

```
1  linux的基本原则
2  1、由目的单一的小程序组成，组合小程序完成复杂任务
3  2、一切皆文件
4  3、尽量避免捕获用户接口
5  4、配置文件保存为纯文本格式
6
7  命令格式
8      命令 选项... 参数
9
10  选项：
11      短选项
12      长选项
13  参数：
14
15  list: ls
16  ls
17      -l:长格式
18          文件类型
19              -:普通文件(f)
20              d:目录文件
21              b:块设备文件
22              c:字符设备文件
23              l:符号链接文件
24              p:命令管道文件
25              s:套接字文件
26          文件权限
27          文件硬链接的次数
28          文件的属主
29          文件的数组
30          文件大小，单位为字节
31          时间戳：最近一次被修改的时间
32              访问
33              修改：文件内容发生改变
34              改变：改变元数据
35          文件名
36      -h:做单位转换
37      -a:显示以.开头的文件
38          .:表示当前目录
39          ..:表示父目录文件
40      -A:显示所有文件但是不显示.and..
41      -d:显示目录属性
42      -i:index node,inode 显示索引节点号
43      -r:逆序显示
44      -R:递归显示
45
46  cd :change directory切换当前目录
47
48  type: 显示指定命令属于哪种类型
49
50
51  列出，列表
52
53  文件夹：
54
55  命令类型：
56      内置命令(shell内置)，内部，内建
57      外部命令：在文件系统的某个路径下有一个与命令名称相应的可执行文件
58
59  环境变量：命名的内存空间
60      变量赋值
61          NAME=JERRY
62
63      printenv: 显示环境变量
64
65      PATH: 使用冒号分割的路径
66
67      hash: 缓存0(1)
68
69
70  date:时间管理
71  LINUX: RTC
72  ntp: 网络时间协议
73
```

```
74 硬件时钟: clock hwclock
75 系统时钟: date
76 获得命令的使用帮助:
77 内部命令:
78     help COMMAND
79 外部命令: COMMAND --help
80 命令手册:
81  man COMMAND
82
83  whatis COMMAND 显示命令摘要信息
84
85
86 分章节
87 1: 用户命令 (/bin,/usr/bin,/usr/local/bin)
88 2: 系统调用 (字符集不匹配时: export LANG=en)
89 3: 库用户
90 4: 特殊文件 (设备文件)
91 5: 文件格式 (配置文件的语法)
92 6: 游戏
93 7: 杂项
94 8: 管理命令 (/sbin,/usr/sbin,/usr/local/sbin)
95
96
97 <>必须给出的内容
98 []可选
99 ...可以出现多次
100 |: 多选一
101 {}: 分组
102
103 man:
104     NAME: 命令名称
105     SYNOPSIS (大纲摘要): 用法说明, 包括可用的选项
106     DESCRIPTION: 命令功能的相近说明, 可能包括每个选项的意义
107     OPTION: 说明每一个选项的意义
108     FILES: 此命令的配置文件
109     BUGS:
110     EXAMPLES: 使用示例
111     SEE ALSO: 另外参照
112
113 DATE: 可以修改系统时间
114
115 翻页:
116     向后翻一屏: SPACE
117     向前翻一屏: b
118     向后翻一行: ENTER
119     向前翻一行: k
120 查找:
121 /KEYWORD: 向前
122 n: 下一个
123 N: 前一个
124 ? KEYWORD: 向后
125
126 q: 退出
127
128 date +"This year is %Y.%n Today is %d."
129
130 将系统时间同步到硬件时间: hwclock -w
131 将硬件时间同步到系统时间: hwclock -s
132
133 hwclock
134     -w:
135     -s:
136 在线文档:
137 info COMMAND
138
139 /usr/share/doc : 主要看内核说明文档
140
141 cal: 日历
142
143 查看是内部命令还是外部命令: type echo
144 内部命令的帮助: help echo
145 echo: 显示一行、
146 echo - n不显示换行
```

```
147 printf
148
149 Windows: PE
150 Linux: ELF
151
152 根文件系统详解:
153 文件系统:
154 rootfs: 根文件系统
155 FHS:Linux文件系统层级标准
156 /boot: 系统启动相关的文件, 如内核、磁盘, 以及grub (bootloader)
157 /dev: 设备文件
158     设备文件:
159         块设备: 随机访问的设备, 数据块
160         字符设备: 线性访问, 按字符为单位
161         !!!!!!!!!!!!!!!!!!!!!!!!!!!!! (移除耳机) !!!!!!!!!!!!!!!!!!!!!!!!!!!!!
162         设备号: 主设备号, 次设备号
163 /etc: 配置文件
164 /home: 用户的家目录, 每一个用户的家目录通常默认为/home/USERNAME
165 /root: 管理员的家目录
166 /lib: 库文件和内核模块文件
167     /lib/modules: 内核模块文件
168     静态库, .a
169     动态库, .dll, .so(shared object)
170 /media:挂载点目录, 通常挂载移动设备 (对于centos来说该目录挂载的为虚拟镜像)
171 /mnt: 额外的临时文件系统
172 /opt: 可选目录, 第三程序安装目录
173 /proc: 伪文件系统, 内核映射文件
174 /sys: 伪文件系统, 根硬件设备相关的属性映射文件
175 /tmp: 临时文件
176 /var: 可变化的文件
177 /bin: 可执行文件, 用户命令, 启动相关
178 /sbin: 管理命令, 与启动相关
179 /usr: shared , read-only 全局共享的只读文件
180     /usr/bin
181     /usr/sbin
182     /usr/lib
183 /usr/local: 第三方软件安装, 非关键性
184     /usr/local/bin
185     /usr/local/sbin
186     /usr/local/lib
187
188 命名规则:
189 1、长度不能超过255个字符。
190 2、不能使用/当文件名
191 3、严格区分大小写
192
193 相对路径:
194 绝对路径:
195
196 使用一个操作系统时所用到的功能和工作
197 编辑文件
198 目录管理
199 运行程序
200 设备管理
201 软件管理
202 进程管理
203 网络管理
204
205 文件管理
206 ls
207 cd
208 pwd
209 mkdir: 创建空目录
210     -p: 递归创建
211     -v: verbose详细信息
212 mkdir -pv /mnt/test/x/m /mnt/test/y
213 mkdir -pv /mnt/test/{x/m, y}
214 花括号: 命令行展开
215 mkdir {a,d}_{b,c}可以创建四个文件夹
216 波浪线展开
217
218 删除目录: rmdir 删除目录, 只能删除空目录
219     rmdir -p只能删除一线单传的
```

220
221 文件的创建和删除：
222 touch: 用来改变时间戳
223 -a: 访问时间
224 -m: 更改时间
225 -t: 修改到指定的时间
226 -c: 不创建文件
227
228 stat: 查看文件的状态
229
230 创建文件，还可以使用文件编辑器
231
232 ASCII: 美国标准信息交换代码
233 128不同的字符：
234 二进制：
235
236
237 file: 确定文件类型
238
239 \rm使用原命令
240 -i提示是否删除
241 -f强制删除
242 -r递归删除所有文件
243 rm -rf
244
245 复制和移动文件
246 cp:
247 只能复制一个文件到一个文件
248 多个文件到一个目录
249 cp -r 复制目录
250 -a : 归档复制，常用于备份q
251 -p : 保留属主
252 -L : 复制文件而不是链接
253 -P : 如果为链接则为复制链接
254 -f : 强行复制
255 -i : 提示
256
257
258 mv: 移动文件(目录均可)
259
260
261 install 只能为文件而不能为目录
262 -d : 创建目录
263 : 复制之后具有执行权限
264 -m : 指定权限
265
266 //2-4end
267
268 //3-1
269
270 tree: 查看目录树
271
272 目录管理:
273 ls cd pwd mkdir rmdir tree
274 文件管理
275 touch stat file rm cp mv nano
276 日期时间
277 date clock hwclock cal
278
279
280 查看文本:
281 cat tac more less head tail
282
283 cat: 链接并显示
284 -n : 显示行号
285 -e : 显示换行符
286 -v :
287
288 Ctrl+c : 结束命令
289
290 分屏显示:
291 more less head
292 less: 类似man命令显示格式比较好

293 tail -f 查看文件尾部，不退出，等待显示后续加至此文件的新内容
294
295 文本处理：
296 cut join sed awk
297
298 cut：
299 -d: 指定字段分隔符，默认是空格
300 -f: 指定要显示的字段
301 -f 1: 显示一
302 -f 1,3: 显示一三
303 -f 1-3: 显示一到三
304 //3-2
305 文本排序: sort
306 -n: 按着数值的大小排序
307 -r: 反向排序
308 -t: 字段分隔符
309 -k: 以哪个字段为关键字进行排序
310 -u: 重复的只显示一次 这个命令可以 sort -u \$(find /) 这样重复的就只显示一次
311 -f: 排序时忽略字符大小写
312 uniq: 相邻的重复行进行忽略
313 -d: 打印重复的行
314 -c: 统计某一行重复的次数
315
316 文本统计: wc (word count)
317 行数 单词数 字节数
318 -c: 字节数
319 -m: 字符数
320 -l: 行数
321 -w: 单词数
322 -L: 最长的一行包含多少个字符
323
324 字符处理命令: tr--实现转换或删除字符
325 tr 'a' 'A' < 文件
326 tr -d : 删除字符
327
328
329 bash及其特性:
330 shell: 外壳
331 gui: gnome, kde, xfce
332 cli: sh, csh, ksh, bash, tcsh, zsh
333
334 进程: 每个进程看来, 当前主机上只存在和内核和当前进程
335 进程是程序的副本, 进程是程序执行实例
336
337 用户工作环境:
338 bash: 设置可能不同
339 管理员#
340 普通用户\$
341 pstree: 显示层次
342
343 bash: 特性
344 1、命令历史
345 2、管道、重定向
346 3、命令别名
347 4、命令行编辑
348 5、命令行展开
349 6、文件名通配
350 7、变量
351 8、编程
352
353 命令行编辑:
354 光标跳转:
355 ctrl+a: 跳到命令行首
356 ctrl+e: 跳到命令行尾
357 ctrl+u: 删除光标以前的内容
358 ctrl+k: 删除光标以后的内容
359 ctrl+左右箭头: 一次跳一个单词
360 ctrl+l: 清屏
361 命令历史:
362 上下箭头
363 history: 查看命令历史
364 -c: 清空命令历史
365 -d: 删除指定位置的命令 history -d 500 3 删除三个历史命令

```
366         -w: 保存命令历史至历史文件中
367
368 环境变量
369 PATH: 命令搜索路径
370 HISTSIZE: 命令历史大小缓冲区大小  echo $HISTSIZE显示大小
371
372 命令历史的使用技巧:
373 ! n 执行命令历史中第n条命令
374 ! -n执行倒数第几行
375 !! 执行上次的命令
376 ! string: 执行最近一次字符串开头的命令
377 ! $: 引用上一个命令的参数
378 esc .
379 alt+.
380
381 命令别名:
382 alias CMDALIAS='COMMAND'
383 unalias 撤销命令别名
384 \CMD不使用参数
385
386 命令替换: $(CMD),反引号 `CMD`
387 把命令中某个子命令替换为其执行结果的过程
388 echo "The current directory is $(pwd)."
389
390 file-2013-02-28-14-53-31.txt
391
392 touch ./file-$(date +%F-%H-%M-%S).txt
393
394 bash支持的引号:
395 ``: 命令替换
396 `"`: 弱引用, 可以实现变量替换
397 `': 强引用, 不完成变量替换
398
399 文件名通配, globbing
400 *: 任意长度的任意字符
401 ?: 任意单个字符
402 [:]: 匹配指定范围内的任意单个字符
403 [abc], [a-m], [a-z], [A-Z], [0-9], [a-zA-Z], [0-9a-zA-Z]
404 [^]: 匹配指定范围之外的任意单个字符
405 文件中包含空白字符: touch 'a b'
406 [[:space:]]: 空白字符
407 [[:punct:]]: 标点符号
408 [[:lower:]]: 小写字母
409 [[:upper:]]: 大写字母
410 [[:alpha:]]: 大小写字母
411 [[:digit:]]: 数字
412 [[:alnum:]]: 数字和大小写字母
413
414 例如识别字母开头中间有空格字母结尾的字符串
415 ls [[:alpha:]]*[[:space:]]*[[:alpha:]]
416
417 man 7 glob : 获得上面的列表帮助
418
419 linux用户及权限
420
421 安全上下文
422
423 权限:
424 r, w, x
425 文件:
426 r: 可读, 可以使用类似cat等命令查看文件内容
427 w: 可写, 可以编辑或删除此文件
428 x: 可执行, executable, 可以在命令提示符下当作命令提交给内核运行
429
430 目录:
431 r: 可以对此目录执行ls以列出内部的所有文件
432 w: 可以在此目录创建文件
433 x: 可以使用cd切换进此目录, 也可以使用ls -l 查看内部文件的详细信息
434 默认文件一般不给执行权限, 但是目录因该有
435
436 用户: UID, /etc/passwd
437 组: GID, /etc/group
438
```

```
439 影子口令：
440 用户： /etc/shadow
441 组： /etc/gshadow
442
443 用户类别：
444 管理员： id号为0
445 普通用户： 1-65535
446 系统用户： 1-499    不允许登录系统
447 普通用户： 500-60000
448
449 用户组：
450 管理员组：
451 普通组：
452 系统组：
453 一般组：
454
455 用户组类别：
456 基本组： 用户的默认组
457
458 私有组： 创建用户组时，如果没有为其指定所属的组，系统会自动为其创建一个与用户名同名的组
459 附加组，额外组：
460
461 进程： tom tom
462 对象：
463
464 /etc/passwd
465 account： 账户
466 password： 密码
467 UID：
468 GID： 基本组ID
469 comment： 注释
470 HOME DIR： 家目录
471 shell： 用户的默认shell
472
473 cat /etc/shadow
474 cat /etc/shells
475 cat /etc/group
476 cat /etc/gshadow
477
478 /etc/shadow
479 account： 登录名
480 encrypted password： 加密的密码
481
482 加密方法：
483 对称加密： 加密和解密使用同一个密码
484 公钥加密： 每个密码都成对出现，一个为私钥一个为公钥
485 单向加密，散列加密： 提取数据特征码，常用于数据完整性校验
486 1、雪崩效应    md5sum 就可以计算其特征码
487 2、定长输出
488    md5sum： Message Digest(信息摘要)，128(一位表示4位)位定长输出
489    shasum： Secure Hash Aigorithm ，160位定长输出
490
491 useradd 用户名
492 tail -1 /etc/passwd
493 tail -1 /etc/group
494 tail -1 /etc/shadow
495
496 cd /etc/default
497 cat /etc/default/adduser 确定添加用户时的默认选项
498
499 groupadd
500
501 用户管理：
502 useradd userdel usermod passwd chsh chfn finger id chage
503 组管理：
504 groupadd groupdel groupmod gpasswd
505 权限管理：
506 chwon chgrp chmod umask
507
508 更改ssh登录提示格式： /etc/update-motd.d
509 //4-01
```

```

510
511 useradd
512     -u UID: 1000
513     -g GID: 基本组
514     -G GID: 附加组或额外组可以有多个中间用逗号隔开
515     -c "注释信息"
516     -d /path/to/somedirectory
517 useradd -c "Tony Blare" -d /home/blare user4
518     -s shell:给用户分配默认shell
519 useradd -s /sbin/nologin user5
520 useradd -s /bin/tcsh user6
521     -m -k :强制创建家目录 etc/skel
522     -M : /etc/login.defs 强制不创建家目录即使默认配置中有家目录
523     -r : 添加系统帐号
524 /etc/shells:指定当前系统可用的安全shell
525
526 userdel :删除用户
527     -r : 删除其家目录
528
529 id: 查看用户id号
530     -n : 显示用户名
531     -u : 显示uid
532     -g : 显示gid
533     -G : 显示额外组
534 finger username: 查看用户帐号信息
535
536 修改用户信息:
537 usermod:
538     -u : 修改id号
539     -g : 修改基本组
540     -G : 修改附加组 与-a一起使用增加额外附加组 不使用-a则会覆盖以前的额外组
541     -c : 修改注释信息
542     -d : 与-m一起使用, 即创建新的家目录还将以前的家目录文件拷贝过去
543     -s : 修改shell
544     -l : 修改登录名
545     -e : 过期时间
546     -f : 非活动时间
547     -L : 锁定帐号
548     -U : 解锁帐号
549
550 chsh : chsh user2 : 修改用户的默认shell
551
552 chfn : 修改用户的注释信息
553     : chfn user3
554
555 密码管理:
556 passwd [USERNAME]
557     --stdin:从标准输入接口接受密码
558     echo "radhat" | passwd --stdin user4 管道输入 ubuntu竟然不支持
559     -l: 锁定用户帐号 即在/etc/shadow文件中密码的前面加入!!即可禁止登陆
560     -u: 解锁用户帐号
561     -d: 删除用户密码
562
563 pwck: 检查用户帐号完整性
564
565
566 组管理:
567 创建组: groupadd
568     -g: 指定Gid
569     -r: 添加系统用户
570
571 groupmod: 修改组
572     -g : gid
573     -n : 组名
574
575 groupdel: 删除组
576
577 gpasswd : 给组设定密码
578
579 newgrp: 登录到一个新组中
580
581 练习
582 1、创建一个用户mandriva, 其ID为2002, 基本组为distro (组ID为3003), 附加组为linux。

