr/bin/bzcat
/usr/bin/bzcmp
```

```
/usr/bin/bzdiff
/usr/bin/bzgrep
/usr/bin/bzip2
/usr/bin/bzip2recover
/usr/bin/bzless
/usr/bin/bzmore
/usr/share/doc/bzip2-1.0.6
/usr/share/doc/bzip2-1.0.6/CHANGES
/usr/share/doc/bzip2-1.0.6/LICENSE
/usr/share/doc/bzip2-1.0.6/README
/usr/share/man/man1/bunzip2.1.gz
/usr/share/man/man1/bzcat.1.gz
/usr/share/man/man1/bzcmp.1.gz
/usr/share/man/man1/bzdiff.1.gz
/usr/share/man/man1/bzgrep.1.gz
/usr/share/man/man1/bzip2.1.gz
/usr/share/man/man1/bzip2recover.1.gz
/usr/share/man/man1/bzless.1.gz
/usr/share/man/man1/bzmore.1.gz
```

RAR

```
sudo apt-get install rar unrar
```

7-Zip

p7zip - 7z file archiver with high compression ratio

<http://www.7-zip.org/>

如果你仅仅是解压文件，只需安装下面的包即可

```
$ sudo apt-get install p7zip
```

如果你要创建7zip文件就需要安装p7zip-full

```
$ sudo apt-get install p7zip-full
```

压缩

```
$ 7z a test.7z /etc/*
```

浏览压缩包

```
$ 7z l test.7z
```

解压

```
$ 7z e test.7z
```

Creates self extracting archive.

创建自解压文件

```
7z a -sfx a.7z *.txt
```

解压

```
./a.7z
```

RAR

```
$ unrar test.rar
```

xz, unxz, xzcat, lzma, unlzma, lzcat - Compress or decompress .xz and .lzma files

```
[root@localhost ~]# echo "Hello" > test
[root@localhost ~]# xz -z test
[root@localhost ~]# ll test.xz
-rw----- 1 root root 1436 2019-01-16 06:13 test.xz
[root@localhost ~]# xz -d test.xz
[root@localhost ~]# cat test
Hello
```

tar 用法

```
[root@localhost ~]# tar Jcvf test.tar.xz test
test
[root@localhost ~]# ll test.tar.xz
-rw-r--r-- 1 root root 1528 2019-03-19 04:32 test.tar.xz

[root@localhost ~]# tar Jxvf test.tar.xz
test
```


6. 日期和时间

date and time

日期格式

自定义格式化显示日期

```
%n : 下一行
%t : 跳格
%H : 小时(00..23)
%I : 小时(01..12)
%k : 小时(0..23)
%l : 小时(1..12)
%M : 分钟(00..59)
%p : 显示本地 AM 或 PM
%r : 直接显示时间 (12 小时制, 格式为 hh:mm:ss [AP]M)
%s : 从 1970 年 1 月 1 日 00:00:00 UTC 到目前为止的秒数
%S : 秒(00..61)
%T : 直接显示时间 (24 小时制)
%X : 相当于 %H:%M:%S
%Z : 显示时区 %a : 星期几 (Sun..Sat)
%A : 星期几 (Sunday..Saturday)
%b : 月份 (Jan..Dec)
%B : 月份 (January..December)
%c : 直接显示日期与时间
%d : 日 (01..31)
%D : 直接显示日期 (mm/dd/yy)
%h : 同 %b
%j : 一年中的第几天 (001..366)
%m : 月份 (01..12)
%U : 一年中的第几周 (00..53) (以 Sunday 为一周的第一天情形)
%w : 一周中的第几天 (0..6)
%W : 一年中的第几周 (00..53) (以 Monday 为一周的第一天情形)
%x : 直接显示日期 (mm/dd/yy)
%y : 年份的最后两位数字 (00..99)
%Y : 完整年份 (0000..9999)
```

2010/06/18 17:57:38

```
$ date '+%Y/%m/%d %H:%M:%S'
```

2010-06-18 17:57:58

```
$ date '+%Y-%m-%d %H:%M:%S'
```

```
$ date '+%Y-%m-01 00:00:01'  
2010-10-01 00:00:01
```

```
[root@netkiller ~]# date +%F  
2015-07-30
```

```
[root@netkiller ~]# date +%Y-%m-%d-%H-%M  
2015-07-30-13-49
```

weekday name

```
$ date +%a  
Fri
```

```
$ date +%A  
Friday
```

-d --date=

```
# date -d next-day +%Y%m%d  
20060328  
# date -d last-day +%Y%m%d  
20060326  
# date -d yesterday +%Y%m%d  
20060326  
# date -d tomorrow +%Y%m%d
```

```
20060328
# date -d last-month +%Y%m
200602
# date -d next-month +%Y%m
200604
# date -d next-year +%Y
2007
```

date 命令的另一个扩展是 -d 选项

1) 2周后的日期 和一天前的日期

```
[root@netkiller ~]# date -d '2 weeks'
2015年 08月 13日 星期四 13:53:06 HKT
```

```
[root@netkiller ~]# date -d '1 day ago'
2015年 07月 29日 星期二 13:53:14 HKT
```

```
[root@netkiller ~]# date -d yesterday
2015年 07月 29日 星期三 13:53:26 HKT
```

2) 下周一的日期

```
[root@netkiller ~]# date -d 'next monday'
2015年 08月 03日 星期一 00:00:00 HKT
```

3) 使用负数以得到相反的时间

```
[root@netkiller ~]# date -d '-1 weeks'
2015年 07月 23日 星期四 13:59:43 HKT
```

```
[root@netkiller ~]# date -d '1 weeks'
2015年 08月 06日 星期四 13:59:50 HKT
```

上个月的一周前

```
[root@netkiller ~]# date -d 'last-month -1 week'
2015年 06月 23日 星期二 14:00:59 HKT
```

相对于6月30号的前两周

```
[root@netkiller ~]# date -d 'jun 30 -2 weeks'
2015年 06月 16日 星期二 00:00:00 HKT
```

```
[root@netkiller ~]# date -d 'jun 30 -2 weeks' +%Y_%m_%d
2015_06_16
```

日期偏移量

昨天 (前一天)

```
date --date='1 days ago' "+%Y-%m-%d"
```

```
date -d '1 days ago' "+%Y-%m-%d"
```

```
date -d yesterday "+%Y-%m-%d"
```

明天 (後一天)

```
date --date='1 days' "+%Y-%m-%d"
```

```
date -d '1 days' "+%Y-%m-%d"
```

```
date -d tomorrow "+%Y-%m-%d"
```

day

```
$ date -d '-1 day' '+%Y-%m-%d 00:00:01'
```

```
2010-10-14 00:00:01
```

```
$ date -d '+5 day' '+%Y-%m-%d 00:00:01'
```

```
2010-10-20 00:00:01
```

month

```
$ date -d '+2 month' '+%Y-%m-%d 00:00:01'
```

```
2010-12-15 00:00:01
```

```
$ date -d '-1 month' '+%Y-%m-%d 00:00:01'
```

```
2010-09-15 00:00:01
```

year

```
$ date -d '-5 year' '+%Y-%m-%d'
```

```
2005-10-15
```

```
$ date -d '+1 year' '+%Y-%m-%d'
```

```
2011-10-15
```

时间偏移

```
1小時前
date --date='1 hours ago' "+%Y-%m-%d %H:%M:%S"
1小時後
date --date='1 hours' "+%Y-%m-%d %H:%M:%S"
1分鐘前
date --date='1 minutes ago' "+%Y-%m-%d %H:%M:%S"
1分鐘後
date --date='1 minutes' "+%Y-%m-%d %H:%M:%S"
1秒前
date --date='1 seconds ago' "+%Y-%m-%d %H:%M:%S"
1秒後
date --date='1 seconds' "+%Y-%m-%d %H:%M:%S"
```

时间戳

1 计算当天的时间戳

```
[root@netkiller ~]# date +%s
1440641485
```

2 计算指定日期的时间戳

```
[root@netkiller ~]# date -d "2015-08-05 09:45:44" +%s
1438739144
```

3 时间戳转换成时间

```
[root@netkiller ~]# date -d @1438739144
2015年 08月 05日 星期三 09:45:44 HKT
```

格式输出

```
[root@mygitlab ~]# date -d @1440661395 "+%Y-%m-%d-%H-%M"
2015-08-27-00-43
```

RFC 2822

RFC 2822 的日期与时间输出格式, RFC 2822 的格式像这样: 星期, 日-月-年, 小时:分钟:秒 时区

时区 +0800 等同于 GMT +8

```
[root@netkiller ~]# date -R
Thu, 30 Jul 2015 11:29:00 +0800
```

UTC

UTC time 以UTC形式显示日期和时间

```
$ datetime=$(date -u '+%Y%m%d %H:%M:%S')
$ echo $datetime
20091203 06:22:03
```

```
[root@netkiller ~]# date -u
2015年 07月 30日 星期四 03:35:01 UTC
```

字符串转日期

```
[root@netkiller ~]# date -d "$(echo 20220103T1430 | sed 's/T/ /')"
```

```
Fri Jan  3 14:30:00 CST 2022
```

```
[root@netkiller ~]# date -d "$(echo 20220103T143011 | sed -r 's/(.*)T(..)(..)(..)/\1 \2:\3:\4/')
```

```
Mon Jan  3 14:30:11 CST 2022
```



7. 数值与运算

数值运算

```
echo $((3+5))  
expr 6 + 3  
awk 'BEGIN{a=(3+2)*2;print a}'
```

seq - print a sequence of numbers

```
[neo@test ~]$ seq 10  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
[neo@test ~]$ seq 5 10  
5  
6  
7  
8  
9  
10
```

等差列, 步长设置


```
$ seq 1 1 10
1
2
3
4
5
6
7
8
9
10

$ seq 1 2 10
1
3
5
7
9

# seq 0 2 10
0
2
4
6
8
10
```

分隔符

```
# seq -s : -w 1 10
01:02:03:04:05:06:07:08:09:10

# seq -s '|' -w 1 10
01|02|03|04|05|06|07|08|09|10
```

等宽，前导字符用0填充

```
# seq -w 1 10
01
02
03
04
05
06
07
08
09
10
```

bc - An arbitrary precision calculator language

```
$ echo "4*5" | bc
```

```
# more calc.txt
3+2
4+5
8*2
10/4
# bc calc.txt
5
9
16
2
```

8. 文本处理

iconv - Convert encoding of given files from one encoding to another

cconv - A iconv based simplified-traditional chinese conversion tool

cconv是建立在iconv之上，可以UTF8编码直接转换，并增加了词转换。

```
sudo apt-get install cconv
```

使用cconv进行简繁转换的方法为：

```
cconv -f UTF8-CN -t UTF8-HK zh-cn.txt -o zh-hk.txt
```

uconv - convert data from one encoding to another

安装

```
sudo apt-get install libicu-dev
```

例子

```
$ uconv -f cp1252 -t UTF-8 -o file_in_utf8.txt  
file_in_cp1252_encoding.txt
```

字符串处理命令**expr**

字符串处理命令**expr**用法简介：

名称：expr

用途：求表达式变量的值。

语法：expr Expression

实例如下：

例子1：字符串长度

```
shell>> expr length "this is a test content";
```

```
22
```

例子2:求余数

```
shell>> expr 20 % 9
```

2

例子3:从指定位置处截取字符串

```
shell>> expr substr "this is a test content" 3 5
```

is is

例子4:指定字符串第一次出现的位置

```
shell>> expr index "testforthe game" s
```

3

例子5:字符串真实重现

```
shell>> expr quote thisisatestformela
```

thisisatestformela

cat - concatenate files and print on the standard output

-b	不对空白行编号。
-e	使用 \$ 字符显示行尾。
-n	从 1 开始对所有输出行编号。
-q	使用静默操作（禁止错误消息）。
-r	将所有多个空行替换为单行（“压缩”空白）。
-t	将制表符显示为 ^I。
-u	不对输出进行缓冲。
-v	可视地显示非打印控制字符。

-s, --squeeze-blank suppress repeated empty output lines

-S 将多个空白行压缩到单行中（与 -r 相同）

```
$ cat >> /tmp/test <<EOF
```

Line1

Line2

Line3

Line4

Line5

```
EOF

$ cat -s /tmp/test
Line1

Line2

Line3

Line4

Line5
```

-v, --show-nonprinting use ^ and M- notation, except for LFD and TAB

显示控制字符。例如Tab等，下面例子查看文件结尾换行符类型

```
[neo@netkiller ~]# cat -v file.txt
GRANT USAGE ON *.* TO 'esouser'@'localhost' IDENTIFIED BY xxxxxxxx; ^M
^M
file^M
2059^M
```

与管道配合使用

```
[log@logging tmp]$ cat <<EOF | grep 'm'
İsmail
Ahmet
Ali
Elif
Mehmet
EOF
İsmail
Ahmet
Mehmet
```

多管道

```
cat <<EOF | grep 'm' | tee matched_names.txt
İsmail
Ahmet
Ali
Elif
Mehmet
EOF
```

nl - number lines of files

```
$ nl /etc/issue
 1 CentOS release 5.4 (Final)
 2 Kernel \r on an \m
```

tr - translate or delete characters

```
[ :alnum: ] : 所有字母字符与数字
[ :alpha: ] : 所有字母字符
[ :blank: ] : 所有水平空格
[ :cntrl: ] : 所有控制字符
[ :digit: ] : 所有数字
[ :graph: ] : 所有可打印的字符 (不包含空格符)
[ :lower: ] : 所有小写字母
[ :print: ] : 所有可打印的字符 (包含空格符)
[ :punct: ] : 所有标点字符
[ :space: ] : 所有水平与垂直空格符
[ :upper: ] : 所有大写字母
[ :xdigit: ] : 所有 16 进位制的数字
```

替换字符

":"替换为"\n"

```
$ cat /etc/passwd |tr ":" "\n"
```

```
[root@gitlab ~]# echo "/opt/netkiller.cn/www.netkiller.cn" | tr -- '/' ':'  
:-  
:opt:netkiller-cn:www-netkiller-cn
```

英文大小写转换

使用 `tr '[:lower:]' '[:upper:]'` 将小写字母替换成大写字母

```
[root@localhost ~]# echo "Helloworld" | tr '[:lower:]' '[:upper:]'  
HELLOWORLD
```

替换整段文字

```
[root@localhost ~]# cat /etc/redhat-release  
CentOS Linux release 7.5.1804 (Core)  
  
[root@localhost ~]# cat /etc/redhat-release | tr '[:lower:]' '[:upper:]'  
CENTOS LINUX RELEASE 7.5.1804 (CORE)
```

```
[root@localhost ~]# echo "Netkiller" | tr 'a-z' 'A-Z'  
NETKILLER
```

[CHAR*] 和 [CHAR*REPEAT]

```
[root@localhost ~]# echo "1234567890" | tr '1-5' '[A*]'  
AAAAA67890  
  
[root@localhost ~]# echo "1234567890" | tr '1-9' '[A*5]BCDE'  
AAAAABCDE0
```

-s, --squeeze-repeats replace each input sequence of a repeated character that is listed in SET1 with a single occurrence of that character

删除重复的字符

```
[root@localhost ~]# echo "My      nickname  is      netkiller." | tr -s ' '
My nickname is netkiller.

[root@localhost ~]# echo "aaaabbbbccccdddd." | tr -s 'a'
abbbbccccdddd.

[root@localhost ~]# echo "aaaabbbbccccdddd." | tr -s 'a-z'
abcd.
```

-d, --delete delete characters in SET1, do not translate

删除字符

```
[root@localhost ~]# echo "My nickname is netkiller" | tr -d ' '
Mynicknameisnetkiller

[root@localhost ~]# md5sum /etc/issue | tr -d [0-9]
ffedfcfbdcabdec /etc/issue
```

删除控制字符

```
[root@netkiller ~]# cat file | tr -d [:cntrl:]
```

cut - remove sections from each line of files

列操作

```
$ last | grep 'neo' | cut -d ' ' -f1
```



```
$ cat /etc/passwd | cut -d ':' -f1
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy

$ cat /etc/passwd | cut -d ':' -f1,3,4

# cat /etc/passwd | cut -d ':' -f1,6
root:/root
bin:/bin
daemon:/sbin
adm:/var/adm
lp:/var/spool/lpd
sync:/sbin
shutdown:/sbin
halt:/sbin
mail:/var/spool/mail
uucp:/var/spool/uucp
operator:/root
games:/usr/games
gopher:/var/gopher
ftp:/var/ftp
nobody:/
vcsa:/dev
saslauth:/var/empty/saslauth
postfix:/var/spool/postfix
sshd:/var/empty/sshd
rpc:/var/cache/rpcbind
rpcuser:/var/lib/nfs
nfsnobody:/var/lib/nfs
ntp:/etc/ntp
nagios:/var/log/nagios
```

行操作

```
$ cat /etc/passwd | cut -c 1-4
root
daem
```

```
bin:
sys:
sync
game
man:

$ echo "No such file or directory" | cut -c4-7
such

$ echo "No such file or directory" | cut -c -8
No such

$ echo "No such file or directory" | cut -c-8
No such
```

printf - format and print data

```
printf "%d\n" 1234
```

```
$ printf "\033[1;33m TEST COLOR \n\033[m"
```

Free 'recode' converts files between various character sets and surfaces.

Following will convert text files between DOS, Mac, and Unix line ending styles:

```
$ recode /cl../cr <dos.txt >mac.txt
$ recode /cr.. <mac.txt >unix.txt
$ recode ../cl <unix.txt >dos.txt
```

/dev/urandom 随机字符串

```
[neo@test .deploy]$ echo `< /dev/urandom tr -dc A-Z-a-z-0-9 | head -c 8`
GidAuuNN
[neo@test .deploy]$ echo `< /dev/urandom tr -dc A-Z-a-z-0-9 | head -c 8`
UyGaWSKr
```

我常常使用这样的随机字符初始化密码

