Linux知识

一、常用命令与参数

1）locale：显示目前支持的语系

```
[rtm@worker9 ~]$ locale
LANG=en_US.UTF-8
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_COLLATE="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
LC_PAPER="en_US.UTF-8"
LC_NAME="en_US.UTF-8"
LC_ADDRESS="en_US.UTF-8"
LC_TELEPHONE="en_US.UTF-8"
LC_MEASUREMENT="en_US.UTF-8"
LC_IDENTIFICATION="en_US.UTF-8"
LC_ALL=
[rtm@worker9 ~]$ LANG=en_US.UTF-8 # 更改输出讯息语系
[rtm@worker9 ~]$ export LC_ALL=en_US.UTF-8 # 更改所有讯息语系
```

2）date：显示日期

```
[rtm@worker9 ~]$ date
Fri Sep  9 14:54:31 CST 2022
[rtm@worker9 ~]$ date +%Y/%m/%d
2022/09/09
[rtm@worker9 ~]$ date +%H:%M
14:55
```

3）cal：显示日历

```
[rtm@worker9 ~]$ cal
   September 2022
Su Mo Tu We Th Fr Sa
             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
```

4）bc：计算器

```
[rtm@worker9 ~]$ bc
bc 1.06.95
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
scale=3 # 小数点精确至3位数
1/3
.333
quit
```

5）--help、man、info展示命令信息

```
[rtm@worker9 ~]$ date --help
[rtm@worker9 ~]$ man date
[rtm@worker9 ~]$ info date
[rtm@worker9 ~]$ man -k man # 信息内包含man的都展示
bsd_signal (3)       - signal handling with BSD semantics
cproj (3)            - project into Riemann Sphere
cprojf (3)           - project into Riemann Sphere
cprojl (3)           - project into Riemann Sphere
fallocate (2)        - manipulate file space
getopt (3)           - Parse command-line options
getopt_long (3)      - Parse command-line options
getopt_long_only (3) - Parse command-line options
```

6）sync、shutdown、reboot、poweroff、halt

```
[rtm@worker9 ~]$ sync # 将缓存、内存数据同步写入磁盘
[rtm@worker9 ~]$ /sbin/shutdown -h 10 'I will shutdown after 10 minutes'
# 通知其他用户，十分钟后关机
[rtm@worker9 ~]$ /sbin/shutdown -r 10 'I will reboot after 10 minutes'
[rtm@worker9 ~]$ /sbin/shutdown -k 10 'I will shutdown after 10 minutes'
# 警告消息，并不会真正关机
[rtm@worker9 ~]$ reboot
[rtm@worker9 ~]$ poweroff
[rtm@worker9 ~]$ halt # 系统停止，还保留最后的页面
```

7）chgrp、chown、chmod：改变文件属性

```
[rtm@worker9 ~]$ chgrp rtm zookeeper.out
[rtm@worker9 ~]$ chgrp -R rtm zookeeper # 改变文件夹下所有文件群组
[rtm@worker9 ~]$ chown rtm zookeeper.out
[rtm@worker9 ~]$ chown -R rtm zookeeper.out
[rtm@worker9 ~]$ chown user:group zookeeper.out
[rtm@worker9 ~]$ chmod 777 zookeeper.out
[rtm@worker9 ~]$ chmod u=rwx,gp=rx zookeeper.out
[rtm@worker9 ~]$ chmod a+w zookeeper.out # a表示三种类型权限
```

8）uname：查看Linux系统参数

```
[rtm@worker9 ~]$ uname -a
Linux worker9 3.10.0-862.el7.x86_64 #1 SMP Fri Apr 20 16:44:24 UTC 2018 x86_64
x86_64 x86_64 GNU/Linux
```

9）rmdir：删除空目录

```
[rtm@worker9 test]$ rmdir test3
```

10）echo $PATH：查看系统环境目录

```
[rtm@worker9 test]$ echo $PATH
/opt/spark//bin:/opt/hadoop/bin:/opt/hadoop/sbin:/usr/java/latest/bin:/usr/local
/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/phoenix/bin:/home/rtm/.local/bin:/h
ome/rtm/bin
[rtm@worker9 test]$ PATH="{$PATH}:/home/rtm/app"
```

11）ls：查看文件列表

```
[rtm@worker9 test]$ ls -alh # a包括隐藏文件，l表示长数据行，h表示大小可读性展示
total 8.0K
drwxr-xr-x   4 rtm rtm    32 Sep  9 15:48 .
drwx------. 42 rtm rtm  4.0K May  6 14:35 ..
drwxr-xr-x   2 rtm root   40 Dec  3  2021 test1
drwxr-xr-x   2 rtm root   23 Dec  3  2021 test2
```