```



```

583 groupadd -g 3003 distro
584 groupadd linux
585 useradd -u 2002 -g distro -G linux mandriva
586
587 2、创建一个用户fedora，其全名为Fedora Community，默认shell为tcsh
588 useradd -c "Fedora Community" -s /bin/tcsh fedora
589
590 3、修改mandriva的ID号为4004，基本组为linux，附加组为distro和fedora
591 usermod -u 4004 -g linux -aG distro,fedora mandriva
592
593 4、给fedora加密码，并设定其密码最短使用期限为2天，最长为50天
594 echo "fedora" | passwd --stdin fedora
595 passwd -n 2 -x 50 fedora
596 5、将mandriva的默认shell改为/bin/bash;
597 chsh mandriva
598 6、添加系统用户hbase，且不允许其登录系统;
599 useradd -r hbase -s /sbin/nologin
600
601 chage:
602     -d: 最近一次的修改时间
603     -E: 过期时间
604     -I: 非活动时间
605     -m: 最短使用期限
606     -M: 最长使用期限
607     -w: 警告时间
608 //04-02
609 权限管理:
610 r:
611 w:
612 x:
613
614 三类用户:
615 u: 属主
616 g: 属组
617 o: 其他用户
618
619 chown: 改变文件属主（只有管理员可以使用此命令）
620
621 -R: 修改目录及其内部文件或内部文件的属主
622 --reference=/path/to/somefile file: 设置为与路径下相同的文件的属主信息
623 chgrp: 修改组名
624 -R: 递归修改
625 --reference=/path/to/somefile file: 设置为与路径下相同的文件的属组信息
626
627 chown USERNAME:GROUPNAME file
628 chown USERNAME,GROUPNAME file
629
630 chmod: 修改文件权限
631
632 修改三类用户的权限
633 -R:
634 --reference=/path/to/somefile file
635
636 修改某类用户或某些类用户权限
637 u, g, o, a
638 chmod 用户类别=MODE file
639 chmod u=rwx file
640 chmod g=rw-
641 chmod o=rx file
642 chmod g=r, u=r
643 chmod g0=w file
644
645 修改某类用户某位或某些权限
646 u, g, o, a
647 chmod 用户类别+ - -mod file
648
649 chmod u-x
650 chmod u+x, g-x
651 chmod a+x
652 chmod +x//三个用户同时操作
653 chmod u-wx file
654
655 练习

```

```
656 1、新建一个没有家目录的用户openstack
657 useradd -M openstack
658 2、复制/etc/skel为/home/openstack
659 cp -r /etc/skel /home/openstack
660 3、改变/home/openstack及其内部文件的属主属组为openstack
661 chown -R openstack:openstack openstack/
662 4、/home/openstack及其内部的文件，属组和其他用户没有任何访问权限
663 、 、 chmod 700 openstack/ ：一般不能让文件具有执行权限
664 chmod -R go= openstack
665
666
667 手动加一个用户hive，基本组为hive （5000）附加组为mygroup
668 nano /etc/group
669 nano /etc/passwd
670 nano /etc/shadow
671 cp -r /etc/skel /home/hive
672
673 openssl passwd
674 whatis passwd
675 man sslpasswd
676 openssl passwd -l -salt
677
678
679
680 umask:遮罩码
681 创建文件：666-umask
682 创建目录：777-umask
683
684 root用户umask为022
685 普通用户umask为002
686
687 umask 022 修改当前用户的umask
688
689 umask 023
690 文件：666-023=643
691 ：计算结果默认文件不能带有执行权限，当计算结果有执行权限时自动加一 所以结果为644
692 目录：777-023=754
693
694 站在用户的登录角度来说，SHELL的类型：
695 登录式shell
696 正常通过其他终端登录
697 su - USERNAME
698 su -l USERNAME
699
700 非登录式shell：
701 su USERNAME
702 图形终端下打开命令窗口
703 自动执行的shell脚本
704
705 bash的配置文件
706 全局配置
707 /etc/profile ， /etc/profile.d/*.sh， /etc/bashrc
708 个人配置
709 ~/.bash_profile ， ~/.bashrc
710
711 profile类的文件
712 设定环境变量
713 运行命令或脚本
714
715 bashrc类的文件
716 设定本地变量
717 定义命令别名
718
719 登录式shell如何获取配置文件
720 /etc/profile --> /etc/profile.d/*.sh --> ~/.bash_profile --> ~/.bashrc-->/etc/bashrc
721
722 非登录式shell如何配置文件
723 ~/.bashrc --> /etc/basrc --> /etc/profile.d/*.sh
724
725 bash:脚本解释器
726
727 计算机组成
```

728
729 运算器、控制器、CPU
730 存储器：RAM
731 输入设备/输出设备
732
733 程序：指令和数据
734
735 控制器：指令
736 运算器：
737 存储器：
738
739 地址总线：内存寻址
740 数据总线：传输数据
741 控制总线：控制指令
742
743
744 寄存器：cpu暂时存储器
745
746 I/O：硬盘，
747
748 程序
749
750 INPUT设备：
751
752 OUTPUT设备
753
754
755 系统设定
756 默认输出设备：标准输出 STDOUT 用1表示
757
758 默认输入设备：标准输入 STDIN 用0表示
759
760 标准错误输出：STDERR 2
761 标准输入：键盘
762 标准输出和错误输出：显示器
763
764 I/O重定向：
765
766 对于linux：
767 输出重定向：> 覆盖输出：ls /etc/var > /tmp/var.out cat /etc/fstab > /tmp/var.out
768 >>:追加输出，可以保留原文件内容，而且继续保留原文件内容
769
770 set
771 -C：开启功能 如果文件中初始有数据禁止使用覆盖方式写入
772
773 +C：关闭功能
774 set -C时强制覆盖输出 则使用 >| 即可
775
776 2>:重定向错误输出
777 2>>:追加输出
778
779 实现错误和正确的都定向输出：ls /var > /tmp/var3.out 2> /tmp/err.out
780
781 &>:重定向标准输出或错误输出至同一个文件
782
783
784 输入重定向：<
785 tr 'a-z' 'A-Z' < /etc/fstab
786
787 <<：在此处生成文档
788
789 cat << END
790
791 cat >> /tmp/myfile.txt << EOF :将此处生成的文件重定向至指定文件
792
793 管道：前一个命令的输出，作为后一个命令的输入
794
795 echo "hello ,world" | tr 'a-z' 'A-Z'
796 echo "radhat" | passwd --stdin USER
797 cat /etc/passwd | sort
798 cut -d : -f1 /etc/passwd | sort
799 cut -d : -f1 /etc/passwd | sort | tr 'a-z' 'A-Z' >> /tmp/username.txt
800

```

801 命令1 | 命令2 | 命令3
802
803
804 tee:即保存又输出至屏幕中
805
806 echo "hello.world" | tee /tmp/hello.out
807
808 wc -l /etc/passwd | cut -d / -f1 :管道实现只输出行数
809 wc -l /etc/passwd | cut -d \ ' -f1 也可以实现
810 ls -l /usr/bin | head -1
811 ls /usr/bin/ | wc -l
812
813 练习:
814 1、统计/usr/bin/ 目录下的文件个数
815    ls /usr/bin | wc -l
816 2、取出当前系统上所有用户的shell, 要求, 每种shell只显示一次, 并且按顺序进行显示
817    cut -d : -f 7 /etc/passwd | sort -u
818 3、思考: 如何显示/var/log目录下每个文件的内容类型
819    file /var/log/*
820    file $(ls /var/log)
821 4、取出/etc/inittab文件的第六行
822    head -6 /etc/inittab | tail -1
823 5、取出/etc/passwd
824    文件中倒数第9个用户的用户名和shell, 显示到屏幕上并将其保存至/tmp/users文件中
825    sort -r /etc/passwd | cat -n 先逆序输出然后显示行号
826    tail -9 /etc/passwd | head -1 | tee /tmp/users
827    tail -9 /etc/passwd | head -1 | cut -d : -f 1,7 | tee /tmp/users
828 6、显示/etc目录下所有以pa开头的文件, 并统计其个数
829    ls -d /etc/pa* | wc -l 要显示为目录不然ls会显示目录中的内容
830 7、不使用文本编辑器, 将alias cls=clear 一行内容添加至当前用户的.bashrc文件中
831    echo "alias cls=clear" >> .bashrc
832
833 文本查找的需要
834 grep, egrep, fgrep
835
836 grep: 根据模式, 搜索文本, 并将符合模式的文本行显示出来
837 pattern: 模式 文本字符和正则表达式的元字符组合而成的匹配条件
838 grep 'root' /etc/passwd
839    -i: 忽略字符大小写
840    grep --color 'root' /etc/passwd
841    -v: 反向查找, 显示没有被模式匹配到的行
842    -o: 只显示被模式匹配到的串
843
844 *:
845 ? :
846 []
847 [^]:
848
849 正则表达式: regular expression
850 元字符:
851 .: 匹配任意单个字符
852 []:匹配指定范围内的任意单个字符
853 [^]:匹配指定范围外的任意单个字符
854    字符集合: [:digit:],[:lower:],[:upper:],[:punct:],[:space:],[:alpha:],[:alnum:]
855
856 grep 'r..t' /etc/passwd
857
858 字符次数(贪婪模式):
859 *: 匹配其前面的字符任意次
860    a, b, ab, aab, acb, adb, amnb
861    a*b :
862    a.*b :ab , aab, acb, adb, amnb
863    .*:任意长度的任意字符
864
865 \?: 匹配其前面的字符1次或0次
866
867 \{m,n\}:匹配其前面的字符至少m次, 至多n次
868    \{1,\}:至少一次
869    \{0,3\}:最多三次
870
871 grep 'a\{1,3\}b' test.txt :匹配a至少一次之多三次
872 grep 'a.\{1,3\}b' test.txt :匹配任意字符至少一次之多三次

```

873 位置锁定：
874 ^:锁定行首，此字符后面的任意内容必须出现在行首
875 \$:锁定行尾，此字符前面的任意内容必须出现在行尾
876 ^\$: 空白行
877 grep '[[[:digit:]]\$' /etc/inittab 查找以数字结尾的行
878
879 \<或者\b:锚定词首，其后面的任意字符必须作为单词首部出现
880 \>或者\b:锚定词尾，其前面的任意字符必须作为单词的尾部出现
881
882 \<root\>:只寻找root这个单词
883
884 grep "root\>" test2.txt
885
886 分组：
887 \(\)
888 \ (ab\)*:将ab作为一个整体可以出现任意次
889 后向引用
890 \1:引用第一个左括号以及与之对应的右括号所包括的所有内容
891 \2:
892 \3:
893
894 grep "\ (l..e\).*\1" test3.txt : 这样实现前后一致才可以匹配
895 grep "\ ([0-9]\).*\1\$" /etc/inittab
896
897
898 He love his lover.
899 She like her liker.
900 he like his lover
901
902 He love his lover.
903 She like her lover.
904 He like his liker.
905 She love her liker.
906
907 grep "l..e" test3.txt
908
909 grep '\ ([0-9]\).*\1\$' /etc/inittab
910
911 l..e
912
913 //5-1
914 正则表达式：
915 Basic REGEXP: 基本正则表达式
916 Extended REGEXP: 扩展正则表达式
917
918 grep: 使用基本正则表达式定义的模式来过滤文本的命令
919 -i: 忽略字符大小写
920 -v: 反向搜索
921 -o: 只显示匹配到的字符串
922 --color : 显示颜色
923 -E: 使用扩展正则表达式
924 -A #: grep -A 2 "^core id" /proc/cpuinfo 表示附加后面两行
925 -B #: 附加显示前面#行
926 -C #: 上下都#行
927
928 扩展的正则表达式：
929 字符匹配：
930 与基本相同
931 .
932 []
933 [^]
934
935
936
937 次数匹配
938 *:任意次
939 ?:匹配零次或一次
940 +:匹配其前面的字符至少一次 等于基本表达式\{1,\}
941 {m, n} : 次数匹配
942
943 锚定字符相同
944 ^:行首
945 \$:行尾

```

946 \<:词首
947 \>:词尾
948
949 分组:
950 ():真正意义的分组
951 \1,\2,\3,...
952
953 或者
954 |:a or b
955
956 C|cat :Cat or cat .是 C或cat
957 grep -E 'C|cat' test4.txt
958 grep -E "(C|c)at" test4.txt 分组找到Cat或cat
959 grep -E "^[[:space:]]+" /boot/grub/grub.conf :以至少一个空白字符开头的
960 grep -E =egrep
961
962 找出ifconfig命令结果中的1-255之间的整数
963
964 \<([1-9] | [1-9] [0-9] | 1[0-9] [0-9] | 2[0-4][0-9] | 25 [0-5])\>
965
966 ifconfig | egrep '\<([1-9] | [1-9][0-9] | 1[0-9][0-9] | 2[0-4][0-9] | 25[0-5])\>'
967
968 \.
969
970
971 上面的是错的下面的是对的。
972 egrep '\<([0-9] | [1-9][0-9] | 1[0-9][0-9] | 2[0-4][0-9] | 25[0-5])\>'
973 ifconfig | egrep
974 '\<([0-9] | [1-9][0-9] | 1[0-9][0-9] | 2[0-4][0-9] | 25[0-5])\>\.\<([0-9] | [1-9][0-9] | 1[0-9][0-9] | 2[0-4][0-9] | 25[0-5])\>\.\<([0-9] | [1-9][0-9] | 1[0-9][0-9] | 2[0-4][0-9] | 25[0-5])\>'
975
976 ifconfig | egrep -o
977 '\<([0-9] | [1-9][0-9] | 1[0-9][0-9] | 2[0-4][0-9] | 25[0-5])\>\.) {3}\<([0-9] | [1-9][0-9] | 1[0-9][0-9] | 2[0-4][0-9] | 25[0-5])\>'
978
979 ipv4:
980 5类: A B C D E
981
982 A:1-127
983 B:128-191
984 C:192-223
985
986
987 只找IP地址:
988
989 egrep -o
990 '\<([1-9] | [1-9][0-9] | 1[0-9][0-9] | 2[01][0-9] | 22[0-3])\>(\.\<([0-9] | [1-9][0-9] | 1[0-9][0-9] | 2[0-4][0-9] | 25[0-5])\>){2}\.\<([1-9] | [1-9][0-9] | 1[0-9][0-9] | 2[0-4][0-9] | 25[0-5])\>'
991
992 //05-02
993 grep, egrep,
994 fgrep: fast快速的grep, 不支持正则表达式
995
996 shell编程:
997
998 编程语言: 机器语言, 汇编语言, 高级语言
999
1000 静态语言: 编译型语言
1001 强类型(变量)
1002 关键字:
1003 事先转换成可执行格式
1004 c, c++, java, c#
1005
1006 动态语言: 解释型语言, on the fly
1007 边解释边执行
1008 PHP、SHELL、PYTHON、PERL
1009
1010 面向过程: shell, c
1011 面向对象: java, Python, perl, c++