```
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:alnum:] | head -c 8`  
xig8Meym  
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:alnum:] | head -c 8`  
23Ac1vZg  
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:digit:] | head -c 8`  
73652314  
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:graph:] | head -c 8`  
GO_o>OnJ  
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:graph:] | head -c 10`  
iGy0FS/a05  
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:graph:] | head -c 50`  
;`E^{5(T4v~5$YovW.?$_?9la<`+qPcRh@7mD\!Whx;MJZVQ\K  
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:print:] | head -c 50`  
fy$[#:'(')jt'gp1/g-)d~p]8 :r9i;MO2d!8M<?Qs3t:QgK$0  
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:graph:] | head -c 50`  
6SivJ5y$/FTi8mf}rrqE&s0"WkA}r;uK-=MT!Wp0U1L_lF0|bL
```

批量生成

```
for i in {1..10}  
do  
echo `< /dev/urandom tr -dc A-Z-a-z-0-9 | head -c 8`  
done
```

```
# cat /dev/urandom | tr -cd [:alnum:] | fold -w30 | head -n 20  
AVqROzjF6ZATJGv2J6PzDHP3jLpKV4  
ONt68UFNDwgXpSnLBV7oRDX3VLRYSX  
EZTWCGvZc3mIEeuw9sxMtV8ZkzVRJv  
BhUiv0a7utsjZFLYpKGZrY5aDXcZL4  
5YfU12hmDT1O9X61DRYg4wSp4lXoXX  
ykyPJxH47PzxnNGlujiUF98ZtB01H0  
QyP53mksQN8bCNNo1fSD3RtqhhEGfa  
u2RkT1M9GUQF4a6O18tG5WD97OOXze  
Whm5X7398Q8L9BONN8k2oLy9CL37JO  
TmGQz7WB6WnkjhyB4wrBHBj3HMIRyf
```

```
hww43yvddUDYUnbNOKjhv3sLhCA4YD
uY6zQtBC6miwLUl3jkCVVA0Xu8ASgj
jv58qu46VW7LvRIq4txNE8bG9NB1Zl
pzaMkydAiCHCF5H2oQVqMn4DTTYgNL
yoN2A9LyrCwLfjPlad9HMAwxExJL5i
J27iy2L90m9dpcPLJ8tl46GGb9xqmQ
6YwFCvuPHyyEwnctUTpqLFcvUafVZ2
Nuq9XgIgRQGynjlVqGLMOpO0MkGpsn
tChkRG7eoRuKVXgW7ccTGx45E54K3Y
qPv48XqdGlOrdULCOGZ45kwJ1v5kVX
```

col - filter reverse line feeds from input

清除 ^M 字符

```
$ cat oldfile | col -b > newfile
```

apg - generates several random passwords

```
sudo apt-get install apg

$ apg

Please enter some random data (only first 16 are significant)
(eg. your old password):>
imlogNukcel5 (im-log-Nuk-cel-FIVE)
Drocdafl (Droc-daf-ONE)
fagJook0 (fag-Jook-ZERO)
heabugJer4 (heab-ug-Jer-FOUR)
5OsEsudy (FIVE-Os-Es-ud-y)
IrjOgneagOc9 (Irj-Og-neag-Oc-NINE)

$ apg -M SNCL -m 16
WoidWemFut6dryn,
byRowpEus-Flutt0
|QuogCagFaycsic0
ojHoadCyct4Freg_
Vir9blir`orhohoo
bapOip?Ibreawov2
```

head/tail

```
head -c 17 | tail -c 1
```

彩色输出

```
printf "%s" $(printf '\033[0;31m'); tail /etc/passwd
```

```
tail -f example.log | sed \
-e "s/FATAL/"$'\e[31m'&"$'\e[m'"/" \
-e "s/ERROR/"$'\e[31m'&"$'\e[m'"/" \
-e "s/WARNING/"$'\e[33m'&"$'\e[m'"/" \
-e "s/INFO/"$'\e[32m'&"$'\e[m'"/" \
-e "s/DEBUG/"$'\e[34m'&"$'\e[m'"/"
```

跳过 **n** 行，输出后面内容

首先看看源文件内容

```
[root@netkiller ~]# head -n 5 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

现在跳过第一行，显示后面所有内容

```
[root@netkiller ~]# tail -n +2 /etc/passwd
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
```

```
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
systemd-coredump:x:999:996:systemd Core Dumper:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
tss:x:59:59:Account used for TPM access:/dev/null:/sbin/nologin
polkitd:x:998:995:User for polkitd:/:/sbin/nologin
sssd:x:997:994:User for sssd:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/usr/share/empty.sshd:/sbin/nologin
systemd-oom:x:992:992:systemd Userspace OOM Killer:/:/usr/sbin/nologin
mysql:x:27:27:MySQL Server:/var/lib/mysql:/sbin/nologin
chrony:x:991:991:/:/var/lib/chrony:/sbin/nologin
docker:x:990:990:Container Administrator:/home/docker:/bin/bash
```

尾部剪掉 n 行

```
[root@netkiller ~]# nmap -F 121.196.46.109
Starting Nmap 7.91 ( https://nmap.org ) at 2022-08-01 14:52 CST
Nmap scan report for 121.196.46.109
Host is up (0.016s latency).
Not shown: 97 filtered ports
PORT      STATE SERVICE
113/tcp    closed ident
2000/tcp   open  cisco-sccp
5060/tcp   open  sip

Nmap done: 1 IP address (1 host up) scanned in 4.38 seconds
[root@netkiller ~]# nmap -F 121.196.46.109 | tail -n +5
PORT      STATE SERVICE
113/tcp    closed ident
2000/tcp   open  cisco-sccp
5060/tcp   open  sip

Nmap done: 1 IP address (1 host up) scanned in 1.83 seconds
[root@netkiller ~]# nmap -F 121.196.46.109 | tail -n +5 | head -n -1
PORT      STATE SERVICE
113/tcp    closed ident
2000/tcp   open  cisco-sccp
5060/tcp   open  sip
```

反转字符串或文件内容

rev - reverse lines of a file or files

反转字符串

```
# echo hello | rev
olleh

# echo "hello world" | rev
dlrow olleh
```

反转文件内容

```
# rev /etc/passwd
hsab/nib/:toor/:toor:0:0:x:toor
nigolon/nibs/:nib/:nib:1:1:x:nib
nigolon/nibs/:nibs/:nomead:2:2:x:nomead
nigolon/nibs/:mda/rav/:mda:4:3:x:mda
nigolon/nibs/:dpl/loops/rav/:pl:7:4:x:pl
cnys/nib/:nibs/:cnys:0:5:x:cnys
nwodtuhs/nibs/:nibs/:nwodtuhs:0:6:x:nwodtuhs
tlah/nibs/:nibs/:tlah:0:7:x:tlah
nigolon/nibs/:liam/loops/rav/:liam:21:8:x:liam
nigolon/nibs/:pcuu/loops/rav/:pcuu:41:01:x:pcuu
nigolon/nibs/:toor/:rotarepo:0:11:x:rotarepo
nigolon/nibs/:semag/rsu/:semag:001:21:x:semag
nigolon/nibs/:rehpog/rav/:rehpog:03:31:x:rehpog
nigolon/nibs/:ptf/rav/:resU PTF:05:41:x:ptf
nigolon/nibs/:/:ydoN:99:99:x:ydoN
nigolon/nibs/:ved/:renwo yromem elosnoc lautriv:96:96:x:ascv
nigolon/nibs/:ptn/cte/::83:83:x:ptn
nigolon/nibs/:htualsas/ytpme/rav/:"resu dhtualsaS":67:994:x:htualsas
nigolon/nibs/:xiftsop/loops/rav/::98:98:x:xiftsop
nigolon/nibs/:dhss/ytpme/rav/:HSS detarapes-egelvirP:47:47:x:dhss
hsab/nib/:lqsym/bil/rav/:revres LQSyM:994:894:x:lqsym
hsab/nib/:www/:noitacilppA beW:08:08:x:www
nigolon/nibs/:xnign/ehcac/rav/:resu xnign:894:794:x:xnign
```

TAB符号与空格处理

expand - convert tabs to spaces

转换 TAB 字符为空格

```

root@netkiller /var/log % yum --showduplicates list httpd | expand
Repository epel is listed more than once in the configuration
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
Available Packages
httpd.x86_64                2.4.6-67.el7.centos
os
httpd.x86_64                2.4.6-67.el7.centos.2
updates

```

unexpand - convert spaces to tabs

转换空格为TAB符

```

root@netkiller /var/log % cat /etc/fstab | unexpand -t 16
/dev/vda1          /          ext3          noatime,acl,user_xattr 1 1
proc              /proc      proc          defaults      0 0
sysfs             /sys       sysfs         noauto       0 0
debugfs           /sys/kernel/debug debugfs       noauto       0 0
devpts            /dev/pts   devpts        mode=0620,gid=5 0 0

```

将16个空格替换为一个TAB符

grep, egrep, fgrep, rgrep - print lines matching a pattern

删除空行

```
$ cat file | grep '.'
```

-v, --invert-match

grep -v "grep"

```

[root@development ~]# ps ax | grep httpd
 6284 ?        Ss      0:10 /usr/local/httpd-2.2.14/bin/httpd -k start
 8372 ?        S        0:00 perl ./wrapper.pl -chdir -name httpd -class
com.caucho.server.resin.Resin restart
19136 ?        S        0:00 /usr/local/httpd-2.2.14/bin/httpd -k start
19749 pts/1    R+      0:00 grep httpd
31530 ?        Sl      0:57 /usr/local/httpd-2.2.14/bin/httpd -k start

```



```

31560 ?      Sl      1:12 /usr/local/httpd-2.2.14/bin/httpd -k start
31623 ?      Sl      1:06 /usr/local/httpd-2.2.14/bin/httpd -k start
[root@development ~]# ps ax | grep httpd | grep -v grep
 6284 ?      Ss      0:10 /usr/local/httpd-2.2.14/bin/httpd -k start
 8372 ?      S        0:00 perl ./wrapper.pl -chdir -name httpd -class
com.caucho.server.resin.Resin restart
19136 ?      S        0:00 /usr/local/httpd-2.2.14/bin/httpd -k start
31530 ?      Sl      0:57 /usr/local/httpd-2.2.14/bin/httpd -k start
31560 ?      Sl      1:12 /usr/local/httpd-2.2.14/bin/httpd -k start
31623 ?      Sl      1:06 /usr/local/httpd-2.2.14/bin/httpd -k start

```

输出控制 (Output control)

显示行号

```

[root@localhost ~]# grep -n 'ftp' /etc/passwd
12:ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin

```

-o, --only-matching show only the part of a line matching PATTERN

```

$ curl -s http://www.example.com | egrep -o '<a href="(.)">.*</a>' |
sed -e 's/.*href="(^[^"]*)"/.*\/1/'

```

```

$ mysqlshow | egrep -o "|\\w(\\.*)\\w|"
Databases
information_schema
test

```

```

$ cat file.html | grep -o \
-E '\b(([\\w-]+://?|www[.])[^\s(<>)]+(?:\([\\w\d]+\))|
([^\[:punct:\]\\s]|/)))'

$ cat file.html | grep -o -E 'href="(^[^"]+)"'

$ cat sss.html | grep -o -E 'thunder://([<]+)'

```

```

neo@MacBook-Pro ~/project % cat WikiTest.java | grep '@Api'
    @Api(method = GET, uri = "/project/:projectName/wikis/page")
    @Api(method = POST, uri = "/project/:projectName/wiki")
    @Api(method = POST, uri = "/project/:projectName/wiki")
    @Api(method = POST, uri = "/project/:projectName/wiki")

neo@MacBook-Pro ~/project % cat WikiTest.java | egrep -o
'method\s=\s\w+,\suri\s=\s\w+'
method = GET, uri = "/project/:projectName/wikis/page"
method = POST, uri = "/project/:projectName/wiki"
method = POST, uri = "/project/:projectName/wiki"
method = POST, uri = "/project/:projectName/wiki"

```

IP 地址

```

# grep rhost /var/log/secure | grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b"

```

UUID

```

neo@MacBook-Pro ~ % curl -s -X POST --user 'api:secret' -d
'grant_type=password&username=netkiller@msn.com&password=123456'
http://localhost:8080/oauth/token | grep -o -E '"access_token":("[0-9a-
f-]+)"'
"access_token":"863ef5df-6448-40a6-8809-f6f4b680689b"

```

行列转换

```

$ grep -o . <<< "Helloworld"
H
e
l
l
o
w
o
r

```

```
l  
d
```

递归操作

递归查询

```
$ sudo grep -r 'neo' /etc/*
```

递归替换

```
<![CDATA[  
for file in $( grep -rl '8800.org' * | grep -v .svn ); do  
    echo item: $file  
    [ -f $file ] && sed -e 's/8800\.org/sf\.net/g' -e  
's/netkiller/neo/g' $file >$file.bak; cp $file.bak $file;  
done
```

-c, --count print only a count of matching lines per FILE

```
$ cat /etc/resolv.conf  
nameserver localhost  
nameserver 208.67.222.222  
nameserver 208.67.220.220  
nameserver 202.96.128.166  
nameserver 202.96.134.133  
$ grep -c nameserver /etc/resolv.conf  
5
```

```
# grep -c GET /www/logs/access.log  
188460  
  
# grep -c POST /www/logs/access.log  
421
```

binary file matches

```
log@logging ~/netkiller> grep '1052302282228360003' spring.2023-02-28.log
grep: spring.2023-02-28.log: binary file matches
```

虽然这是文本文件，但是文件中含有二进制内容输出，导致 grep 误以为是二进制文件

解决方法 -a, --text equivalent to --binary-files=text

```
log@logging ~/netkiller> grep -a '1052302282228360003' spring.2023-02-28.log
```

Context control

-A, --after-context=NUM print NUM lines of trailing context

返回匹配当前行至下面N行

```
# grep -A1 game /etc/passwd
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin

# grep -A2 game /etc/passwd
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
```

-B, --before-context=NUM print NUM lines of leading context

返回匹配当前行至上面N行

```
# grep -B1 game /etc/passwd
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin

# grep -B2 game /etc/passwd
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
```

```
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
```

-C, --context=NUM print NUM lines of output context

-NUM same as **--context=NUM**

```
neo@neo-OptiPlex-380:~$ grep -C 1 new /etc/passwd
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh

neo@neo-OptiPlex-380:~$ grep -C 5 new /etc/passwd
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh

# grep -3 game /etc/passwd
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
```

--color

```
# grep --color root /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
```

可以通过alias别名启用--color选项

```
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
```

加入.bashrc中，每次用户登录将自动生效

```
# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval
"$(dircolors -b)"
    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi
```

Regexp selection and interpretation

n 开头

```
$ grep '^n' /etc/passwd
news:x:9:9:news:/var/spool/news:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
neo:x:1000:1000:neo chan,,,:/home/neo:/bin/bash
nagios:x:116:127::/var/run/nagios2:/bin/false
```

bash 结尾

```
$ grep 'bash$' /etc/passwd
root:x:0:0:root:/root:/bin/bash
neo:x:1000:1000:neo chan,,,:/home/neo:/bin/bash
postgres:x:114:124:PostgreSQL
administrator,,,:/var/lib/postgresql:/bin/bash
cvsroot:x:1001:1001:cvsroot,,,:/home/cvsroot:/bin/bash
svnroot:x:1002:1002:subversion,,,:/home/svnroot:/bin/bash
```

中间包含 root

```
$ grep '.*root' /etc/passwd
root:x:0:0:root:/root:/bin/bash
cvsroot:x:1001:1001:cvsroot,,,:/home/cvsroot:/bin/bash
svnroot:x:1002:1002:subversion,,,:/home/svnroot:/bin/bash
```

.*

```
$ curl -s http://www.example.com | egrep -o '<a href=(.*)>.*</a>'
```

2010:(13|14|15|16)

regular 匹配一组

```
egrep "2010:(13|14|15|16)" access.2010-11-18.log > apache.log
```

```
ps ax |grep -E "mysqld|httpd|resin"
```

```
neo@MacBook-Pro-Neo ~> cat /etc/passwd | egrep -e "root|daemon"
root:!:0:0:System Administrator:/var/root:/bin/sh
daemon:!:1:1:System Services:/var/root:/usr/bin/false
_cvmsroot:!:212:212:CVMS Root:/var/empty:/usr/bin/false
```

[]与{}

源文件

```
# cat /etc/fstab
#
# /etc/fstab
# Created by anaconda on Sat Sep 10 00:25:46 2011
```

```
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more
info
#
UUID=091f295e-ea6d-4f57-9314-e2333f7ebff7 /                               ext4
defaults          1 1
UUID=b3661a0b-2c50-4e18-8030-be2d043cbfc4 /www                          ext4
defaults          1 2
UUID=4d3468de-a2ac-451c-b693-3bdca8832096 swap                        swap
defaults          0 0
tmpfs              /dev/shm                      tmpfs    defaults
0 0
devpts             /dev/pts                      devpts   gid=5,mode=620
0 0
sysfs              /sys                        sysfs    defaults
0 0
proc               /proc                      proc     defaults
0 0
```

匹配每行包含4个连续字符的字符串的行。

```
# grep '[A-Z]\{4\}' /etc/fstab
UUID=091f295e-ea6d-4f57-9314-e2333f7ebff7 /                               ext4
defaults          1 1
UUID=b3661a0b-2c50-4e18-8030-be2d043cbfc4 /www                          ext4
defaults          1 2
UUID=4d3468de-a2ac-451c-b693-3bdca8832096 swap                        swap
defaults          0 0
```

-P, --perl-regexp Perl正则表达式

Interpret PATTERN as a Perl regular expression. This is highly experimental and grep -P may warn of unimplemented features.

```
[neo@netkiller nginx]$ grep -Po '\w+\.js' www.netkiller.cn.access.log
index.js
min.js
min.js
mCustomScrollbar.js
min.js
ajax_gd.js
ajax.js
validation.js
```



```
AC_RunActiveContent.js
WdatePicker.js
cookie.js
msg_modal.js
all.js
common.js
commonjs.js
swfobject.js
dateutil.js
form.js
live800.js
lang.js
cycle2.js
min.js
carousel.js
tabify.js
image.js
min.js
ctrl.js
packed.js
min.js
common.js
```

fgrep

^M 处理

```
fgrep -rl `echo -ne '\r'` .
find . -type f -exec grep $'\r' {} +
```

egrep

egrep = grep -E 在egrep中不许看使用转意字符，例如

```
# grep '\(oo\).*\1' /etc/passwd
root:x:0:0:root:/root:/bin/bash

# grep -E '(oo).*\1' /etc/passwd
root:x:0:0:root:/root:/bin/bash

# egrep '(oo).*\1' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

