目录

Linux shell

Shell Start

shell 的启动依赖于用户账户的配置 `/etc/passwd`
用户条目： `gouchao:x:1000:1000:gouchao,,,:/home/gouchao:/bin/bash`
这些字段包括：

- 用户名
- 密码（仅仅是占位符）
- 用户UID
- 用户GID
- 用户信息
- 用户目录
- 用户默认shell程序

> 默认情况下Bash Shell 会自动处理用户目录下的.bashrc文件，但是该文件又加载了特殊的公用文件，这些公用文件放在 `/etc/bashrc` 目录下
> 默认的Bash提示符是 `$`

Ubuntu的BashShell提示符是

```
gc@gc:~$
```

- 启动shell的用户名
- 当前用户的主机名
- 当前目录 ~

两个环境变量是控制提示符格式的: PS1 ， PS2

```
gouchao@gouchao:~$ echo $PS1
\[\e]0;\u@\h: \w\a\]${debian_chroot:+($debian_chroot)}\u@\h:\w\$
gouchao@gouchao:~$ echo $PS2
>
gouchao@gouchao:~$
```

特殊提示符：

- \a 报警符
- \d 日 月 年 格式显示日期
- \e ASCII转义字符
- \h 本地主机名
- \H 完全限定域名
- \j shell当前管理的任务数
- \l shell的终端设备名中的基名
- \n ASCII换行符
- \r ASCII回车符
- \s Shell的名称
- \t HH:MM:SS \24 当前时间
- \T HH:MM:SS \12 当前时间
- \@ am / pm \12 当前时间
- \u 当前用户的用户名
- \v Base Shell的版本
- \V Bash Shell的发行版
- \w 当前工作目录
- \W 当前工作目录的基名
- $ 普通用户$ root用户#
- \\ 反斜线
- \[开始一个控制字符的序列
- \] 结束一个控制字符的序列

> 注意，设置PS1的时候不能插入空格，不然shell会以cmd的方式解读

```
gouchao@gouchao:~$ PS1 = [\t]\$
PS1：未找到命令
gouchao@gouchao:~$ PS1="[\t]\$"
[11:22:25]$
```

查询命令的相关信息用 `man + command` 的方式，比如 `man ls` 查询ls的相关信息

File

常见的Linux 虚拟目录名称:

- `/` 根目录
- `/bin` 二进制目录，用来存储GNU用户级的工具
- `/boot` 启动目录，用来放启动文件
- `/dev` 设备目录
- `/etc` 系统配置文件目录
- `/home` 用户主目录
- `/lib` 库目录，存放系统和应用程序的库
- `/media` 媒体目录，存放移动设备的挂载点
- `/mnt` 另一个存放移动设备挂载点的地方
- `/opt` 存放可选的包
- `/root` 跟目录
- `/sbin` 存放GNU管理员级的工具
- `/tmp` 临时文件的目录
- `/usr` 用户安装软件的目录
- `/var` 可变文件目录，如Log

file命令：

- cd
- ls
    - -F 显示出文件还是目录
    - -a
    - -R 递归显示文件
    - -l 显示文件信息
        - 文件类型 ： d目录，- 文件，c 字符型文件，b 块文件
        - 权限
        - 硬链接总数
        - 属主用户名
        - 属组组名

- 文件大小：字节
- 上次修改的时间
- 名

ll ep:

```
drwxrwxr-x  5 gouchao gouchao      4096  4月 22 10:42 wor
kspace
drwxr-xr-x  2 gouchao gouchao      4096  4月 12 15:53 公共
的
drwxr-xr-x  2 gouchao gouchao      4096  4月 12 15:53 模板
drwxr-xr-x  2 gouchao gouchao      4096  4月 12 15:53 视频
drwxr-xr-x  3 gouchao gouchao      4096  4月 27 18:03 图片
drwxr-xr-x  5 gouchao gouchao      4096  4月 27 16:32 文档
drwxr-xr-x  8 gouchao gouchao      4096  4月 27 17:05 下载
drwxr-xr-x  2 gouchao gouchao      4096  4月 12 15:53 音乐
drwxr-xr-x  2 gouchao gouchao      4096  4月 13 14:23 桌面
```

