linux常用命令

1. 删除N天前最老的目录或文件

删除3天前最老的目录或文件，find时仅限搜索一层。 -- 感觉这玩意有时候不灵，不太好用。

```
find /your_dir/ -maxdepth 1 -mtime +3 | head -1 | xargs rm -rf
```

2. 解压

```
Main operation mode:
       -c, --create
              create a new archive
       -x, --extract, --get
              extract files from an archive

       -v, --verbose
              verbosely list files processed
       -f, --file=ARCHIVE
              use archive file or device ARCHIVE

       -z, --gzip
              filter the archive through gzip
       -j, --bzip2
              filter the archive through bzip2
       -J, --xz
              filter the archive through xz


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

- 解压.tar到指定目录

  ```
  tar -xvf xxx.tar.gz -C /tmp/
  ```

- 解压tar.gz到指定目录

  ```
  tar -zxvf xxx.tar.gz -C /tmp/
  ```

- 解压bz2到指定目录

  ```
  tar -jxvf xxx.tar.bz2 -C /tmp/
  ```

3. cksum

linux文件crc完整性check工具：

```
cksum [FILE]...
```

一般网上下载软件的时候都需要check crc值，来确保下载数据有没有问题，有时候即使文件大小□crc也是check不过的，下载下来的数据包有问题。

4. mount

4.1 NFS

```
mount -t nfs 192.168.2.22:/home/example/rootfs /mnt/nfs
```

4.2 cdrom

```
mount -t iso9660 /dev/scd0 /media/dvd1
mount -t iso9660 /dev/scd1 /media/dvd2
```

5. cdls

linux cd命令后自动ls目录

bash

```
alias cd='new() { cd $1; ls; }; new'
```

csh/tcsh

```
alias cd 'cd \!*; ls'

alias cd 'cd \!:^; ls'

使用时需要在前面加\
!:^ Inserts the first argument from the command line
!:1 表示第1个参数
!:2 表示第2个参数
!* 表示所有参数
!:* Inserts all of the arguments from the command line
!:x-y Inserts a range of words; from word x to word y.
!:x* Like x-$( $ is the last argument) above.
!:x- Like x* except that word$ is omitted.
```

6. 管道

格式

```
command1 | command2
command1 | command2 [ | commandN... ]
```

当在两个命令之间设置管道时，管道符|左边命令的输出就变成了右边命令的输入。只要第一个命令向标准输出写入，而第二个命令是从标准输入读取，那么这两个命令就可以形成一个管道。

7. bshell prompt

```
Sequence Value Displayed
\a ASCII bell. This makes the computer beep when it is encountered.
\d Current date in day, month, date format. For example, “Mon May
26.”
\h Hostname of the local machine minus the trailing domain name.
\H Full hostname.
\j Number of jobs running in the current shell session.
\l Name of the current terminal device.
\n A newline character.
\r A carriage return.
\s Name of the shell program.
\t Current time in 24 hour hours:minutes:seconds format.
\T Current time in 12 hour format.
\@ Current time in 12 hour AM/PM format.
\A Current time in 24 hour hours:minutes format.
\u username of the current user.
\v Version number of the shell.
\V Version and release numbers of the shell.
\w Name of the current working directory.
\W Last part of the current working directory name.
\! History number of the current command.
\# Number of commands entered during this shell session.
\$ This displays a “$” character unless we have superuser privileges.
In that case, it displays a “#” instead.
\[ Signals the start of a series of one or more non-printing characters.
This is used to embed non-printing control characters which
manipulate the terminal emulator in some way, such as moving the
159
13 – Customizing The Prompt
cursor or changing text colors.
\] Signals the end of a non-printing character sequence.
```

8. cshell prompt

```
prompt
The string which is printed before reading each command from the terminal.
prompt may include any of the following formatting sequences (+), which are
replaced by the given informa- tion:

%/ The current working directory.

%~ The current working directory, but with one's home directory represented
by '~' and other users' home directories represented by '~user' as per
Filename substitution. '~user' substi- tution happens only if the shell has
already used '~user' in a pathname in the current session.

