Linux命令

Linux基础命令

- 参考文章： Linux Tools Quick Tutorial

命令帮助

1. whatis
2. info
3. man
4. which
5. whereis

文件及目录管理

1. mkdir、rm、mv、cp

```
cp source_file dest_file
cp -r source_dir  dest_dir

mv source_file(文件) dest_directory(目录)
```

2. cd、pwd

```
切换到上一个工作目录
cd -
切换到home目录
cd cd!
```

3. ls

```
查看隐藏文件
ls -a
按时间排序，以列表的方式显示目录项
ls -lrt

9个字母为该文件的权限标识，3个为一组，分别表示文件所属用户、用户所在组、其它用户的读写和执行权限
eg:-rwxrw-r-- colin king 725 2013-11-12 15:37 /home/colin/a
ls-l
```

4. alias

```
alias lsl='ls -lrt'
```

5. find
   - 实时查找
6. locate
   - 更快的查询，为文件系统建立索引数据库

```
更新数据库，获得最新的文件索引信息
updatedb
寻找包含有string的路径
locate string
```

7. cat

```
显式行号
cat -n

```

8. head、tail

```
显示文件前10行
head - 10 filename
显示文件第一行
head -1 filename
显示文件倒数第五行
tail -5 filename
```

9. diff

```
diff file1 file2
```

10. more
11. egrep
12. chown、chmod、chgrp
    - 文件权限：rwxrwxrwx；当前用户权限｜用户组权限｜额外的其他用户权限
    - chmod -R 777
    - rwx->111->7

```
改变文件拥有者
chown
改变文件读、写、执行属性
chomd
```

13. |;&&||

```
xxx1成功，xxx2执行；xxx1失败，xxx2不执行
```

```
xxx1 && xxx2
xxx1失败，xxx2执行
xxx1 || xxx2
```

文本处理（待补充）

- Linux 三剑客
- grep：适合用于单纯的查找与匹配
- awk：适合对文本进行复杂的格式化处理
- sed：适合修改匹配到的文本

磁盘管理

1. df（disk free）：显示所有文件系统

```
human：以易读的方式显示结果
df -h

```

2. du（disk used）：显示目录或文件的大小

```
-s 递归整个目录的大小
du -sh
```

3. free
4. 查找大文件

```
1、查询服务器中大于1G的文件
find / -type f -size +1G
2、查询服务器中大于1G的文件及属性信息
find / -type f -size +1G  -print0 | xargs -0 ls -l
3、查询大文件并排序
find / -type f -size +1G  -print0 | xargs -0 du -h | sort -nr
```

    - 文章

5. tar、gzip
    - 打包生成的是.tar
    - 压缩生成的是.gz/.bz2/xz

```
打包
-c打包选项，-v显示打包进度，-f使用档案文件，
-x解包选项
-z解压gz文件
-i解压bz2文件
-j解压xz文件
tar -cvf
解包
tar -xvf xxx.tar
```

```
10
11
12
13 压缩
14 gzip xxx.txt
15 解压缩gz
16 gunzip demo.tar.gz
17
18 解压缩bz2
19 tar jxvf demo.tar.bz2
20 =（如果tar不支持j选项的话）
21 bzip2 -d demo.tar.bz2
22 tar -xvf demo.tar
```

进程管理

1. ps

```
1 查询正在运行的进程信息
2 ps -ef：标准格式显示
3 ps -aux：bsd格式显示
```

2. top：显示进程信息
    - top命令如何实现的
3. perf
    - perf top：实时显示占用CPU时钟最多的函数或者指令，因此可以用来查找热点函数
    - perf record 提供了保存数据的功能，保存后的数据，
    - perf report 来解析perf record展示
4. lsof（list open files）：一切皆文件
    - 文章：lsof 一切皆文件
    - 实例

```
1 列出系统所有打开的文件
2 lsof | more
3
4 查找某个文件相关的进程
```

5. kill
    - kill -2：类似于Ctrl+C，程序在结束之前，保存相关数据，然后再退出
    - kill -9：直接强制结束程序
    - nohub挂起后，想要结束程序最好使用kill -2
6. pmap

```
1 输出进程内容状况，用来分析线程堆栈
2 pmap PID
```