```
$ snmpwalk -v2c -c public 172.16.1.254 | egrep -i 'if(in|out)'  
  
for pid in $(ps -axf |grep 'php-cgi' | egrep egrep "0:00.  
(6|7|8|9)""{print $1}'); do kill -9 $pid; done  
  
for pid in $(ps -axf |grep 'php-cgi' | egrep "0:(0|1|2|3|4|5)0.  
(6|7|8|9)" |awk '{print $1}'); do kill -9 $pid; done
```

匹配多个条件

```
[root@localhost src]# egrep "^r|^d" /etc/group  
root:x:0:  
daemon:x:2:  
disk:x:6:  
dialout:x:18:  
dbus:x:81:  
render:x:998:  
docker:x:991:www,gitlab-runner
```

sort - sort lines of text files

```
$ du -s * | sort -k1,1rn
```

```
$ rpm -q -a --qf '%10{SIZE}\t%{NAME}\n' | sort -k1,1n  
$ dpkg-query -W -f='${Installed-Size;10}\t%{Package}\n' | sort -k1,1n
```

对列排序

sort -k 具体说来, 你可以使用 -k1,1 来对第一列排序, -k1来对全行排序

```
# sort -t ':' -k 1 /etc/passwd
```

```
ort -n -t ' ' -k 2 file.txt
```

多列排序

```
$ sort -n -t ' ' -k 2 -k 3 file.txt
```

-s, --stable stabilize sort by disabling last-resort comparison

例如: 如果你要想对两列排序, 先是以第二列, 然后再以第一列, 那么你可以这样. `sort -s` 会很有用

```
sort -k1,1 | sort -s -k2,2
```

uniq

```
history | cut -c 8- | sort -r | uniq -u
```

```
# netstat -ant|fgrep ":"|cut -b 77-90|sort |uniq -c
  1 CLOSE_WAIT
  1 CLOSING
 88 ESTABLISHED
  7 FIN_WAIT1
  7 FIN_WAIT2
  3 LAST_ACK
  4 LISTEN
  1 SYN_RECV
  1 SYN_SENT
177 TIME_WAIT
```

awk

内置变量

ARGC	命令行参数个数
ARGV	命令行参数排列
ENVIRON	支持队列中系统环境变量的使用
FILENAME	awk浏览的文件名
FNR	浏览文件的记录数
FS	设置输入域分隔符, 等价于命令行 <code>-F</code> 选项

NF	浏览记录的域的个数
NR	已读的记录数
OFS	输出域分隔符
ORS	输出记录分隔符
RS	控制记录分隔符

处理列

```
# cat /etc/fstab | awk '{print $1}'
```

printf

%d 十进制有符号整数
 %u 十进制无符号整数
 %f 浮点数
 %s 字符串
 %c 单个字符
 %p 指针的值
 %e 指数形式的浮点数
 %x, %X 无符号以十六进制表示的整数
 %o 无符号以八进制表示的整数
 %g 自动选择合适的表示法
 \n 换行
 \f 清屏并换页
 \r 回车
 \t Tab符
 \xhh 表示一个ASCII码用16进表示,其中hh是1到2个16进制数

说明:

(1). 可以在"%"和字母之间插进数字表示最大场宽。

例如: %3d 表示输出3位整型数, 不够3位右对齐。

%9.2f 表示输出场宽为9的浮点数, 其中小数位为2, 整数位为6, 小数点占一位, 不够9位右对齐。

%8s 表示输出8个字符的字符串, 不够8个字符右对齐。

如果字符串的长度、或整型数位数超过说明的场宽, 将按其实际长度输出. 但对浮点数, 若整数部分位数超过了说明的整数位宽度, 将按实际整数位输出; 若小数部分位数超过了说明的小数位宽度, 则按说明的宽度以四舍五入输出。

另外, 若想在输出值前加一些0, 就应在场宽项前加个0。

例如: %04d 表示在输出一个小于4位的数值时, 将在前面补0使其总宽度为4位。

如果用浮点数表示字符或整型量的输出格式, 小数点后的数字代表最大宽度, 小数点前的数字代表最小宽度。

例如: %6.9s 表示显示一个长度不小于6且不大于9的字符串。若大于9, 则第9个字符以后的内容将被删除。

```

echo 1.7 > 2
awk '{printf ("%d\n",$1)}' 2
1
awk '{printf ("%f\n",$1)}' 2
1.700000
awk '{printf ("%3.1f\n",$1)}' 2
1.7
awk '{printf ("%4.1f\n",$1)}' 2
1.7
awk '{printf ("%e\n",$1)}' 2

```

print 拼装rm命令实现，查找文件并删除

```

#!/bin/sh
LOCATE=/home/samba
find $LOCATE -name '*.eml'>log
find $LOCATE -name '*.nws'>>log
gawk '{print "rm -rf \"$1}"' log > rmfile
chmod 755 rmfile
./rmfile

```

Pattern(字符匹配)

输出包含（不包含）特定字符的行（sed也可以完成该功能）：

```

:~$ awk '/[a-c]/ { print }' file.txt
daemon x 1 1 daemon /usr/sbin /bin/sh
bin x 2 2 bin /bin /bin/sh
sys x 3 3 sys /dev /bin/sh
sync x 4 65534 sync /bin /bin/sync
games x 5 60 games /usr/games /bin/sh
man x 6 12 man /var/cache/man /bin/sh
lp x 7 7 lp /var/spool/lpd /bin/sh
mail x 8 8 mail /var/mail /bin/sh
news x 9 9 news /var/spool/news /bin/sh
uucp x 10 10 uucp /var/spool/uucp /bin/sh
proxy x 13 13 proxy /bin /bin/sh
www-data x 33 33 www-data /var/www /bin/sh
backup x 34 34 backup /var/backups /bin/sh
list x 38 38 Mailing List Manager /var/list /bin/sh
irc x 39 39 ircd /var/run/ircd /bin/sh
gnats x 41 41 Gnats Bug-Reporting System (admin) /var/lib/gnats /bin/sh
nobody x 65534 65534 nobody /nonexistent /bin/sh
libuuid x 100 101 /var/lib/libuuid /bin/sh
syslog x 101 103 /home/syslog /bin/false
sshd x 102 65534 /var/run/sshd /usr/sbin/nologin

```

```
landscape x 103 108 /var/lib/landscape /bin/false
mysql x 104 112 MySQL Server,,, /var/lib/mysql /bin/false
ntpd x 105 114 /var/run/openntpd /bin/false
postfix x 106 115 /var/spool/postfix /bin/false
nagios x 107 117 /var/lib/nagios /bin/false
chun x 1003 1003 Li Fu Chun,,, /home/chun
munin x 108 118 /var/lib/munin /bin/false
```

```
$ awk '!/[a-c]/ { print }' file.txt
root x 0 0 root /root
neo x 1000 1000 neo,,, /home/neo
```

采用判断来输出特定的列数据：

```
neo@monitor:~$ sed -e 's:/ /g' /etc/passwd | awk '$1 == "neo" { print $1 }'
neo
```

部分包含，不包含指定的字符：

```
$ awk '$1 ~ /[a-d]/ { print }' file.txt
$ awk '$1 !~ /[a-d]/ { print }' file.txt
```

Pattern, Pattern

```
# awk '/www/,/Web/ {print}' /etc/passwd
www:x:80:80:Web User:/www:/bin/bash

# awk '/www/,/[Ww]eb/ {print}' /etc/passwd
www:x:80:80:Web User:/www:/bin/bash
```

```
cat /var/log/rinetd.log | awk -F' ' '$7 ~ /0/ {print $1"\t"$2"\t"$7"\t"$8"\t"$9}'
```

```
# cat /var/log/rinetd.log | awk -F' ' '$7 ~ /(210|209|210)/ {print $1"\t"$2"\t"$7"\t"$8"\t"$9}'
```

Built-in Variables (NR/NF)

例如：awk 读入第一笔数据行
"aaa bbb ccc ddd" 之后，程序中：
\$0 之值将是 "aaa bbb ccc ddd"

```
$1 之值为 "aaa"  
$2 之值为 "bbb"  
$3 之值为 "ccc"  
$4 之值为 "ddd"  
$NF 之值为 4  
$NR 之值为 1
```

NR

NR=n 指定n行号

```
# awk -F':' 'NR==1 {print $(1)}' /etc/passwd  
root  
  
# awk -F':' 'NR==2 {print $(1)}' /etc/passwd  
bin
```

取 1, 3, 4行

```
# awk 'NR==1; NR==3; NR==4 {print $1}' /etc/passwd  
root:x:0:0:root:/root:/bin/bash  
daemon:x:2:2:daemon:/sbin:/sbin/nologin  
adm:x:3:4:adm:/var/adm:/sbin/nologin
```

awk ... '{if(NR=1){...}else{exit}}'

```
$ awk -F' ' '{if(NR==1) print $1}' /etc/issue  
Ubuntu
```

NF

```
# echo "aaa bbb ccc ddd" | awk '{print $(NR)}'  
aaa  
# echo "aaa bbb ccc ddd" | awk '{print $(NR+1)}'  
bbb  
# echo "aaa bbb ccc ddd" | awk '{print $(NR+2)}'  
ccc  
# echo "aaa bbb ccc ddd" | awk '{print $(NF)}'  
ddd  
# echo "aaa bbb ccc ddd" | awk '{print $(NF-1)}'  
ccc
```

```
# echo "aaa bbb ccc ddd" | awk '{print $(NF-2)}'
bbb

uptime | awk '{print $(NF-2)}'
```

```
[root@netkiller ~]# netstat -na |awk '/^tcp/ {print NF}' | head -n 1
6

[root@netkiller ~]# netstat -ant |awk '/^tcp/ {print $NF}' | tail -n 5
TIME_WAIT
CLOSE_WAIT
CLOSE_WAIT
LISTEN
LISTEN

[root@netkiller ~]# netstat -ant |awk '/^tcp/ {print $(NF-5)}' | tail -n
5
tcp
tcp
tcp
tcp6
tcp6
```

练习

使用 ss 命令统计 TCP 状态

```
[root@netkiller ~]# ss -ant | awk '{++S[$1]} END {for(a in S) print a,
S[a]}'
LISTEN 13
CLOSE-WAIT 42
ESTAB 95
State 1
FIN-WAIT-2 20
LAST-ACK 44
SYN-SENT 10
TIME-WAIT 403
```

```
[root@netkiller ~]# ss -ant | awk 'BEGIN {stats["CLOSE-
WAIT"]=0;stats["ESTAB"]=0;stats["FIN-WAIT-1"]=0;stats["FIN-WAIT-
2"]=0;stats["LAST-ACK"]=0;stats["SYN-RCV"]=0;stats["SYN-
SENT"]=0;stats["TIME-WAIT"]=0} {++stats[$1]} END {for(a in stats) print
a, stats[a}]'
```



```
LISTEN 6
SYN-RECV 0
ESTAB 4
CLOSE-WAIT 0
State 1
FIN-WAIT-1 0
LAST-ACK 0
FIN-WAIT-2 0
TIME-WAIT 3
SYN-SENT 0
```

TCP/IP Status

```
netstat -ant | awk '/^tcp/ {++state[$NF]} END {for(key in state) print
key,"\t",state[key]]}'
```

```
TIME_WAIT 88
CLOSE_WAIT 6
FIN_WAIT1 9
FIN_WAIT2 9
ESTABLISHED 303
SYN_RECV 126
LAST_ACK 5
```

```
ss | awk '$1 !~ /State/ {++state[$1]} END {for(key in state) print
key,"\t",state[key]]}'
```

```
LAST-ACK 1
ESTAB 5
FIN-WAIT-2 1
CLOSE-WAIT 13
```

用户shell统计

```
# cat /etc/passwd | awk -F':' '{++shell[$NF]} END {for(key in shell)
print key,"\t",shell[key]]}'
```

```
/sbin/shutdown 1
/bin/sh 1
/bin/bash 3
/sbin/nologin 20
/sbin/halt 1
/bin/sync 1
```

access.log POST与GET统计

```
# cat /www/logs/access.log | egrep -o 'GET|POST' | awk '{++method[$NF]}
END {for(num in method) print num, method[num]}'
POST 422
GET 188571

# cat /www/logs/access.log | egrep -o 'GET|POST' | awk '{++method[$1]}
END {for(num in method) print num, method[num]}'
POST 422
GET 188573
```

Built-in Functions

length

```
# awk -F: 'length($1)<4 {print NR , $1}' /etc/passwd
2 bin
4 adm
5 lp
14 ftp
20 ntp
22 rpc
25 www
```

toupper() 转为大写字母

```
[root@localhost ~]# awk '{print toupper($1)}' /etc/passwd
ROOT:X:0:0:ROOT:/ROOT:/BIN/BASH
BIN:X:1:1:BIN:/BIN:/SBIN/NOLOGIN
DAEMON:X:2:2:DAEMON:/SBIN:/SBIN/NOLOGIN
ADM:X:3:4:ADM:/VAR/ADM:/SBIN/NOLOGIN
LP:X:4:7:LP:/VAR/SPOOL/LPD:/SBIN/NOLOGIN
SYNC:X:5:0:SYNC:/SBIN:/BIN/SYNC
SHUTDOWN:X:6:0:SHUTDOWN:/SBIN:/SBIN/SHUTDOWN
HALT:X:7:0:HALT:/SBIN:/SBIN/HALT
MAIL:X:8:12:MAIL:/VAR/SPOOL/MAIL:/SBIN/NOLOGIN
OPERATOR:X:11:0:OPERATOR:/ROOT:/SBIN/NOLOGIN
GAMES:X:12:100:GAMES:/USR/GAMES:/SBIN/NOLOGIN
FTP:X:14:50:FTP
NOBODY:X:99:99:NOBODY:/:/SBIN/NOLOGIN
SYSTEMD-NETWORK:X:192:192:SYSTEMD
DBUS:X:81:81:SYSTEM
POLKITD:X:999:997:USER
```

```
POSTFIX:X:89:89::/VAR/SPOOL/POSTFIX:/SBIN/NOLOGIN
CHRONY:X:998:996::/VAR/LIB/CHRONY:/SBIN/NOLOGIN
SSHD:X:74:74:PRIVILEGE-SEPARATED
NTP:X:38:38::/ETC/NTP:/SBIN/NOLOGIN
DHCPD:X:177:177:DHCP
WWW:X:80:80:WEB
NGINX:X:997:995:NGINX
MYSQL:X:27:27:MYSQL
REDIS:X:1000:1000::/VAR/LIB/REDIS:/BIN/FALSE
ETHEREUM:X:1001:1001::/HOME/ETHEREUM:/BIN/BASH
MONGOD:X:996:991:MONGOD:/VAR/LIB/MONGO:/BIN/FALSE
```

tolower() 转为小写字母

```
[root@localhost ~]# awk -F '\n' '{print tolower($1)}' /etc/redhat-release
centos linux release 7.5.1804 (core)
```

rand() 随机数生成

```
neo@MacBook-Pro ~ % awk 'BEGIN{print rand()*10000000}'
840188
neo@MacBook-Pro ~ % awk 'BEGIN{srand(); print rand()}'
0.0334342
neo@MacBook-Pro ~ % awk 'BEGIN{srand(); print rand()*1000000}'
759412
```

过滤相同的行

```
grep 'Baiduspider' access.2011-02-22.log | awk '{print $1}' | awk '!a[$0]++'
```

```
awk '! a[$0]++' 1.txt >2.txt
这个是删除文件中所有列都重复的记录
```

```
awk '! a[$1]++' 1.txt >2.txt
```

删除文件中第一列重复的记录

```
awk '! a[$1,$2]++' 1.txt >2.txt
```

删除文件中第一，二列都重复的记录

数组演示

```
[root@localhost ~]# awk -F ':' 'BEGIN {count=1;} {name[count] = $1;count++;}; END{for (i = 1; i < NR; i++) print i, name[i}]' /etc/passwd
1 root
2 bin
3 daemon
4 adm
5 lp
6 sync
7 shutdown
8 halt
9 mail
10 operator
11 games
12 ftp
13 nobody
14 systemd-network
15 dbus
16 polkitd
17 postfix
18 chrony
19 sshd
20 ntp
21 dhcpcd
22 www
23 nginx
24 mysql
25 redis
26 ethereum
```

sed

<http://sed.sourceforge.net/>

查找与替换

find and replace

```

sed -n 's/root/admin/p' /etc/passwd
sed -n 's/root/admin/2p' /etc/passwd
#在每行的第2个root作替换
sed -n 's/root/admin/gp' /etc/passwd
sed -n '1,10 s/root/admin/gp' /etc/passwd
sed -n 's/root/AAA&BBB/2p' /etc/passwd
#将root替换成AAArootBBB, &作反向引用, 代替前面的匹配项
sed -ne 's/root/AAA&BBB/' -ne 's/bash/AAA&BBB/p' /etc/passwd #-e将多个命令连接起来, 将root或bash行作替换
sed -n 's/root/AAA&BBB;/s/bash/AAA&BBB/p' /etc/passwd #与上命令功能相同
sed -nr 's/(root)(.)(bash)/\3\2\1/p' /etc/passwd #将root与bash位置替换, 两标记替换 或sed -n 's/root.*bash/\3\2\1/p' /etc/passwd

```

```

ls -l *.html | awk '{printf "sed \047s/ADDRESS/address/g\047 %s\n", $1, $1, $1, $1;}' | bash

for f in `ls -l *.html`; do [ -f $f ] && sed 's/<\BODY>/<script src="http://www.google-analytics.com/urchin.js" type="text/javascript"></script>\n<script type="text/javascript">\n_uacct = "UA-2033740-1";\nurchinTracker();\n</script>\n<\BODY>/g' $f >$f.sed;mv $f.sed $f; done;

```