```

1012 内存：遍址的存储单元
1013
1014 进程：
1015 1+100;
1016 1+1000000
1017
1018 变量：内存空间
1019 1,10000所占去的内存不同
1020 字符型10：16bit
1021 数值10：1010 4bit
1022
1023 变量类型：事先确定数据的存储格式和长度
1024 字符
1025 数值
1026 整形：
1027 浮点型：11.23
1028 2013/10/10 ，64bit
1029 99999：24bit
1030 真、假
1031 逻辑：1+1>2
1032 逻辑运算：与、或、非、异或
1033 1：真
1034 0：假
1035
1036 1 & 0 = 0
1037 0 & 1 = 0
1038 0 & 0 = 0
1039 1 & 1 = 1
1040
1041 或
1042
1043 非：单目运算符
1044 ! 真 = 假
1045 ! 假 = 真
1046
1047 异或：
1048
1049 相同为假
1050 不同为真
1051
1052
1053
1054 整形，8bit ：256
1055 0-255,溢出
1056
1057 shell编程：弱类型编程语言
1058
1059 强：变量在使用前，必须事先声明，甚至还需初始化
1060 弱类型：变量用时申明，甚至不区分类型
1061
1062 11+c =
1063
1064 显示转换
1065 隐式转换
1066
1067 变量赋值：var_name=value
1068
1069 脚本编程
1070
1071 变量名称：
1072 1、只能包含字母数字和下划线，并且不能数字开头
1073 2、不应该跟系统中已有的环境变量重名
1074 3、最好做到见名知意
1075
1076
1077 bash变量类型
1078 环境变量
1079 本地变量（局部变量）
1080 位置变量
1081 特殊变量
1082
1083 本地变量：整个bash进程
1084 bash：

```
1085 (set) NAME=jerry
1086 echo $(NAME)
1087 声明局部变量
1088 local VARNAME=VALUE: 作用域为当前代码段
1089
1090 环境变量: 作用域为当前shell进程及其子进程
1091 export VARNAME=VALUE
1092 "导出"
1093
1094 脚本在执行时会启动一个shell进程
1095 命令行中启动的脚本会继承当前shell环境变量
1096 系统自动执行的脚本(非命令行启动)就需要自我定义需要各环境变量
1097
1098 位置变量:
1099 $1,$2,...
1100
1101 特殊变量:
1102 $?:上一个命令的执行状态返回值
1103
1104 程序执行, 可能有两类返回值
1105 程序执行结果
1106 程序状态返回代码
1107 0: 正确执行
1108 1-255: 返回错误类型 1, 2, 127 系统预留
1109
1110 输出重定向
1111 >
1112 >>
1113 2>
1114 2>>
1115 &>
1116
1117 /dev/null :软件设备 , bit bucket位桶, 数据黑洞
1118
1119 撤销变量:
1120 unset VARNAME
1121
1122 查看当前shell中的所有变量: set
1123
1124 查看当前shell中的环境变量:
1125 printenv
1126 env
1127 export
1128
1129 定义一堆字符串
1130 ANIMALS=pig
1131 ANIMALS=$ANIMALS:goar
1132
1133 export PATH=$PATH:/usr/local/apache/bin 一种追加赋值的方法
1134
1135 shell默认定义为字符串变量
1136 A=2
1137 B=3
1138 C=$A+$B
1139 echo $C
1140 显示结果为2+3
1141
1142 引用变量: ${VARNAME}, 括号有时可省略,
1143 ``: 弱引用, 用``时显示的内部变量不会被引用
1144 ``: 强引用, ``里面的东西直接被引用
1145
1146 脚本: 命令的堆砌, 按实际需要, 结合命令流程控制机制实现的源程序
1147
1148 shebang: 魔数
1149 #! /bin/bash
1150 # 注释行, 不执行
1151
1152 练习: 写一个脚本完成一下任务
1153 1、添加5个用户, user1,...,user5
1154 2、每个用户的密码同用户名, 而且要求, 添加密码完成后不显示passwd命令的执行结果信息
1155 3、每个用户添加完成后, 都要显示用户某某已经添加成功
1156
1157 useradd user1
```



```

1158 echo 'user1' | passwd --stdin user1 > /dev/null
1159 echo "user1添加成功"
1160 useradd user2
1161 echo 'user2' | passwd --stdin user2 > /dev/null
1162 echo "user2添加成功"
1163 useradd user3
1164 echo 'user3' | passwd --stdin user3 > /dev/null
1165 echo "user3添加成功"
1166
1167 练习：写一个脚本，完成一下任务
1168 1、使用一个变量保存一个用户名
1169 2、删除此变量中的用户，并且一并删除其家目录：
1170 3、显示“用户删除完成”类的信息
1171
1172 NAME=user1
1173 userdel -r $NAME
1174 echo "用户$NAME删除成功"
1175
1176 NAME=user2
1177 userdel -r $NAME
1178 echo "用户$NAME删除成功"
1179
1180 NAME=user3
1181 userdel -r $NAME
1182 echo "用户$NAME删除成功"
1183 //5-3
1184 条件判断：
1185     如果用户不存在
1186         添加用户，给密码并显示添加成功
1187     否则
1188         显示如果已经存在，没有添加
1189
1190 如果用户存在，就显示用于已存在，否则，就添加用户
1191 id user1 &> /dev/null && echo "user1 is a user" || useradd user1
1192
1193 如果用户不存在，就添加，否则显示其已存在
1194 ! id user1 &> /dev/null && useradd user1 || echo "user1 exists"
1195
1196 如果用户不存在，就添加并给密码：否则，显示其已经存在
1197 ! id user1 &> /dev/null && useradd user1 && echo "user1" | passwd --stdin user1 ||
echo "user1 exists"
1198
1199 bash中如何实现条件判断？
1200 条件测试类型：
1201     整数测试
1202     字符测试
1203     文件测试
1204
1205 条件测试的表达式：
1206     [ expression ]
1207     [[ expression ]]
1208     test expression
1209
1210 整数比较：
1211     -eq：测试两个整数是否相等：比如[ $A -eq $B ]
1212     -ne：测试两个整数是否不等，不等为真，相等为假
1213     -gt：测试一个数是否大于另一个数：大于为真，否则为假
1214     -lt：测试一个数是否小于另一个数：小于为真，否则为假
1215     -ge：大于或等于
1216     -le：小于或等于
1217
1218 命令间的逻辑关系：
1219     逻辑与：&&
1220         第一个条件为假时，第二个条件不再判断，最终结果已经确定
1221         第一个条件为真时，第二个条件必须得判断
1222     逻辑或：||
1223         第一个为假时，
1224         !：取反
1225
1226 id user1 &> /dev/null && echo "hello.student"
1227
1228 如果用户不存在，就添加用户user6
1229 ! id user6 && useradd user6

```

```

1230 id user6 || useradd user6
1231
1232 如果/etc/inittab文件的行数大于100，就显示好大的文件
1233 [ `wc -l /etc/inittab | cut -d ' ' -f 1` -gt 100 ] && echo "It is a big file" ||
echo "It is a small file"
1234
1235 变量名称：
1236     1、只能包含字母，数字和下划线，并且不能以数字开头；
1237     2、不应该跟系统中已有的环境变量重名；
1238     3、最好做到见名知意；
1239
1240 如果用户存在，就显示用户已经存在；否则，就添加此用户；
1241 id zhangshuo &> /dev/null && echo "zhangshuo is exist" || useradd zhangshuo
1242
1243 如果用户不存在，就添加；否则，显示其已经存在
1244 ! id zhangshuo &> /dev/null && useradd zhangshuo || echo "zhangshuo exists"
1245
1246 如果用户不存在，添加并且给密码，否则，显示其已经存在
1247 ! id zhangshuo &> /dev/null && useradd zhangshuo && echo "zhangshuo" | passwd
--stdin zhangshuo || echo "zhangshuo exists"
1248
1249 练习，写一个脚本，完成一下要求；
1250 1、添加3个用户user1，user2,user3，但要先判断用户是否存在，不存在而后再添加；
1251 2、添加完成后，显示一共添加了几个用户；当然，不能包括因为实事先存在而没有添加的；
1252 3、最后显示当前系统一共有多少个用户
1253
1254 练习，写一个脚本，完成以下要求：
1255 给定一个用户：
1256     1、如果其UID为0，就显示此为管理员
1257     2、否则，就显示其为普通用户；
1258
1259 [ $(id -u root) -eq 0 ] && echo "此用户为管理员" || echo "该用户为普通用户"
1260 "
1261
1262 如果UID为0那么
1263     显示为管理员
1264 否则
1265     显示普通用户
1266
1267 NAME=user16
1268 USERID=$(id -u $NAME)
1269 if [ $USERID -eq 0 ];then
1270     echo "Admin"
1271 else
1272     echo "common user"
1273 fi
1274
1275
1276
1277
1278 练习：写一个脚本完成一下任务
1279 1、使用一个变量保存一个用户名
1280 2、删除此变量中的用户，并一并删除其家目录
1281 3、显示“用户删除完成”类的信息
1282
1283 条件判断，控制结构
1284
1285 单分支的if语句
1286 if 判断条件; then
1287     statement1
1288     statement2
1289     ...
1290 fi
1291
1292 双分支的if语句：
1293 if 判断条件; then
1294     statement1
1295     statement1
1296
1297 else
1298     statement3
1299
1300 fi

```

```

1301
1302 如果用户存在则显示用户存在
1303 #!/bin/bash
1304 NAME=user1
1305 if id $NAME &> /dev/null;then
1306     echo "$NAME 用户存在"
1307 else
1308     echo "$NAME 用户不存在"
1309 fi
1310
1311 如果用户存在显示用户存在，用户不存在则建立用户并命名密码
1312 if id $NAME &> /dev/null ;then
1313     echo "$NAME 用户存在"
1314 else
1315     useradd user1
1316     echo "user1" | passwd --stdin user1
1317     echo "$NAME 用户不存在，所以建立了此用户并赋予与用户名相同的密码"
1318 fi
1319
1320 练习：写一个脚本
1321 判断当前系统上是否有用户的默认shell为bash
1322 如果有，就显示有多少个这类用户；否则，就显示没有这类用户
1323
1324 #!/bin/bash
1325 USERBASHNUM=`grep "\<bash\>" /etc/passwd | wc -l`
1326 if [ $USERBASHNUM -eq 0 ];then
1327     echo "no such user"
1328 else
1329     echo "$USERBASHNUM users"
1330 fi
1331
1332
1333 练习：写一个脚本
1334 给定一个文件，比如/etc/inittab
1335 判断这个文件中是否有空白行
1336 如果有，则显示空白行数否则，就显示没有空白行
1337
1338 #!/bin/bash
1339 #by zhangshuo 2018/01/15
1340 FILENAME=/etc/inittab
1341 grep "^$" $FILENAME &> /dev/null
1342 RES=$?
1343 if [ $RES -eq 0 ];then
1344     BACKNUM=`grep "^$" $FILENAME | wc -l`
1345     echo "$BACKNUM backline in $FILENAME"
1346 else
1347     echo "no back line"
1348 fi
1349
1350 练习：写一个脚本
1351 给定一个用户，判断其UID与GID是否一样
1352 如果一样，就显示此用户为“GOOD GUY”：否则，就显示此用户为“BAD GUY”
1353
1354 #!/bin/bash
1355 #by zhangshuo 2018/01/15
1356 USERNAME=zhangshuo
1357 USER_UID=`id -u $USERNAME`
1358 USER_GID=`id -g $USERNAME`
1359 if [ $USER_UID -eq $USER_GID ];then
1360     echo "good guy"
1361 else
1362     echo "bad guy"
1363 fi
1364
1365 进一步要求，不使用id命令获得其id号
1366
1367 练习：写一个脚本
1368 给定一个用户，获取其密码的警告期限：
1369 而后判断用户最近一次修改密码时间距离今天是否已经小于警告期限
1370 提示：算数运算的方法${A-$B}：表示变量A的值减去变量B的值的结果
1371 如果小于，则显示“warning”：否则，就显示“OK”
1372
1373 #!/bin/bash

```

```

1374 #
1375 USERNAME=zhangshuo
1376 DATEWARNING=`grep "$USERNAME" /etc/shadow | cut -d : -f 5`
1377
1378 DATENOW=$(echo "`date +%s`/86400" | bc)
1379
1380 DATECHAGE=`grep "$USERNAME" /etc/shadow | cut -d : -f 3`
1381
1382 let DATEERR=$DATENOW-$DATECHAGE
1383
1384 if [ $DATEERR -gt $DATEWARNING ];then
1385     echo "warning"
1386 else
1387     echo "ok"
1388 fi
1389
1390 练习：写一个脚本
1391 判定命令历史中命令的总条目是否大于1000;如果大于，就显示"some command will gone
1392 ": 否则显示"OK"
1393
1394 shell中如何进行算数运算
1395 A=3
1396 B=6
1397 1、let 算数运算表达式
1398     let C=$A+$B
1399 2、$[算数运算表达式]
1400     C=$[A+B]
1401 3、$((A+B))
1402 4、expr 算数运算表达式，表达式中各操作数及运算符之间要有空客，而且要使用命令引用
1403     C=$(expr $A + $B)
1404     C=`expr $A + $B`
1405
1406 A=3
1407 B=6
1408 let C=$A+$B
1409
1410 //06-01
1411 exit:退出脚本
1412 exit #
1413
1414 如果脚本没有明确定义退出状态码，那么，最后执行的一条命令的退出码即为脚本的退出状态码
1415
1416 测试方法：
1417 [ expression ]
1418 [[           ]]
1419 test expression
1420
1421
1422 bash中常用的测试条件有三种
1423 整数测试
1424     -gt
1425     -le
1426     -ne
1427     -eq
1428     -ge
1429     -lt
1430
1431 INT1=63
1432 INT2=77
1433 [ $INT1 -eq $INT2 ]
1434 [[ $INT1 -eq $INT2 ]]
1435 test $INT1 -eq $INT2
1436
1437 if [ grep "^$USERNAME\>" /etc/passwd ];then 此条件不要使用中括号
1438
1439
1440 文件测试：
1441 -e FILE: 测试文件是否存在
1442 -f FILE: 测试文件是否为普通文件
1443 -d FILE: 测试指定路径是否为目录
1444 -r FILE: 测试当前用户为指定文件是否有读权限
1445 -w FILE: 测试当前用户为指定文件是否有写权限

```

```

1446 -x FILE: 测试当前用户为指定文件是否有执行权限
1447
1448 测试脚本是否有语法错误:
1449 bash -n 脚本好像不好用
1450 bash -x 显示每一步执行的程序并显示出来
1451
1452
1453
1454 [ -e /etc/inittab ]:测试inittab文件是否存在
1455 [ -x /etc/rc.d/rc.sysinit ]:测试指定文件是否有执行权限
1456
1457 FILE=/etc/inittab
1458 if [ ! -e $FILE ];then
1459     echo "NO $FILE"
1460     exit 8
1461     : 这个数字是自己定义的退出码, 如果没有定义退出码, 则最后执行的一条命令的退出码即为脚
      本的退出状态码
1462 fi
1463
1464 知识补充: 多分支if语句
1465 if 判断条件1;then
1466     state
1467 elif 判断条件2;then
1468     state
1469 elif 判断条件;then
1470     state
1471 else
1472     state
1473 fi
1474
1475 练习: 写一个脚本
1476 给定一个文件:
1477 如果是一个普用文件, 就显示之
1478 如果是一个目录, 亦显示之
1479 否则, 此为无法识别之文件
1480
1481 #!/bin/bash
1482 #by zhangshuo 2018/01/15
1483
1484 FILENAME=/etc/inittab
1485
1486 if [ -e $FILENAME ];then
1487     if [ -f $FILENAME ];then
1488         echo "it is a common file"
1489     elif [ -d $FILENAME ];then
1490         echo "it is a dir"
1491     else
1492         echo "cat recognize"
1493     fi
1494 else
1495     echo "no such file"
1496     exit 4
1497 fi
1498
1499
1500
1501 bash变量的类型:
1502 本地变量 (局部变量)
1503 环境变量
1504 位置变量 : $1,$2...
1505         shift n
1506 特殊变量
1507
1508 ./filetest.sh /etc/fatab /etc/inittab
1509 $1:/etc/fatab
1510 $2:/etc/inittab
1511
1512 练习: 写一个脚本
1513 能接受一个参数(文件路径)
1514 判定: 此参数如果是一个存在的文件;并显示其文件类型, 否则就显示"no such file "
1515
1516 #!/bin/bash

```

```

1517 #by zhangshuo 20180115
1518
1519 if [ $# -eq 0 ];then
1520     echo "usage !!!!!!!!!!!!"
1521     exit 2
1522 fi
1523
1524 if [ -e $1 ];then
1525     if [ -f $1 ];then
1526         echo "common file"
1527     elif [ -d $1 ];then
1528         echo "dir"
1529     else
1530         echo "unkonw"
1531     fi
1532 else
1533     echo "no such file"
1534     exit 3
1535 fi
1536
1537

```

特殊变量:

```

1538     $?: 返回码
1539     $#: 参数的个数
1540     $*: 参数列表
1541     $@: 参数列表
1542
1543

```

练习: 写一个脚本

给脚本传递两个参数{整数};
显示此两者之和。之积

```

1544
1545
1546
1547
1548 #!/bin/bash
1549 #by zhangshuo 20180115
1550 if [ $# -lt 2 ];then
1551     echo "usage!!!!!!!!!!!!!!!"
1552     exit 2
1553 fi
1554
1555 let SUM=$1+$2
1556 let SQR=$1*$2
1557
1558 echo "SUM is $SUM"
1559 echo "SQR is $SQR"
1560
1561 //06-02
1562 处理文本: 工具
1563 grep, sed (数据流编辑器), awk
1564

```

sed基本用法:

```

1566 sed : stream editor
1567     行编辑器 (全屏编辑器: vi)
1568

```

sed: 模式空间

默认不编辑原文件, 仅对模式空间中的数据做处理

```

1571
1572 sed [options] 'AddressCommand' file ...
1573     -n: 静默模式 不显示模式空间中的内容 不然匹配到的行会显示两次
1574     -i: 直接修改原文件
1575     -e SCRIPT -e SCRIPT...: 可以同时执行多个脚本
1576     -f /PATH/TO/SED_SCRIPT
1577         sed -f /path/to/scripts file :挨个处理脚本中的操作
1578     -r : 使用扩展正则表达式
1579

```

有三种方式可以指定命令行上的多重指令:

1、用分号分隔指令

```

1582     sed 's/ MA/ ,Massachusetts;/s/ PA/, Pennsylvania/' list
1583

```

2、在每个指令前放置-e

```

1584     sed -e 's/ MA/ ,Massachusetts/' -e 's/ PA/, Pennsylvania/' list
1585

```

3、使用bash的分行指令功能。在输入单引号后按RETURN键, 就会出现多行输入的提示符(>).

```

1586     sed '
1587         s/ MA/, Massachusetts/
1588         s/ PA/, Pennsylvania/
1589         s/ CA/, California/' list

```

```

1590
1591 Address:
1592 1.StartLine,EndLine
1593 比如1,100
1594 $:最后一行
1595 $-1: 倒数第二行
1596 2./RegExp/
1597 /^root/
1598 3./pattern1/,/pattern2/:表示被第一次模式匹配到的行到第二次模式匹配到的行中间的所有行
1599 4.LineNumber :表示精确某行
1600 5.StartLine,+N
1601 从startLine开始, 向后的N行
1602
1603 command:
1604 d: 删除符合条件的行
1605 sed '1,2d' /etc/fstab
1606 sed '/oot/d' /etc/fstab
1607 sed '1,+2d' /etc/fstab
1608 p:显示符合条件的行
1609 sed '/^\\//d' /file :删除以斜线开头的行
1610 sed -n '/^\\//p' /file :只显示匹配到的行
1611 a \\string:在指定的行后面追加新行, 内容为"string"
1612 sed '/^\\#/a \\# hello world' /etc/fstab :
1613 i \\string :在指定的行前面添加新行, 内容为string
1614 r file :将指定的文件的内容添加至符合条件的行处
1615 sed '2r /etc/issue' /etc/fstab
1616 sed '$r /etc/issue' /etc/fstab
1617 sed '1,2r /etc/issue' /etc/fstab
1618 w file:将地址指定范围内的内容另存至符合条件的行处
1619 sed '/oot/w /tmp/oot.txt' /etc/passwd
1620 s/pattern/string/:查找并替换 :默认每行中第一次被模式匹配到的串
1621 sed 's/oot/OOT/' /etc/passwd
1622 修饰符
1623 g: 全局替换
1624 sed 's/oot/OOT/g' /etc/passwd
1625 sed 's/^\\//#/g' /etc/passwd
1626 i:忽略字符大小写
1627 s///:s###:s@@@:s!!!
1628
1629 后向引用
1630 \\(\\),\\1,\\2
1631
1632 l..e:like-->liker
1633 love-->lover
1634
1635 &:引用模式匹配整个串
1636
1637 sed 's#l..e#&r#' sed.txt
1638 sed 's#\\(l..e\\)#\\lr#' sed.txt :后向引用
1639 sed 's#l\\(..e\\)#L\\l#' file :引用一部分
1640
1641 history | sed 's#[[:space:]]##g'
1642 history | sed 's/^[[[:space:]]*// ' | cut -d ' ' -f 1
1643
1644 sed练习:
1645 1、删除/etc/grub.conf文件中行首的空白字符
1646 sed 's#^[[[:space:]]*##g' sed.txt
1647 sed -r 's#^[[[:space:]]*##g' sed.txt
1648 2、替换/etc/inittab文件中"id:3:initdefault:"一行中的数字为5
1649 sed 's#\\(id:\\)[0-9]\\(\\(:initdefault\\)\\)#\\15\\2#g' /etc/inittab
1650 3、删除/etc/inittab文件中的空白行
1651 sed '/^$/d' sed.txt
1652
1653 删除/etc/inittab文件中开头的#号
1654 sed 's@^#@g' /etc/inittab
1655
1656 4、删除/etc/inittab文件中开头的#号, 但要求#号后面必须有空白字符
1657 sed 's@^#[[:space:]]*+@g' /etc/inittab
1658
1659 5、删除某文件中开头的#号及后面的空白字符, 但要求#号后面必须有空白字符
1660
1661 6、删除某文件中以空白字符后面跟#类的行中的开头的空白字符及#
1662