用ls指定过滤文件：

ls -l xxxxx

过滤器识别标准统配符：

- ? 一个字符
- * 代表零个或多个字符
-

- mkdir 创建目录
- rmdir 删除目录
- touch 创建文件
- cp
    - -a 归档文件，保留现有属性
    - -b 创建已经存在目标文件备份并且覆盖它
    - -d 保留
    - -f 强制覆盖，不提示
    - -i 提示
    - -l 创建连接不复制
        - 默认创建硬链接，创建出来的连接都指向同一个文件：索引相同
        - 修改其中一个文件，另一个文件也修改
        - 不能在不同挂载点间创建硬链接，可以创建软连接
    - -p 复制文件，尽可能的保留文件属性
    - -r 地柜复制文件
    - -R 递归复制目录

    - -u 源文件比当前文件新则覆盖
    - -v 详细模式
    - -s 创建软连接
        - 软连接是真是的创建出了一个文件，但是文件里仅仅存储了源文件的信息，不存储数据

> 在复制连接的时候需要注意，当cp一个链接到另外一个源文件中的时候复制的是副本而不是链接，如果想复制链接，直接创建一个源文件的链接即可，切记不要轻易的创建链接链，不但容易断，还容易出现各种bug

- mv
    - mv文件最后两个文件保存相同的索引节点
    - 不要mv软链接，不然会造成 指针指向非法内存
    - mv可以移动目录，不用加参数，mv要比cp块的多
- rm
    - 删除文件之后，软连接变成了指向无效文件，硬链接还能访问到，因为硬链接指向的是索引节点
- stat filename 查看文件的详细信息
- file 显示文件类型
    - 文本文件，包含可以打印字符的文件
    - 可执行文件
    - 数据文件 包含不可打印的二进制文件，又不能在系统上运行的文件
- cat
    - -n 加上行号
    - -s 压缩空白行
    - -T 取消制表符
- more
    - q 退出
    - Enter 下一行
    - space 下一屏
    - b 上一屏
    - / expression 根据表达式查询本页
    - n 查找下一处
    - v 在当前行启动vim
    - = 显示当前行在文件中的行号
    - . 重复前一个命令
- less
    - less明显比cat more要高级（less is more）
    - less不用读取整个文件（这点就比cat more要好）

- tail 默认显示文件后10行
  - -n 显示文件后几行
  - -f 让tail一直运行，当有新的内容添加到文件末尾，就显示出来

- head
  - -n

系统管理命令

- ps
    - 默认显示当前控制台下属于当前用户的进行
    - -A 显示所有进程
    - -a 显示除控制进程以及无终端的进程外所有进程
    - -d 显示出控制进程外所有的进程
    - -e 显示所有进程
    - -f 完整格式的输出
    - -l 长格式
        - S 进程状态
            - S 休眠
            - R 可运行
            - 0 正在运行
            - Z 僵化

    - -ef

- top 实时监控
    - NI 进程谦让读
    - PR 优先级
    - VIRT 虚拟内存
    - RES 真是内存
    - SHR 共享内存

- kill PID
    - -s + 信号 + PID
        - HUP
        - INT
        - QUIT
        - KILL
        - TERM
        - STOP

    - killall + commandname（可以使用通配符）

检测磁盘空间

- mount 挂载媒体，默认情况下会输出当前系统上挂在的设备列表

    - 媒体设备文件名
    - 虚拟目录挂载点
    - 文件系统类型
    - 访问状态
    - `mount -t type device directory` root身份手动挂载设备
        - type指定文件类型
            - vfat windows长文件（大多数windows格式化的文件系统）
            - ntfs windows高级文件系统
            - iso9660 CD-ROM文件系统
            - 存储设备的设备文件位置
            - 挂载点在虚拟目录中位置
    - -o loop 可以挂在一个.iso到系统中

- umount [directory|device] 取消挂在
- lsof + name|path 查看占用设备的进程

> 如果umount失败，可能有其他进程占用设备，先通过lsof来查看占用的进程

- df 查看剩余磁盘空间（kb：或者说1024字节的块）
    - -h 以人类可读的方式显示
- du 显示目录磁盘使用情况，列出所有的目录文件和子文件
- sort 对数据进行排序
    - -n 讲数字识别成数字，而不是字符
    - -M 将时间识别成时间，而不是字符
    - -r 反序排序
- grep 在文件中查询一行
    - -v 反向搜索
        - `grep -v t filename` 在filename文件中查询 不 包含t的行
    - -n 显示行号
    - -c 知道有多少行包含
    - -e 指定多个模式，得到其中一个结果即可
        - `grep -e t -e f filename`
        - 默认grep采用linux风格的正则表达式，unix的正则表达式需要加特殊字符
            - `grep [tf] filename`
- bzip2 压缩文件
    - 压缩文件之后替换原来的文件并且以。bz2结尾
    - bzip2压缩之后的文件可以用bzcat查看，但是不能用cat查看
- bunzip2 解压文件