12）cp、rm、mv

```
[rtm@worker9 test1]$ cp -i test1.cli test3.cli # i表示询问是否覆盖
cp: overwrite ‘test3.cli’? y
[rtm@worker9 test1]$ cp -p test3.cli test4.cli # p表示复制权限和时间
[rtm@worker9 test1]$ cp -a test4.cli test5.cli # a表示复制权限和时间还有隐藏属性
[rtm@worker9 test1]$ cp -u test4.cli test5.cli # u表示前文件必须比后文件新才替换
[rtm@worker9 test1]$ mv -u test4.cli ./ # u表示前文件必须比后文件新才替换
```

13）basename、dirname

```
[rtm@worker9 test1]$ basename test1.cli
test1.cli
[rtm@worker9 test1]$ dirname /home/rtm/test/test1/test1.cli
/home/rtm/test/test1
```

14）cat、tac、nl、more、less、head、tail、od查看文件

```
[rtm@worker9 test1]$ cat -A test1.cli # A表示展示特殊字符
[rtm@worker9 test1]$ tac test1.cli # 反向输出文件
[rtm@worker9 test1]$ nl test1.cli # 带行号输出
[rtm@worker9 test1]$ more test1.cli # 翻页输出
[rtm@worker9 test1]$ less test1.cli # 翻页输出
[rtm@worker9 test1]$ tail -f -n 100 test1.cli # n输出尾部100条，f同步
[rtm@worker9 test1]$ head -n 100 test1.cli # n输出头部100条
[rtm@worker9 test1]$ od -t oCc test1.cli # 按照特殊格式输出文件
```

15）touch：新建空文件或修改文件时间

```
[rtm@worker9 test1]$ touch -d "2 days ago" test1.cli
```

16）umask：文件默认权限

```
[rtm@worker9 test1]$ umask -S
u=rwx,g=rx,o=rx
```

17）chattr、lsattr：隐藏属性

```
[rtm@worker9 test1]$ chattr [+-=][ASacdistu] 文件
# A表示存取此文件，不修改access time，优化性能
# S表示写入文件直接同步到磁盘
# a表示只能添加新数据进文件，不能删除或修改
# c文件读取自动解压，写入后自动压缩
# d dump时不会dump该文件
# i表示不能对文件做任何改动
[rtm@worker9 test1]$ lsattr 文件
```

18）which：查找脚本文件

```
[rtm@worker9 test1]$ which ifconfig # $PATH下搜索
/usr/sbin/ifconfig
```

19）whereis、find、locate：查找文件

```
[rtm@worker9 test1]$ whereis ifconfig # 仅在sbin或man下插找
ifconfig: /usr/sbin/ifconfig /usr/share/man/man8/ifconfig.8.gz
[rtm@worker9 test1]$ locate -i -l 5 ifconfig # 要求/var/lib/mlocate数据库内的数据
/home/rtm/cpri/python3/lib/python3.6/site-packages_move/sphinx/ext/ifconfig.py
/home/rtm/cpri/python3/lib/python3.6/site-
packages_move/sphinx/ext/__pycache__/ifconfig.cpython-37.pyc
/usr/sbin/ifconfig
/usr/sbin/pifconfig
/usr/share/man/de/man8/ifconfig.8.gz
[rtm@worker9 test1]$ updatedb # 更新/var/lib/mlocate数据库内的数据
[rtm@worker9 test1]$ find [PATH] [option] [action]
# -mtime n表示n天前一天之内改动的文件，-mtime +n表示n天之前被改动文件，-newer file表示比
file新的文件
# -user name表示查询特定用户的文件，-group name，-name name按照文件名称去找
```

20）df、du：查看目录占用

```
[rtm@worker9 ~]$ df -h
Filesystem                                Size  Used Avail Use% Mounted on
/dev/mapper/centos-root                    50G   21G   30G  42% /
/dev/sda1                                1014M  145M  870M  15% /boot
/dev/mapper/centos-home                   600G  184G  417G  31% /home
/dev/mapper/centos-data                   1.2T  729G  467G  61% /data
# a表示所有文件系统，h表示可读性展示，T表示展示filesystem名称，i表示inode数量

[rtm@worker9 ~]$ du -h
3.4G    ./backup
1.7G    ./20200311
1.6G    ./20200318
1.3G    ./20200323
29G     .
# a表示所有文件和目录容量，h表示可读性展示，s表示仅列出总量，S表示不包括子目录统计
```

21）ln：实体连接和符号链接

```
[rtm@worker9 ~]$ ln /etc/crontab .
# hard link，相当于新建一个inode指向旧inode指向的block，不能对目录使用，不能跨fileSystem
[rtm@worker9 ~]$ ln -s /etc/crontab crontab2
# -s表示symbolic link，相当于建立快捷方式，新建inode，指向的data block指向旧inode
```