%c[[0]n], %.[[0]n] The trailing component of the current working directory,
or n trailing components if a digit n is given. If n begins with '0', the
number of skipped components precede the trailing com- ponent(s) in the
format '/<skipped>trailing'. If the ellipsis shell variable is set, skipped
components are represented by an ellipsis so the whole becomes
'...trailing'. '~' substitution is done as in '%~' above, but the '~'
component is ignored when counting trailing compo- nents.

%C Like %c, but without '~' substitution.

%h, %!, ! The current history event number.

%M The full hostname.

%m The hostname up to the first '.'.

%S (%s) Start (stop) standout mode.

%B (%b) Start (stop) boldfacing mode.

%U (%u) Start (stop) underline mode.

%t, %@ The time of day in 12-hour AM/PM format.

%T Like '%t', but in 24-hour format (but see the ampm shell variable).

%p The 'precise' time of day in 12-hour AM/PM format, with seconds.

%P Like '%p', but in 24-hour format (but see the ampm shell variable).

\c c is parsed as in bindkey.

^c c is parsed as in bindkey.

%% A single '%'.

%n The user name.

%d The weekday in 'Day' format.
```

```
%D The day in 'dd' format.

%w The month in 'Mon' format.

%W The month in 'mm' format.

%y The year in 'yy' format.

%Y The year in 'yyyy' format.

%l The shell's tty.

%L Clears from the end of the prompt to end of the display or the end of the line.

%$ Expands the shell or environment variable name immediately after the '$'.

%# '>' (or the first character of the promptchars shell variable) for normal users, '#' (or the second character of promptchars) for the superuser.

%{string%} Includes string as a literal escape sequence. It should be used only to change terminal attributes and should not move the cursor location. This cannot be the last sequence in prompt.

%? The return code of the command executed just before the prompt.

%R In prompt2, the status of the parser. In prompt3, the corrected string. In history, the his- tory string.

'%B', '%S', '%U' and '%{string%}' are available in only eight-bit-clean shells; see the version shell variable.

The bold, standout and underline sequences are often used to distinguish a superuser shell. For example,

> set prompt = "%m [%h] %B[%@]%b [%/] you rang? "

tut [37] [2:54pm] [/usr/accts/sys] you rang? _

If '%t', '%@', '%T', '%p', or '%P' is used, and noding is not set, then print 'DING!' on the change of hour (i.e, ':00' minutes) instead of the actual time.

Set by default to '%# ' in interactive shells.
```

9. 修改terminal title

参考文件：http://tldp.org/HOWTO/Xterm-Title-4.html

使用csh时，可以在.cshrc文件中添加如下设置 set prompt = “%{\033]0;%C2\007%}%/”

其中红色部分是设置title名字为%C2绿色是命令行提示符为%/。

10. cshrc

```
set title = "%{\033]0;%C2\007%}"
set prompt = "$title%B%m %/ %b"

###################### alias #########################
# alias   ls          "ls --color"
alias   cd          "cd \!*;ls"
alias   ll          "ls -l"
alias   lrt         "ls -lrt"
alias   lrs         "ls -lrSh"
alias   la          "ls -a"
alias   lla         "ls -l -a"
alias   hh          "history"
alias   h           "history 30"

alias   .           "pwd"
alias   ..          "cd .."
alias   g           "gvim"
```

11. 不同机器之间手动同步时间

```
/usr/sbin/ntpdate 192.168.0.115
```

12. sort

```
1. sort   [选项参数]   [-o<输出文件>]   [-t<分隔字符>]   [+<起始栏位>   -<结束栏位>]   [文件]
2.
3. -c ===> 检查文件是否已经按照顺序排序
4.
5. -b ===> 忽略每行前面开始处的空格字符
6. -i ===> 排序时，除了040至176之间的ASCII字符外，忽略其他的字符
7. -d ===> 排序时，处理英文字母、数字及空格字符外，忽略其他的字符
8. -f ===> 排序时，将小写字母视为大写字母
9. -M ===> 将前面3个字母依照月份的缩写进行排序
10. -n ===> 依照数值的大小排序
11. -g ===> 按通用数值排序，支持科学计数法
12. -r ===> 以相反的顺序来排序
13.
14. -u ===> 对排序后的数据去重
```