```

my=/root/dir
str="/root/dir/file1 /root/dir/file2 /root/dir/file3 /root/dir/file/file1"
echo $str | sed "s:$my::g"

```

正则

```

sed s/[[:space:]]//g filename          删除空格

```

aaa="bbb" 提取bbb

```
$ echo "aaa=\"bbb\"" | sed 's/.*=\"\(.*\)\\"/\1/g'
$ curl -s http://www.example.com | egrep -o '<a href="(.)">.*</a>' |
sed -e 's/.*href=\"\([^"]*\)\".*\/\1/'
```

Mac 地址转换

```
echo 192.168.2.1-alf4.40c1.5756 | sed -r 's|(.*)-(.)(.)(.)(.)(.)(.)|  
|.)|\1\2:\3:\4:\5:\6:\7|g'
```

"aaa": "bbb" 提取bbb

数据样本

```
[root@localhost ~]# curl -s https://registry.hub.docker.com/v1/repositories/centos/tags | jq
[
  {
    "layer": "",
    "name": "latest"
  },
  {
    "layer": "",
    "name": "5"
  },
  {
    "layer": "",
    "name": "5.11"
  },
  {
    "layer": "",
    "name": "6"
  },
  {
    "layer": "",
    "name": "6.10"
  },
  {
    "layer": "",
    "name": "6.6"
  },
]
```

```
{
  "layer": "",
  "name": "6.7"
},
{
  "layer": "",
  "name": "6.8"
},
{
  "layer": "",
  "name": "6.9"
},
{
  "layer": "",
  "name": "7"
},
{
  "layer": "",
  "name": "7.0.1406"
},
{
  "layer": "",
  "name": "7.1.1503"
},
{
  "layer": "",
  "name": "7.2.1511"
},
{
  "layer": "",
  "name": "7.3.1611"
},
{
  "layer": "",
  "name": "7.4.1708"
},
{
  "layer": "",
  "name": "7.5.1804"
},
{
  "layer": "",
  "name": "7.6.1810"
},
{
  "layer": "",
  "name": "7.7.1908"
},
{
  "layer": "",
  "name": "7.8.2003"
```

```
},
{
  "layer": "",
  "name": "7.9.2009"
},
{
  "layer": "",
  "name": "8"
},
{
  "layer": "",
  "name": "8.1.1911"
},
{
  "layer": "",
  "name": "8.2.2004"
},
{
  "layer": "",
  "name": "8.3.2011"
},
{
  "layer": "",
  "name": "8.4.2105"
},
{
  "layer": "",
  "name": "centos5"
},
{
  "layer": "",
  "name": "centos5.11"
},
{
  "layer": "",
  "name": "centos6"
},
{
  "layer": "",
  "name": "centos6.10"
},
{
  "layer": "",
  "name": "centos6.6"
},
{
  "layer": "",
  "name": "centos6.7"
},
{
  "layer": "",
```



```
    "name": "centos6.8"
  },
  {
    "layer": "",
    "name": "centos6.9"
  },
  {
    "layer": "",
    "name": "centos7"
  },
  {
    "layer": "",
    "name": "centos7.0.1406"
  },
  {
    "layer": "",
    "name": "centos7.1.1503"
  },
  {
    "layer": "",
    "name": "centos7.2.1511"
  },
  {
    "layer": "",
    "name": "centos7.3.1611"
  },
  {
    "layer": "",
    "name": "centos7.4.1708"
  },
  {
    "layer": "",
    "name": "centos7.5.1804"
  },
  {
    "layer": "",
    "name": "centos7.6.1810"
  },
  {
    "layer": "",
    "name": "centos7.7.1908"
  },
  {
    "layer": "",
    "name": "centos7.8.2003"
  },
  {
    "layer": "",
    "name": "centos7.9.2009"
  },
  {
```

```

    "layer": "",
    "name": "centos8"
  },
  {
    "layer": "",
    "name": "centos8.1.1911"
  },
  {
    "layer": "",
    "name": "centos8.2.2004"
  },
  {
    "layer": "",
    "name": "centos8.3.2011"
  },
  {
    "layer": "",
    "name": "centos8.4.2105"
  }
]

```

提取方法

```

[root@localhost ~]# curl -s
https://registry.hub.docker.com/v1/repositories/centos/tags | sed
's/}/}\n/g' | sed -e 's/.*"name": "\(^[#]*\)".*/\1/'
latest
5
5.11
6
6.10
6.6
6.7
6.8
6.9
7
7.0.1406
7.1.1503
7.2.1511
7.3.1611
7.4.1708
7.5.1804
7.6.1810
7.7.1908
7.8.2003
7.9.2009
8

```

```
8.1.1911
8.2.2004
8.3.2011
8.4.2105
centos5
centos5.11
centos6
centos6.10
centos6.6
centos6.7
centos6.8
centos6.9
centos7
centos7.0.1406
centos7.1.1503
centos7.2.1511
centos7.3.1611
centos7.4.1708
centos7.5.1804
centos7.6.1810
centos7.7.1908
centos7.8.2003
centos7.9.2009
centos8
centos8.1.1911
centos8.2.2004
centos8.3.2011
centos8.4.2105
```

首字母大写

```
$ cat /etc/passwd | cut -d: -f1 | sed 's/\b[a-z]/\U&/g'
Root
Daemon
Bin
Sys
Sync
Games
Man
Lp
Mail
News
Uucp
Proxy
Www-Data
Backup
```

```
List
Irc
Gnats
Nobody
Libuuid
Syslog
Messagebus
Whoopsie
Landscape
Sshd
Neo
Ntop
Redis
Postgres
Colord
Mysql
Zookeeper
```

insert 插入字符

i 命令插入一行，并且在当前行前面有两个空格

在root行前插入一个admin

```
sed '/root/i admin' /etc/passwd
```

33 行处插入字符

```
sed -i "33 i \ \ authorization: enabled" /etc/mongod.conf
```

追加字符

在root行后追加一个admin行

```
sed '/root/a admin' /etc/passwd
```

修改字符

将root行替换为admin

```
sed '/root/c admin' /etc/passwd
```

删除字符

删除含有root的行

```
sed '/root/d' /etc/passwd
```

delete

删除空行

```
sed /^$/d          filename
sed '/./!d' filename
```

行操作

模式空间中的内容全部打印出来

定位行:

```
sed -n '12,~3p' pass #从第12行开始, 直到下一个3的倍数行 (12-15行)
sed -n '12,+4p' pass #从第12行开始, 连续4行 (12-16行)
sed -n '12~3p' pass  #从第12行开始, 间隔3行输出一次 (12, 15, 18, 21...)
sed -n '10,$p' pass  #从第10行至结尾
sed -n '4!p' pass    #除去第4行
```

打印3~6行间的内容

```
$ sed -n '3,6p' /etc/passwd
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
```

打印35行至行尾

```
$ sed -n '35,$p' /etc/passwd
sshd:x:116:65534::/var/run/sshd:/usr/sbin/nologin
mysql:x:117:126:MySQL Server,,,:/nonexistent:/bin/false
uidd:x:100:101::/run/uidd:/bin/false
libvirt-gemu:x:118:128:Libvirt Qemu,,,:/var/lib/libvirt:/bin/false
libvirt-dnsmasq:x:119:129:Libvirt
Dnsmasq,,,:/var/lib/libvirt/dnsmasq:/bin/false
redis:x:120:130::/var/lib/redis:/bin/false
```

编辑文件

```
-i[SUFFIX], --in-place[=SUFFIX]
                        edit files in place (makes backup if extension
supplied)
```

下面例子是替换t.php中的java字符串为php

```
$ cat t.php
<?java

$ sed -i 's/java/php/g' t.php

$ cat t.php
<?php
```

```
find -name "*.php" -exec sed -i '/<?.*eval(gzinflate(base64.*?>/ d' '{}'\
\; -print
```

指定查找替换的行号

```
sed -i "7,7 s/#server.host: \"localhost\"/server.host: \"0.0.0.0\"/"
/etc/kibana/kibana.yml
```

正则表达式

正则：'/正则式/'

```
sed -n '/root/p' /etc/passwd
sed -n '/^root/p' /etc/passwd
sed -n '/bash$/p' /etc/passwd
sed -n '/ro.t/p' /etc/passwd
sed -n '/ro*/p' /etc/passwd
sed -n '/[ABC]/p' /etc/passwd
sed -n '/[A-Z]/p' /etc/passwd
sed -n '/[^ABC]/p' /etc/passwd
sed -n '/^[^ABC]/p' /etc/passwd
sed -n '/\<root/p' /etc/passwd
sed -n '/root\>/p' /etc/passwd
```

扩展正则：

```
sed -n '/root\|yerik/p' /etc/passwd #拓展正则需要转义
sed -nr '/root|yerik/p' /etc/passwd #加-r参数支持拓展正则
sed -nr '/ro(ot|ye)rik/p' /etc/passwd #匹配rootrik和royerik单词
sed -nr '/ro?t/p' /etc/passwd #?匹配0-1次前导字符
sed -nr '/ro+t/p' /etc/passwd #匹配1-n次前导字符
sed -nr '/ro{2}t/p' /etc/passwd #匹配2次前导字符
sed -nr '/ro{2,}t/p' /etc/passwd #匹配多于2次前导字符
sed -nr '/ro{2, 4}t/p' /etc/passwd #匹配2-4次前导字符
sed -nr '/(root)*p' /etc/passwd #匹配0-n次前导单词
```

管道操作

```
cat <<! | sed '/aaa=(bbb\|ccc\|ddd\)/!s/(aaa=\.*/\1xxx/'
> aaa=bbb
> aaa=ccc
> aaa=ddd
> aaa=[something else]
!
aaa=bbb
aaa=ccc
aaa=ddd
```

```
aaa=xxx
```

字母大小写转换

```
[root@localhost ~]# echo "netkiller" | sed 's/[a-z]/\u&/g'
NETKILLER
[root@localhost ~]# echo "NETKILLER" | sed 's/[A-Z]/\l&/g'
netkiller
```

perl

```
sed -i -e 's/aaa/bbb/g' *
perl -p -i -e 's/aaa/bbb/g' *
```

案例

HTML 转 文本

```
# Remove HTML Tags from a File in Linux
sed 's/<[^>]*>//g ; /^$/d' htmlpage.html

# Convert HTML to Text in Linux
sed 's/<[^>]*>//g ; /^$/d' htmlpage.html > output.txt
```


9. 表格操作/行列转换

column - columnate lists

列格式化

下面举一个例子，mount 执行结果

```
[www@netkiller www.netkiller.cn]$ mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
devtmpfs on /dev type devtmpfs
(rw,nosuid,size=1931400k,nr_inodes=482850,mode=755)
securityfs on /sys/kernel/security type securityfs
(rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts
(rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,mode=755)
tmpfs on /sys/fs/cgroup type tmpfs
(ro,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup
(rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/usr/lib/s
ystemd/systemd-cgroups-agent,name=systemd)
pstore on /sys/fs/pstore type pstore
(rw,nosuid,nodev,noexec,relatime)
cgroup on /sys/fs/cgroup/perf_event type cgroup
(rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup
(rw,nosuid,nodev,noexec,relatime,cpuacct,cpu)
cgroup on /sys/fs/cgroup/devices type cgroup
(rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/net_cls type cgroup
(rw,nosuid,nodev,noexec,relatime,net_cls)
cgroup on /sys/fs/cgroup/blkio type cgroup
(rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/hugetlb type cgroup
(rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/freezer type cgroup
(rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/memory type cgroup
```

```
(rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/cpuset type cgroup
(rw,nosuid,nodev,noexec,relatime,cpuset)
configfs on /sys/kernel/config type configfs (rw,relatime)
/dev/xvda1 on / type ext4 (rw,relatime,nobarrier,data=ordered)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs
(rw,relatime,fd=33,pgrp=1,timeout=300,minproto=5,maxproto=5,direct)
mqueue on /dev/mqueue type mqueue (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime)
/dev/xvdb1 on /opt type btrfs (rw,relatime,ssd,space_cache)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc
(rw,relatime)
none on /proc/xen type xenfs (rw,relatime)
tmpfs on /run/user/0 type tmpfs
(rw,nosuid,nodev,relatime,size=361892k,mode=700)
/dev/xvdb1 on /var/ftp type btrfs (rw,relatime,ssd,space_cache)
```

使用 column 格式化后

```
[www@netkiller www.netkiller.cn]$ mount | column -t
sysfs          on /sys                      type sysfs
(rw,nosuid,nodev,noexec,relatime)
proc           on /proc                    type proc
(rw,nosuid,nodev,noexec,relatime)
devtmpfs       on /dev                     type devtmpfs
(rw,nosuid,size=1931400k,nr_inodes=482850,mode=755)
securityfs     on /sys/kernel/security    type securityfs
(rw,nosuid,nodev,noexec,relatime)
tmpfs          on /dev/shm                type tmpfs
(rw,nosuid,nodev)
devpts         on /dev/pts                 type devpts
(rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs          on /run                    type tmpfs
(rw,nosuid,nodev,mode=755)
tmpfs          on /sys/fs/cgroup          type tmpfs
(ro,nosuid,nodev,noexec,mode=755)
cgroup         on /sys/fs/cgroup/systemd   type cgroup
(rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/usr/lib/systemd/systemd-cgroups-agent,name=systemd)
pstore         on /sys/fs/pstore           type pstore
(rw,nosuid,nodev,noexec,relatime)
```

```

cgroup      on  /sys/fs/cgroup/perf_event  type  cgroup
(rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup      on  /sys/fs/cgroup/cpu,cpuacct  type  cgroup
(rw,nosuid,nodev,noexec,relatime,cpuacct,cpu)
cgroup      on  /sys/fs/cgroup/devices  type  cgroup
(rw,nosuid,nodev,noexec,relatime,devices)
cgroup      on  /sys/fs/cgroup/net_cls  type  cgroup
(rw,nosuid,nodev,noexec,relatime,net_cls)
cgroup      on  /sys/fs/cgroup/blkio  type  cgroup
(rw,nosuid,nodev,noexec,relatime,blkio)
cgroup      on  /sys/fs/cgroup/hugetlb  type  cgroup
(rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup      on  /sys/fs/cgroup/freezer  type  cgroup
(rw,nosuid,nodev,noexec,relatime,freezer)
cgroup      on  /sys/fs/cgroup/memory  type  cgroup
(rw,nosuid,nodev,noexec,relatime,memory)
cgroup      on  /sys/fs/cgroup/cpuset  type  cgroup
(rw,nosuid,nodev,noexec,relatime,cpuset)
configfs    on  /sys/kernel/config  type  configfs
(rw,relatime)
/dev/xvda1  on  /  type  ext4
(rw,relatime,nobarrier,data=ordered)
systemd-1   on  /proc/sys/fs/binfmt_misc  type  autofs
(rw,relatime,fd=33,pgrp=1,timeout=300,minproto=5,maxproto=5,direct)
mqueue      on  /dev/mqueue  type  mqueue
(rw,relatime)
debugfs     on  /sys/kernel/debug  type  debugfs
(rw,relatime)
hugetlbfs   on  /dev/hugepages  type  hugetlbfs
(rw,relatime)
/dev/xvdb1  on  /opt  type  btrfs
(rw,relatime,ssd,space_cache)
binfmt_misc on  /proc/sys/fs/binfmt_misc  type  binfmt_misc
(rw,relatime)
none        on  /proc/xen  type  xenfs
(rw,relatime)
tmpfs       on  /run/user/0  type  tmpfs
(rw,nosuid,nodev,relatime,size=361892k,mode=700)
/dev/xvdb1  on  /var/ftp  type  btrfs
(rw,relatime,ssd,space_cache)

```

```
$ (printf "PERM LINKS OWNER GROUP SIZE MONTH DAY HH:MM/YEAR
NAME\n" ; ls -l | sed 1d) | column -t

$ cat /etc/passwd |tr ':' ' ' | column -t

$ cat /etc/passwd |tr ':' ' ' | column -t | colrm 20 20
```

paste - merge lines of files

```
# vim test
aaaaa   bbbbbb   cccccc   ddddd
1111    2222    3333    444

# paste -s test
aaaaa   bbbbbb   cccccc   ddddd   1111    2222    3333    444
```

join

join 命令就是一个根据关键字合并数据文件的命令(join lines of two files on a common field),类似于数据库中两张表关联查询.

```
内连接 (inner join)                                格式: join <FILE1>
<FILE2>
左连接 (left join, 左外连接, left outer join)      格式: join -a1
<FILE1> <FILE2>
右连接 (right join, 右外连接, right outer join)    格式: join -a2
<FILE1> <FILE2>
全连接 (full join, 全外连接, full outer join)      格式: join -a1 -a2
<FILE1> <FILE2>

// 注意 使用 join 来合并两个文件的数据行时, 这两个文件必须要被正确排序.

1) 差集
[root@test23 ~]# cat a.txt | sort > file1; cat b.txt | sort >
```

```
file2; join -v 1 file1 file2
1.1.1.1
3.3.3.3
[root@test23 ~]# cat a.txt | sort > file1; cat b.txt | sort >
file2; join -v 2 file1 file2
4.4.4.4
a.b.c.d
```

2) 并集

```
[root@test23 ~]# cat a.txt | sort > file1; cat b.txt | sort >
file2; join -a1 -a2 file1 file2
1.1.1.1
1.2.3.4
2.2.2.2
3.3.3.3
4.4.4.4
a.b.c.d
```

3) 交集

// 不指定任何参数的情况下使用join命令,相当于数据库中的内连接,(关键字,默认用第一列[使用空格作为分割符]作为关键字)不匹配的行不会输出.