22）gzip、bzip2、xz：常用压缩

```
[rtm@worker9 ~]$ gzip -c services > services.gz
# c表示将压缩数据通过数据流重导向来处理（可避免默认压缩会导致源文件消失）
# d表示解压缩，t表示检验，v表述输出压缩比，数字表示压缩等级，9最好，默认6
# bzip2、xz使用方法类似
```

23）gcat、gless、gmore：直接对纯文本压缩后文件直接读取

```
[rtm@worker9 ~]$ zcat services.gz
# 对纯文本文档压缩后文件，不需要解压也可以直接读取，类似的还有zmore、zless、zgrep
# 对应bzip2有bzcat、bzmore、bzless、bzgrep，对应xz有xzcat、xzmore、xzless、xzgrep
```

24）tar：打包

```
[rtm@worker9 ~]$ tar -zcv -f filename source
# z表示使用gzip进行压缩/解压，j表示使用bzip2，J表示使用xz
# c表示建立打包文件，v表示查看处理的文件名，x表示解压，cv和xv互斥
# f表示新建的文档名称，C表示解压缩后放到哪个目录
# --exclude=FILE表示不打包某个文件
```

25）history：查看执行命令历史

```
[rtm@worker9 ~]$ history | grep less # 此次登录用户的执行历史
# c表示情况当前shell所有history内容，n表示展示数量，w表示写入histfiles，a表示新增入
histfiles
[rtm@worker9 ~]$ cat ~/.bash_history # 前一次登录执行的命令
[rtm@worker9 ~]$ !! # 执行上一条命令
[rtm@worker9 ~]$ !66 # 执行history中66号指令
[rtm@worker9 ~]$ !al # 执行最近的以al开头的指令
```

26）alias、unalias、type：对命令进行别名

```
[rtm@worker9 ~]$ alias lm='ls -al' # 直接输入alias会展示所有别名的命令
[rtm@worker9 ~]$ unalias lm
[rtm@worker9 ~]$ type -a ls # 查看命令信息
ls is aliased to `ls --color=auto'
ls is /usr/bin/ls
```

27）echo、=、unset：变量操作命令

```
[rtm@worker9 ~]$ echo ${PATH}
/opt/spark//bin:/opt/hadoop/bin:/opt/hadoop/sbin:/usr/java/latest/bin:/usr/local
/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/phoenix/bin:/home/rtm/.local/bin:/h
ome/rtm/bin
[rtm@worker9 ~]$ PATH=${PATH}:/usr/local/bin # =后面不能直接加空格，用使用''括起来或者
\转义，"${PATH}"可使用变量，单引号不行
[rtm@worker9 ~]$ export PATH # 普通变量只能在本地shell中使用，export后可将变量变成环境变
量
[rtm@worker9 ~]$ unset PATH
[rtm@worker9 ~]$ echo `uname -r` # ``内的指令会被优先执行，作为输入源，效果与$(指令)相同
3.10.0-862.el7.x86_64
[rtm@worker9 ~]$ echo $(uname -r)
3.10.0-862.el7.x86_64
```

28）env、set：展示变量

```
[rtm@worker9 ~]$ env # 展示系统变量，export不加变量时也可以展示系统变量
[rtm@worker9 ~]$ set # 展示当前所有变量
```

29）read、declare、typeset：读取键盘输入给变量赋值

```
[rtm@worker9 ~]$ read -p "Please enter atest:" atest
# p表示提示字符，t表示等待时间，时间等待后无输入则为空
Please enter atest:123
[rtm@worker9 ~]$ echo ${atest}
123
[rtm@worker9 ~]$ declare -xi asum=1+2+3
# x表示将变量变成系统变量，i表示定义为整数，a表示定义为数组，r表示定义为只读
# typeset和declare相同
[rtm@worker9 ~]$ env | grep asum
asum=6
[rtm@worker9 ~]$ declare +x asum # 取消asum的系统变量属性，变成普通变量
[rtm@worker9 ~]$ anum[1]=2 # 设定数组，${anum[1]}读取
```

30）login-shell和non-login-shell差别

- login-shell会去读取两个配置文件，/etc/profile（系统整体配置），~/.bash_profile或~/.bash_login或~/.profile（个人配置）
- non-login-shell会读取~/.bashrc

```
[rtm@worker9 ~]$ source ~/.bashrc # 使当前配置直接生效
```