```

```

1663     sed 's@^[[[:space:]]\+#@@g' /etc/inittab
1664
1665 7、取出一个文件路径的目录名称
1666
1667     echo "/etc/rc.d" | sed -r 's@^(/.*\/) [^/]+/?@l@g'
1668     取出文件的基名
1669     echo "/etc/rc.d" | sed -r 's@^/.*/([^\/]+/?)@l@g'
1670
1671 #abc
1672 # hello world
1673 # hi world
1674
1675 练习：
1676 传递一个用户名参数给脚本，判断此用户的用户名跟其基本组的组名是否一致，并将结果显示出来。
1677 #!/bin/bash
1678 #by zhangshuo 20180116
1679
1680 if [ $# -eq 0 ];then
1681     echo "usage!!!!"
1682     exit 2
1683 fi
1684
1685 USERUID=`id -un $1`
1686 USERGID=`id -gn $1`
1687
1688 if [ $USERUID == $USERGID ];then
1689     echo "same"
1690 else
1691     echo "dont same"
1692 fi
1693 字符测试：
1694 ==：相等为真不等为假（等号两段必须有空格）
1695 !=：不等为真，等则为假
1696 >
1697 <
1698 -s string:测试字符串是否为空，空则真，不空则假
1699 -n string:测试指定字符串是否不空，不空为真，空则为假
1700
1701 写一个脚本
1702 传递一个参数（单字符串就行）给脚本，如果参数为q、Q、quit或Quit,就退出脚本；否则，就显示用户的参数：
1703 #!/bin/bash
1704 #by zhangshuo 20180116
1705
1706 if [ $# -eq 0 ];then
1707     id
1708     exit 2
1709 fi
1710
1711 if echo $1 | egrep '\<(q|Q|quit|Quit)\>' &> /dev/null ;then
1712     exit 2
1713 else
1714     echo $1
1715 fi
1716
1717 if [ $1 == 'q' -o $1 == 'Q' -o $1 == 'quit' -o $1 == 'Quit' ];then
1718     exit 2
1719 else
1720     echo $1
1721 fi
1722
1723 练习：
1724 传递三个参数给脚本，第一个为整数，第二个为算数运算符，第三个为整数，将计算结果显示出来，
1725 要求保留两位精度。形如： ./calc.sh 5/2
1726 echo $(echo "scale=2;$1$2$3" | bc)
1727 bc <<< "scale=2;5/2"
1728
1729 #!/bin/bash
1730 #by zhangshuo 20180116
1731
1732 echo "使用*时需要使用逃逸字符即\"

```



```

1733 echo `echo "scale=2;$1$2$3" | bc`
1734
1735
1736 练习: (line1491.sh)
1737 传递三个参数给脚本, 参数均为用户名, 将这些用户的账号信息提取出来后放置于/tmp/testusers
    .txt文件中, 并要求
1738 每一行行首有行号。
1739
1740 写一个脚本: (line1495.sh)
1741 判断当前主机的CPU生产商, 其信息在/proc/cpuinfo文件中vendor_id 一行中。
1742 如果其生产商为AUTHENTICamd, 就显示其为AMD公司;
1743 如果其生产商为GenuineInter, 就显示其为Inter公司;
1744 否则, 就说其为非主流公司;
1745
1746 写一个脚本: (line1501.sh)
1747 给脚本传递三个参数, 判断其中的最大数和最小数, 并显示出来。
1748
1749 循环: 进入条件, 退出条件
1750 for
1751 while
1752 until
1753
1754 for 变量 in 列表 ;do
1755     循环体
1756 done
1757
1758 遍历完成之后, 退出:
1759
1760 如何生成列表:
1761 {1..100} // 整数列表
1762
1763 seq 5 10
1764 seq 10
1765 seq 启示数 步进长度 结束数
1766
1767 1,...,100
1768
1769 declare -i SUM=0
1770
1771 for I in $(seq 1 1 100);do
1772     let SUM=${SUM+$I}
1773 done
1774
1775 echo "the sum is :$SUM"
1776
1777 #!/bin/bash
1778 #by zhangshuo 20180116
1779
1780 declare -i SUM=0
1781 for I in `seq 1 1 100`;do
1782     let SUM+=I
1783 done
1784
1785 echo $SUM
1786
1787 写一个脚本: (line1532.sh)
1788 1、设定变量FILE的值为/etc/passwd
1789 2、依次向/etc/passwd中的每个用户问好, 并显示对方的shell, 型如:
1790     Hello, root, your shell : /bin/bash
1791 3、统计一共有多少个用户
1792
1793 #!/bin/bash
1794 #by zhangshuo 20180116
1795
1796 declare -i NUM=0
1797
1798 for I in `cat /etc/passwd | cut -d : -f 1`;do
1799     #echo "helo $I,your shell :`grep "$I" /etc/passwd | cut -d : -f 7`"
1800     USERSHELLS=$(grep "^<$I:" /etc/passwd | cut -d : -f 7)
1801     if [ $USERSHELLS == '/bin/bash' ];then
1802         echo "helo $I,your shell : $USERSHELLS"
1803         let NUM+=1
1804     fi

```

```

1805 done
1806
1807 echo $NUM
1808
1809 只向默认shell为bash的用户问好
1810
1811 写一个脚本: (line1540.sh)
1812 1、添加10个用户user1到user10, 密码同用户名 但是要求只有用户不存在的条件下才能添加
1813
1814 扩展:
1815 接受一个参数:
1816 add: 添加用户user1..user10
1817 del: 删除用户user1..user10
1818 其他: 退出      再继续扩展: --add user1,user2,user3,hello,hi (line1547.sh)
1819 #!/bin/bash
1820 #by zhangshuo 20180116
1821
1822 if [ $# -eq 0 ];then
1823     echo "usage!!!!"
1824     exit 2
1825 fi
1826
1827 if [ $1 == '--add' ];then
1828     for I in `seq 1 1 10`;do
1829         if ! id user$I &> /dev/null;then
1830             echo "add user$I"
1831             useradd user$I
1832             echo "user$I" | passwd --stdin user$I &> /dev/null
1833         else
1834             echo "user$I exisit"
1835         fi
1836     done
1837 elif [ $1 == '--del' ];then
1838     for I in {1..10};do
1839         if id user$I &> /dev/null;then
1840             echo "del user$I"
1841             userdel -r user$I
1842         else
1843             echo "no user to del"
1844         fi
1845     done
1846 else
1847     exit 0
1848 fi
1849
1850 #!/bin/bash
1851 #by zhangshuo 20180117
1852
1853 if [ $# -lt 2 ];then
1854     echo "usage!!!!!!"
1855     exit 2
1856 fi
1857
1858 if [ $1 == '--add' ];then
1859     echo "add users"
1860     for I in `echo $2 | sed 's@, @g'`;do
1861         if id $I &> /dev/null;then
1862             echo "user $I exisit"
1863         else
1864             echo "add user $I done"
1865         fi
1866     done
1867 fi
1868
1869 写一个脚本: (line1549.sh)
1870 计算100以内所有能被3整除正整数的和
1871 取模, 取余 %
1872 3%2=1
1873 100%55=45
1874
1875 写一个脚本: (line1555.sh)
1876 计算100以内所有奇数的和以及所有偶数的和, 分别显示之:
1877

```

```

1878 #by zhangshuo 20180116
1879
1880 declare -i JSUM=0
1881 declare -i OSUM=0
1882
1883 for I in `seq 1 100`;do
1884     let MID=$I%2
1885     if [ $MID -eq 0 ];then
1886         let OSUM+=I
1887     else
1888         let JSUM+=I
1889     fi
1890
1891 done
1892
1893 echo "偶数的和为: $OSUM"
1894 echo "素数的和为: $JSUM"
1895
1896 let I=${I+1} 相当于 let I++
1897 SUM=${SUM+1}
1898
1899 let SUM+=I
1900
1901 -=
1902 /-
1903 %=
1904
1905
1906
1907
1908 写一个脚本: (line1558.sh)
1909 分别显示当前系统上所有默认shell为bash的用户和默认shell为/sbin/nologin的用户, 并统计各
    类shell下
    的用户总数。显示结果形如:
1910 BASH,3users, they are:
1911 root, redhat, gentoo
1912 NOLOGIN,2users, they are:
1913 bin, ftp
1914
1915
1916 declare -i SUM=0 //声明变量类型
1917     -i : 整型
1918     -x : 声明为环境变量
1919
1920
1921 测试:
1922 整数测试
1923     -le
1924     -lt
1925     -ge
1926     -gt
1927     -eq
1928 字符串测试
1929     ==
1930     !=
1931     >
1932     <
1933     -n
1934     -z
1935 文件测试
1936     -e
1937     -f
1938     -d
1939     -r
1940     -w
1941     -x
1942 组合测试条件
1943     -a:与关系&&
1944     -o:或关系||
1945     ! :非关系
1946
1947 1<$#<=3
1948 [ $# -gt 1 -a $# -le 3 ] 或 [ $# -gt 1 ] && [ $# -le 3 ] 写法
1949

```

1950
1951 vim编辑器
1952
1953
1954 文本编辑器，字处理器
1955 ASCII
1956
1957 nano , sed,
1958
1959 vi: Visual Interface 可视化接口
1960 vim: vi improved
1961
1962 全屏编辑器，模式化编辑器
1963
1964 vim模式：
1965 编辑模式（命令模式）
1966 输入模式
1967 末行模式
1968
1969 默认处于编辑模式
1970
1971
1972 模式转换：
1973 编辑模式-->输入模式：
1974 i: 在当前光标所在字符的前面，转为输入模式
1975 a: 在当前光标所在字符的后面，转为输入模式
1976 o: 在当前光标所在行的下方，新建一行，并转为输入模式
1977
1978 I: 在当前光标所在行的行首，转换为输入模式
1979 A: 在当前光标所在行的行尾，转换为输入模式
1980 O: 在当前光标所在行的上方，新建一行，并转为输入模式
1981
1982 输入模式-->编辑模式
1983 ESC
1984
1985 编辑模式-->末行模式：
1986 :
1987 10d: 删除第十行
1988 10,20d: 删除第十到第二十行
1989
1990 末行模式-->编辑模式
1991 ESC, ESC
1992
1993 一、打开文件
1994 # vim /path/to/somefile
1995 vim +# file : 光标直接显示到#行
1996 vim + file: 光标处在最后一行上
1997 vim +/PATTERN : 打开文件，定位至第一次被PATTERN匹配到的行的行首
1998
1999 二、关闭文件
2000 1、末行模式下关闭文件
2001 w: 保存
2002 q: 退出
2003 wq: 保存并退出
2004 q! : 不保存并退出
2005 w! : 强行保存（只有管理员才可以）
2006 wq --> : x 相同的功能
2007 2、编辑模式下退出
2008 zz: 保存退出
2009
2010 三、移动光标（编辑模式）
2011 1、逐字符自动
2012 h: 左
2013 l: 右
2014 j: 下
2015 k: 上
2016 #h: 移动#个字符
2017
2018 2、按单词为单位移动
2019 w: 移至下一个单词的词首
2020 e: 跳至当前或下一个单词的词尾
2021 b: 跳至当前或前一个单词的词尾
2022

2023 #w:
2024
2025 3、行内跳转:
2026 0: 绝对行首
2027 ^: 行首的第一个空白字符
2028 \$: 绝对行尾
2029
2030 4、行间跳转
2031 #G: 跳转至第#行
2032 G: 跳至结尾
2033
2034 末行模式下, 直接给出行号即可实现行间跳转
2035
2036 四、翻页操作
2037 Ctrl+f: 向下翻一屏
2038 Ctrl+b: 向上翻一屏
2039
2040 Ctul+d: 向下翻半屏
2041 Ctul+u: 向上翻半屏
2042
2043 五、删除单个字符
2044 x: 删除光标所在处的单个字符
2045 #x: 删除光标所在处及向后的共#个字符
2046
2047 六、删除命令: d
2048 d命令跟行内跳转命令组合使用
2049 #d跳转字符: 5dw: 删除五个单词 d0: 删除至行首 d\$: 删除至行尾
2050
2051 dd: 删除当前光标所在行
2052 #dd: 删除包括当前光标所在内的#行:
2053
2054 末行模式下:
2055 StartADD, EndADDd: 删除多行
2056 .: 表示当前行
2057 \$: 最后一行
2058 +#: 相对表示法
2059 \$-#: 到最后一行之前多少行
2060
2061
2062 七、粘贴命令 p
2063 P: 如果删除或复制为整行内容, 则粘贴至光标所在行的下方, 如果复制或删除的内容为非整行,
2064 则粘贴至光标所在字符的后面
2065 p: 如果删除或复制为整行内容, 则粘贴至光标所在行的上方, 如果复制或删除的内容为非整行,
2066 则粘贴至光标所在字符的下面
2067
2068 八、复制命令 y
2069 用法同d命令
2070
2071 九、修改: 先删除内容, 在转换为输入模式
2072 c: 用法同d
2073
2074 十、替换 : r替换单个字符
2075 R: 替换模式
2076
2077 十一、撤销编辑操作 u
2078 u: 撤销前一次的操作 可以连续使用u 最多可以撤销50次操作
2079
2080 #u: 直接撤销最近#次编辑
2081
2082 撤销最近一次撤销: Ctrl+r
2083
2084 十二、重复前一次编辑操作
2085 .
2086
2087 十三、可视化模式
2088 v: 按字符选取
2089 V: 按矩形选取
2090
2091 十四、查找
2092 /PATTERN
2093 ?PATTERN
2094 n
2095 N

```
2094
2095 十五、查找并替换
2096 在末行模式下使用s命令
2097 ADDR1,ADDR2s@PATTER@STRING@gi
2098 .,$-1s/he/HE/g
2099
2100 %:表示全文
2101
2102 练习：将/etc/yum.repos.d/server.repo文件中的ftp://instructor.example.com/pub替换为http
2103 ://172.16.0.1/yum
2104
2104 %s/ftp:\\\\instructor\\.example\\.com\\/pub/http:\\\\172.16.0.1\\/yum/g
2105 %s@ftp://instructor.example.com/pub@http://172.16.0.1/yum@gi
2106
2107 十六、如何使用vim编辑多个文件
2108 vim FILE1 FILE2 FILE3
2109 : next切换至下一个文件
2110 : prev切换至前一个文件
2111 : last切换至最后一个文件
2112 : first切换至第一个文件
2113
2114 : qa 全部退出
2115
2116 十七、分屏显示一个文件
2117 Ctrl+w, s: 水平拆分窗口
2118 Ctrl+w, v: 垂直拆分窗口
2119 Ctrl+w, ARROW: 窗口间切换
2120
2121 : qa 关闭所有窗口
2122
2123 十八、分窗口显示多个文件
2124 vim -o : 水平分割
2125 vim -O: 垂直分割显示
2126
2127 十九、将当前文件中的部分内容另存为零为一个文件
2128 末行模式下使用w命令
2129 : w
2130 : ADDR1,ADDR2w /path/to/somewhere
2131
2132 二十、将另外一个文件的内容填充在当前文件中
2133 : r /path/to/somefile
2134
2135
2136 二十一、跟shell交互
2137 : ! COMMAND
2138
2139 二十二、高级话题
2140 1、显示或取消显示行号
2141 : set number
2142 : set nu
2143
2144 : set nonu
2145
2146 2、显示或忽略区分字符大小写
2147 : set ignorecase
2148 : set ic
2149
2150 : set noignorecase
2151 : set noic
2152
2153 3、设定自动缩进
2154 : set autoindent
2155 : set ai
2156 : set noai
2157
2158 4、查找到的文本高亮显示或取消
2159 : set hlsearch
2160 : set nohlsearch
2161
2162 5、语法高亮
2163 : syntax on
2164 : syntax off
2165
```

```

2166
2167 二十三、配置文件
2168 /etc/vimrc
2169 ~/.vimrc
2170
2171 Vimtutor
2172
2173
2174 //07-03
2175 文件查找:
2176 locate:
2177     非实时, 查找是根据全系统文件数据库进行的;
2178     locate passwd
2179 # updatedb, 手动生成文件数据库
2180 优点 速度快
2181
2182 find:
2183     实时
2184     精确
2185     支持众多查找标准
2186     遍历指定目录中的所有文件完成查找, 速度慢;
2187
2188 find 查找路径 查找标准 查找到以后的处理动作
2189 查找路径: 默认为当前目录
2190 查找标准: 默认为制定路径下的所有文件
2191 处理动作: 默认为显示
2192
2193 匹配标准:
2194     -name 'FILENAME' 对文件名做精确匹配
2195     # find /etc -name 'passwd'
2196         文件名通配
2197             *: 任意长度的任意字符
2198             ?: 任意单个字符
2199             []: 匹配范围内的字符
2200     -iname 'FILENAME' 文件名匹配时不区分大小写
2201     -regex PATTERN 基于正则表达式进行文件名匹配
2202
2203     -user USERNAME :根据文件的属主查找文件
2204     # find /etc -user zhangshuo
2205     -group GROUPNAME:根据组查找
2206
2207     -uid UID:根据UID查找
2208     -gid GID:根据GID查找
2209
2210     -nouser: 查找没有属主的文件
2211     -nogroup: 查找没有属组的文件
2212
2213     -type 文件类型查找
2214         f:普通文件
2215         d:目录文件
2216         c:字符设备
2217         b:块设备
2218         l:符号链接
2219         p:管道设备
2220         s:套接字设备
2221     # find /etc -type d
2222     # find /tmp -type s
2223
2224     -size 文件大小查找
2225         [+|-]#k:
2226         [+|-]#M:
2227         [+|-]#G:
2228     find /etc -size 10k -ls
2229
2230 组合条件: 默认为&&条件
2231     -a:&&
2232     -o:||
2233     -not:!
2234     # find /tmp -nouser -type d
2235     # find /tmp -not -type d //表示查找非目录的文件
2236 查找/tmp目录下, 不是目录, 并且还不能是套接字类型的文件
2237     #find /tmp -not -type d -a -not -type s
2238     #find /tmp -not \( -type d -o -type s\)