```
[root@test23 ~]# cat a.txt | sort > file1; cat b.txt | sort >
file2; join file1 file2
1.2.3.4
2.2.2.2
```

join 其他用法

-t <CHAR> 指定分隔符,比如: -t ':' 使用冒号作为分隔符,默认的分隔符是空白.

-o <FILENO.FIELDNO> ... 指定输出字段 其中FILENO=1表示第一个文件,FILENO=2表示第二个文件,FIELDNO表示字段序号,从1开始编号.默认会全部输出,但关键字列只输出一次.

10. standard input/output

xargs - build and execute command lines from standard input

xargs命令是 给其他命令传递参数的一个过滤器,也是组合多个命令的一个工具 它擅长将标准输入数据转换成命令行参数,xargs能够处理管道或者stdin并将其转换成特定命令的命令参数. xargs也可以将单行或多行文本输入转换为其他格式,例如多行变单行,单行变多行. xargs的默认命令是echo,空格是默认定界符;这意味着通过管道传递给xargs的输入将会包含换行和空白,不过通过xargs的处理,换行和空白将被空格取代.

xargs命令用法

格式化

xargs用作替换工具，读取输入数据重新格式化后输出。

定义一个测试文件，内有多行文本数据：

```
cat >> test.txt <<EOF
```

```
a b c d e f g
h i j k l m n
o p q
r s t
u v w x y z
```

EOF

```
# cat test.txt
```

```
a b c d e f g
h i j k l m n
o p q
r s t
u v w x y z
```

多行输入一行输出:

```
# cat test.txt | xargs
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

等效

```
# cat test.txt | tr "\n" " "
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

standard input

```
# xargs < test.txt
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

```
# cat /etc/passwd | cut -d : -f1 > users
# xargs -n1 < users echo "Your name is"
```

```
Your name is root
Your name is bin
Your name is daemon
Your name is adm
Your name is lp
Your name is sync
Your name is shutdown
Your name is halt
Your name is mail
Your name is operator
Your name is games
Your name is ftp
Your name is nobody
Your name is dbus
Your name is polkitd
Your name is avahi
Your name is avahi-autoipd
Your name is postfix
Your name is sshd
Your name is neo
Your name is ntp
Your name is opendkim
Your name is netkiller
Your name is tcpdump
```

-I 替换操作

-I R same as **--replace=R**

复制所有图片文件到 /data/images 目录下:

```
ls *.jpg | xargs -n1 -I cp {} /data/images
```

读取stdin, 将格式化后的参数传递给命令xargs的一个选项**-I**, 使用**-I**指定一个替换字符串{}, 这个字符串在xargs扩展时会被替换掉, 当**-I**与xargs结合使用, 每一个参数命令都会被执行一次:

```
# echo "name=Neo|age=30|sex=T|birthday=1980" | xargs -d"|" -n1 |  
xargs -I {} echo "select * from tab where {} "  
select * from tab where name=Neo  
select * from tab where age=30  
select * from tab where sex=T  
select * from tab where birthday=1980
```

```
# xargs -I user echo "Hello user" <users  
Hello root  
Hello bin  
Hello daemon  
Hello adm  
Hello lp  
Hello sync  
Hello shutdown  
Hello halt  
Hello mail  
Hello operator  
Hello games  
Hello ftp  
Hello nobody  
Hello dbus  
Hello polkitd  
Hello avahi  
Hello avahi-autoipd  
Hello postfix
```



```
Hello sshd
Hello netkiller
Hello neo
Hello tss
Hello ntp
Hello opendkim
Hello noreply
Hello tcpdump
```

-I 使用-I指定一个替换字符串,这个字符串在xargs扩展时会被替换掉,当-I与xargs结合使用,每一个参数命令都会被执行一次。

```
mysql -u root -predhat -s -e "show databases" | egrep
"^mt4_user_equity_" | xargs -I "@@" mysql -u root -predhat -e
"DROP DATABASE \@@" ;"
```

-n, --max-args=MAX-ARGS use at most MAX-ARGS arguments per command line

-n 参数來指定每一次执行指令所使用的参数个数上限值.

-n选项多行输出:

```
# cat test.txt | xargs -n3
a b c
d e f
g h i
j k l
m n o
p q r
s t u
v w x
y z
# cat test.txt | xargs -n4
a b c d
e f g h
i j k l
m n o p
q r s t
u v w x
y z
```

```
# cat test.txt | xargs -n5
a b c d e
f g h i j
k l m n o
p q r s t
u v w x y
z

[neo@netkiller test]# echo 'a b c d e 1 2 3 4 5' | xargs -n 5
a b c d e
1 2 3 4 5
```

-t, --verbose print commands before executing them

-t 参数可以让 xargs 在执行指令之前先显示要执行的指令

```
[neo@netkiller test]# echo a b c d e f | xargs -t
/bin/echo a b c d e f
a b c d e f
```

-d, --delimiter=CHARACTER items in input stream are separated by CHARACTER, not by whitespace; disables quote and backslash processing and logical EOF processing

-d 自定义一个定界符 默认是空格

```
[neo@netkiller test]# echo 'abc' | xargs -d b
a c
```

-d选项可以自定义一个定界符:

```
# echo "name|age|sex|birthday" | xargs -d "|"
name age sex birthday
```

结合-n选项使用:

```
# echo "name=Neo|age=30|sex=T|birthday=1980" | xargs -d "|" -n1
```

```
name=Neo
age=30
sex=T
birthday=1980
```

-0, --null items are separated by a null, not whitespace; disables quote and backslash processing and logical EOF processing

-0 是以null字符结尾的,而不是以白空格(whitespace)结尾的且引号和反斜杠,都不是特殊字符;

每个输入的字符,都视为普通字符禁止掉文件结束符,被视为别的参数.当输入项可能包含白空格,引号,反斜杠等情况时,才适合用此参数

```
[neo@netkiller test]# touch "Mr liu"
[neo@netkiller test]# ls M*
Mr liu
[neo@netkiller test]# find -type f -name "Mr*" | xargs rm -f
[neo@netkiller test]# ls M*
Mr liu
[neo@netkiller test]# find -type f -name "Mr*" | xargs -t rm -f
rm -f ./Mr liu
// 这个时候我们可以将 find 指令加上 -print0 参数,另外将 xargs 指令加上
-0 参数,改成这样:
[neo@netkiller test]# find -type f -name "Mr*" -print0 | xargs -
t -0 rm -f
rm -f ./Mr liu
[neo@netkiller test]# ls M*
ls: 无法访问M*: 没有那个文件或目录
```

-r, --no-run-if-empty if there are no arguments, then do not run COMMAND; if this option is not given, COMMAND will be

-r 如果标准输入不包含任何非空格,请不要运行该命令.

```
[neo@netkiller test]# echo a b c d e f | xargs -p -n 3
/bin/echo a b c ?...n
/bin/echo d e f ?...n
/bin/echo ?...n
//当我们使用 -p 参数时，如果所有的指令都输入 n 跳过不执行时候，最后还会出现
一个没有任何参数的 echo 指令，
如果想要避免以这种空字符串作为参数来执行指令，可以加上 -r 参数
[neo@netkiller test]# echo a b c d e f | xargs -p -n 3 -r
/bin/echo a b c ?...n
/bin/echo d e f ?...n
```

-p, --interactive prompt before running commands

-p 确认操作选项,具有可交互性:

-P 修改最大的进程数, 默认是1.为 0 时候为 as many as it can.

11. flock - manage locks from shell scripts

flock

当多个进程可能会对同样的数据执行操作时,这些进程需要保证其它进程没有在操作,以免损坏数据.通常,这样的进程会使用一个“锁文件”,也就是建立一个文件来告诉别的进程自己在运行,如果检测到那个文件存在则认为有操作同样数据的进程在工作.

这样的问题是,进程不小心意外死亡了,没有清理掉那个锁文件,那么只能由用户手动来清理了.

flock 是对于整个文件的建议性锁;也就是说如果一个进程在一个文件(inode)上放了锁,那么其它进程是可以知道的,(建议性锁不强求进程遵守)最棒的一点是,它的第一个参数是文件描述符,在此文件描述符关闭时,锁会自动释放;而当进程终止时,所有的文件描述符均会被关闭.于是,很多时候就不用考虑解锁的事情.

flock分为两种锁:

一种是共享锁 使用-s参数

一种是独享锁 使用-x参数

选项和参数:

-s --shared: 获取一个共享锁,在定向为某文件的FD上设置共享锁而未释放锁的时间内,其他进程试图在定向为此文件的FD上设置独占锁的请求失败,而其他进程试图在定向为此文件的FD上设置共享锁的请求会成功.

-x, -e, --exclusive: 获取一个排它锁,或者称为写入锁,为默认项

-u, --unlock: 手动释放锁,一般情况不必须,当FD关闭时,系统会自动解锁,此参数用于脚本命令一部分需要异步执行,一部分可以同步执行的情况.

-n, --nb, --nonblock: 非阻塞模式,当获取锁失败时,返回1而不是等待.

-w, --wait, --timeout seconds : 设置阻塞超时,当超过设置的秒数时,退出阻塞模式,返回1,并继续执行后面的语句.

-o, --close : 表示当执行command前关闭设置锁的FD,以使command的子进程不保持锁.

-c, --command command : 在shell中执行其后的语句.

<>打开\${LOCK_FILE} (打开LOCK_FILE文件,与文件描述符101绑定),原因是定向文件描述符是先于命令执行的.因此假如在您要执行的语句段中需要读 LOCK_FILE 文件,例如想获得上一个脚本实例的pid,并将此次的脚本实例的pid写入 LOCK_FILE ,此时直接用>打开 LOCK_FILE 会清空上次存入的内容,而用<打开 LOCK_FILE 当它不存在时会导致一个错误.

example

> ntp

```

#!/bin/bash
#
#author junun
#description this script for start or stop check sever time
from an ntp server every 1s
#please add in /etc/rc.local
#
script_0=$0
script_name=${script_0##*/}
lockfile=/var/lock/subsys/$script_name
pidfile=/var/run/$script_name

start() {
    [ -f $lockfile ] && echo -e "\033[31m$script_name is
running...\033[0m" && exit 1
    while true ;do
        /usr/sbin/ntpdate clock.isc.org > /dev/null 2>&1
        echo $$ > $pidfile
        touch $lockfile
        sleep 1
    done
}

stop() {
    [ ! -f $lockfile ] && echo -e "\033[31m$script_name is not
running...\033[0m" && exit 1
    kill -TERM `cat $pidfile`
    rm -rf $lockfile
}

case "$1" in
    start)
        $1
        ;;
    stop)
        $1
        ;;
    *)
        echo $"Usage: $0 {start|stop}"
        exit 2
esac
exit $?

```

```
*/10 * * * * /usr/bin/flock -xn /var/run/check_time.lock -c  
'/usr/local/bin/monitor/check_time start &' > /dev/null 2>&1
```

```
>2 monitor
```

```
#!/bin/bash
```

```
#
```

```
#
```

```
SHELL_DIR=$(cd $(dirname $0);pwd)
```

```
LOCK_FILE=/dev/shm/`echo ${SHELL_DIR}|sed  
's!/!.!g;s!..!!'\`.monitor.lock
```

```
{
```

```
    flock -n 100 || { exit 2; }
```

```
    cd ${SHELL_DIR}
```

```
    function monitor() {
```

```
        while true;do
```

```
            ./run.sh monitor
```

```
            sleep 3
```

```
        done
```

```
    }
```

```
    monitor >> ../logs/monitor.log 2>&1 &
```

```
} 100<>${LOCK_FILE}
```

```
#!/bin/bash
```

```
#
```

```
ulimit -c unlimited
```

```
ulimit -u unlimited
```

```
ulimit -HSn 655350
```

```
SERVER_NAME='changed_order_deal'
```

```
SHELL_DIR=$(cd $(dirname $0);pwd)
```

```
BASE_DIR=$(cd $(dirname $0);cd ../pwd)
```

```
SHELL_FILE="${SHELL_DIR}/run.sh"
```

```
SERVER_BIN=${SHELL_DIR}/${SERVER_NAME}
```

```
LOG_DIR=${BASE_DIR}/logs
```

```
PID_FILE=${LOG_DIR}/PID
```

```

CONF_FILE=${BASE_DIR}/conf/${SERVER_NAME}.conf
LOCK_FILE=/dev/shm/`echo ${SERVER_BIN}|sed
's!/!!!g;s!!!!'`.monitor.lock

start() {
    if [ ! -f "${SERVER_BIN}" ];then
        echo `date +"%F %T"` - ERROR - Can not find
${SERVER_BIN} ...
        exit 1
    fi

    PID=`/sbin/pidof ${SERVER_BIN}`
    if [ x"${PID}" == x"" ];then
        cd ${SHELL_DIR}
        mkdir -p ${LOG_DIR}
        nohup ${SERVER_BIN} -flagfile=${CONF_FILE} >>
${LOG_DIR}/${SERVER_NAME}.stdout.log 2>&1 &
        # place the following shell sentence right after the
nohup statement
        /sbin/pidof ${SERVER_BIN} > ${PID_FILE}          #进程
pid写入文件
        echo "`date +"%F %T"` - start ${SERVER_BIN} "
    else
        ps aux|grep pt_auth
        echo "`date +"%F %T"` - ERROR - PID:${PID} exist.
${SERVER_BIN} is already running."
    fi
}

stop() {
    PID=`cat ${PID_FILE}`
    if [ x"${PID}" == x"" ];then
        echo "`date +"%F %T"` - ERROR - ${SERVER_BIN} is not
running..."
    else
        kill -15 $PID
        while true
        do
            if test $( ps aux | awk '{print $2}' | grep -w
"$PID" | grep -v 'grep' | wc -l ) -eq 0;then
                echo "`date +"%F %T"` - SUCCESS - ${SERVER_BIN}
has been stopped..."
                > ${PID_FILE}
                break
            else

```



```

        echo "`date +%F %T"` - wait to stop..."
        sleep 1
    fi
done
fi
}

kill9() {
    PID=`cat ${PID_FILE}`
    if [ x"${PID}" == x"" ];then
        echo "`date +%F %T"` - ERROR - ${SERVER_BIN} is not
running..."
        exit 1
    else
        kill -9 $PID
    fi
}

restart() {
    stop
    start
}

monitor() {
    check_num=`ps ax -o pid,cmd|grep "$SERVER_BIN"|grep -v
grep|wc -l`
    if [ $check_num -eq 0 ];then
        start
        echo "`date +%F %T"` - restart.
    fi
}

case "$1" in
    "start")
        start;
        ;;
    "stop")
        stop;
        ;;
    "restart")
        restart;
        ;;
    "kill9")
        kill9;
        ;;

```

```
    "monitor")
        monitor;
    ;;
*)
    echo "Usage: $(basename "$0")
start/stop/restart/kill9/monitor"
    exit 1
esac

* * * * * /srv/bin/monitor.sh &> /dev/null
```

12. 进制转换 - 16进制 - 8进制 - 二进制

od - dump files in octal and other formats

16进制

```
neo@netkiller ~ % echo "helloworld" | od -x
00000000      6568      6c6c      776f      726f      646c      000a
00000013

neo@netkiller ~ % echo "helloworld" | od -x -An
      6568      6c6c      776f      726f      646c      000a
```

使用 od 随机生成密码

```
neo@netkiller ~ % od -vN 32 -An -tx1 /dev/urandom | tr -d ' \n'
a6bf6dad8ed860a234046b66d550008f61c36e9cb2630c22d935dac5e20d7920
```

hexdump, hd -- ASCII, decimal, hexadecimal, octal dump

以十六进制方式显示二进制文件

```
neo@netkiller ~ % hexdump -n 256 -C ./coutput/HelloWorld.bin
00000000  36 30 36 30 36 30 34 30  35 32 33 34 31 35 36 31  |6060604052341561|
00000010  30 30 30 66 35 37 36 30  30 30 38 30 66 64 35 62  |000f57600080fd5b|
00000020  36 31 30 32 65 33 38 30  36 31 30 30 31 65 36 30  |6102e38061001e60|
00000030  30 30 33 39 36 30 30 30  66 33 30 30 36 30 36 30  |00396000f3006060|
00000040  36 30 34 30 35 32 36 30  30 34 33 36 31 30 36 31  |6040526004361061|
00000050  30 30 34 63 35 37 36 30  30 30 33 35 37 63 30 31  |004c576000357c01|
00000060  30 30 30 30 30 30 30 30  30 30 30 30 30 30 30 30  |0000000000000000|
*
00000090  30 30 30 30 30 30 30 30  39 30 30 34 36 33 66 66  |00000000900463ff|
000000a0  66 66 66 66 66 66 31 36  38 30 36 33 34 65 64 33  |ffffffff1680634ed3|
000000b0  38 38 35 65 31 34 36 31  30 30 35 31 35 37 38 30  |885e146100515780|
000000c0  36 33 36 64 34 63 65 36  33 63 31 34 36 31 30 30  |636d4ce63c146100|
000000d0  61 65 35 37 35 62 36 30  30 30 38 30 66 64 35 62  |ae575b600080fd5b|
000000e0  33 34 31 35 36 31 30 30  35 63 35 37 36 30 30 30  |341561005c576000|
000000f0  38 30 66 64 35 62 36 31  30 30 61 63 36 30 30 34  |80fd5b6100ac6004|
00000100
```

xxd - make a hexdump or do the reverse.

```
neo@MacBook-Pro ~/workspace % xxd -b netkiller.dat
00000000: 00000000 00000000 00000000 11111111 00000000 00000000  ....
00000006: 00000000 00000000 11111111 11111111 11111111 11111111  ....
```

指定每行的列数

```
neo@MacBook-Pro ~ % xxd -c 2 -b netkiller.bin
00000000: 10010110 01001000  .H
```

跳过字节

跳过两个字节，三列显示

```
neo@MacBook-Pro ~ % xxd -s 2 -c 3 -b netkiller.txt
00000002: 11101001 10011001 10001000  ...
00000005: 11100110 10011001 10101111  ...
00000008: 11100101 10110011 10110000  ...
```

binutils

```
$ sudo apt-get install binutils
```

strings - print the strings of printable characters in files.

```
tcpdump -i eth0 -s 0 -l -w - dst port 80 | strings
```