7. linux如何后台运行进程，而且不随终端关闭而关闭

```
&（后台执行）：加在一个命令的最后，可以把这个命令放到后台执行
nohub（即使终端被关还是继续执行）：让程序始终在后台执行，即使关闭当前的终端也执行（之前的&做不到）
jobs：查看当前终端有多少在后台运行的命令
fg：将后台中的命令调至前台继续运行
bg：将一个在后台暂停的命令，变成在后台继续执行。
```

8. 查看线程数的几种方法

```
top -H：查看系统中的总线程数
pstreee -p pid |wc -l：查看系统中的总线程数
cat /proc/pid/status：
ll /proc/pid/task/|wc -l
```

9. 查看某个进程的线程

```
ps -T -p <pid>：由进程号为<pid>的进程创建的所有线程
top -H -p <pid>
htop
```

性能监控

1. sar
2. free
3. watch
4. vmstat n m

网络工具

1. netstat

```
netstat -a：all，显示所有选项
-t：tcp
-u：udp
-n：拒绝显示别名
netstat -at
```

   a. 查找某个具体的端口号
      - netstat -a | grep 3306
      - lsof -i:8000
   b. netstat -an|awk '/tcp/ {print $6}'|sort|uniq -c
2. route
3. ping
4. traceroute

5. host
6. wget
7. ssh/sftplftp
8. scp 网络复制
9. iptables
    - NAT

```
1 iptables [-t tables] [-L] [-nv]
2 -t : 后面接 table ，例如 nat 或 filter ，若省略此项目，则使用默认的 filter
3 -L : 列出某个 table 的所有链或某个链的规则
4 -n : 直接显示 IP，速度会快很多
5 -v : 列出更多的信息，包括通过该规则的数据包总位数、相关的网络接口等
6
7 $ sudo iptables -t nat -L -n
```

用户管理工具

1. useradd -m username
2. userdel -r username
3. sud username
4. groups usermod
5. chmod
6. chown
7. chgrp
8. 环境变量
    - bashrc与profile都用于保存用户的环境信息，
    - bashrc用于交互式non-loginshell
    - 而profile用于交互式login shell。
    - /etc/profile，/etc/bashrc 是系统全局环境变量设定
    - ~/.profile，~/.bashrc用户目录下的私有环境变量设定
    - 读取环境设置脚本步骤：
        - 首先读入的是全局环境变量设置文件/etc/profile，然后根据其内容读取额外的文档，如/etc/profile.d和/etc/inputrc
        - 读取当前登录用户Home目录下的文件~/.bash_profile，其次读取~/.bash_login，最后读取~/.profile，这三个文档设定基本上是一样的，读取有优先关系
        - 读取~/.bashrc
    - ~/.profile与~/.bashrc的区别
        - 这两者都具有个性化定制功能
        - ~/.profile可以设定本用户专有的路径，环境变量，等，它只能登入的时候执行一次
        - ~/.bashrc也是某用户专有设定文档，可以设定路径，命令别名，每次shell script的执行都会使用它一次

系统资源以及IPC资源管理

1. uname
2. sar
3. arch显示架构
4. date显示当前系统时间
5. ipcs查看系统使用的IPC资源
6. ulimit

GDB

- 参考文章：
  - 程序调试GDB
  - gdb 调试利器

```
g++ -g hello.cpp -o hello
gdb <program>
```

1. GDB交互命令
   - run：运行程序
   - continue：继续执行
   - next：单步跟踪程序
   - step：单步跟踪，next进入函数，step不进去函数内部
   - until：退出循环体
   - until + 行号：
   - finish：运行程序
   - call 函数名字(参数)：调用函数
   - quit：退出gdb
   - break 行号/函数名：设置断点
   - delete 断点号：删除断点
   - list 行号/函数名：查看源代码
   - print 表达式/字符串/变量值：打印表达式

   - pstack：跟踪栈空间
   - strace：分析系统调用
   - backtrace（打印函数调用栈的信息）
   - frame n（没有n查看当前栈帧的信息，有n打印指定栈帧）
2. 如果程序崩溃的话，怎么使用GDB调试?
   - 参考

```
./a.test
run 执行文件（可以看到崩溃的提示信息）
```

```
where 查看可能出错的位置
list 查看附近代码
break 设置断点
run 重新运行程序
print 打印变量的值
```

core dump

- 从core文件中分析原因，通过gdb看出程序挂再哪里，分析前后的变量，找出问题的原因
- 文章

Valgrind

- 检查内存泄漏工具
- memwatch