```

```

2239 查找/tmp/test目录下，属主不是user1，也不是user2的文件
2240      find /tmp/test -not \( -user user1 -o -user user2 \)
2241
2242      -mtime:修改时间(天)
2243      -ctime:改变时间
2244      -atime:访问时间
2245          [+|-]#
2246          # find /tmp -mtime +5 -a -type -f -a -nouser
2247
2248      -mmin:修改时间(分钟)
2249      -cmin:改变时间
2250      -amin:访问时间
2251          [+|-]#
2252 查找/tmp目录下，一周之前访问的文件
2253 # find /etc -atime +7
2254
2255      -perm MODE:精确匹配给定权限
2256          /MODE: 只要包含一个相同的权限即可
2257      -MODE: 文件权限要完全包含给定权限时才匹配
2258      find ./ -perm -001
2259
2260 动作:
2261      -print:显示
2262      -ls:类似ls -l 的形式显示每一个文件的详细信息
2263      -ok COMMAND {} \; 每次操作都需要用户确认
2264      -exec COMMAND {} \;
2265
2266      find ./ -perm -006 -exec chmod o-x {} \;
2267      find ./ -type d -exec chmod a+x {} \;
2268
2269 查找组有读权限的文件并改名为~.new
2270 # find ./ -perm -020 -exec mv {} {}.new \;
2271
2272 找到目录下所有文件名为.sh结尾的并将其他用户的执行权限去掉
2273 # find ./ -name '*.sh' -a -perm -111 -exec chmod o-x {} \;
2274
2275 练习:
2276 1、查找/var目录下属主为root并且属组为mail的所有文件
2277 # find /var -user root -a -group mail
2278 2、查找/usr目录下不属于root, bin, 或student的文件
2279 # find /usr -not \( -user root -a -user bin -a -user zhangshuo \)
2280 3、查找/etc目录下最近一周内内容修改过且不属于root及student用户的文件
2281 # find /etc -atime -7 -a -not -user root -a -not -user zhangshuo -ls
2282 4、查找当前系统上没有属主或属组且最近一天内曾被访问过的文件，并将其属主属组都写改为root
2283 # find / -nouser -a -nogroup -a -atime -7 -exec chown root:root {} \;
2284 5、查找/etc目录下大于1M的文件，并将其文件名写入/tmp/etc.largefiles文件中
2285 # find /etc -size +1M -exec ls {} > /tmp/etc.largefiles \;
2286 6、查找/etc目录下所有用户都没有写权限的文件，显示出其详细信息
2287 # find /etc -not -perm /222 -ls
2288
2289
2290 特殊权限
2291 passwd;s
2292
2293 SUID:运行某程序时，相应进程的属主是程序文件自身的属主，而不是启动者；
2294      chmod u+s FILE
2295      chmod u-s FILE
2296      如果FILE本身原来就有执行权限，则SUID显示为s, 否则显示为S；
2297 SGID:运行某程序时，相应进程的属组是程序文件自身的属组，而不是启动者所属的基本组；
2298      chmod g+s FILE
2299      chmod g-s FILE
2300 sticky:在一个公共目录，每个都可以创建文件，删除自己的文件，但不能删除别人的文件。
2301      chmod o+t DIR
2302      chmod o-t DIR
2303
2304 000
2305 001
2306 ...
2307 111
2308
2309 chmod 2755 /backup/test 在以前的基础上右多了一个位
2310

```



```

2311 umask 0022
2312
2313
2314 文件系统访问列表
2315 tom
2316     tom,tom基本组
2317 JERRY:other: rw-
2318
2319 普通用户没有权限执行chmod
2320
2321  ACL:Filesystem Access Control List
2322 利用文件扩展保存额外的访问控制权限
2323
2324 加入ACL之后文件的访问顺序:
2325     Owner -> facl.user -> Group -> facl,group -> Other
2326
2327 setfacl
2328     -m: 设定
2329         u:UID:perm
2330         d:u:UID:perm 给目录创建默认的访问列表
2331         g:GID:perm
2332         d:g:UID:perm
2333     -x: 取消
2334         u:UID
2335         g:GID
2336
2337 getfacl
2338
2339 setfacl -m u:hadoop:rw /path/to/some/file
2340 setfacl -m g:mygroup:rw /path/to/some/file
2341
2342
2343
2344 关于几个命令
2345
2346 whoami 我是谁
2347
2348 who //显示当前系统登陆的用户有哪些
2349     -r:显示用户级别
2350
2351 写一个脚本:
2352 每隔5秒钟,就来查看hadoop是否已经登陆,如果登陆,显示已经登陆,并退出;
2353 sleep 睡眠
2354
2355 w: 显示当前登录的用户并显示其动作
2356
2357 last: 显示登陆日志其实为 显示/var/log/wtmp文件 显示用户登陆历史及系统重启历史
2358     -n #:显示最近#次的相关信息
2359 lastb, 显示/var/log/btmp,显示用户错误的登陆尝试
2360     -n #:
2361
2362 lastlog:显示每一个用户最近一次成功登陆信息
2363     -u USERNAME:显示特定用户最近的登陆信息
2364
2365 basename /etc/abc/me
2366     $0:执行脚本时的脚本路径及名称
2367
2368 mail 邮件服务
2369
2370 hostname:显示或更改主机名
2371 如果当前主机的主机名不是www.magedu.com,就将其改为www.magedu.com
2372
2373 [ $(hostname) != 'www.magedu.com' ] && hostname www.magedu.com
2374
2375 如果当前主机的主机名是localhost,就将其改为www.magedu.com
2376 [ $(hostname) == 'www.magedu.com' ] && hostname www.magedu.com
2377
2378 如果当前主机的主机名为空,或者为(none),或者为localhost,就将其改为www.magedu.com
2379 [ -z $(hostname) ] || [ $(hostname) == '(none)' -o $(hostname) == 'localhost' ] &&
hostname www.magedu.com
2380
2381 生成随机数
2382 RANDOM:0-32768

```

```

2383
2384 随机数生成器：熵池
2385 /dev/random
2386 /dev/urandom
2387
2388 写一个脚本：(line2031.sh) (random.sh)
2389 利用RANDOM生成10个随机数，并找出其最大值（排序）
2390
2391 #!/bin/bash
2392 #by zhangshuo 20180118
2393
2394 MAXNUM=0
2395 MINNUM=32768
2396
2397 for I in {1..10};do
2398     RANDOMNUM=$RANDOMNUM:`echo $RANDOM`
2399 done
2400
2401 echo $RANDOMNUM
2402
2403 for I in `echo $RANDOMNUM | sed 's@:@ @g'`;do
2404     echo $I
2405     if [ $I -gt $MAXNUM ];then
2406         MAXNUM=$I
2407     elif [ $I -lt $MINNUM ];then
2408         MINNUM=$I
2409     fi
2410 done
2411
2412 echo "MAXNUM is $MAXNUM"
2413 echo "MINNUM is $MINNUM"
2414
2415
2416
2417 终端类型：
2418     console：控制台
2419     pty：物理终端（VGA）
2420     tty：虚拟终端（VGA）
2421     ttys：串行终端
2422     pts/#：伪终端
2423
2424
2425 面向过程
2426     控制结构
2427         顺序结构
2428         选择结构
2429         循环结构
2430
2431 选择结构：
2432 if 单分支、双分支、多分支
2433 if CONDITION ;then
2434
2435
2436 case语句：选择结构 (case.sh)
2437
2438 case SWITCH in
2439 value1)
2440     statement
2441     ...
2442     ;;
2443 value2)
2444     statement
2445     ...
2446     ;;
2447 *)
2448     statement
2449     ...
2450     ;;
2451 esac
2452
2453 #!/bin/bash
2454 #by zhangshuo 20180118
2455

```

```

2456 if [ $# -eq 0 ];then
2457     echo "usage!!!!!"
2458     exit 2
2459 fi
2460
2461 case $1 in
2462 [a-z])
2463     echo "$1 is a char"
2464     ;;
2465 [0-9])
2466     echo "$1 is a num"
2467     ;;
2468 *)
2469     echo "unknow"
2470     ;;
2471 esac
2472
2473 写一个脚本(line2074.sh)
2474 只接受参数start ,stop,restart,status其中之一并显示之
2475
2476 #!/bin/bash
2477 #by zhangshuo 20180118
2478
2479 if [ $# -eq 0 ];then
2480     echo "usage!!!!!"
2481     exit 2
2482 fi
2483
2484 case $1 in
2485 'start')
2486     echo "start"
2487     ;;
2488 'stop')
2489     echo "stop"
2490     ;;
2491 'restart')
2492     echo "restart"
2493     ;;
2494 'status')
2495     echo "status"
2496     ;;
2497 *)
2498     echo "usage:`basename $0` {start|stop|restart|status}"
2499     ;;
2500 esac
2501
2502 写一个脚本，可以接受选项及参数，而后能够获取每一个选项，及选项的参数，并能根据选项及参
2503 数做出
2504 特定的操作，比如：
2505 adminusers.sh --add tom,jerry --del tom,blair -v | --verbose -h|--help
2506
2507 #!/bin/bash
2508 #by zhangshuo 20180118
2509
2510 DEBUG=0
2511 ADDUSER=0
2512 DELUSER=0
2513
2514 for I in `seq $#`;do
2515 if [ $# -gt 0 ];then
2516     case $1 in
2517         '-v'|'--verbose')
2518         DEBUG=1
2519         shift 1
2520         ;;
2521         '--add')
2522         ADDUSER=1
2523         ADDUSERS=$2
2524         shift 2
2525         ;;
2526         '--del')
2527         DELUSER=1

```

```

2528         DELUSERS=$2
2529         shift 2
2530         ;;
2531     *)
2532         echo "gg"
2533         ;;
2534     esac
2535 fi
2536 done
2537
2538 echo $DEBUG $ADDUSER $DELUSER $ADDUSERS $DELUSERS
2539
2540 if [ $ADDUSER -eq 1 ];then
2541     for USER in `echo $ADDUSERS | sed 's@, @g'`;do
2542         if id $USER &> /dev/null;then
2543             [ $DEBUG -eq 1 ] && echo "user $USER exist"
2544         else
2545             [ $DEBUG -eq 1 ] && echo "add $USER finished"
2546         fi
2547     done
2548 elif [ $DELUSER -eq 1 ];then
2549     for USER in `echo $DELUSERS | sed 's@, @g'`;do
2550         if id $USER &> /dev/null;then
2551             [ $DEBUG -eq 1 ] && echo "del $USER finished"
2552         else
2553             [ $DEBUG -eq 1 ] && echo "no such user"
2554         fi
2555     done
2556 fi
2557
2558 练习：写一个脚本showlogged.sh，其用法格式为：
2559 showlogged.sh -v -c -h| --help
2560 其中，-h选项只能单独使用用于显示帮助信息；-c选项时，显示当前系统上登陆的所有用户数：如
2561 果同时使用了-v选项，则既显示同时登陆的用户数，又显示登陆的用户的相关信息：如
2562
2563 Logged users: 4.
2564 they are:
2565 Root tty2 feb 18 02:41
2566
2567 #!/bin/bash
2568 #by zhangshuo 20180118
2569
2570 declare -i SHOWNUM=0
2571 declare -i SHOWUSERS=0
2572
2573 for I in `seq $#`;do
2574     if [ $# -gt 0 ];then
2575         case $1 in
2576             '-h'|'-help')
2577                 echo "usage :`basename $0` -h|--help -c|--count"
2578                 exit 0
2579                 ;;
2580             '-c'|'--count')
2581                 let SHOWNUM=1
2582                 shift
2583                 ;;
2584             '-v'|'--verbose')
2585                 let SHOWUSERS=1
2586                 shift
2587                 ;;
2588             *)
2589                 echo "usage :`basename $0` -h|--help -c|--count"
2590                 exit 2
2591                 ;;
2592         esac
2593     fi
2594 done
2595
2596 if [ $SHOWNUM -eq 1 ];then
2597     echo "Logged users:`who | wc -l`"
2598 fi
2599
2600 if [ $SHOWUSERS -eq 1 ];then
2601     echo "They are:"

```

```
2600         who
2601     fi
2602
2603     分区partition
2604     文件系统
2605     MBR 主引导记录位于零磁道零扇区（512byte）
2606     master boot record
2607     main boot record
2608         446byte: bootloader, 程序,
2609         64bytes:
2610             16bytes: 标识一个分区
2611     最后两个字节:
2612         magic number : 模数
2613         标记mbr是否有效
2614
2615     文件系统管理
2616     重新创建文件系统会损坏原有文件
2617
2618     硬链接, 符号链接
2619     符号链接权限都为777
2620     ln [-s -v] SRC DEST
2621
2622     硬链接:
2623         1、只能对文件创建, 不能应用于目录
2624         2、不能跨文件系统
2625         3、创建硬链接会增加文件被链接的次数。
2626
2627     符号链接:
2628         1、可应用与目录
2629         2、可以跨文件系统
2630         3、不会增加被硬链接的次数
2631         4、其大小为指定的路径所包含的字符个数
2632
2633     du -s 显示文件夹的大小
2634
2635     df -h 显示分区的使用情况
2636         -i 显示INODE
2637         -p 不换行显示
2638
2639     设备文件
2640         块设备: 按块为单位, 随机访问的设备
2641         字符设备: 按字符为单位, 线性设备
2642
2643     ls -l /dev
2644         字符设备的两个号码
2645         第一个为: 主设备号
2646         第二个为: 次设备号
2647
2648     mknod : 创建块设备或字符设备
2649     mknod -m MODE (权限)
2650
2651     使用tty命令可以查看当前使用的客户端类型
2652     echo "hello" >> /dev/pts/6
2653
2654     硬盘设备的设备文件名:
2655     IDE ,ATA : hd
2656     STAT:sd
2657     SCSI:sd
2658     USB:sd
2659         a,b,c,....来区别同一种类型下的不同设备
2660
2661     IDE:
2662         第一个IDE口: 主、从
2663             /dev/hda /dev/hdb
2664         第二个IDE口: 主、从
2665             /dev/hdc /dev/hdd
2666
2667     fdisk -l 查看当前系统分区情况
2668
2669     Linux支持的文件系统
2670     VFS: 虚拟文件系统
2671
2672     cat /proc/partitions :查看当前内核已经识别的分区
```

```

2673
2674 partprobe: 通知内核重读分区表
2675
2676 管理磁盘分区
2677 fdisk /dev/sdb
2678     p:显示当前硬盘的分区情况
2679     w:保存退出
2680     n:创建新分区
2681         e:扩展分区
2682         p:主分区
2683     d:删除一个分区
2684     q:不保存退出
2685     t:修改分区类型
2686     l:显示文件系统类型
2687
2688
2689 mkfs:make file system
2690     -t:FSTYPE:
2691         ext2
2692         ext3
2693
2694 mkfs -t ext2 = mkfs.ext2
2695 mkfs -t ext3 = mkfs.ext3
2696
2697 专门管理ext系列文件系统
2698
2699 mke2fs :创建ext2格式文件系统
2700     -j :创建日志文件系统即ext3
2701     -b BLOCK_SIZE:指定块大小，默认为4096 可用取值为1024 2048 4096
2702     -L LABEL: 指定分区卷标
2703     -m #:制定预留给超级用户的块数百分比
2704     -i
2705     #:用于指定为多少个字节空间创建一个inode，默认为8192，这里给出的数值应该为块大小的2
2706     ^n倍
2707     -N: 指定inode个数
2708     -F: 强制创建文件系统
2709     -E: 用户指定额为文件系统属性
2710
2711 cat /proc/filesystems 查看当前内核支持那些文件系统
2712
2713 对于VFAT文件系统的优盘来说
2714 #vi /etc/mtools.conf
2715 在最后一行增加 mtools_skip_check=1。
2716 修改卷标命令为: mlabel -i /dev/sdb1 ::name
2717
2718 blkid: 查询或查看磁盘设备的相关属性
2719     UUID
2720     TYPE
2721     LABEL
2722
2723 e2label :用于查看或定义卷标
2724     e2label:设备文件 卷标: 设定卷标
2725
2726 tune2fs:调整文件系统的相关属性
2727     -j:不损害原有数据，将ext2升级为ext3
2728     -L LABEL: 设定或修改卷标
2729     -m #:调整预留百分比
2730     -r #:指定预留块数
2731     -o:设定默认挂在格式
2732         acl
2733     -c #:指定挂在次数达到#次之后进行自检，0或-1表示关闭此功能
2734     -i #:每挂载使用多少天后进行自检: 0或-1表示关闭此功能
2735     -l :显示超级快中的信息
2736
2737 dumpe2fs:显示文件属性信息
2738     -h:只显示超级块信息
2739
2740
2741 fsck: 检查并修复Linux文件系统
2742     -t FSTYPE:文件系统类型
2743     -a :自动修复

