

# Linux常用命令简介

# man

- 使用方式: `man XXX`
- 查看命令的帮助信息，遇到不会的命令时请优先使用这个
- 另一个查看帮助的方法是每个命令都有的  
`xxx --help`

# 管道

- 用于把一个命令的输出作为下一个命令的输入
- 示例
  - `cmd1 *.txt | cmd2 | cmd3`

# 输入输出重定向

- 把输入或输出定向到另一个地方
  - > 输出重定向到一个文件或设备 覆盖原来的文件
  - >> 输出重定向到一个文件或设备 追加原来的文件
  - < 输入重定向到一个程序
- 示例：
  - `ls -l > ls_result` 把ls的结果存在ls\_result文件中
  - `ls -l >> ls_result` 把ls的结果追加到ls\_result文件中
  - `mysql -uroot < test.sql` 在mysql控制台中执行test.sql中的语句
  - `./test > all_result 2 > &1` 把test程序的stderr输出重定向到stdout中，然后都存储到all\_result文件中

# date

- 获取时间的命令
- 常用用法：
  - 格式化输出: `date +%Y%m%d` -> 20160731
  - 自定义时间: `date -d "1 days ago"`

# cat

- 连接文件，并输出到stdout
- 使用方法： `cat file_name1 file_name2 ...`
- 示例：
  - `cat file1 file2 > file3` 连接file 1和file 2，并输出到file 3中
  - `cat file1 >> file2` 把文件1的内容追加到文件2中

# head和tail

- head
  - 取出文件的头部数据
  - 用法示例：
    - `head -n 10 data.txt` 从data.txt中取出前10行数据并输出到stdout中
- tail
  - 取出文件尾部数据
  - 用法：
    - `tail -n 10 data.txt` 从data.txt中取出最后10行数据并输出到stdout中
    - `tail -f data.txt` 持续监控data.txt，一旦data.txt尾部写入了新数据，则将新数据输出到stdout中，否则等待data.txt更新

# more和less

- more
  - 类似于cat，一页一页地展示文件内容，按enter或者pagedown翻页，适用于较大的文本文件
  - `hadoop fs -cat /user/large.txt | more`
- less
  - 和more功能类似，但是more只能向下翻，less可以向上回滚



# sort

- 给文本文件排序
- 命令参数: `sort [-arg]... [file]...`
  - `-t` 指定分隔符, 如果分隔符是 `\t`, 需要写成 `-t$'\t'`
  - `-kn, m[参数]`, 把第 `n` 列到第 `m` 列的数据按照[参数]中的方法来排列, 默认是字典序增序拍
    - `n` 按照整数排列 (不支持小数)
    - `g` 按照通用的计数方法来拍 (支持小数)
    - `r` 按照递减的顺序来排
  - 示例:
    - `sort -t'#' -k3r -k1,2n data.txt` 把 `data.txt` 中的数据按照 `#` 分隔后, 前两列按照数字顺序, 先按第三列按减序排列, 再按第一和第二列按照数字增序排列

# find

- 用途：查找文件
- 常用用法：
  - `find path -name fn` 在路径`path`下查找名为`fn`的文件，并将找到的路径输出到`stdout`
  - `find path -name fn -exec cmd {} \;` 对找到的每个文件执行`cmd`命令

# grep

- 查找输入中符合条件的字符串
- 用法：
  - `grep 'pattern' filename` 在文件filename中查找符合pattern的所有字符串
  - 加上-E参数，可支持正则表达式

# awk

- 处理文本的神器
- 基本用法
  - `awk -F' ' '{expr}' file_path`
    - -F 后面跟着分隔符，把每一行按分隔符分为多列
    - 对读取的每一行数据执行 `expr`
      - 支持条件，循环等语句
      - 内置变量：\$0 表示整行的文本，\$i 表示第i列的数据，NF 表示该行的列数，NR 表示已读取的行数
    - 示例：  
`cat file1 file2 | awk -F'\t' '{if (length($1) > 5 && NF >= 2 ) print $2}'`

# awk

- 带初始化和结束

- `awk -F" 'BEGIN{expr1}{expr2}END{expr3}' file`

- 示例：

- 统计一组正整数中最大的数

- ```
awk 'BEGIN{max=0}{if ($0 > max)
max=$0}END{print max}'
```

# crontab

- 定时运行任务
  - 使用说明：
    - `crontab -e` 编辑crontab表，每个用户的crontab表都不同
    - 格式
- #分 时 日 月 周 |任务的完整命令行
- \* \* \* \* \* /home/user/cmd.sh
- 举例：
- \* /5 \* \* \* \* 每5分钟运行一次
- 0 2 1 4 \* 每年4月1日凌晨2点运行
- 0 14 \* \* 5,6 每个星期五或星期六的下午2点