13. ed, red - text editor

行寻址

. 此选项对当前行寻址（缺省地址）。
number 此选项对第 number 行寻址。可以按逗号分隔的范围 (first,last) 对行寻址。0 代表缓冲区的开头（第一行之前）。
-number 此选项对当前行之前的第 number 行寻址。如果没有 number，则减号对紧跟在当前行之前的行寻址。
+number 此选项对当前行之后的第 number 行寻址。如果没有 number，则加号对紧跟在当前行之后的行寻址。
\$ 此选项对最后一行寻址。
, 此选项对第一至最后一行寻址，包括第一行和最后一行（与 1,\$ 相同）。
; 此选项对当前行至最后一行寻址。
/pattern/ 此选项对下一个包含与 pattern 匹配的文本的行寻址。
?pattern? 此选项对上一个包含与 pattern 匹配的文本的行寻址。

命令描述

a 此命令在指定的地址之后追加文本。
c 此命令将指定的地址更改为给定的文本。
d 此命令删除指定地址处的行。
i 此命令在指定的地址之前插入文本。
q 此命令在将缓冲区保存到磁盘后终止程序并退出。
r file 此命令读取 filespec 的内容并将其插入指定的地址之后。
s/pattern/replacement/ 此命令将匹配 pattern 的文本替换为指定地址中的 replacement 文本。
w file 此命令将指定的地址写到 file。如果没有 address，则此命令缺省使用整个缓冲区。

实例，删除passwd中的neo用户

```
ed -s passwd <<EOF
/neo/
d
wq
EOF
```

```
ed -s mfsmetallogger.cfg <<EOF
,s/^# //
wq
EOF
```

删除尾随空格

```
$ cat -vet input.txt

This line has trailing blanks.      $
This line does not.$

$ (echo ',s/ *$//'; echo 'wq') | ed -s input.txt

$ cat -vet input.txt

This line has trailing blanks.$
This line does not.$
```

14. vim

查找与替换

s%/aaa/bbb/g

```
Starting Nmap 5.21 ( http://nmap.org ) at 2012-02-02 17:03 CST
NSE: Script Scanning completed.
Nmap scan report for 10.10.1.1
Host is up (0.0072s latency).
The 1 scanned port on 10.10.1.1 is filtered

Nmap scan report for 10.10.1.2
Host is up (0.0064s latency).
The 1 scanned port on 10.10.1.2 is closed

Nmap scan report for 10.10.1.3
Host is up (0.0071s latency).
The 1 scanned port on 10.10.1.3 is closed

Nmap scan report for 10.10.1.4
Host is up (0.0072s latency).
PORT      STATE SERVICE
3306/tcp  open  mysql
| mysql-info: Protocol: 10
| Version: 5.1.54-log
| Thread ID: 37337702
| Some Capabilities: Long Passwords, Connect with DB, Compress,
ODBC, Transactions, Secure Connection
| Status: Autocommit
|_Salt: y0!QV;ekiN)"kx;\=Y+g

Nmap scan report for 10.10.1.5
Host is up (0.0081s latency).
PORT      STATE SERVICE
3306/tcp  open  mysql
| mysql-info: Protocol: 10
| Version: 5.1.48-community-log
| Thread ID: 6655211
| Some Capabilities: Long Passwords, Connect with DB, Compress,
```

```
ODBC, Transactions, Secure Connection
| Status: Autocommit
|_Salt: i3ap1?+UL^q>$5~=UqYJ

Nmap scan report for 10.10.1.6
Host is up (0.0073s latency).
The 1 scanned port on 10.10.1.6 is closed

Nmap scan report for www.example.com (10.10.1.7)
Host is up (0.0074s latency).
The 1 scanned port on www.example.com (10.10.1.7) is closed
</screen>
<para>删除closed上面2行</para>
<screen>
:%s:.*\n.*\n.*closed$:g
:%s/\n\n\n//g
```

删除操作

删除指定行

```
:158,158d
```

插入文件

当前光标处插入文件

```
:r /etc/passwd
```

第十行处插入文件


```
:10 r /etc/passwd
```

批处理

test script

```
vim test.txt <<end > /dev/null 2>&1  
:%s/neo/neo chen/g  
:%s/hello/hello world/g  
:wq  
end
```

test.txt

```
begin  
neo  
test  
hello  
world  
end
```

test result

```
$ ./test  
$ cat test.txt  
begin  
neo chen  
test
```

```
hello world
world
end
neo@netkiller:/tmp$
```

vi 批处理

```
for i in file_list
do
vi $i <<-!
:g/xxxx/s//XXXX/g
:wq
!
done
```

line()

加入行号

```
:g/^/ s//\=line(' '). ' '/
```

set fileformat

加入行号

```
vim    set fileformat
执行 set fileformat 会返回当前文件的 format 类型 如:fileformat=dos
也可执行 set line
```

空格与TAB转换

ts 是tabstop的缩写，设TAB宽度为4个空格。

softtabstop 表示在编辑模式的时候按退格键的时候退回缩进的长度，当使用 expandtab 时特别有用。

shiftwidth 表示每一级缩进的长度，一般设置成跟 softtabstop 一样。

expandtab表示缩进用空格来表示，noexpandtab 则是用制表符表示一个缩进。

autoindent自动缩进

空格tab 长度设置

```
set ts=4
set softtabstop=4
set shiftwidth=4
set expandtab
set autoindent
```

上面配置可以添加到 vim 配置文件中：/etc/virc 和 /etc/vimrc

TAB替换为空格

```
:set ts=4
:set expandtab
:%retab!
```

空格替换为TAB

```
:set ts=4  
:set noexpandtab  
:%retab!
```

15. Wget - The non-interactive network downloader.

wget各种选项分类列表

```
* 启动
-V, --version 显示wget的版本后退出
-h, --help 打印语法帮助
-b, --background 启动后转入后台执行
-e, --execute=COMMAND 执行`.wgetrc'格式的命令, wgetrc格式参见/etc/wgetrc或~/.wgetrc
* 记录和输入文件
-o, --output-file=FILE 把记录写到FILE文件中
-a, --append-output=FILE 把记录追加到FILE文件中
-d, --debug 打印调试输出
-q, --quiet 安静模式(没有输出)
-v, --verbose 冗长模式(这是缺省设置)
-nv, --non-verbose 关掉冗长模式, 但不是安静模式
-i, --input-file=FILE 下载在FILE文件中出现的URLs
-F, --force-html 把输入文件当作HTML格式文件对待
-B, --base=URL 将URL作为在-F -i参数指定的文件中出现的相对链接的前缀
--sslcertfile=FILE 可选客户端证书
--sslcertkey=KEYFILE 可选客户端证书的KEYFILE
--egd-file=FILE 指定EGD socket的文件名
* 下载
--bind-address=ADDRESS 指定本地使用地址(主机名或IP, 当本地有多个IP或名字时使用)
-t, --tries=NUMBER 设定最大尝试链接次数(0 表示无限制).
-O --output-document=FILE 把文档写到FILE文件中
-nc, --no-clobber 不要覆盖存在的文件或使用.#前缀
-c, --continue 接着下载没下载完的文件
--progress=TYPE 设定进程条标记
-N, --timestamping 不要重新下载文件除非比本地文件新
-S, --server-response 打印服务器的回应
--spider 不下载任何东西
-T, --timeout=SECONDS 设定响应超时的秒数
-w, --wait=SECONDS 两次尝试之间间隔SECONDS秒
--waitretry=SECONDS 在重新链接之间等待1...SECONDS秒
--random-wait 在下载之间等待0...2*WAIT秒
-Y, --proxy=on/off 打开或关闭代理
-Q, --quota=NUMBER 设置下载的容量限制
--limit-rate=RATE 限定下载输率
* 目录
```

```
-nd, -no-directories 不创建目录
-x, -force-directories 强制创建目录
-nH, -no-host-directories 不创建主机目录
-P, -directory-prefix=PREFIX 将文件保存到目录 PREFIX/...
-cut-dirs=NUMBER 忽略 NUMBER层远程目录
* HTTP 选项
-http-user=USER 设定HTTP用户名为 USER.
-http-passwd=PASS 设定http密码为 PASS.
-C, -cache=on/off 允许/不允许服务器端的数据缓存 (一般情况下允许).
-E, -html-extension 将所有text/html文档以.html扩展名保存
-ignore-length 忽略 `Content-Length`头域
-header=STRING 在headers中插入字符串 STRING
-proxy-user=USER 设定代理的用户名为 USER
-proxy-passwd=PASS 设定代理的密码为 PASS
-referer=URL 在HTTP请求中包含 `Referer: URL`头
-s, -save-headers 保存HTTP头到文件
-U, -user-agent=AGENT 设定代理的名称为 AGENT而不是 Wget/VERSION.
-no-http-keep-alive 关闭 HTTP活动链接 (永远链接).
-cookies=off 不使用 cookies.
-load-cookies=FILE 在开始会话前从文件 FILE中加载cookie
-save-cookies=FILE 在会话结束后将 cookies保存到 FILE文件中
* FTP 选项
-nr, -dont-remove-listing 不移走 `.listing`文件
-g, -glob=on/off 打开或关闭文件名的 globbing机制
-passive-ftp 使用被动传输模式 (缺省值).
-active-ftp 使用主动传输模式
-retr-symlinks 在递归的时候, 将链接指向文件(而不是目录)
* 递归下载
-r, -recursive 递归下载 -- 慎用!
-l, -level=NUMBER 最大递归深度 (inf 或 0 代表无穷).
-delete-after 在现在完毕后局部删除文件
-k, -convert-links 转换非相对链接为相对链接
-K, -backup-converted 在转换文件x之前, 将之备份为 x.orig
-m, -mirror 等价于 -r -N -l inf -nr.
-p, -page-requisites 下载显示HTML文件的所有图片
* 递归下载中的包含和不包含(accept/reject)
-A, -accept=LIST 分号分隔的被接受扩展名的列表
-R, -reject=LIST 分号分隔的不被接受的扩展名的列表
-D, -domains=LIST 分号分隔的被接受域的列表
-exclude-domains=LIST 分号分隔的不被接受的域的列表
-follow-ftp 跟踪HTML文档中的FTP链接
-follow-tags=LIST 分号分隔的被跟踪的HTML标签的列表
-G, -ignore-tags=LIST 分号分隔的被忽略的HTML标签的列表
-H, -span-hosts 当递归时转到外部主机
-L, -relative 仅仅跟踪相对链接
-I, -include-directories=LIST 允许目录的列表
```

```
-X, --exclude-directories=LIST 不被包含目录的列表  
-np, --no-parent 不要追溯到父目录
```

Logging and input file

-i, --input-file=FILE download URLs found in local or external FILE.

准备输入文件，将要下载的连接放入文件中，例如：

```
$ vim file.lst  
  
http://www.example.com/file1.txt  
http://www.example.com/file2.txt  
...  
http://www.example.com/file10.txt
```

开始下载

```
$ wget -i file.lst
```

下载相关参数

-O, --output-document=FILE write documents to FILE 保存到文件

```
wget -q  
https://raw.githubusercontent.com/oscm/shell/master/web/tomcat/systemd/tomcat.service -O /usr/lib/systemd/system/tomcat.service
```

HTTP options (HTTP 选项)

--post-data=STRING use the POST method; send STRING as the data.

```
wget -O - -q --post-  
data="user=neo&password=pasw0rd&title=test&message=helloworld"  
http://localhost/index.php
```

header HTTP头定义

`--header=STRING` 在headers中插入字符串 STRING

```
wget --no-cookies --header "Cookie: oraclelicense=accept-  
securebackup-cookie" http://download.oracle.com/otn-  
pub/java/jdk/8u131-b11/d54c1d3a095b4ff2b6607d096fa80163/server-  
jre-8u131-linux-x64.tar.gz
```

Recursive download

`-r, --recursive` specify recursive download.

使用-r是应该注意，很多网页有外站链接，-r会将外站一同下载(旧版本)

```
wget -r http://netkiller.github.com
```

`-m, --mirror` shortcut for `-N -r -l inf --no-remove-listing`.

我们通常使用-m可以下载整个网站例如我的网站上有很多电子书，你想一次下载下来离线阅读

```
wget -m http://netkiller.github.com/index.html
```


--no-passive-ftp disable the "passive" transfer mode.

```
$ wget ftp://ftp:59bde6@42.120.45.123/test.zip
--2012-04-05 15:48:47--
ftp://ftp:*password*@42.120.45.123/test.zip
      => `test.zip'
Connecting to 42.120.45.123:21... connected.
Logging in as ftp ... Logged in!
==> SYST ... done.      ==> PWD ... done.
==> TYPE I ... done.   ==> CWD not needed.
==> SIZE 20120404.zip ... 42023258
==> PASV ...
```

程序一直停留在 PASV 处

```
$ wget --no-passive-ftp
ftp://ftp:26d9a0dd@42.120.45.123/test.zip
--2012-04-05 15:50:15--
ftp://ftp:*password*@42.120.45.123/test.zip
      => `test.zip'
Connecting to 42.120.45.123:21... connected.
Logging in as ftp ... Logged in!
==> SYST ... done.      ==> PWD ... done.
==> TYPE I ... done.   ==> CWD not needed.
==> SIZE test.zip ... 42023258
==> PORT ... done.     ==> RETR test.zip ... done.
Length: 42023258 (40M) (unauthoritative)

100%
[=====
=====>]
42,023,258    691K/s    in 62s

2012-04-05 15:51:18 (657 KB/s) - `test.zip' saved [42023258]
```

下载一组连续的文件名

地址如下

```
http://news.netkiller.cn/2018/1/index.html  
http://news.netkiller.cn/2018/2/index.html  
...  
...  
http://news.netkiller.cn/2018/12/index.html
```

下载方法

```
wget -c http://news.netkiller.cn/2018/{1..12}/index.html
```

16. CURL - transfer a URL

基本用法

```
curl http://www.google.com/
```

提交表单数据

post 表单数据

```
curl -d "user=neo&password=chen" http://www.example.com/login.php  
curl --data "user=neo&password=chen" http://www.example.com/login.php
```

上传文件

```
curl -F "upload=@card.txt;type=text/plain"  
"http://www.example.com/upload.php"
```

使用 CURL 上传 OAuth2 + Jwt 认证的 Restful 接口

```
curl -s -H "Authorization: Bearer ${TOKEN}" -X POST -F  
"file=@/etc/hosts" http://localhost:8080/upload/single
```

connect-timeout

```
curl -o /dev/null --connect-timeout 30 -m 30 -s -w %{http_code}
```

```
http://www.google.com/
```

max-time

-m, --max-time SECONDS Maximum time allowed for the transfer

```
curl -o /dev/null --max-time 10 http://www.netkiller.cn/
```

compressed

--compressed Request compressed response (using deflate or gzip)

```
curl --compressed http://www.example.com
```

代理服务器

vhosts 测试

有时候你需要设觉察/etc/hosts文件才能访问vhost,下面例子可以不设置/etc/hosts

```
curl -x 127.0.0.1:80 your.exmaple.com/index.php
```

socks5 服务器

```
$ curl -v -x socks5://username:password@IP:1080 http://www.google.com/
```

-w, --write-out <format> 输出格式定义

计时器	描述
-----	----

time_connect 建立到服务器的 TCP 连接所用的时间
time_starttransfer 在发出请求之后,Web 服务器返回数据的第一个字节所用的时间
time_total 完成请求所用的时间
time_namelookup DNS解析时间,从请求开始到DNS解析完毕所用时间(记得关掉 Linux 的 nsd 的服务测试)
speed_download 下载速度,单位-字节每秒。

```
curl -o /dev/null -s -w %{time_connect}:%{time_starttransfer}:%  
{time_total} http://www.example.net  
curl -o /dev/null -s -w "Connect: %{time_connect}\nTransfer: %  
{time_starttransfer}\nTotal: %{time_total}\n"  
https://www.netkiller.cn/index.html  
  
curl -o /dev/null -s -w "Connect: %{time_connect} \nTransfer: %  
{time_starttransfer}\nTotal: %{time_total}\nNamelookup: %  
{time_namelookup}\nDownload: %{speed_download}\n"  
https://www.netkiller.cn/index.html  
Connect: 0.024241  
Transfer: 0.117727  
Total: 0.117842  
Namelookup: 0.004367  
Download: 129877.000
```

测试页面所花费的时间

```
date ; curl -s -w 'Connect: %{time_connect} TTFB: %{time_starttransfer}  
Total time: %{time_total} \n' -H "Host: www.example.com"  
http://172.16.0.1/webapp/test.jsp ; date ;
```

```
curl -o /dev/null -s -w %{time_connect}, %{time_starttransfer}, %  
{time_total}, %{time_namelookup}, %{speed_download}  
http://www.netkiller.cn
```

返回HTTP状态码

```
curl -s -I http://netkiller.sourceforge.net/ | grep HTTP | awk '{print  
$2} "$3}'  
curl -o /dev/null -s -w %{http_code} http://netkiller.sourceforge.net/  
  
curl --connect-timeout 5 --max-time 60 --output /dev/null -s -w %  
{response_code} http://www.netkiller.cn/
```

```
# curl -w '\nLookup time:\t%{time_namelookup}\nConnect time:\t%{time_connect}\nPreXfer time:\t%{time_pretransfer}\nStartXfer time:\t%{time_starttransfer}\n\nTotal time:\t%{time_total}\n' -o /dev/null -s http://www.netkiller.cn

Lookup time: 0.125
Connect time: 0.125
PreXfer time: 0.125
StartXfer time: 0.125

Total time: 0.126
```

-A/--user-agent <agent string>

设置用户代理，这样web服务器会认为是其他浏览器访问

```
curl -A "Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)"
http://www.example.com
```

referer[illegible]

```
Speed
100 172k    0 172k    0    0 10.2M    0 --:--:-- --:--:-- --:--:--
11.9M* Connection #0 to host www.your.com left intact

* Closing connection #0
```

-v

-o, --output FILE Write output to <file> instead of stdout

```
curl -o /dev/null http://www.example.com
curl -o index.html http://www.example.com
```

-L, --location

```
curl -L --retry 5 --retry-delay 3
https://github.com/hyperledger/fabric/releases/download/v2.0.1/hyperledg
er-fabric-linux-amd64-2.0.1.tar.gz | tar xz
```

-H/--header <line> Custom header to pass to server (H)