```

2744 e2fsck:专用于修复ext2/ext3文件系统
2745 -f:强制检查
2746 -p:自动修复
2747 -a:自动修复
2748
2749
2750 挂载: 将新的文件系统关联至当前文件系统
2751 卸载: 将某文件系统与当前根文件系统的关联关系予以移除
2752
2753 mount :挂载
2754 mount 设备 挂载点
2755 设备文件: /dev/sdb1
2756 卷标: LABEL=""
2757 UUID: UUID=""
2758 挂载点: 目录
2759 要求:
2760 1、此目录没有被其他进程使用
2761 2、目录要事先存在
2762 3、目录中的原有的文件将会暂时隐藏
2763
2764 挂载完成后要通过挂载点访问文件
2765
2766 umount: 卸载某文件系统
2767 umount 设备
2768 umount 挂载点
2769
2770 卸载注意事项:
2771 挂载的设备没有进程使用
2772
2773 mount: 显示当前系统已经挂载的设备及挂载点
2774 mount [options] [-o options] DEVICE MOUNT POINT
2775 -a:表示挂载/etc/fstab 文件中定义的所有文件系统
2776
2777 -n:默认情况下, mount命令每挂载一个设备, 都会把挂载的设备信息保存至/etc/mtab文件;
2778 使用
2779 -n选项意味着不写入
2780 -t
2781 FSTYPE:指定正在挂载设备上的文件系统的类型, 不使用次选项时, mount会调用blkid命令获取
2782 对应文件系统类型
2783 -r:只读挂载 挂载光盘时常用此选项
2784 -w:读写挂载
2785
2786 -o:指定额外的挂载选项, 也即指定文件系统启用的属性
2787 remount: 重新挂载当前文件系统
2788 ro:挂在为只读
2789 rw: 读写挂载
2790
2791 mount -o remount, ro
2792
2793 swap: 交换空间
2794
2795 增加交换分区:
2796 1、新建一个分区
2797 2、更改分区类型为交换空间 82
2798 3、通知内核重新读取 partprobe /dev/sdb
2799 4、创建交换分区类型文件系统
2800
2801 创建交换分区:
2802 mkswap /dev/sdb2
2803 -L LABEL: 卷标
2804
2805 swapon /dev/sdb2 : 启用交换空间
2806 -a:启用所有的定义在/etc/fstab文件中的交换设备
2807 swapoff /dev/sdb2 :关闭交换空间
2808
2809 free:查看物理内存与交换分区的使用情况
2810 -m: 以m的形式显示
2811
2812 回环设备
2813 loopback, 使用软件来模拟实现硬件
2814
2815 创建一个镜像文件, 1G

```

2813
2814 dd命令
2815     if=数据来源
2816     of=数据存储目标
2817
2818     bs=1 blocksize 块大小单位为字节
2819     count=2
2820 dd if=/dev/sda of=/mnt/usb/mbr.backup bs=512 count=1 将MBR保存到优盘文件
2821 dd if=/mnt/usb/mbr.backup of=/dev/sda bs=512 conut=1
2822
2823 cat /dev/cdrom > /root/rhel5.iso 直接制作镜像文件
2824
2825 dd if=/dev/zero of=/var/swapfile bs=1M count=1024
2826 dd if=/dev/zero of=/var/swapfile seek=1023 bs=1M count=1024
2827
2828 mount命令，可以挂载iso镜像
2829 mount DEVICE MONT POINT
2830     -o loop:挂载本地回环设备
2831 mount -o loop ubuntu-16.04-desktop-amd64.iso /media/zhangshuo/iso/ 挂载iso镜像文件
2832 umount /media/zhangshuo/iso 卸载iso镜像文件
2833
2834 wget ftp://172.16.0.1/pub/isos/rhci-5-8-1.iso
2835
2836 文件系统的配置文件 /etc/fstab
2837     os在初始化时，会自动挂载此文件中定义的每一个文件系统
2838
2839 使用空白隔开的六个字段
2840
2841 要挂在的设备    挂载点    文件系统类型    挂载选项    转储频率（每多少天做一次备份）
2842 文件系统检测次序（只有根可以为1，0表示不检查）
2843 /dev/sdb1    /rum/media/zhangshuo/usb1    vfat    defaults    0    0
2844
2845 mount -a :挂载/etc/fstab文件中定义的所有文件系统
2846
2847 fuser :验证进程正在使用的文件或套接字文件
2848     -v:查看某文件上正在运行的进程
2849     -k: kill
2850     -m:name
2851
2852     fuser -km MOUNT_POINT:终止正在访问此挂载点的所有进程
2853
2854 练习：
2855 1、创建一个5G的分区，文件系统为ext3，卷标为MYDATE，块大小为1024，预留管理空间为磁盘分
2856 区的3%，要求开机后可以自动挂载至/data 目录，并且自动挂载的设备要使用卷标进行引用
2857 2、创建一个本地回环文件/var/swaptmp/seapfile来用于swap，要求大小为512MB，卷标为SWAP-F
2858 ILE，并且
2859     开机自动启动此交换设备
2860     mkdir /var/swaptmp
2861     dd if=/dev/zero of=/var/swaptmp/swapfile bs=1M count=512
2862     mkswap LABEL=SWAP-FILE /var/swaptmp/swapfile
2863
2864     /etc/fstab
2865     /var/swaptmp/swapfile swap swap defaults 0 0
2866 3、上述第一问，如何让其自动挂载的同时启动ACL功能
2867     /ect/fstab
2868     LABEL='MYDATE' /data ext3 defaults,acl 0 0
2869
2870 压缩、解压缩命令
2871 压缩格式：gz, bz2,xz,zip,z
2872
2873 压缩算法：算法不同，压缩比也会不同；
2874
2875 comprees:FILENAME.z
2876 uncompress
2877
2878 gzip:.gz
2879     gzip PATH/TO/SOMEFILE:压缩完成后会删除源文件
2880     -d:解压缩
2881     -#: 1-9: 指定压缩比，默认为6
2882 gunzip:

```



```
2883 gunzip /PATH/TO/SOMEFILE.gz :压缩完成后会删除源文件
2884
2885 zcat:不解压的情况下, 查看压缩后的文件内容
2886 zcat test.gz
2887
2888 bzip2:.bz2
2889 比gzip有着更大压缩比的压缩工具, 使用格式近似gzip
2890 bzip2 /PATH/TO/SOMEFILE
2891 -d: 解压缩
2892 -#: 1-9 , 默认为6的压缩比
2893 -k:压缩时保留源文件
2894
2895 bunzip2 /PATH/TO/SOMEFILE:解压缩文件
2896 bzipcat /PATH/TO/SOMEFILE 不用解压缩直接查看文件内容
2897 xz:.xz
2898 xz /PATH/TO/SOMEFILE
2899 -d:解压缩文件
2900 -#:1-9,默认为6
2901 -k:压缩时保留源文件
2902
2903 unxz /PATH/TO/SOMEFILE:解压缩文件
2904 xzcat:
2905 xzdec:
2906
2907 zip:即归档又压缩的工具
2908 zip FILENAME.zip FILE1 FILE2 ...: 压缩后不删除源文件
2909 unzip FILENAME.zip 解压缩文件
2910
2911
2912 archive:归档, 归档本身并不意味者压缩
2913
2914 tar:归档工具
2915 -c:创建归档文件
2916 -f FILE.tar : 操作的归档文件
2917 -x:展开归档
2918 --xattrs:归档时, 保留文件的扩展属性信息
2919 -t:不展开归档, 直接查看归档了哪些文件
2920
2921 -zcf:归档并调用gzip压缩
2922 -zxf:调用gzip解压缩并展开归档
2923
2924 -jcf: bzip
2925 -jxf:
2926
2927 -Jcf:xz
2928 -Jxf:
2929
2930 -jtf:不解压缩直接查看压缩包内容
2931
2932 RAR压缩和解压文件
2933 1、解压文件
2934 rar x FILENAME.rar
2935 2、压缩文件
2936 rar a FILENAME.rar /path
2937
2938 cpio:归档工具
2939
2940 #!/bin/bash
2941 #by zhangshuo 20180121
2942
2943 echo -n "input 2 num:"
2944 read NUMBER1 NUMBER2
2945 echo "$NUMBER1 plus $NUMBER2 equal:${$NUMBER1 + $NUMBER2}"
2946
2947 read: 读取键盘内容并赋值给变量
2948 -p:直接提示内容
2949 -t:超时事件限制
2950 #!/bin/bash
2951 #by zhangshuo 20180121
2952
2953 read -t 3 -p "input 2 num:" NUMBER1 NUMBER2
2954
2955 [ -z $NUMBER1 ] && NUMBER1=1 && echo -n "1 + "
```

```

2956
2957 [ -z $NUMBER2 ] && NUMBER2=1 && echo "1"
2958
2959 echo "$NUMBER1 plus $NUMBER2 equal:${NUMBER1 + $NUMBER2}"
2960
2961
2962 练习：写一个脚本（line2397.sh）
2963 从键盘让用户输入几个文件，脚本能够将此几个文件归档压缩成一个文件
2964
2965 #!/bin/bash
2966 #by zhangshuo 20180121
2967
2968 read -p "three files: " FILE1 FILE2 FILE3
2969
2970 read -p "name: " FILENAME
2971
2972 read -p "type: " TYPE
2973
2974 case $TYPE in
2975 gzip)
2976     tar -zvcf ${FILENAME}.tar.gz $FILE1 $FILE2 $FILE3
2977     ;;
2978 bzip2)
2979     tar -jvcf ${FILENAME}.tar.bz2 $FILE1 $FILE2 $FILE3
2980     ;;
2981 xz)
2982     tar -Jvcf ${FILENAME}.tar.xz $FILE1 $FILE2 $FILE3
2983     ;;
2984 *)
2985     echo "unkonw"
2986     exit 2
2987     ;;
2988 esac
2989
2990
2991 while循环，适用于循环次数未知的场景,要有退出条件
2992 语法：
2993     while CONDITION ;do
2994         statement
2995         ...
2996     done
2997
2998 就算100以内所有整数的和（line2411.sh）
2999
3000 #!/bin/bash
3001 #by zhangshuo 20180121
3002
3003 declare -i I=1
3004 declare -i SUM=0
3005
3006 while [ $I -le 100 ];do
3007     let SUM+=I
3008     let I++
3009 done
3010
3011 echo $SUM $I
3012
3013 给定一个字符串并转换为大写，当输入quit时退出while quit （line2413.sh）
3014
3015 #!/bin/bash
3016 #by zhangshuo 20180121
3017
3018
3019 read -p "input some string: " STRING
3020
3021 while [ $STRING != 'quit' ];do
3022     echo $STRING | tr 'a-z' 'A-Z'
3023     read -p "input some string: " STRING
3024
3025 done
3026
3027 每隔五分钟查看一次某用户是否登陆系统（line2415.sh）
3028

```

```

3029 #!/bin/bash
3030 #by zhangshuo 20180121
3031
3032 RES=1
3033
3034 while [ $RES -eq 1 ];do
3035     if who | grep "jialei" &> /dev/null;then
3036         RES=0
3037         echo " jialei in"
3038     else
3039         sleep 5
3040     fi
3041 done
3042
3043 写一个脚本：（line2417.sh）
3044 1) 显示一个菜单给用户
3045 d|D) show disk usages
3046 m|M) show memory usages
3047 s|S) show swap usages
3048 *)quit
3049 2) 当用户给定选项后显示相应的内容；
3050
3051 扩展：
3052     当用户选择完成，显示相应消息后，不退出；而让用户再一次选择，再次显示相应内容
3053     除非用户使用quit退出
3054
3055 #!/bin/bash
3056 #by zhangshuo 20180121
3057
3058 #echo "d|D) show disk usages"
3059 #echo "m|M) show memory usages"
3060 #echo "s|S) show swap usages"
3061 #echo "*) quit"
3062
3063 cat << EOF
3064 d|D) show disk usages
3065 m|M) show memory usages
3066 s|S) show swap usages
3067 *) quit
3068 EOF
3069
3070 read -p "chose one: " CHOSICE
3071
3072 while [ $CHOSICE != 'quit' ];do
3073 case $CHOSICE in
3074 d|D)
3075     df -h
3076     ;;
3077 m|M)
3078     free
3079     ;;
3080 s|S)
3081     free
3082     ;;
3083 *)
3084     echo "unknow"
3085 esac
3086 read -p "chose one: " CHOSICE
3087 done
3088
3089
3090 echo -e "\033[31mhello\033[0m,world" 用于控制输出的颜色设置
3091
3092 IDE:133Mbps
3093 STAT:300Mbps      600Mbps 6Gbps
3094 usb3.0: 480Mbps
3095 SCSI:small computer system interface
3096     10000,15000
3097
3098 RAID:
3099 级别，仅代表磁盘组织方式不同，没有上下之分
3100 0: 条带
3101     性能提升：读，写

```

```

3102     冗余能力：无
3103 1: 镜像
3104     性能提升：写能力下降，读性能提升
3105     冗余能力：有
3106 2:
3107 3:
3108 4:
3109 5:
3110     性能表现：读写提升
3111     冗余能力：有
3112     空间利用率：  $(n-1)/n$ 
3113 10:
3114     性能表现：读写提升
3115     冗余能力：有
3116     空间利用率：0.5
3117 01:
3118     性能表现：读写提升
3119     冗余能力：有
3120     空间利用率：0.5
3121 50:
3122     性能表现：读写提升
3123     冗余能力：有
3124     空间利用率：  $(n-2)/n$ 
3125
3126 JBOD:
3127     性能表现：无
3128     冗余能力：无
3129     空间利用率：1
3130 Hadoop
3131     HDFS
3132
3133 硬件RAID
3134 软件RAID
3135
3136 逻辑RAID
3137 /dev/md0 表示raid逻辑设备
3138 /dev/md1
3139
3140 当一个系统完成磁盘RAID后如果系统崩溃则数据无法找回，所以制作RAID时应将要制作的设备标记
    为内核可识别的设备，即在磁盘分区时就行该指定磁盘类型。RAID的类型为FD。
3141
3142 md: multi disks 软件raid
3143 mdadm: 将任何块设备做成RAID
3144 模式化的命令：
3145     创建模式
3146         -C
3147     专用选项
3148         -l: 级别
3149         -n: 设备个数
3150         -a (yes no) : 自动为其创建设备文件
3151         -c: chunk大小即数据块大小 默认为64k
3152         -x: 指定空闲盘个数
3153 管理模式
3154     --add, --remove
3155     mdadm /dev/md# --fail /dev/sdb7 模拟将/dev/sdb7进行损坏
3156     mdadm /dev/md1 -f /dev/sdb7 模拟硬盘损毁
3157     mdadm /dev/md1 -r /dev/sdb7 卸载坏的硬盘
3158     mdadm /dev/md1 -a /dev/sdb7 增加一块好的硬盘到raid
3159 监控模式
3160     -F
3161 增长模式
3162     -G
3163 装配模式
3164     -A
3165
3166 创建RAID0
3167     2G:
3168         4: 512MB
3169         2: 1G
3170
3171 创建raid0    mdadm -C /dev/md0 -a yes -l 0 -n 2 /dev/sdb{5,6}
3172 查看当前raid    cat /proc/mdstat
3173

```

```

3174 创建raid1
3175     2G:
3176         2: 2G
3177
3178 mdadm -D /dev/md1 :查看磁盘阵列的详细信息
3179
3180 停用阵列:
3181     mdadm -S /dev/md#
3182         --stop
3183 启动阵列
3184     mdadm -A /dev/md#
3185
3186 完整删除RAID信息
3187 1、查看RAID磁盘阵列信息，确认一下要关闭哪个。如：关闭md0这个阵列
3188     cat /proc/mdstat
3189 2、卸载md0这个阵列的挂载点
3190     umount /mnt/RAID5
3191 3、停止md0这个阵列，并释放与该阵列相关的所有资
3192     mdadm -S /dev/md0
3193 4、清除成员磁盘当中阵列的超级块信息，这一步很重要！
3194     mdadm --zero-superblock /dev/sdb[1-3,5]
3195 5、删除或注释/etc/fstab上的挂载信息
3196 6、删除或注释/etc/mdadm.conf对应的RAID信息
3197
3198
3199 watch: 周期性的执行命令，并以全屏方式显示结果
3200     -n #: 指定周期长度，单位为秒，默认为2
3201     watch -n # 'COMMAND'
3202
3203 mdadm -D --scan > /etc/mdadm.conf 将raid信息保存至文件以后便可开机装配
3204
3205 mke2fs -E stride 16 :格式化时指定条带大小这样可以提高raid性能。chunk大小默认为64k
    磁盘块大小为4k所以这里数字为16.注意用优盘做raid时它的chunk大小为512k。
3206
3207 RAID5:
3208     2G: 3, 1G
3209     madam -C /dev/md2 -a yes -l 5 -n 3 -x 1 /dev/sdb{1,2,3}
3210
3211 lsmod:显示已经加载到内核中的模块状态信息
3212
3213 mdadm: 用户空间工具只是管理工具，对于raid其实还是内核中raid模块进行的管理。
3214
3215 MD, DM
3216     /dev/md#
3217     meta device
3218
3219 DM:Device Mapper
3220     逻辑设备
3221         RAID, LVM2
3222
3223 DM:LVM2
3224     线性 用md也可实现，类似jbod
3225     镜像 md也可以实现既raid
3226     快照 数据备份
3227     多路径
3228
3229
3230 逻辑卷的快照必须和逻辑卷在同一个卷组中切必须留有足够的空间给快照卷
3231
3232 将物理卷加入到卷组中时会给物理卷分块类似chunk此时的块叫做pe: physical extend
    物理盘区 既逻辑存储单位
3233
3234 将pe放到逻辑卷中，即逻辑卷的存储空间是由若干个pe组成的。此时的pe叫做le: logical
    extend 逻辑盘区
3235
3236 physical volume pv:物理卷
3237     pvcreate, pvremove, pvscan, pvdisplay, pvmove
3238     pvcreate /dev/sdb{11,12}
3239     pvs :查看当前系统的pv
3240     pvdisplay
3241 volume group vg:卷组类似扩展分区
3242     vgcreate, vgremove, vgextend, vgreduce, vgs, vgdisplay, vgscan
3243     vgs