31）> >> 2> 2>> < << tee：数据重导向

```
[rtm@worker9 ~]$ ll > ~/llresult
# >表示标准输出，>>表示标准输出追加，不会覆盖之前的内容
[rtm@worker9 ~]$ find111 2> ~/errorresult
# 2>表述标准错误输出，2>>表示标准错误输出追加
[rtm@worker9 ~]$ find111 2> dev/null # /dev/null表示垃圾桶
[rtm@worker9 ~]$ find /home /root123 -name .bashrc > result 2>&1
[rtm@worker9 ~]$ find /home /root123 -name .bashrc &> result
# 即标准输出和错误输出都放到一个文件中，且保持次序。若>result 2>result无法保持次序
[rtm@worker9 ~]$ cat > catfile < .bashrc
# <表示标准输入，从文件中提取数据，然后输出到catfile中
[rtm@worker9 ~]$ cat > catfile << "eof"
# <<表示结束字符，使用>从键盘获取数据，然后输入eof则表述输入结束
[rtm@worker9 test]$ last | head -n 3 | tee last.list | cut -d " " -f 1
rtm
rtm
rtm
# tee双向重导向，除了输出到指定设备文件，还会输出到标准输出即屏幕，a表示追加
```

32）；&& ||：命令执行判断依据

- ;分割的命令之间没有联系，前者失败不影响后者执行
- && 前者成功才会执行后者的命令
- || 前者成功则不执行后者的命令

33）|：管线

- 管线命令只能够接收standard output，无法接收standard error output
- 管线命令必须能够接收前一个指令的数据成为standard input继续处理才行，如cat less more等

34）cut、grep：截取命令，针对一行一行的信息

```
# cut提取信息中的需要部分
[rtm@worker9 ~]$ echo ${PATH}
/opt/spark//bin:/opt/hadoop/bin:/opt/hadoop/sbin:/usr/java/latest/bin:/usr/local
/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/phoenix/bin:/home/rtm/.local/bin:/h
ome/rtm/bin
[rtm@worker9 ~]$ echo ${PATH} | cut -d ':' -f 2 # -d后面接分割字符，-f后面接取出第几
段，可以2-5或者2，3取出数据，df合起来用
/opt/hadoop/bin
[rtm@worker9 ~]$ export | head -n 1
declare -x CLASSPATH=".:JAVA_HOME/lib/dt.jar:/usr/java/latest/lib/tools.jar"
[rtm@worker9 ~]$ export | head -n 1 | cut -c 12- # -c后面接数字表示字符范围，12-表示
12到最后一个字符
CLASSPATH=".:JAVA_HOME/lib/dt.jar:/usr/java/latest/lib/tools.jar"
# grep提取满足条件的信息
[rtm@worker9 ~]$ last | head -n 10 | tail -n 2
rtm      pts/0        100.98.141.74    Tue Sep  6 16:03 - 18:15  (02:11)
root     pts/0        100.98.141.64    Tue Jul 26 11:20 - 11:24  (00:03)
[rtm@worker9 ~]$ last | head -n 10 | tail -n 2 | grep root # i表示忽略大小写，n表示
输出行号，c表示计算符合条件次数，v表示反向选择
root     pts/0        100.98.141.64    Tue Jul 26 11:20 - 11:24  (00:03)
```

35）sort、uniq、wc：排序指令

```
# sort，f表示忽略大小写，b表示忽略最前面空格，M表示按照月份JAN排序，n表示使用纯数字排序，r表示
反向排序，u表示仅出现重复数据的一条，t表示分隔符，k表示使用哪个区间
[rtm@worker9 ~]$ cat /etc/passwd | head -n 3 | sort -t ':' -k 3
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
# uniq，i表示忽略大小写，c表示计数，但是要注意必须排序后再进行去重，目前看起来是会跟上一行数据
进行对比看是否重复，不排序会导致异常
[rtm@worker9 ~]$ last | cut -d ' ' -f 1 | sort | uniq -c
      1
     11 reboot
     51 root
    863 rtm
      1 wtmp
# wc，l表示行数，w表示字数，m表示字符数
[rtm@worker9 ~]$ cat /etc/passwd | wc
     37      69    1874
```