```
15. -m ===> 将几个排序好的文件进行合并
16.
17. -t ===> <分隔字符> 指定排序时所用的栏位分隔字符
18. -k ===> POS1[,POS2] （配合-t，排序从POS1开始，若指定POS2，则POS2结束，否则以pos1排序
19.  + ===> <起始栏位> -<结束栏位> 以指定的栏位来排序，范围由起始栏位到结束栏位的前一栏位（从0开始）
20. -o ===> <输出文件> 将排序后的结果存入指定的文件
21. ————————————————
22. 版权声明：本文为CSDN博主「IT蓝月」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。
23. 原文链接：https://blog.csdn.net/ITlanyue/article/details/105013828
```

13. tail

tail - output the last part of files

```
tail -n 20 file; # 输出文件的最后20行
tail -n +20 file; # 从文件的第20行开始输出
```

14. head

head - output the first part of files

```
head -n 20 file; # 输出文件的前20行内容
```

15. sed

15.1 基本用法

sed 常用命令

```
sed -i 's/regexp/replace' xxx.txt;    替换文件内容，并修改文件
sed -n 'p' xxx.txt    ;    # 全部打印输出文件内容
sed 's/regexp/replace/' xxx.txt;   将文件内容替换打印输出，但不修改文件。
sed -n '1,3p' xxx.txt;     #只打印输出文件的1到3行
sed -n '2p' xxx.txt;     #只打印输出文件第2行
```

```
sed 's/regexp/replacement/' distros.txt

# regexp expand方式的匹配，可以理解为ERE
sed -r 's/(regexp)/new_\1' distros.txt

#和上面效果一样，是普通的regexp，可以理解为BRE
```

```
sed 's/\(regexp\)/new_\1/' distros.txt


# -i 将替换后的结果写回原文件中
sed -i 's/\(regexp\)/new_\1/' distros.txt
```

对某个字符串进行替换

```
[me@linuxbox ~]$ echo "front" | sed 's/front/back/'
back
```

15.2 多行处理

这个是高级用法

- N -- 将数据流的下一行也加入到模式匹配中， 当前行与下一行包含一个换行符 \n
- D -- 删除多行中的一行
- P -- 打印多行中的一行

```
sed '/ee/N;   s/\n/ /' tmp.txt;   -- 按行匹配到ee后，读出下一行内容形成两行匹配，将其中的换行符替换为 空格。
```

16. basename

```
# csh
set a = '/home/xxx'
set name = `basename $a`
echo $name
> xxx
```

17. cut

cut是一个简单功能的字段截取工具，推荐使用更强功能的awk awk语法速查

```
cut OPTION... [FILE]...
-c 仅显示行中指定范围的字符；
-d 指定字段的分隔符，默认的字段分隔符为“TAB”
-f 显示指定字段的内容；
```

```
# cat test.txt
No Name Mark Percent
01 tom 69 91
02 jack 71 87
03 alex 68 98

# 截取字符1到4
# cut -c1,4 test.txt
```

```
No N
01 t
02 j
03 a


#打印第一列数据
# cut -f 1 test.txt
No
01
02
03

#打印第2，3列数据
# cut -f2,3 test.txt
Name Mark
tom 69
jack 71
alex 68


# cat test2.txt
No;Name;Mark;Percent
01;tom;69;91
02;jack;71;87
03;alex;68;98

#指定;符号为分隔符，只能指定单个符号为分隔符
# cut -f2 -d";" test2.txt
Name
tom
jack
alex
```

18. wget批量下载文件

使用 wget 完成批量下载

如果想下载一个网站上目录中的所有文件, 我需要执行一长串wget命令, 但这样做会更好:

wget -nd -r -l1 --no-parent http://www.foo.com/mp3/

这条命令可以执行的很好, 但有时会下载像 index.@xx 这样一些我不想要的文件. 如果你知道想要文件的格式, 可以用下面的命令来避免下载那些多余的文件:

wget -nd -r -l1 --no-parent -A.mp3 -A.wma http://www.foo.com/mp3/

我来简单的介绍一下命令中指定选项的作用.

-nd 不创建目录, wget默认会创建一个目录

-r 递归下载

-l1 (L one) 递归一层,只下载指定文件夹中的内容, 不下载下一级目录中的.

–no-parent 不下载父目录中的文件

19. iconv 文件编码转换

```
 iconv - convert text from one character encoding to another
SYNOPSIS
       iconv [options] [-f from-encoding] [-t to-encoding]
       [inputfile]...
```