```

```

3244     vgcreate:创建卷组
3245         -s: 可以指定pe大小默认为4m 该选项可以带单位用时man一下
3246     vgcreate myvg /dev/sdb{11,12}
3247
3248     vgrename vg_name : 移除该vg
3249
3250     vgcreat VG_NAME /PATH/TO/PV
3251         -s #:PE大小, 默认为4MB
3252
3253     vgcreate -s 8M myvg /dev/sdb{11,12}
3254
3255
3256 缩小一个vg的过程
3257 1、pvmove 移除一个pv该过程会自动复制数据
3258     pvmove /dev/sdb12
3259 2、vgreduce myvg /dev/sdb12
3260 3、pvremove 真正移除pv
3261
3262
3263 增加一个vg的过程
3264 1、pvcreate /dev/sdb12
3265 2、vgextend myvg /dev/sdb12
3266
3267 logical volume: 逻辑卷
3268     lvcreate,lvremove,lvextend,lvreduce,lvresize,lvs,lvdisplay
3269     lvs
3270
3271     lvcreate -n LV_NAME -L #G VG_NAME
3272
3273 lv创建好以后其设备目录为/dev/vgname/lvname, 但是其实真正的lv设备是在/dev/mapper下面
3274
3275 删除一个lv
3276 1、卸载
3277     umount /run/media/zhangshuo/vg0/
3278 2、删除
3279     lvremove /dev/myvg/testlv
3280
3281 练习: 创建一个由两个物理卷组成的大小为20G的卷组myvg, 要求其PE大小为16M, 而后在此卷组中
3282 创建一个
3283 大小为5G的逻辑卷lv1, 此逻辑卷要能在开机后自动挂载至/users目录, 且支持ACL功能
3284
3285 1、扩展逻辑卷:
3286     lvextend : 扩展其物理边界
3287         -L [+]# /PATH/TO/LV
3288
3289     resize2fs : 扩展其文件系统边界
3290     resize2fs /PATH/TO/LV 5G 修复逻辑边界到5g
3291     resize2fs -p /PATH/TO/LV 扩展至最大
3292
3293
3294
3295 2、缩减逻辑卷
3296 注意:
3297     1、不能在线缩减, 要先卸载再缩减
3298     2、确保缩减后的空间大小依然能存储原有的所有数据
3299     3、在缩减之间应强制检查文件, 以确保文件系统处于一致性状态:
3300
3301     df -lh 确保大小足够
3302     umount 卸载挂载
3303     e2fsck -f /PATH/TO/LV 检查文件系统
3304
3305     resize2fs
3306         resize2fs /PATH/TO/PV 3G 缩小至3G
3307
3308     lvreduce -L [-]# /PATH/TO/LV
3309
3310 重新挂载
3311
3312 3、快照卷
3313 1、生命周期为整个数据时常: 在这段时常内, 数据的增长量不能超出快照卷大小:
3314 2、快照卷应该是只读的
3315 3、跟原卷在同一卷组内

```

```

3316
3317 lvcreate
3318     -s :指定为快照卷
3319     -p r|w 指定权限，一般快照用于数据备份使用所以快照卷不允许修改所以应该为只读
3320
3321 lvcreate -L # -n SLV_NAME -s -p r /PATH/TO/LV
3322
3323
3324 脚本编程控制结构
3325     顺序
3326     选择
3327         if
3328         case
3329     循环
3330         for
3331         while
3332         until
3333
3334 while CONDITION; do
3335     statment
3336 done
3337
3338 进入循环，条件满足
3339 退出循环，条件不满足
3340
3341 until CONDITIONL;do
3342     statement
3343 done
3344
3345 进入循环，条件不满足
3346 退出循环，条件满足
3347     (line2647.sh)
3348
3349 每隔五秒钟查看zhangshuo是否登陆通过until实现 (line2649.sh)
3350
3351
3352 for (( expr1 ; expr2 ;expr3 ));do
3353     循环体
3354 done
3355
3356 100以内所有整数的和for另一种用法实现 (line2656.sh)
3357
3358 写一个脚本 (line2658.sh)
3359 1、通过ping命令测试192.168.0.151到192.168.0.254之间的所有主机时候在线
3360     如果在线，就显示"IP is up 。"其中的IP要换为真正的IP地址，且以绿色显示：
3361     如果不在线就显示"ip is down"，其中的IP要换为真正的IP地址，且以红色显示
3362
3363 要求：分别使用while， until， 和for循环实现
3364 echo -e "\033[31mhello\033[0m,world" 用于控制输出的颜色设置
3365
3366 ping
3367     -c: 指定ping次数
3368     -w: 指定等待时间，超时则退出
3369
3370 awk 'pattern{action}' file
3371     print $1 :显示第一段
3372
3373 Df -Ph | awk '{print $1}'
3374 awk -F : '{print $1,$3}' /etc/passwd 以： 为分割字段，输出第一三列
3375
3376 fdisk -l 2> /dev/null | grep "^Disk /dev/[sh]d[a-z]":显示设备信息
3377
3378 fdisk -l 2> /dev/null | grep "^Disk /dev/[sh]d[a-z]" | awk -F : '{print
3379 $1}':只显示设备名称
3380
3381 写一个脚本（前提，请为虚拟机新增一块硬盘，假设它为/dev/sdb），为指定的硬盘创建分区
3382 1、列出当前系统上所有的磁盘，让用户选择，如果用户选择quit则退出脚本，如果用户选择错误
    ，就让用户重新选择
3383 2、当用户选择后，提醒用户确认接下来的操作可能会毁坏数据，并请用户确认：如果用户选择y就
    继续，否则，让用户重新选择
3384 3、抹除那块硬盘上的所有分区（提示，抹除所有分区后执行sync命令，并让脚本睡3秒后再分区）
    ；并为其创建三个分区，第一个为20M，第二个为

```

512M, 第三个为128M, 且第三个为swap分区类型, (提示, 将分区命令通过echo传送给fdisk即可实现)

```
3385
3386 fdisk -l 2> /dev/null | grep "^Disk /dev/[sh]d[a-z]"
3387 fdisk -l 2> /dev/null | grep "^Disk /dev/[sh]d[a-z]" | awk -F : '{print $1}'
取出第一段
3388 脚本来实现 (line2675.sh)
3389
3390 dd if=/dev/zero of=/dev/sdb bs=512 count=1 : 抹除硬盘的MBR既删除所有分区
3391 执行上述命令之后执行
3392 sync将数据同步到磁盘
3393 sleep 3 睡3秒防止操作太快磁盘无法正常工作
3394
3395 #!/bin/bash
3396 #by zhangshuo 20180123
3397 #
3398 fdisk -l 2> /dev/null | grep "^Disk /dev/[sh]d[a-z]" | awk -F : '{print $1}'
3399
3400 read -p "input you disk: " DISK
3401
3402 until [ $DISK == 'quit' ] || fdisk -l 2> /dev/null | grep "^Disk /dev/[sh]d[a-z]" |
3403     awk -F : '{print $1}' | grep "^Disk ${DISK}$" &> /dev/null ;do
3404     read -p "input you disk: " DISK
3405 done
3406
3407 echo "your choice is $DISK"
3408
3409 echo "waring the dask will gg!!!!,continu?"
3410
3411 read -p "y/n: " CHOICE
3412
3413 until [ $CHOICE == 'n' -o $CHOICE == 'y' ];do
3414     read -p "y/n: " CHOICE
3415 done
3416
3417 echo $CHOICE
3418
3419 if [ $CHOICE == 'y' ];then
3420     #当操作某个设备时应首先保证该设备是处于没有挂载状态
3421     for I in `mount | grep "$DISK" | awk '{print $1}'`;do
3422         umount $I
3423         echo "umount $I done"
3424     done
3425     #卸载完设备以后便可以开始格式化
3426     echo "stat repart $DISK"
3427     #清除但前磁盘所有分区
3428     dd if=/dev/zero of=$DISK bs=512 count=1 &> /dev/null
3429     sync
3430     sleep 3
3431     echo 'n
3432     p
3433     1
3434
3435     +1G
3436     n
3437     p
3438     2
3439
3440     +1G
3441     n
3442     p
3443     3
3444
3445     +1G
3446     n
3447     e
3448
3449
3450     n
3451
3452     +1G
3453     n
```



```

3454
3455 +1G
3456 n
3457
3458 +1G
3459 n
3460
3461 +1G
3462 n
3463
3464
3465 t
3466 1
3467 fd
3468 t
3469 2
3470 fd
3471 t
3472 3
3473 fd
3474 t
3475 5
3476 fd
3477 t
3478 6
3479 fd
3480 t
3481 7
3482 fd
3483 t
3484 8
3485 fd
3486 w'| fdisk $DISK &> /dev/null
3487 sync
3488 sleep 3
3489
3490 partprobe
3491
3492 echo "done"
3493
3494 fdisk -l $DISK
3495
3496 #此时分区结束，开始建立RAID及LVM
3497
3498 else
3499     exit 0
3500 fi
3501
3502
3503 网络管理
3504
3505 点分十进制：
3506 0-255
3507
3508 221.34.23.12
3509
3510 Ip地址与端口的绑定就是套接字文件
3511
3512 网络地址：
3513 主机地址：
3514
3515 A类：第一段表示网络地址后三段表示主机地址    255.0.0.0    8位长度的掩码
3516 0    000 0001 - 0    111 1111
3517 127个A类，127用于回环，1-126
3518 容纳多少个主机： $2^{24}-2$ 
3519 主机位全0：网络地址
3520 主机位全1：广播地址
3521 B类：前两段表示网络地址，后两段表示主机地址 255.255.0.0 16位长度的掩码
3522 10 00 0000 - 10 11 1111
3523 128-191
3524 64个B类网络， $2^{14}$ 方个
3525 容纳多少个主机： $2^{16}-2$ 
3526 C类：前三段表示网络地址，最后一段表示主机地址    255.255.255.0 24位长度的掩码

```

3527 110 0 0000 - 110 1 1111
3528 192-223
3529 32个C类地址， 2^{21} 次方个C类网络
3530 容纳多少个主机： 2^8-2
3531 D
3532 1110 0000 - 1110 1111
3533 224-239
3534 E
3535
3536 私有地址用于建立局域网可以随意使用的地址，但是不可以连接到外网：
3537 A类：10.0.0.0/8
3538 B类：172.16.0.0/16-172.31.0.0/16
3539 C类：192.168.0.0/24-192.168.255.0/24
3540
3541 路由，选路
3542
3543
3544 主机接入网络需要配置如下信息：
3545
3546 IP：
3547 NETMASK掩码：
3548 GATEWAY网关：
3549 HOSTNAME主机名：
3550 DNS1：服务器地址
3551 DNS2：备用dns
3552 DNS3：
3553 以上所有配置手动指定
3554 DHCP：
3555 随机自动分配给的地址169.254
3556
3557 路由
3558
3559 Linux：
3560 lo：本地回环
3561 以太网卡：ethX
3562 pppX：点对点连接
3563
3564 在系统中如果直接操作设备文件会很麻烦，所以linux引入了设备名称，这样可以直接引用该设备
3565 RHEL6中定义网卡名称的文件：/etc/udev/rules.d/70-persistent-net.rules
3566 RHEL5中定义网卡别名的文件：/etc/modprobe.conf 通过alias别名定义的
3567
3568 修改Centos7ifconfig命令的网卡名称
3569 1、vim /etc/default/grub
3570 在GRUB_CMDLINE_LINUX的最后，加上 net.ifnames=0 biosdevname=0 的参数
3571 GRUB_CMDLINE_LINUX="rd.lvm.lv=rootvg/usrlv rd.lvm.lv=rootvg/swaplv crashkernel=auto
vconsole.keymap=us rd.lvm.lv=rootvg/rootlv vconsole.font=latacyrheb-sun16 rhgb
quiet net.ifnames=0 biosdevname=0"
3572 2、grub2-mkconfig -o /boot/grub2/grub.cfg
3573 3、mv /etc/sysconfig/network-scripts/ifcfg-enp0s3
/etc/sysconfig/network-scripts/ifcfg-eth0
3574 4、将复制过来的文件内容替换为eth0
3575 5、reboot
3576
3577 ifconfig
3578 -a:显示所有接口的配置信息
3579
3580 ifconfig ethX IP/MASK {up|down}
3581 配置的地址立即生效，但重启网络服务或主机，都失效
3582
3583 网络服务：
3584 RHEL5: /etc/init.d/network {start|stop|restart|status}
3585 RHEL6: /etc/init.d/NetworkManager {start|stop|restart|status}
3586
3587 网关：
3588 route
3589 add:添加
3590 -host:主机路由
3591 -net:网络路由
3592 -net 0.0.0.0
3593 J 表示可以通过192这个网关到达10网络
3594 route add -net|-host DEST gw NEXTHOP
3595 route add default gw NEXTHOP 添加默认路由
3596

```
3597 del:删除
3598     -host
3599     -net
3600
3601 route del -net 10.0.0.0/8 [gw NEXTHOP]
3602 route del -net 0.0.0.0 删除默认路由
3603 route del default
3604
3605 所做出的改动或重启网络服务后失效;
3606
3607
3608 查看:
3609     route -n : 以数字方式显示各主机或端口等相关信息
3610
3611 查看网卡UUID:
3612 nmcli con show      或      nmcli con list
3613
3614 查看网卡mac地址:
3615 nmcli dev show      或      nmcli dev list
3616
3617 查看当前网卡速度命令:
3618 dmesg | grep "eth0"
3619
3620 查看硬件设备生产商
3621 lspci
3622
3623 网络配置文件:
3624 /etc/sysconfig/network
3625
3626 网络接口配置文件:
3627 /etc/sysconfig/network-scripts/ifcfg-INTERFACE_NAME
3628 DEVICE=:关联的设备名称, 要与文件名的后半部"INTERFACE_NAME"保持一致;
3629 BOOTPROTO={static|none|dhcp|bootp}:引导协议: 要使用静态地址, 使用static或none; dhcp表示使用DHCP服务器获取地址;
3630 IPADDR=: IP地址
3631 NETMASK=: 子网掩码
3632 GATEWAY=: 设定默认网关
3633 ONBOOT=: 开机时是否自动激活此网络接口
3634 HWADDR=: 硬件地址, 要与硬件中的地址保持一致, 可省:
3635 USERCTL={yes|no}:是否允许普通用户控制此接口
3636 PEERDNS={yes|no}:是否在BOOTPROTO为dhcp时接受由DHCP服务器指定的DNS地址:
3637
3638 不会立即生效, 但重启网络服务或主机都会生效:
3639
3640
3641 路由:
3642 /etc/sysconfig/network-scripts/route-ethx
3643 添加格式一: 新建文件
3644 DEST{目标} via{下一跳}      NEXTHOP{}
3645 192.168.10.0/24 via 10.10.10.254
3646
3647 添加格式二:
3648 ADDRESS0=
3649 NETMASK0=
3650 GATEWAY0=
3651
3652 DNS服务器指定方法只有一种:
3653 /etc/resolv.conf
3654 nameserver DNS_IP_1
3655 nameserver DNS_IP_2
3656
3657 指定本地解析:
3658 /etc/hosts
3659 主机IP 主机名 主机别名
3660 172.16.0.1 www.zhangshuo.com www
3661
3662 DNS-->/etc/hosts-->DNS
3663
3664
3665 配置主机名:
3666 hostname HOSTNAME
3667 立即生效, 但不是永久生效
3668
```

```

3669 /etc/sysconfig/network
3670 HOSTNAME=
3671
3672
3673 ifconfig :老旧
3674
3675 iproute2
3676     ip
3677         link:配置网络接口属性
3678             show
3679                 -s:显示统计信息
3680                 ip -s link show
3681             set
3682                 ip link set DEV{up|down}
3683         addr: 协议地址
3684             add
3685                 ip addr add ADDRESS dev DEV label ethx:x
3686                 ip addr add 10.3.3.3/8 dev eth1 label eth1:1
3687             del
3688                 ip addr del ADDRESS dev DEV label ethx:x
3689             show
3690
3691             flush 清除
3692                 ip addr flush eth1 to 10/8
3693
3694         route: 路由
3695
3696
3697
3698 一块网卡可以使用多个地址:
3699 网络设备可以别名:
3700 eth0
3701     etchX:X
3702     ifconfig eth0:0 192.168.1.105/24
3703
3704 配置方法:
3705     ifconfig     ethx:x IP/NETMASK
3706
3707     /etc/sysconfig/network-scripts/ifcfg-ethx:x
3708     DEVICE=ethx:x
3709
3710     非主要地址不能使用DHCP动态获取
3711
3712
3713
3714 软件包管理
3715
3716 应用程序:
3717     程序与, Architecture体系架构有很大的关系
3718 c语言: 源代码-->(编译) 二进制格式
3719 脚本: 解释器 (二进制程序)
3720
3721 源代码: -->编译-->链接-->运行
3722     程序:
3723         库
3724             静态
3725             动态
3726
3727             静态链接
3728             动态链接
3729             共享库.so
3730
3731 配置文件:
3732 dir=/path/to/somewhere
3733
3734 程序的组成部分:
3735     二进制程序
3736     库
3737     配置文件
3738     帮助文件
3739
3740 /boot
3741 /etc