36）tr、col、join、paste、expand、unexpand：字符转换命令

```
# tr，d表示删除，s表示替换
[rtm@worker9 test]$ last | head -n 3 | tr [a-z] [A-Z]
RTM      PTS/1        100.98.141.74    WED SEP 14 13:59   STILL LOGGED IN
RTM      PTS/0        100.98.141.74    WED SEP 14 13:50   STILL LOGGED IN
RTM      PTS/0        100.98.141.74    WED SEP 14 11:08 - 13:22  (02:13)
# col，b表示不输出空格，x表示把tab换成对等空格键
[rtm@worker9 test]$ last | head -n 3 | col -b | cat -A
rtm^I pts/1^I      100.98.141.74    Wed Sep 14 13:59^I  still logged in$
rtm^I pts/0^I      100.98.141.74    Wed Sep 14 13:50^I  still logged in$
rtm^I pts/0^I      100.98.141.74    Wed Sep 14 11:08 - 13:22^I (02:13)$
```

```
# join，i表示忽略大小写，t后面接分隔符，1后面接数字表示第一个文件使用哪个字段，2后面接数字表示第二个文件使用哪个字段
# join后的结果会把比较字段放在最前面，并且join前的文件最好是排序过的
[root@master ~] head -n 2 /etc/passwd /etc/group
==> /etc/passwd <==
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin

==> /etc/group <==
root:x:0:gitlab-runner
bin:x:1:
[root@master ~] join -t ':' -1 4 /etc/passwd -2 3 /etc/group 2> /dev/null | head -n 2
0:root:x:0:root:/root:/bin/bash:root:x:gitlab-runner
1:bin:x:1:bin:/bin:/sbin/nologin:bin:x:
# paste，d后面接分隔符，-可替代文件，表示标准输入
[root@master ~] paste /etc/passwd /etc/shadow | head -n 1
root:x:0:0:root:/root:/bin/bash
root:$6$QOL5RmwO$3kyWSG84YtS7LSMqCrwEf.rHegnd79/tOa8kVU2QnadxvU7pSzILPDEf3NuSZjFZiKFG/UlZAWra1tT1UvuWy/:18074:0:99999:7:::
# expand，t后面接数字表示将一个tab换成多少个空格
[root@master ~] grep '^MANPATH' /etc/man_db.conf | head -n 1 | cat -A
MANPATH_MAP^I/bin^I^I^I/usr/share/man$
[root@master ~] grep '^MANPATH' /etc/man_db.conf | head -n 1 | expand -t 6 | cat -A
MANPATH_MAP /bin              /usr/share/man$
```

37）split：大文件拆成小文件

```
[root@master ~] split -b 300k /etc/services services
[root@master ~] ll -h | grep service
-rw-r--r--. 1 root root 300K Sep 14 15:39 servicesaa
-rw-r--r--. 1 root root 300K Sep 14 15:39 servicesab
-rw-r--r--. 1 root root  55K Sep 14 15:39 servicesac
# b后面接分成的文件大小，l后面接行号用于分区，最后的字符串为前缀
[root@master ~] cat services* >> servicesback
# 使用重导向就能将文件重合
```

38）xargs：参数代换

```
[rtm@worker9 test]$ cut -d ':' -f 1 /etc/passwd | head -n 3 | xargs -n 1 id
uid=0(root) gid=0(root) groups=0(root)
uid=1(bin) gid=1(bin) groups=1(bin)
uid=2(daemon) gid=2(daemon) groups=2(daemon)
# xargs可以用来产生某个指令的参数，n后接数字表示每次使用多少个结果做参数，p表示每次执行都进行询问，e后面接字符串表示EOF
```

39）-：代替标准输入输出，替换文件

```
# 前面的-表示将home打包到标准输出stdout中，后面的-表示从前一个指令的标准输出stdout中解开打包文件
[rtm@worker9 ~]$ tar -cvf - /home | tar -xvf - -C /tmp/homeback
```

40）sed：取代、删除、新增、截取等功能（管线命令之一）

```
# sed [-nefr] [n1[,n2]]function
# n表示选择哪几行，r表示支持延伸型正则表达式，f表示写入文件，e表示直接在指令列模式上进行sed操作
# function：d删除，a新增，c取代，i插入，p打印，s取代
[rtm@worker9 ~]$ nl /etc/passwd | sed '2,5d'| head -n 3
     1  root:x:0:0:root:/root:/bin/bash
     6  sync:x:5:0:sync:/sbin:/bin/sync
     7  shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
[rtm@worker9 ~]$ nl /etc/passwd | sed '2a drink tea' | head -n 3
     1  root:x:0:0:root:/root:/bin/bash
     2  bin:x:1:1:bin:/bin:/sbin/nologin
drink tea
[rtm@worker9 ~]$ nl /etc/passwd | sed '2,5c No 2-5 line' | head -n 3
     1  root:x:0:0:root:/root:/bin/bash
No 2-5 line
     6  sync:x:5:0:sync:/sbin:/bin/sync
[rtm@worker9 ~]$ ifconfig em1 | grep 'inet'
        inet 10.163.217.71  netmask 255.255.255.192  broadcast 10.163.217.127
[rtm@worker9 ~]$ ifconfig em1 | grep 'inet' | sed 's/ *inet //g'
10.163.217.71  netmask 255.255.255.192  broadcast 10.163.217.127
[rtm@worker9 ~]$ ifconfig em1 | grep 'inet' | sed 's/ *inet //g' | sed 's/
*netmask.*//g'
10.163.217.71
```