Last-Modified / If-Modified-Since

If-Modified-Since

```
neo@neo-OptiPlex-780:/tmp$ curl -I
http://images.example.com/test/test.html
HTTP/1.0 200 OK
Cache-Control: s-maxage=7200, max-age=900
Expires: Mon, 16 May 2011 08:10:37 GMT
```

```
Content-Type: text/html
Accept-Ranges: bytes
ETag: "1205579110"
Last-Modified: Mon, 16 May 2011 06:57:39 GMT
Content-Length: 11
Date: Mon, 16 May 2011 07:55:37 GMT
Server: lighttpd/1.4.26
Age: 604
X-Via: 1.0 ls71:80 (Cdn Cache Server V2.0), 1.0 lydx136:8105 (Cdn Cache
Server V2.0)
Connection: keep-alive
```

```
neo@neo-OptiPlex-780:/tmp$ curl -H
"If-Modified-Since: Fri, 12 May 2011 18:53:33 GMT" -I
http://images.example.com/test/test.html
HTTP/1.0 304 Not Modified
Date: Mon, 16 May 2011 07:56:19 GMT
Content-Type: text/html
Expires: Mon, 16 May 2011 08:11:19 GMT
Last-Modified: Mon, 16 May 2011 06:57:39 GMT
ETag: "1205579110"
Cache-Control: s-maxage=7200, max-age=900
Age: 790
X-Via: 1.0 wzdx168:8080 (Cdn Cache Server V2.0)
Connection: keep-alive
```

ETag / If-None-Match

```
neo@neo-OptiPlex-780:/tmp$ curl -I
http://images.example.com/test/test.html
HTTP/1.1 200 OK
Cache-Control: s-maxage=7200, max-age=900
Expires: Mon, 16 May 2011 09:48:45 GMT
Content-Type: text/html
Accept-Ranges: bytes
ETag: "1984705864"
Last-Modified: Mon, 16 May 2011 09:01:07 GMT
Content-Length: 22
Date: Mon, 16 May 2011 09:33:45 GMT
Server: lighttpd/1.4.26
```

```
neo@neo-OptiPlex-780:/tmp$ curl -H 'If-None-Match: "1984705864"' -I
```



```
http://images.example.com/test/test.html
HTTP/1.1 304 Not Modified
Cache-Control: s-maxage=7200, max-age=900
Expires: Mon, 16 May 2011 09:48:32 GMT
Content-Type: text/html
Accept-Ranges: bytes
ETag: "1984705864"
Last-Modified: Mon, 16 May 2011 09:01:07 GMT
Date: Mon, 16 May 2011 09:33:32 GMT
Server: lighttpd/1.4.26
```

Accept-Encoding: gzip, deflate

```
$ curl -H Accept-Encoding: gzip, deflate -I
http://www.example.com/product/374218.html
HTTP/1.1 200 OK
Date: Mon, 16 May 2011 09:13:18 GMT
Server: Apache
Accept-Ranges: bytes
Content-Encoding: gzip
Content-Length: 16660
Content-Type: text/html; charset=UTF-8
X-Pad: avoid browser bug
Age: 97
X-Via: 1.1 dg44:8888 (Cdn Cache Server V2.0)
Connection: keep-alive
```

```
$ curl -H Accept-Encoding: gzip, deflate
http://www.example.com/product/374218.html | gunzip
```

HOST

```
curl -H HOST:www.example.com -I http://172.16.1.10/product/374218.html
```

HTTP 认证

未认证返回401

```
# curl --compressed http://webservice.example.com/members
<html>
<head><title>401 Authorization Required</title></head>
<body bgcolor="white">
<center><h1>401 Authorization Required</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

`-u/--user <user[:password]>` Set server user and password

使用 `-u`或者 `--user` 指定用户与密码

```
# curl --compressed -u neo:chen
http://webservice.example.com/members
```

Accept

```
-H "Accept: application/json"
```

Content-Type

```
-H "Content-Type: application/json"
```

curl-config

```
curl-config --features
```

指定网络接口或者地址

--interface INTERFACE Use network INTERFACE (or address)

```
curl --interface 127.0.0.1 http://www.netkiller.cn
```

Cookie 处理

cookie 可以从 http header 设置

```
curl -LO -H "Cookie: oraclelicense=accept-securebackup-cookie"  
http://download.oracle.com/otn-pub/java/jdk/8u131-  
b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.rpm
```

curl 还提供两个参数用于处理 cookie

```
-b, --cookie STRING/FILE Read cookies from STRING/FILE (H) 读取 cookie 文件  
-c, --cookie-jar FILE Write cookies to FILE after operation (H) 将  
cookie 写入文件
```

```
curl -c cookie.txt -d "user=neo&password=123456"  
http://www.netkiller.cn/login  
curl -b cookie.txt http://www.netkiller.cn/user/profile
```

Restful 应用 JSON 数据处理

下面提供一些使用 curl 操作 restful 的实例

GET 操作

```
curl http://api.netkiller.cn/v1/withdraw/get/15.json
```

用户认证的情况

```
curl http://test:123456@api.netkiller.cn/v1/withdraw/get/id/815.json
```

POST 操作

```
curl -i -H "Accept: application/json" -H "Content-Type: application/json" -X POST -d '{
  "id": "B0402000000000000000", "name": "Neo", "amount": 12, "password": "12345", "createdate": "2016-09-12 13:10:10"
}' https://test:123456@api.netkiller.cn/v1/withdraw/create.json
```

```
curl -H "Accept: application/json" -H "Content-Type: application/json" -d '{"id":100000, "username":"netkiller", "password":"123456", "email":"netkiller@msn.com"}' curl http://localhost:8080/restful/validation
```

Curl OAuth2

```
URL=https://api.netkiller.cn
token=$(curl -k --cacert -s -X POST --user 'api:secret' -d 'grant_type=password&username=netkiller@msn.com&password=123456' ${URL}/oauth/token | grep -o -E '"access_token": "[0-9a-f-]+" ' | cut -d \" -f 4 )
curl -k -H "Accept: application/json" -H "Content-Type: application/json" -H "Authorization: Bearer ${token}" -X GET ${URL}/search/article/list/22/0/5.json
```

Curl + OAuth2 + Jwt

获取 access_token 字符串

方法一

```
curl -s -X POST --user 'api:secret' -d  
'grant_type=password&username=netkiller@msn.com&password=123456'  
http://localhost:8080/oauth/token | sed 's/.*"access_token": "\'  
([^\"]*)" */\1/g'
```

方法二

```
curl -s -X POST --user 'api:secret' -d  
'grant_type=password&username=netkiller@msn.com&password=123456'  
http://localhost:8080/oauth/token | grep -o -E '"access_token": "(.+)"' |  
cut -d \" -f 4
```

访问自签名证书

```
curl --cacert certs/domain.crt https://www.netkiller.cn/
```

HTTP2

curl 已经支持 HTTP2，使用方法如下

```
neo@MacBook-Pro ~/workspace % curl -I --http2 https://www.google.com  
HTTP/2 200  
date: Tue, 26 Feb 2019 06:36:03 GMT  
expires: -1
```

```
cache-control: private, max-age=0
content-type: text/html; charset=ISO-8859-1
p3p: CP="This is not a P3P policy! See g.co/p3phelp for more info."
server: gws
x-xss-protection: 1; mode=block
x-frame-options: SAMEORIGIN
set-cookie: 1P_JAR=2019-02-26-06; expires=Thu, 28-Mar-2019 06:36:03 GMT;
path=/; domain=.google.com
set-cookie:
NID=160=aQySEvsSa9gVU8qivD3t5qsgi_PRUtD5Ao3nRb48jMyLAzlYA1ViWuF1BaZHChVz
Y6EuknQ00Uz3Z2vhWwrcIzO4WV6BmWgPhz6jowqFF3XCStsyYVwLQp2-
_c0aGioBryAP1bTT0c-PX9iJzk5Zcbm2cFs6kH0Qb2a_3ML7Ioc; expires=Wed, 28-
Aug-2019 06:36:03 GMT; path=/; domain=.google.com; HttpOnly
alt-svc: quic=":443"; ma=2592000; v="44,43,39"
accept-ranges: none
vary: Accept-Encoding
```

HTTP/2 200 表示当前采用 HTTP2 连接

FAQ

Error in TLS handshake, trying SSLv3...

解决方案

```
curl -v --cipher rsa_rc4_128_sha
https://www.mpaymall.com/MPay/MerchantPay.do
```

17. expect

```
$ sudo apt-get install expect
```

命令含义

```
#!/usr/bin/expect
set timeout 30
spawn ssh root@192.168.1.1
expect "password:"
send "mypassword\r"
interact
```

set 设置变量

spawn 执行命令

expect 检测点

send 发送指令

模拟登录 **telnet** 获取**Cisco**配置

例 4.2. example for expect

```
cat tech-support.exp
#!/usr/bin/expect
set timeout 30
spawn telnet 172.16.1.24
expect "Password: "
send "chen\r"
expect "*>"
send "enable\r"
expect "Password: "
send "chen\r"
```

```
expect "*#"
send "sh tech-support\r"
send "!\r"
expect "*-Switch#"
send "exit\r"
expect eof
exit
```

3层设备

```
cisco.exp
#!/usr/bin/expect
set ip [lindex $argv 0]
set username [lindex $argv 1]
set password [lindex $argv 2]
log_file $ip.log
spawn telnet $ip
expect "Username:"
send "$username\r"
expect "Password:"
send "$password\r"
expect "#"
send "show running-config\r"
send "exit\r"
expect eof
```

2层设备

```
$ cat config.exp
#!/usr/bin/expect
set timeout 30
set host [lindex $argv 0]
set password [lindex $argv 1]
set done 0

log_file $host.log
spawn telnet $host
expect "Password:"
send "$password\r"
```



```

expect "*>"
send "enable\r"
expect "Password: "
send "$password\r"
expect "*#"
send "show running-config\r"

while {$done == 0} {
expect {
" --More--" { send -- " " }
"*#" { set done 1 }
eof { set done 1 }
}
}

send "\r"
expect "*#"
send "exit\r"
expect eof
exit

$ cat loop.sh
#!/bin/sh
while read sw
do
    ./config.exp $sw
done <<EOF
172.16.0.240 chen
172.16.0.241 chen
172.16.0.242 chen
172.16.0.243 chen
172.16.0.245 chen
172.16.0.246 chen
EOF

$ chmod +x config.exp loop.sh
$ ./loop.sh

```

模拟登录 ssh

例 4.3. example for expect

```
#!/usr/bin/expect
set timeout 30
spawn ssh root@192.168.1.1
expect "password:"
send "mypassword\r"
interact
```

例 4.4. example 1

```
#!/usr/bin/expect
set password 1234 #密码
#download
spawn scp /www/* root@172.16.1.2:/www/
set timeout 300
expect "172.16.1.2's password:"
set timeout 3000
#exec sleep 1
send "$password\r"
set timeout 300
send "exit\r"
#expect eof
interact
spawn scp /www/* root@172.16.1.3:/www/
set timeout 300
expect "root@172.16.1.3's password:"
set timeout 3000
#exec sleep 1
send "$password\r"
set timeout 300
send "exit\r"
interact
```

例 4.5. *.exp

```
$ expect autossh.exp neo@192.168.3.10 chen "ls /"
```

autossh.exp

```
#!/usr/bin/expect -fset ipaddress [lindex $argv 0]
set ipaddress [lindex $argv 0]
set passwd [lindex $argv 1]
set command [lindex $argv 2]
set timeout 30
spawn ssh $ipaddress
expect {
    "yes/no" { send "yes\r";exp_continue }
    "password:" { send "$passwd\r" }
}
expect ""

send "$command \r"

send "exit\r"

expect eof
exit
```

批量执行

```
password.lst
192.168.0.1 passwd
192.168.0.2 passwd
192.168.0.3 passwd
```

```
#!/bin/bash

cat password.lst | while read line
do
    host=$(echo $line|awk '{print $1}')
    passwd=$(echo $line|awk '{print $2}')
    expect autossh.exp $host $passwd
    sleep 2
done
```

SCP

```
#!/usr/bin/expect -f
spawn scp 1 neo@192.168.0.1:
expect "*password:"
send "your password\r"

expect eof
```

```
#!/bin/expect

spawn scp x.x.x.x

for {} {1} {} {
    expect {
        "password:" {
            send "YourPassWord"
        }
    }
}
```

```
spawn scp 1 neo@172.16.0.1:
for { set i 1 } {$i<5} {incr i} {
    expect {
        "*password:" {send "koven\r"}
        "*(yes/no)*" {send "yes\r"}
    }
}
```

openssl 例子

```
expect -c '
spawn openssl req -newkey rsa:4096 -nodes -sha256 -keyout
domain.key -x509 -days 3650 -out domain.crt
```

```
expect {
    "Country Name" { send "CN\r"; exp_continue}
    "State or Province Name" { send "Guangdong\r" ;
exp_continue}
    "Locality Name" { send "Shenzhen\r"; exp_continue}
    "Organization Name" { send "netkiller\r"; exp_continue}
    "Organizational Unit Name" { send "Neo\r"; exp_continue}
    "Common Name" { send "netkiller.cn\r" ; exp_continue}
    "Email Address" { send "netkiller@msn.com\r" ;
exp_continue}
    eof { exit }
}'
```

18. expect-lite - quick and easy command line automation tool



19. sshpass - noninteractive ssh password provider

sshpass -p 'ssh_password' ssh www.example.org

```
# ssh neo@192.168.6.1
The authenticity of host '192.168.6.1 (192.168.6.1)' can't be
established.
RSA key fingerprint is
c9:97:95:2a:5c:6a:2f:ac:e8:ac:94:24:b0:5c:45:8a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.6.1' (RSA) to the list of
known hosts.
neo@192.168.6.1's password:

[root@centos6]~# sshpass -p 'chen' ssh neo@192.168.6.1
Last login: Wed Nov 13 15:24:50 2013
[neo@NEO ~]$
```

20. Klish - Kommand Line Interface Shell (the fork of clish project)

<http://code.google.com/p/klish/>

Klish是一个命令行补全工具，可以实现类似于CISCO路由器的命令行帮助界面。它是Clish的后续版本，Klish有一个特殊的功能，可以让用户仅使用指定目录中的命令。

安装Klish

```
# cd /usr/local/src/
# wget http://klish.googlecode.com/files/klish-1.6.4.tar.bz2
# tar jxvf klish-1.6.4.tar.bz2
# cd klish-1.6.4/
# ./configure --prefix=/srv/klish-1.6.4
# make
# make install

# cp -r xml-examples /srv/klish-1.6.4/
# export CLISH_PATH=/srv/klish-1.6.4/xml-examples/clish
```

启动clish

```
# /srv/klish-1.6.4/bin/clish

*****
*          CLISH (see-lish)          *
*                                     *
*      WARNING: Authorised Access Only      *
*****

Welcome root it is Mon Feb 18 09:59:06 CST 2013
>
```


为用户指定clish作为默认Shell

```
# vim /etc/passwd
neo:x:1000:1000:neo,,,:/home/neo:/bin/bash
```

改为

```
neo:x:1000:1000:neo,,,:/home/neo:/srv/clish-1.6.4/bin/clish
```

FAQ

clish/shell/shell_expat.c:36:19: fatal error: expat.h: No such file or directory

```
clish/shell/shell_expat.c:36:19: fatal error: expat.h: No such
file or directory
compilation terminated.
make[1]: *** [clish/shell/libclish_la-shell_expat.lo] Error 1
make[1]: Leaving directory `/usr/local/src/clish-1.6.4'
make: *** [all] Error 2
```

解决方案，安装expat开发包

```
# apt-get install libexpat1-dev
```

21. Limited command Shell (lshell)

<https://github.com/ghantoos/lshell>

主要功能就是能够限制那些命令用户可以运行

22. TUI

screen - screen manager with VT100/ANSI terminal emulation

screen 类似 jobs, 前者是对terminal, 后者针对进程。你可以随时再次链接screen会话, 而不用担心中途因网络不稳定造成的中断。

```
sudo apt-get install screen
```

进入

```
screen
```

查看任务

```
screen -ls
```

重新连接会话

```
screen -r 16582
```

退出screen 使用组合键 C-a K 或者

```
screen -wipe
```

tmux — terminal multiplexer

<http://tmux.sourceforge.net/>

查看当前session **\$tmux ls**

```
$ tmux ls
0: 1 windows (created Mon Aug 19 10:12:15 2013) [270x56]
(attached)

$ tmux list-sessions
0: 1 windows (created Mon Aug 19 10:12:15 2013) [270x56]
(attached)
```

创建session

```
tmux new -s session-name
```

结束session

```
$tmux kill-session -t 0

#结束所有session
$tmux kill-server
```

使当前会话进入后台

```
tmux detach
```

恢复session, detach的反向操作

```
tmux attach -t session-id
```

byobu - wrapper script for seeding a user's byobu configuration and launching a text based window manager (either screen or tmux)



htop - interactive process viewer



elinks

chat

finch,irssi

23. jq - Command-line JSON processor

<https://stedolan.github.io/jq/>

```
[root@localhost ~]# curl -s
https://api.github.com/repos/netkiller/netkiller.github.io/comm
its?per_page=5 | jq '[.[] | {message: .commit.message, name:
.commit.committer.name, parents: [.parents[].html_url]}]'
[
  {
    "message": "ethereum",
    "name": "netkiller",
    "parents": [
      "https://github.com/netkiller/netkiller.github.io/commit/4aa040
9b9049c4ff77d047e17514964617d23d26"
    ]
  },
  {
    "message": "ethereum",
    "name": "netkiller",
    "parents": [
      "https://github.com/netkiller/netkiller.github.io/commit/939a62
d6a8a0058025fca4a0226ded30c07f9178"
    ]
  },
  {
    "message": "ethereum",
    "name": "netkiller",
    "parents": [
      "https://github.com/netkiller/netkiller.github.io/commit/111a7d
09089d7a1950d9879239370ca198f0870a"
    ]
  },
  {
    "message": "hyperledger",
    "name": "netkiller",
    "parents": [
```

```
"https://github.com/netkiller/netkiller.github.io/commit/201b88ec4ad328268856ce6e894b860fa4bdd3a7"
  ]
},
{
  "message": "ethereum",
  "name": "netkiller",
  "parents": [

"https://github.com/netkiller/netkiller.github.io/commit/92a052d152ef1333565646c79f12ada2f701003f"
  ]
}
]
```

24. asciinema 终端录屏

asciinema [as-kee-nuh-muh] is a free and open source solution for recording terminal sessions and sharing them on the web.

<https://asciinema.org/>

```
brew install asciinema
```


25. parallel - build and execute shell command lines from standard input in parallel

并行执行shell命令

```
$ sudo apt-get install parallel
```

例 4.6. parallel - build and execute shell command lines from standard input in parallel

```
$ cat *.csv | parallel --pipe  
grep '13113'
```

设置块大小

```
$ cat *.csv | parallel --block  
10M --pipe grep '131136688'
```

26. multitail

```
dnf -y install epel-release
dnf -y update

dnf install -y gcc gcc-c++ make automake autoconf patch
dnf install -y git
dnf install -y python36
dnf install -y ncurses-devel

cd /usr/local/src/
git clone git://github.com/martine/ninja.git
cd ninja/
python3 bootstrap.py
cp ninja /usr/local/bin/

cd /usr/local/src/
git clone https://github.com/flok99/multitail.git
cd multitail/
make install
```

安装出错

```
[root@localhost multitail]# make install
cmake --build ../.build-multitail-Debug --target install
ninja: error: loading 'build.ninja': No such file or directory
make: *** [GNUmakefile:65: install] Error 1
```

27. Logging

logger - a shell command interface to the syslog(3) system log module

```
# logger -p local0.notice -t HOSTIDM -f /dev/idmc
# tail /var/log/messages

# logger -p local0.notice -t passwd -f /etc/passwd
# tail /var/log/syslog

# logger -p user.notice -t neo -f /etc/passwd
# tail /var/log/syslog
# tail /var/log/messages

# logger -i -s -p local3.info -t passwd -f /etc/passwd
# tail /var/log/messages
```

28. Password

Shadow password suite configuration.