```

3742 /usr
3743 /var
3744 /dev
3745 /lib
3746 /tmp
3747 /bin
3748 /sbin
3749 /proc
3750 /sys
3751 /mnt
3752 /media
3753 /home
3754 /root
3755 /misc
3756 /opt
3757 /srv
3758
3759 /usr/share/man: 帮助文件
3761
3762 /etc:配置文件,/bin,/sbin: 二进制程序,/lib: 库文件
3763 系统启动就需要用到的程序, 这些目录不能挂载额外的分区, 必须挂载在根文件系统的分区上
3764
3765 /usr/
3766 bin
3767 sbin
3768 lib
3769
3770 提供操作系统核心功能, 可以单独分区
3771
3772 /usr/local
3773 bin
3774 sbin
3775 lib
3776 etc
3777 man
3778
3779 第三方程序安装目录, 建议单独分区
3780
3781
3782 /proc
3783 /sys
3784 不能单独分区, 默认为空
3785
3786 /dev:设备, 不能单独分区:
3787
udev:利用内核信息动态创建设备文件, 使得设备文件可以按需创建不再是将所有文件都放在
这里
3788 /root:不能单独分区
3789
3790 /var:日志信息和运行信息存放目录 建议单独分区
3791
3792 /boot:内核 建议单独分区因为文件系统一般是靠LVM托管, 开机时bootloader要找内核
3793
3794
3795 程序: 指令+数据
3796 指令: 芯片
3797 CUP: 普通指令, 特权指令
3798 指令集
3799
3800
3801 C语言:
3802 Powerpc CPU 编译为 二进制
3803
3804 不能在x86:运行
3805
3806
3807 软件包管理器
3808 打包成一个文件: 二进制程序, 库文件, 配置文件, 帮助文件
3809 生成数据库, 追踪所安装的每一个文件
3810
3811 软件包管理器的核心功能
3812 1、制作软件包

```
3813 2、安装、卸载、升级、查询、校验：
3814
3815 Redhat , suse, debian
3816
3817 redhat, suse RPM
3818
3819 Debian: dpt
3820
3821
3822 依赖关系：
3823     x-->y-->z
3824 前端工具： yum apt-get
3825 后端工具： rpm, dpt
3826
3827 yum: Yellowdog Update Modifier
3828
3829 rpm命令：
3830     rpm:
3831         数据库/var/lib/rpm
3832     rpmbuid: 用于创建rpm软件包
3833
3834 安装、查询、卸载、升级、校验、数据库重建、验证数据包的合法性等工作；
3835
3836 rpm命名：
3837 包： 组成部分
3838     主包：
3839         bind-9.7.1-1.i586.el5.rpm
3840     子包：
3841         bind-libs-9.7.1-1.i586.el5.rpm
3842 包名格式
3843     name-version-release.arch.rpm
3844     bind-major.minor.release-release.aech.rpm
3845
3846 主版本号： 重大改进
3847 次版本号： 某个子功能发生重大改变
3848 发行号： 修正了部分bug，调整了一点功能
3849
3850 bind-9.7.1.tar.gz这是下载的文件
3851
3852 rpm包：
3853     二进制格式： 安装简单
3854
3855     rpm包作者下载源程序，编译配置完成后，制作成rpm包此时会加上自己的发行号，即后面
3856     的发行号。
3857     源码格式： 能够更适合自己的硬件平台
3858
3859 rpm
3860
3861 1、安装
3862 rpm
3863     -i /PATH/TO/PACKAGE_FILE
3864     -h:以#显示安装进度： 每个#表示2%；
3865     -v:显示详细过程
3866     -vv:更纤细的过程
3867
3868 rpm -ivh /PATH/TO/PACKAGE_FILE
3869
3870     --nodeps:忽略依赖关系；
3871     --replacepks:重新安装，替换原有安装；
3872     --force:强行安装，可以实现重装或降级安装
3873
3874 2、查询
3875 rpm -q PACKAGE_NAME : 查询指定的包是否已经安装
3876     --changelog : rpm包的修改日志
3877     rpm -q bash
3878 rpm -qa :查询已经安装的所有包
3879     rpm -qa python*
3880
3881 rpm -qi PACKAGE_NAME : 查询指定包的说明信息
3882 rpm -ql PACKAGE_NAME:查询指定包安装后生成的文件列表
3883 rpm -qc PACKAGE_NAME:查询安装包的配置文件
3884 rpm -qd PACKAGE_NAME:查询指定包安装的帮助文件；
```

```
3884 rpm -q --scripts PACKAGE_NAME: 查询指定包包含的脚本
3885 脚本包含四部分, 安装前, 安装后, 卸载前, 卸载后
3886
3887 rpm -qf /path/to/somefile:查询指定的文件是由哪个rpm包安装生成的:
3888
3889 如果某rpm包尚未安装, 我们需要查询其说明信息、安装以后会生成的文件:
3890
3891 rpm -qpi /PATH/TO/PACKAGE_FILE
3892
3893 3、升级
3894 rpm -Uvh /PATH/TO/NEW_PACKAGE_FILE: 如果装有老版本的, 则升级: 否则, 则安装
3895 rpm -Fvh /PATH/TO/NEW_PACKAGE_FILE:如果安装有老版本的, 则升级, 否则, 退出
3896
3897 4、降级
3898 rpm -Uvh --oldpackage /PATH/TO/NEW_PACKAGE_FILE
3899 rpm -Fvh --oldpackage /PATH/TO/NEW_PACKAGE_FILE
3900
3901 5、卸载
3902 rpm -e PACKAGE_NAME
3903     --nodeps 如果有安装包依赖此安装包则禁止卸载, 如果要强行卸载则需要解除依赖
3904
3905 6、校验
3906 rpm -V PACKAGE_NAME
3907
3908 校验结果会用如下格式表示
3909
3910     S file Size differs
3911     M Mode differs (includes permissions and file type)
3912     5 digest (formerly MD5 sum) differs
3913     D Device major/minor number mismatch
3914     L readLink(2) path mismatch
3915     U User ownership differs
3916     G Group ownership differs
3917     T mTime differs
3918     P caPabilities differ
3919
3920
3921 7、重建数据库
3922 rpm
3923     --rebuilddb:重建数据库, 一定会重新建立。
3924     --initdb:初始化数据库, 没有才建立, 有就不用建立
3925
3926 8、校验来源合法性, 及软件完整性
3927 加密类型:
3928     对称加密: 加密和解密使用同一个密码
3929     公钥加密: 每个密码都成对出现, 一个为私钥一个为公钥
3930         一对儿密钥, 公钥, 私钥: 公钥隐含与私钥中, 可以提取出来, 并公开出去
3931     单向加密, 散列加密: 提取数据特征码, 常用于数据完整性校验
3932
3933
3934 ls /etc/pki/rpm-gpg 保存红帽对外发布的校验码
3935 RPM-GPG-KEY-redhat-release
3936
3937 rpm --import /etc/pki/RPM-GPG-KEY-redhat-release 导入公钥
3938
3939 rpm -K /PATH/TO/PACKAGE_NAME
3940     dsa, gpg:验证来源合法性, 也即验证签名: 可以使用--nosignature, 忽略此项
3941     sha1, md5:验证软件包完整性: 可以使用--nodigest, 忽略此项
3942
3943 要解决rpm包的依赖关系很是复杂所有制作了yum软件包管理器
3944 该软件包为C/S架构, Client, Servier 即客户端服务器架构
3945
3946 yum repository yum仓库
3947     文件服务
3948         ftp: //
3949         web: //
3950         file: ///
3951
3952 yum自己的客户端
3953     配置文件 (该客户端的各种配置信息)
3954     写明服务器地址
3955
3956 仓库文件中都会存在repodata文件夹, 文件夹中就包含了该yum仓库的信息
```

```

3957
3958 HEML: Hypertext mark language
3959 XML:extended Mark Language
3960
3961 xml,json:半结构化的数据在不同操作系统之间实现数据传输
3962
3963
3964 yum仓库中的元数据文件:
3965 primary(首要的最早的).xml.gx
3966     所有RPM包的列表
3967     依赖关系
3968     每个RPM安装生成的文件列表
3969 filelists.xml.gz
3970     当前仓库中所有RPM包的所有文件列表:
3971 other.xml.ge
3972     额外信息, rpm包的修改日志:
3973 repomd.xml
3974     记录的是上面三个文件的时间戳和校验和:
3975
3976 comps*.xml:RPM包分组信息
3977
3978 man yum
3979
3980 command is one of:
3981     * install package1 [package2] [...]
3982     * update [package1] [package2] [...]
3983     * update-to [package1] [package2] [...]
3984     * update-minimal [package1] [package2] [...]
3985     * check-update
3986     * upgrade [package1] [package2] [...]
3987     * upgrade-to [package1] [package2] [...]
3988     * distribution-synchronization [package1] [package2] [...]
3989     * remove | erase package1 [package2] [...]
3990     * autoremove [package1] [...]
3991     * list [...]
3992     * info [...]
3993     * provides | whatprovides feature1 [feature2] [...]
3994     * clean [ packages | metadata | expire-cache | rpmdb | plugins | all ]
3995     * makecache [fast]
3996     * groups [...]
3997     * search string1 [string2] [...]
3998     * shell [filename]
3999     * resolvedep dep1 [dep2] [...]
4000         (maintained for legacy reasons only - use repoquery or yum provides)
4001     * localinstall rpmfile1 [rpmfile2] [...]
4002         (maintained for legacy reasons only - use install)
4003     * localupdate rpmfile1 [rpmfile2] [...]
4004         (maintained for legacy reasons only - use update)
4005     * reinstall package1 [package2] [...]
4006     * downgrade package1 [package2] [...]
4007     * deplist package1 [package2] [...]
4008     * repolist [all|enabled|disabled]
4009     * repoinfo [all|enabled|disabled]
4010     * repository-packages <enabled-repoid>
4011     <install|remove|remove-or-reinstall|remove-or-distribution-synchronization>
4012     [package2] [...]
4013     * version [ all | installed | available | group-* | nogroups* | grouplist |
4014     groupinfo ]
4015     * history
4016     [info|list|packages-list|packages-info|summary|addon-info|redo|undo|rollback|n
4017     ew|sync|stats]
4018     * load-transaction [txfile]
4019     * updateinfo [summary | list | info | remove-pkgs-ts | exclude-updates |
4020     exclude-all | check-running-kernel]
4021     * fssnapshot [summary | list | have-space | create | delete]
4022     * fs [filters | refilter | refilter-cleanup | du]
4023     * check
4024     * help [command]
4025
4026 yum的配置信息在/etc/yum.conf
4027
4028 [main]
4029 cachedir=/var/cache/yum/$basearch/$releasever

```



```
4024 keepcache=0
4025 debuglevel=2
4026 logfile=/var/log/yum.log
4027 #严格检查平台版本
4028 exactarch=1
4029 #过期的要不要废弃
4030 obsoletes=1
4031 gpgcheck=1
4032 plugins=1
4033 installonly_limit=5
4034 bugtracker_url=http://bugs.centos.org/set_project.php?project_id=23&ref=http://bugs.ce
4035 ntos.org/bug_report_page.php?category=yum
4036 #发行的版本号
4037 distroverpkg=centos-release
4038
4039 # This is the default, if you make this bigger yum won't see if the metadata
4040 # is newer on the remote and so you'll "gain" the bandwidth of not having to
4041 # download the new metadata and "pay" for it by yum not having correct
4042 # information.
4043 # It is esp. important, to have correct metadata, for distributions like
4044 # Fedora which don't keep old packages around. If you don't like this checking
4045 # interrupting your command line usage, it's much better to have something
4046 # manually check the metadata once an hour (yum-updatesd will do this).
4047 # metadata_expire=90m
4048
4049 # PUT YOUR REPOS HERE OR IN separate files named file.repo
4050 # in /etc/yum.repos.d
4051
4052 如何为yum定义repo文件/etc/yum.repos.d
4053 [Repo_ID]
4054 name=Description
4055 baseurl=
4056     ftp://
4057     http://
4058     file:///
4059 enabled={0|1}是否启用
4060 gpgcheck={0|1}是否启用校验
4061 gpgkey=
4062
4063 yum [options] [command] [package ...]
4064     -y:自动回答为yes
4065     --nogpgcheck:
4066
4067 list:列表
4068     all:所有的
4069     availabel:可用的, 仓库中有但尚未安装的
4070     installed: 已经安装的
4071     updates:可用的升级
4072
4073 clean:清理缓存
4074     all:
4075 repolist:[all|enabled|disabled] 列出当前系统的仓库
4076
4077 install: 安装软件包
4078 yum install PACKAGE_NAME
4079
4080 update:升级
4081 update_to : 升级为指定版本
4082
4083 remove | erase :卸载
4084
4085 info :详细信息
4086
4087 provides | whatprovides :查看指定的文件或特性是由哪个包安装生成的;
4088 yum provides /etc/inittab :查看某个文件属于哪个安装包, 即rpm -qf /path/to/somewhere
4089
4090 group info :组详细信息
4091 group list
4092 group install
4093 group remove
4094 group update
4095
```

```
4096 如何创建yum仓库
4097 createrepo /yum/VT/
4098 1、在/etc/yum.repo中创建一个文件
4099 [Base]
4100 name=Centos7 CDROM Server
4101 baseurl=file:///media/zhangshuo/cdrom
4102 enabled=1
4103 gpgcheck=0
4104
4105
4106 RPM安装：
4107     二进制格式：
4108     源程序-->编译-->二进制格式
4109     有些特性是编译选定的，如果编译未选定此特性，将无法使用：
4110     rpm包的版本会落后与源码包，甚至落后很多：bind-9.8.7，bind-9.7.2
4111
4112 定制：手动编译安装
4113
4114 编译环境，开发环境
4115 开发库，开发工具
4116
4117 Linux：C，
4118 GNU：C
4119
4120 C，C++
4121 gcc:GNU C Compiler,c
4122 g++:
4123
4124 如果手动使用gcc编译器，当要编译和链接的文件很多时要指定很多编译次序，此时对于大项目来说会很复杂，所以有了make工具
4125
4126 make:项目管理工具
4127     makefile: 定义了make(gcc,g++)gcc按何种次序去编译这些源程序文件中的源程序
4128
4129 automake,-->makefile.in-->makefile
4130 autoconf,-->configure
4131
4132 100个可选特性
4133
4134 make install
4135
4136
4137 编译安装的三步骤
4138 前提：准备开发环境（编译环境）
4139 安装“Development Tools”和“Development Libraries”
4140
4141 #tar
4142 #cd
4143 #./configure
4144     --help:帮助
4145     --prefix=/PATH/TO/SOMEWHERE:软件安装路径
4146     --sysconfdir=/PATH/TO/CONFIE_PATH:配置文件路径
4147     功能：1、让用户选定编译环境：2、检查编译环境：
4148 #make
4149 #make install
4150
4151 更改环境变量/etc/profile
4152     PATH=$PATH:/usr/local/nginx/sbin加入此行可以改变环境变量
4153     export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL
4154 或
4155 在/etc/profile.d/目录下建立一个以.sh为名称后缀的文件，在里面定义export
4156     PATH=$PATH:/PATH/TO/SOMEWHERE
4157
4158 source /etc/profile :可以通知内核重新读取该文件
4159
4160 # tar xf tensorflow-1.4.2.tar.gz
4161 # cd tensorflow-1.4.2
4162 # ./configure --prefix=/usr/local/tengine --conf-path=/etc/tengine/tengine.conf
4163 # make
4164 # make install
4165
4166 1、修改PATH环境变量，以能够识别此程序的二进制文件路径
4167     修改/etc/profile文件
```

```

4167     在/etc/profile.d/目录建立一个以.sh为名称后缀的文件，在里面定义export
4168     PATH=$PATH:/PATH/TO/SOMEWHERE
4169 2、默认情况下，系统搜索库文件的路径/lib,/usr/lib；要增添额外搜索路径：在
4170     /etc/ld.so.conf.d/中创建以.conf为后缀名的文件，而后把要添加的路径直接写至此文件中
4171     # ldconfig通知系统重新搜寻库文件
4172     -v：显示重新搜寻库的过程
4173 3、头文件：输出给系统
4174     默认：/usr/include
4175     增添头文件搜寻路径，使用链接进行
4176     将/usr/local/tengine/include/    链接到/usr/include/目录下
4177     两种方式
4178     ln -sv /usr/local/tengine/include/* /usr/include/    或
4179     ln -sv /usr/local/tengine/include    /usr/include/tengine
4180 4、man文件路径：安装在--prefix指定的目录下的man目录：/usr/share/man
4181     1、man -M /path/to/man dir command
4182     2、在/etc/man_db.conf 中添加一条MANPATH
4183
4184
4185 删除CentOS更新后的旧内核
4186 1.查看系统当前内核版本：
4187     # uname -a
4188 2.查看系统中全部的内核RPM包：
4189     # rpm -qa | grep kernel
4190 3.删除旧内核的RPM包
4191     yum remove kernel-2.6.18-194.el5
4192     yum remove kernel-devel-2.6.18-194.el5
4193
4194 netstat命令
4195     -r:显示路由表
4196     -n:以数字方式显示
4197
4198     -t:建立的tcp链接
4199     -u:显示udp链接
4200     -l:显示监听状态的链接（多个选项可以同时使用）
4201     -p:显示监听指定的套接字的进程的ID号及进程名
4202
4203 # netstat -tulnp 显示tcp链接udp链接监听状态链接及进程id等所有信息
4204
4205

```