# crontab

- 需要注意的地方

- 谨慎地设置时间

- \* 2 \* \* \*

- crontab的环境变量与系统的不同

- \* \* \* \* \* source ~/.bashrc; /home/user/cmd.sh

# top

- 查看系统资源的使用情况

```
cjhon2 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
cjhon2 x
top - 02:37:25 up 4 days, 14:45, 1 user, load average: 0.00, 0.01, 0.05
Tasks: 138 total, 1 running, 137 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.1 us, 0.1 sy, 0.0 ni, 99.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 16269468 total, 3037724 used, 13231744 free, 204912 buffers
KiB Swap: 0 total, 0 used, 0 free. 2084208 cached Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
 1102 root        20   0   64820   6064   4032  S   0.7   0.0   20:40.29 AliYunDun
    48 root        39  19     0     0     0  S   0.3   0.0    0:04.23 khugepaged
  1116 root        20   0  836536  10692   6780  S   0.3   0.1    9:03.59 AliHids
28998 mysql       20   0 2628044 337236   9108  S   0.3   2.1   12:22.45 mysqld
    1 root        20   0   49692   3844   2316  S   0.0   0.0    0:16.81 systemd
    2 root        20   0     0     0     0  S   0.0   0.0    0:00.02 kthreadd
    3 root        20   0     0     0     0  S   0.0   0.0    0:00.70 ksoftirqd/0
    5 root         0 -20     0     0     0  S   0.0   0.0    0:00.00 kworker/0:0H
    6 root        20   0     0     0     0  S   0.0   0.0    0:00.00 kworker/u8:0
    7 root        rt    0     0     0     0  S   0.0   0.0    0:00.12 migration/0
    8 root        20   0     0     0     0  S   0.0   0.0    0:00.00 rcu_bh
    9 root        20   0     0     0     0  S   0.0   0.0    0:00.00 rcuob/0
   10 root        20   0     0     0     0  S   0.0   0.0    0:00.00 rcuob/1
   11 root        20   0     0     0     0  S   0.0   0.0    0:00.00 rcuob/2
   12 root        20   0     0     0     0  S   0.0   0.0    0:00.00 rcuob/3
   13 root        20   0     0     0     0  S   0.0   0.0   2:54.77 rcu_sched
   14 root        20   0     0     0     0  S   0.0   0.0   1:21.06 rcuos/0
   15 root        20   0     0     0     0  S   0.0   0.0   0:57.97 rcuos/1
   16 root        20   0     0     0     0  S   0.0   0.0   0:57.95 rcuos/2
```

Ready ssh2: AES-256-CTR 26, 97 26 Rows, 97 Cols Linux CAP NUM



# ps

- 查看当前运行的进程的信息
- 常用用法：
  - `ps axu | grep httpd` 查找所有进程中进程包含  
httpd的进程

# kill

- 向指定的进程发送信号
- 常用用法：
  - `Kill -SIGKILL pid` 强行杀死进程号为pid的进程

# xargs

- 把输入的每一行作为命令的参数分别执行
- 一个例子：
  - `ps axu | grep java | awk -F' ' { print $2 } | xargs kill`

# chmod

- 改变文件的权限
- 文件的权限：
  - r: 读; w: 写; x: 执行
  - 本用户 : 用户所在组的其他用户: 其他用户

```
[root@iZ25u46ftbdZ zabbix]# ll -tr
total 36
-rw-r----- 1 root    zabbix 14882 Jul 26 20:05 zabbix_server.conf
drwxr-x--- 2 apache  apache  4096 Jul 26 21:18 web
drwxr-xr-x 2 root    root     4096 Jul 26 21:28 zabbix_agentd.d
-rw-r--r-- 1 root    root    10340 Jul 28 23:41 zabbix_agentd.conf
```

- 用法：
  - chmod +x test.sh
  - chmod -r test.txt
  - chmod 644 key

# nohup

- 让程序在后台继续运行
  - 原理：忽略SIGHUP信号
- 用法：
  - `nohup cmd & > log 2>&1`
  - 不重定向则把输出默认追加到nohup.out中

# screen

- 命令行终端切换
  - 也可以用来在后台持续运行程序
- 用法：
  - 创建新的screen会话: `screen -S name`
  - 脱离当前screen会话: `Ctrl + A`, 再按D
  - 恢复之前创建的screen: `screen -r name`
  - 创建一个新的screen会话: `Ctrl+A`, 再按C
  - 跳转到第n号screen会话: `Ctrl+A`, 再按相应的数字  
n
  - 关闭当前的screen会话: `Ctrl+A`, 再按D
  - 查看当前系统所有运行着的screen: `screen -ls`

# 练习

- 写个定时任务，每10分钟执行一次，找到当前CPU使用率和内存使用率最高（先排CPU后排内存）的前10个进程，记录用户名，进程id，CPU和内存使用率，追加写入到一个文件中，每次执行时打上时间戳