41）awk：数据处理工具

```
# awk '条件类型1{动作1} 条件类型2{动作2}...' filename
# awk针对的是每一行的字段的数据，sed针对的是每一行的数据
# awk默认以空格或tab区分，$1表示第一个字段，$0表示一整行，NF表示每一行$0字段数，NR表示当前是
第几行，FS表示当前分隔符
[rtm@worker9 ~]$ last |head -n 3 | awk '{print $1 "\t" $3 "\t line number: " NR
"\t columns:" NF }'
rtm     100.98.141.74    line number: 1  columns:10
rtm     100.98.141.74    line number: 2  columns:10
rtm     100.98.141.74    line number: 3  columns:10
# FS修改分隔符，如果不加入BEGIN，则在读取了第一行之后才会生效，此时第一行数据会异常
[rtm@worker9 ~]$ cat /etc/passwd | head -n 3 | awk 'BEGIN {FS=":"} $3 < 10
{print $1 "\t" $3}'
root    0
bin     1
daemon  2
```

42）diff、patch：文档对比与补丁

```
# b表示忽略一行中的空白差异，i表示忽略大小写差异，B表示忽略空白行差异
[rtm@worker9 ~]$ diff passwd.old passwd.new
4d3 # 左边第四行在右边被删除(d)了，以右边第3行为基准
< adm:x:3:4:adm:/var/adm:/sbin/nologin
6c5 # 左边第6行在右边被替换了，以右边第5行为基准
< sync:x:5:0:sync:/sbin:/bin/sync
---
> no dix line
[rtm@worker9 ~]$ diff passwd.old passwd.new > passwd.patch
[rtm@worker9 ~]$ patch -p0 < passwd.patch # 将passwd.old更新成passwd.new
[rtm@worker9 ~]$ patch -R -p0 < passwd.patch # 将更新后的passwd.old恢复原样
```

二、shell script相关

1）shell script执行相关知识

- 第一行的#!/bin/bash几乎是必须存在的，要通知系统以什么shell来执行，且能加载bash的相关环境配置文件
- exit后面接回传的值，一般用0表示正确执行
- 使用sh或者直接执行script都会使用新的bash环境来执行脚本命令，使用source来执行则是在原本的bash环境下执行
- $((运算内容))与declare -i total=${num1}*${num2}效果一致
- 默认变量，$0表示脚本名称，$1表示第一个参数，$#表示参数数量，$@表示["$1" "$2" "$3" "$4"]
- shift会导致变量向左移动，即前几个变量消失

2）test：测试指令

```
[rtm@worker9 ~]$ test -e bin2/ && echo "exist" || echo "No exist"
No exist
# e表示文档是否存在，f表示是否存在且为文件，d表示是否存在且为目录
# rwx表示文档是否可读、写、执行
[rtm@worker9 ~]$ test file1 -nt file2 || echo "No"
No
# nt表示file1是否比file2新，ot表示是否比后者旧，ef表示是否为同一文件
[rtm@worker9 ~]$ test 1 -gt 2 || echo "No"
No
# gt表示是否大于，ne表示是否不相等，类似的还有lt，eq，ge，le
[rtm@worker9 ~]$ test -z "" && echo "Yes"
Yes
# z表示字符串是否为空，n表示是否不为空，==和!=表示是否等于或不等于
[rtm@worker9 ~]$ test -z "" -a -n "1" && echo "Yes"
Yes
# a表示多条件组合and，o表示or，!表示非
```

3）[]：判断符号

```
[rtm@worker9 ~]$ [ -z "" ] && echo "Yes"
Yes
# 要注意[]内部每个组件都要用空格隔开，变量最好用双引号括起来，如[ "${name}" == "JIA" ]
```

4）if then：判断

```
if [条件判断式]; then
  指令
elif [条件判断式2]; then
  指令
else
  指令
fi
# [条件1] || [条件2]与[条件1 -o 条件2]效果一致
```

5）case esac：判断

```
case $变量 in
     "value1")
     指令
     ;; # ;;表示一个类别指令的结束
     "value2")
     指令
     ;;
     *) # *表示默认情况
     指令
     ;;
esac
```

6）function：函数

```
function fname(){
     指令
}
# 注意函数内部的$1为函数执行时的参数，不是script的参数
```