```
# cat /etc/login.defs
# *REQUIRED*
# Directory where mailboxes
reside, _or_ name of file, relative to the
# home directory. If you _do_
define both, MAIL_DIR takes precedence.
# QMAIL_DIR is for Qmail
#
#QMAIL_DIR Maildir
MAIL_DIR /var/spool/mail
#MAIL_FILE .mail

# Password aging controls:
#
# PASS_MAX_DAYS Maximum number
of days a password may be used.
# PASS_MIN_DAYS Minimum number
of days allowed between password changes.
# PASS_MIN_LEN Minimum
acceptable password length.
# PASS_WARN_AGE Number of days
warning given before a password expires.
#
PASS_MAX_DAYS 99999
PASS_MIN_DAYS 0
PASS_MIN_LEN 5
PASS_WARN_AGE 7

#
# Min/max values for automatic
uid selection in useradd

#
UID_MIN 500
UID_MAX 60000

#
# Min/max values for automatic
```

```

gid selection in groupadd
                                #
                                GID_MIN 500
                                GID_MAX 60000

                                #
                                # If defined, this command is
run when removing a user.
                                # It should remove any
at/cron/print jobs etc. owned by
                                # the user to be removed
(passed as the first argument).
                                #
                                #USERDEL_CMD
/usr/sbin/userdel_local

                                #
                                # If useradd should create home
directories for users by default
                                # On RH systems, we do. This
option is overridden with the -m flag on
                                # useradd command line.
                                #
                                CREATE_HOME yes

                                # The permission mask is
initialized to this value. If not specified,
                                # the permission mask will be
initialized to 022.
                                UMASK 077

                                # This enables userdel to
remove user groups if no members exist.
                                #
                                USERGROUPS_ENAB yes

                                # Use MD5 or DES to encrypt
password? Red Hat use MD5 by default.
                                MD5_CRYPT_ENAB yes

                                ENCRYPT_METHOD MD5

```

newusers - update and create new users in batch

```
# cat userfile.txt

www00:x:520:520::/home/www00:/sbin/nologin
www01:x:521:521::/home/www01:/sbin/nologin
www02:x:522:522::/home/www02:/sbin/nologin
www03:x:523:523::/home/www03:/sbin/nologin
www04:x:524:524::/home/www04:/sbin/nologin
www05:x:525:525::/home/www05:/sbin/nologin
www06:x:526:526::/home/www06:/sbin/nologin
www07:x:527:527::/home/www07:/sbin/nologin
www08:x:528:528::/home/www08:/sbin/nologin
www09:x:529:529::/home/www09:/sbin/nologin

# newusers userfile.txt
```

chpasswd - update passwords in batch mode

echo "user:password" | chpasswd

```
[root@dev1 ~]# adduser test
[root@dev1 ~]# echo
"test:123456" | chpasswd
```

```
# cat passwd.txt
neo:neopass
jam:jampass
```

```
# cat passwd.txt | chpasswd
```

```
# chpasswd -c < passwd.txt
```

passwd 命令实现相同功能

```
echo "mypassword" | passwd --stdin neo
```

sshpas - noninteractive ssh password provider

```
sudo apt install -y sshpass
```

```
root@ubuntu:~# sshpass -v
```

```
Usage: sshpass [-f|-d|-p|-e] [-hV] command parameters
```

```
  -f filename      Take password to use from file
```

```
  -d number        Use number as file descriptor for getting
```

```
password
```

```
  -p password      Provide password as argument (security unwise)
```

```
  -e               Password is passed as env-var "SSHPASS"
```

```
With no parameters - password will be taken from stdin
```

```
  -P prompt        Which string should sshpass search for to  
detect a password prompt
```

```
  -v               Be verbose about what you're doing
```

```
  -h               Show help (this screen)
```

```
  -V               Print version information
```

```
At most one of -f, -d, -p or -e should be used
```

```
sshpas -p Password scp target/*.jar
```

```
root@dev.netkiller.cn:/root/
```

```
sshpass -p Password ssh root@dev.netkiller.cn java -jar  
/root/java-0.0.1-SNAPSHOT.jar
```


29. 信息摘要

cksum, sum -- display file checksums and block counts

```
neo@MacBook-Pro ~ % head -20 /dev/urandom | cksum | cut -f1 -d "
"
1705222024
```

md5sum - compute and check MD5 message digest

```
[root@localhost ~]# md5sum /etc/hosts
54fb6627dbaa37721048e4549db3224d  /etc/hosts
```

```
[root@localhost ~]# shasum /etc/hosts
7335999eb54c15c67566186bdfc46f64e0d5a1aa  /etc/hosts
```

30. envsubst - substitutes environment variables in shell format strings

替代品在shell环境变量的格式字符串，类似模版替换操作

```
[root@localhost tmp]# echo "welcome $HOME ${USER:=a8m}" |  
envsubst  
welcome /root root
```

```
[root@localhost tmp]# cat config.template  
HOME=${HOME}  
USER=${USER}  
  
[root@localhost tmp]# envsubst < config.template > config.conf  
  
[root@localhost tmp]# cat config.conf  
HOME=/root  
USER=root
```

只替换 \${USER} 变量

```
[root@localhost tmp]# envsubst '${USER}' < config.template >  
config.conf  
[root@localhost tmp]# cat config.conf  
HOME=${HOME}  
USER=root
```

模版变量

<code>\${var}</code>	var值(与 \$var 相同)
<code>\${var-\$DEFAULT}</code>	如果未设置 var, 则将表达式计算为 \$DEFAULT
<code>\${var:-\$DEFAULT}</code>	如果未设置var或者为空, 则将表达式计算为
<code>\$DEFAULT</code>	
<code>\${var=\$DEFAULT}</code>	如果未设置 var, 则将表达式计算为 \$DEFAULT
<code>\${var:=\$DEFAULT}</code>	如果未设置var或者为空, 则将表达式计算为
<code>\$DEFAULT</code>	
<code>\${var+\$OTHER}</code>	如果为 var, 则将表达式计算为 \$OTHER,, 否则为
空字符串	
<code>\${var:+\$OTHER}</code>	如果为 var, 则将表达式计算为 \$OTHER,, 否则为
空字符串	

第 5 章 常用命令

获取IP地址

```
[root@localhost ~]# hostname -I|awk '{print $1}'  
192.168.30.12
```

第 6 章 Shell Terminal

dialog, whiptail, gdialog, kdialog and nautilus

1. terminal

resize - set TERMCAP and terminal settings to current xterm window size

显示终端屏幕的尺寸

```
$ resize
COLUMNS=151;
LINES=46;
export COLUMNS LINES;
```

设置终端屏幕的尺寸

```
eval `resize`
```

tset, reset - terminal initialization

```
tset -e ^? 设置Backspace删除前面一个字符
tset -k ^C 设置删除一行
```

建议使用stty替代tset

stty - change and print terminal line settings

```
$ stty
speed 38400 baud; line = 0;
eol = M-^?; eol2 = M-^?; swtch = M-^?;
ixany iutf8

$ stty -a
speed 115200 baud; rows 46; columns 151; line = 0;
```

```
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = M-^?; eol2
= M-^?; swtch = M-^?; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread -clocal -crtscts
-ignbrk brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -
ixoff -iuclc ixany imaxbel iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0
vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -
echoprt echoctl echoke
```

```
OLDCONFIG=`stty -g`      # save configuration
stty -echo               # do not display password
echo "Enter password: \c"
read PASSWD              # get the password
stty $OLDCONFIG          # restore configuration
```

2. tput

为输出着色

tput Color Capabilities:

tput setab [1-7] – Set a background color using ANSI escape

tput setb [1-7] – Set a background color

tput setaf [1-7] – Set a foreground color using ANSI escape

tput setf [1-7] – Set a foreground color

tput Text Mode Capabilities:

tput bold – Set bold mode

tput dim – turn on half-bright mode

tput smul – begin underline mode

tput rmul – exit underline mode

tput rev – Turn on reverse mode

tput smso – Enter standout mode (bold on rxvt)

tput rmso – Exit standout mode

tput sgr0 – Turn off all attributes

Color Code for tput:

0 – Black

1 – Red

2 – Green

3 – Yellow

4 – Blue

5 – Magenta

6 – Cyan

7 – White

```
NORMAL=$(tput sgr0)
```

```
RED=$(tput setaf 1)
```

```
GREEN=$(tput setaf 2; tput bold)
```

```
YELLOW=$(tput setaf 3)
```

```
BLUE=$(tput setaf 4)
```

```
MAGENTA=$(tput setaf 5)
```

```
CYAN=$(tput setaf 6)
```

```
WHITE=$(tput setaf 7)

function exception(){
    echo -e "$WHITE*$NORMAL"
}

function critical() {
    echo -e "$RED*$NORMAL"
}

function info() {
    echo -e "$GREEN*$NORMAL"
}

function warning() {
    echo -e "$YELLOW*$NORMAL"
}

function error(){
    echo -e "$MAGENTA*$NORMAL"
}

function debug(){
    echo -e "$CYAN*$NORMAL"
}

# To print critical
critical "kernel error"

# To print exception
exception "file system exception"

# To print error
error "The configuration file does not exist"

# To print warning
warning "You have to use higher version."

# To print info
info "Task has been completed."

# To print debug
debug "This is a debug message."
```


Change the prompt color using tput

```
$ export PS1="\[$(tput bold)$(tput setb 4)$(tput setaf  
7)\]\u@\h:\w $ \[$(tput sgr0)\]"
```

3. dialog

```
$ sudo apt-get install dialog
```

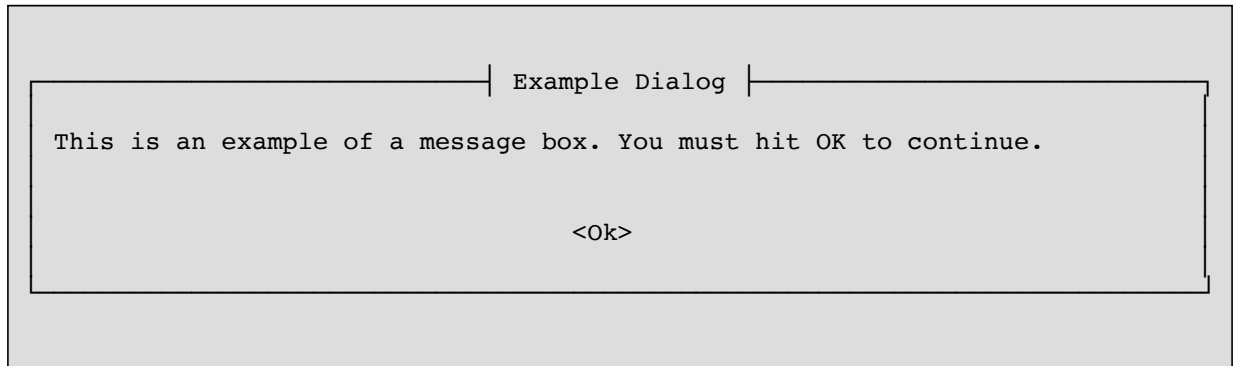
--inputbox

```
result=$(dialog --output-fd 1 --inputbox "Enter some text" 10  
30)  
echo Result=$result
```

4. whiptail - display dialog boxes from shell scripts

--msgbox

```
whiptail --title "Example Dialog" --msgbox "This is an example of a message box.  
You must hit OK to continue." 8 78
```



--infobox

```
whiptail --title "Example Dialog" --infobox "This is an example of a message  
box. You must hit OK to continue." 8 78
```

--yesno

例 6.1. whiptail - yesno

```
#!/bin/bash
# http://archives.seul.org/seul/project/Feb-1998/msg00069.html
if (whiptail --title "PPP Configuration" --backtitle "Welcome to SEUL" --yesno "
Do you want to configure your PPP connection?" 10 40 )
then
    echo -e "\nWell, you better get busy!\n"
elif (whiptail --title "PPP Configuration" --backtitle "Welcome to
SEUL" --yesno "Are you sure?" 7 40)
then
    echo -e "\nGood, because I can't do that yet!\n"
else
    echo -e "\nToo bad, I can't do that yet\n"
fi
```

```
whiptail --title "Example Dialog" --yesno "This is an example of a yes/no box."
8 78
```

```
exitstatus=$?  
if [ $exitstatus = 0 ]; then  
    echo "User selected Yes."  
else  
    echo "User selected No."  
fi  
  
echo "(Exit status was $exitstatus)"
```

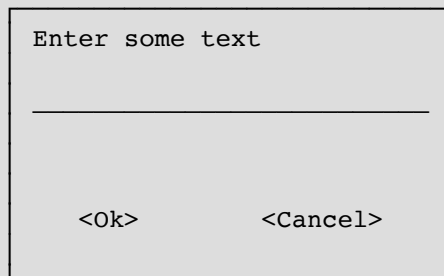
设置--yes-button, --no-button, --ok-button 按钮的文本

```
whiptail --title "Example Dialog" --yesno "This is an example of a message box.  
You must hit OK to continue." 8 78 --no-button 取消 --yes-button 确认
```

--inputbox

例 6.2. whiptail - inputbox

```
result=$(tempfile) ; chmod go-rw $result  
whiptail --inputbox "Enter some text" 10 30 2>$result  
echo Result=$(cat $result)  
rm $result
```



```
COLOR=$(whiptail --inputbox "What is your favorite Color?" 8 78 --title "Example  
Dialog" 3>&1 1>&2 2>&3)  
  
exitstatus=$?  
if [ $exitstatus = 0 ]; then  
    echo "User selected Ok and entered " $COLOR  
else  
    echo "User selected Cancel."  
fi
```

```
echo "(Exit status was $exitstatus)"
```

--passwordbox

例 6.3. whiptail - passwordbox

```
whiptail --title "Example Dialog" --passwordbox "This is an example of a  
password box. You must hit OK to continue." 8 78
```

--textbox

例 6.4. whiptail - passwordbox

```
whiptail --title "Example Dialog" --textbox /etc/passwd 20 60
```

为文本框添加滚动条功能

```
whiptail --title "Example Dialog" --textbox /etc/passwd 20 60 --scrolltext
```

--checklist

例 6.5. whiptail - example 1

```
whiptail --title "Check list example" --checklist \  
"Choose user's permissions" 20 78 16 \  
"NET_OUTBOUND" "Allow connections to other hosts" ON \  
"NET_INBOUND" "Allow connections from other hosts" OFF \  
"LOCAL_MOUNT" "Allow mounting of local devices" OFF \  
"REMOTE_MOUNT" "Allow mounting of remote devices" OFF
```

--radiolist

例 6.6. whiptail - radiolist

```
whiptail --title "Check list example" --radiolist \  

```

```
"Choose user's permissions" 20 78 16 \  
"NET_OUTBOUND" "Allow connections to other hosts" ON \  
"NET_INBOUND" "Allow connections from other hosts" OFF \  
"LOCAL_MOUNT" "Allow mounting of local devices" OFF \  
"REMOTE_MOUNT" "Allow mounting of remote devices" OFF
```

--menu

```
whiptail --title "Menu example" --menu "Choose an option" 22 78 16 \  
"<-- Back" "Return to the main menu." \  
"Add User" "Add a user to the system." \  
"Modify User" "Modify an existing user." \  
"List Users" "List all users on the system." \  
"Add Group" "Add a user group to the system." \  
"Modify Group" "Modify a group and its list of members." \  
"List Groups" "List all groups on the system."
```

```
Menu example  
-- Back      Return to the main menu.  
Add User     Add a user to the system.  
Modify User  Modify an existing user.  
List Users   List all users on the system.  
Add Group    Add a user group to the system.  
Modify Group Modify a group and its list of members.  
List Groups  List all groups on the system.  
  
<Ok>                                <Cancel>
```

--gauge

```
#!/bin/bash  
{  
    for ((i = 0 ; i <= 100 ; i+=30)); do  
        sleep 1  
        echo $i  
    done  
} | whiptail --gauge "Please wait" 5 50 0
```

附录 A. 附录

1. 参考文献

《高级Bash脚本编程指南》

<http://www.linuxsir.org/main/doc/abs/abs3.7cnhtm/index.html>