7）loop：循环

```
while [ condition ]
do
     指令
done
# while是只要条件满足就一直循环，until是除非条件满足否则一直循环
until [ condition ]
do
     指令
done
# for后面可接循环的内容数组
for var in con1 con2 cone # for i in $(seq 1 100)
do
     指令
done
# for的另一种写法，${RANDOM}生成0-32767的随机数，因此要随机1-10的话，应该${RANDOM} * 9 /
32767 + 1
for ((初始值;限制值;执行步阶)) # for (( i = 1; i <= 100; i = i + 1))
do
     指令
done
```

8）shell script的追踪与debug

```
[rtm@worker9 ~]$ sh -x hello.sh
# n表示不执行脚本仅检查是否有语法错误，v表示执行前输出脚本内容，x表示把使用到的脚本内容显示出来，并展示每一步的结果
```

三、例行性工作

1）at：仅执行一次的操作

```
# at必须保证atd这个服务开启才能执行
[rtm@worker9 ~]$ systemctl restart atd # 重启
[rtm@worker9 ~]$ systemctl enable atd # 开机启动
[rtm@worker9 ~]$ at now + 5 minutes
at> echo "hello"
at> <EOT> # ctrl + d
# l相当于atq展示所有at任务，d相当于atrm可以取消某个at任务，v表示明显的时间格式列出at任务，c后面接任务的实际指令
# at后面可直接接时间，格式可参考man at
[rtm@worker9 ~]$ atq # 展示at任务
[rtm@worker9 ~]$ atrm 数字 # 删除某个at任务
```

2）batch：系统空闲时执行操作

```
[rtm@worker9 ~]$ batch
at> echo "hello"
at> <EOT>
# 会在cpu工作负载小于0.8的时候才执行，可通过atq管理
```

3）crontab：循环执行操作

```
[rtm@worker9 ~]$ crontab -e
# e进入编辑crontab工作内容，l查阅工作内容，r移除所有工作任务，u使用某个用户进入
# @reboot sleep 60; 指令 表示开机后进行操作
```

4）anacron：唤醒停机期间的工作任务

- anacron预设以一天、七天、一个月为期去侦测系统未进行的crontab任务，然后执行它
- anacron会在每个小时都被主动执行一次

四、进程管理

1）&：后台运行

```
[rtm@worker9 ~]$ cp file1 file2 &
[1] 14432
[1]+ Done      cp file1 file2 &
# &能将整个命令变成后台运行的命令，在执行完成后才显示完成的消息
# 1为job number，14432为PID
```

2）ctrl+z：暂停并放到后台

```
[rtm@worker9 ~]$ vim ~/.bashrc
[1]+ Stopped        vim ~/.bashrc
# ctrl+z会将当前任务暂停，并放置于后台中。+代表最近一个被丢进后台的任务
```

3）jobs、fg、bg：观察后台任务工作状态

```
[rtm@worker9 ~]$ jobs -l
[1]- 170139 Stopped                 vim file1
[2]+ 170268 Stopped                 find / -print
# l表示除了列出job任务，还列出PID，r表示只列出running状态任务，s表示只列出stopped状态任务
# +表示最近被放到后台的任务，-表示最近第二个被放到后台的任务
[rtm@worker9 ~]$ fg
# 加job任务数字ID表示把job任务恢复到前台运行，不加数字则恢复+的那个任务
# 除了使用数字ID，还可以使用+或-
[rtm@worker9 ~]$ bg
# bg与fg类似，不过bg是将后台的任务变成running状态，不会提到前台中
```

4）kill：关闭后台任务或进程

```
[rtm@worker9 ~]$ kill -signal %jobnumber | PID
# l展示kill后面可以加那些signal，job任务要在数字前加%，不加%默认为PID
# -1相当于reload，-2相当于CTRL+c，-9强制删除，-15正常删除，-19相当于CTRL+Z
[rtm@worker9 ~]$ killall -signal command name
# killall会向所有以这个指令启动的进程的子进程
```

5）nohup：脱机管理

- &可以将任务放到后台进行，但是当用户注销时任务将被中断，为了保证注销后任务正常进行，使用nohup

```
[rtm@worker9 ~]$ nohup 指令 [&]
```

6）ps：查看当前进程运行情况

```
[rtm@worker9 ~]$ ps aux | head -n 3
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 193328  6244 ?        Ss   Jul03  17:38
/usr/lib/systemd/systemd --switched-root --system --deserialize 22
root           2  0.0  0.0      0     0 ?        S    Jul03   0:01 [kthreadd]
# aux查阅所有系统运作情况，-l查阅自己bash进程的运作情况，axjf查看进程树
[rtm@worker9 ~]$ ps axjf | sed '1d' | sort -k2 | awk '$1 == 162570 || $1 ==
162575 {print $0}' | head -5
162570 162575 162575 162575 pts/0    185192 Ss    1000   0:00  |       \_ -bash
162575 170268 170268 162575 pts/0    185192 T     1000   0:00  |           \_
find / -print
162575 185192 185192 162575 pts/0    185192 R+    1000   0:00  |           \_ ps
axjf
162575 185193 185192 162575 pts/0    185192 S+    1000   0:00  |           \_
sed 1d
162575 185194 185192 162575 pts/0    185192 S+    1000   0:00  |           \_
sort -k2
```

```
# 查找对应进程
[rtm@worker9 ~]$ ps aux | grep print
rtm      170268  0.0  0.0 128808  9904 pts/0    T    15:08   0:00 find / -print
rtm      185496  0.0  0.0 112708   972 pts/0    S+   16:07   0:00 grep --
color=auto print
```

7）top：动态监测进程

```
[rtm@worker9 ~]$ top -d 2
top - 16:09:58 up 75 days,  5:09,  1 user,  load average: 0.00, 0.04, 0.08
Tasks: 394 total,   1 running, 392 sleeping,   1 stopped,   0 zombie
%Cpu(s):  0.2 us,  0.1 sy,  0.0 ni, 99.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32659380 total,  2712508 free,  4644748 used, 25302124 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 25547896 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
  1399 root      20   0 3807012 100372  37168 S   6.0  0.3   3570:35 kubelet
  1395 root      20   0 3346092 107284  25356 S   1.0  0.3   1325:15 dockerd
# d后面接秒数表示更新间隔，bn表示进行几次top输出并将结果输出到目录或文件，p表示指定某个PID的任务
# top监测过程中，P表示以CPU使用率排序，N表示以PID排序，M表示以内存排序，T表示以进程使用CPU时间排序
[rtm@worker9 ~]$ top -b -n 2 > /tmp/top.txt
```

8）pstree：进程树

```
[rtm@worker9 ~]$ pstree -Aup | head -2
systemd(1)-+-NetworkManager(1073)-+-{NetworkManager}(1118)
           |                      `-{NetworkManager}(1120)
# A表示以ASCII字符连接，u表示输出进程所属用户，p表示输出PID
```

9）free：观察内存使用情况

```
[rtm@worker9 ~]$ free -m -t
              total        used        free      shared  buff/cache   available
Mem:          31893        4289        2890        1562       24714       25196
Swap:             0           0           0
Total:        31893        4289        2890
# m表示单位Mbytes，同样的还有b|k|g|h，t表示输出总量，s表示间隔几秒输出一次，c表示打印几次
```

10）netstat：网络使用情况

```
[root@master ~] netstat -tunlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 10.163.217.67:10315     0.0.0.0:*               LISTEN
 13590/kube-proxy
tcp        0      0 0.0.0.0:32267           0.0.0.0:*               LISTEN
 13590/kube-proxy
# a表示列出所有联机、监听、socket数据，t/u表示列出tcp/udp网络封包数据，n表示以端口号来显示
# l表示列出目前正在网络监听的服务，p表示列出PID
```

11）vmstat：系统资源监控

```
[root@master ~] vmstat 1 2
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 5700648   5488 34492448    0    0     0     1    0    0  0  0 99  0
 0
 0  0      0 5700848   5488 34492448    0    0     0     4 8363 11226  0  0 100
 0  0
# 统计运行状态，每秒一次，共计两次
```

12）systemctl：服务管理

```
[root@master ~] systemctl command unit
# command，start/stop/restart/enable/disable/status/is-active/is-enable
# unit，表示某个服务
[root@master ~] systemctl | head -5
  UNIT              LOAD   ACTIVE SUB       DESCRIPTION
  service1.service  loaded active plugged   PERC_H730_Mini 1
  device1.device    loaded active plugged   PERC_H730_Mini 2
  device2.device    loaded active plugged   PERC_H730_Mini
# 直接输入systemctl可以展示系统上启动的unit，list-units展示目前启动的unit，list-unit-files列出所有文件
# --type=service/socket/target表示不同类型的unit
[root@master ~] systemctl list-dependencies | head -5
default.target
• ├─abrt-ccpp.service
• ├─abrt-oops.service
• ├─abrt-vmcore.service
• ├─abrt-xorg.service
# 分析服务之间的依赖关系
```

13）rpm、yum：安装

```
[root@master ~] rpm -ivh package_name
# ivh安装某个软件并且展示安装进度，qa查询某个软件是否已经安装，e表示移除某个软件
[root@master ~] yum search raid
# yum基于分析RPM的标头资料，处理各个软件的依赖相关来下载软件，自动处理依附关系
# list列出所有yum管理的软件即版本，search查询某个关键字相关软件，info查询信息
# install安装，update更新，remove移除，clean清除所有旧数据
```