- gzip
- gzcat
- gunzip
    - .gz

- zip
    - -r 将整个目录压缩进文件
        - `zip -r toname fromname`

- zipcloak 加锁
- unzip
- tar
    - -c 创建一个新的归档文件
    - -t 列出已有tar归档文件中的内容
    - -z 输出重定向给gzip来压缩内容
    - -v 处理文件时候显示文件
    - -f 输出结果或文件到file
    - `tar -cvf test.tar test/ test2/`
    - `tar -tf test.tar`
    - `tar -xvf test.tar`
    - `tar -zxvf filename.tgz`

系统环境变量

- printenv 打印全局变量
- export + 局部变量名 ： 将局部变量到出到全局变量里
- unset 删除一个环境变量

登陆系统时候，系统会当做登陆shell进行shell 启动。登陆shell会在四个不同的启动文件中读取命令

- /etc/profile
    - 系统启动时候必须要执行的文件
    - 下面三个是用户自己定制的

- $HOME/.bash_profile
- $HOME/.bash_login
- $HOME/.profile

Linux文件权限

/etc/shadow文件中共有9条记录

- 与/etc/passwd对应的用户名

- 加密后的密码
- 自1997年1月1日（上次修改密码的日期）至当天的天数
- 多少天后才能修改密码
- 多少天后必须修改密码
- 密码过期提前多少天提醒用户修改密码
- 密码过期后多少天后禁用用户密码
- 用户被禁用的日期
- 预留字段，留给将来使用

添加用户 useradd

- -D 查看创建系统用户的默认值
    - ```
      gouchao@gouchao:~$ useradd -D
      GROUP=100 //组
      HOME=/home //用户目录放在这个目录下
      INACTIVE=-1 //是否过期禁用
      EXPIRE= //多久过期
      SHELL=/bin/sh //shell
      SKEL=/etc/skel //将/etc/skel内容复制到home目录下
      CREATE_MAIL_SPOOL=no //在mail目录下创建一个用于接受mail的文件
      gouchao@gouchao:~$
      ```

> 默认情况下 useradd 不会创建home文件，需要加-m 参数就会创建了，并且将/etc/skel目录下的文件全部复制过来

下面创建用户时候可以添加的参数

- -c commend 给新用户添加备注
- -d 为主目录指定一个名字
- -e 以YYYY-MM-DD指定一个过期的日期
- -f 指定过期之后多久被禁用
    - 0 立即被禁用
    - -1 没有这个功能
- -g 指定登陆组名或GID
- -G 指定附加组
- -k 必须跟-m一起使用，将/etc/skel内容复制到home里
- -m 创建用户home目录
- -M 不创建用户home目录
- -r 创建系统用户
- -p 指定密码
- -s 指定shell
- -u 指定UID
      ```

删除用户 `userdel`

- userdel -r gc 删除gc用户并且删除用户目录，不加-r只会删除/etc/passwd中的记录

修改用户信息

- usermod 修改用户账户字段，并且指定主要组以及附加组的关系
    - -c 修改备注字段
    - -e 修改过期日期
    - -g 修改登陆组
    - -l 修改账户的登录名
    - -L 锁定用户
    - -p 修改账户密码
    - -U 解锁用户
- passwd 修改用户密码
    - -e 强中用户下次登陆时修改密码
- chapasswd 从文件中读取登录名密码对。并且更新密码
    - 读取格式为 `username:passwd` 的密码对文件进行修改

```
gouchao@gouchao:~$ sudo chpasswd < /home/gc/a.txt
[sudo] password for gouchao:
gouchao@gouchao:~$ su gc
密码：
gc@gouchao:/home/gouchao$ cd
gc@gouchao:~$ cat a.txt
gc:123456
gc@gouchao:~$
```

- change 修改密码过期日期
    - -d 设置上次修改密码到现在的天数
    - -E 设置密码过期日期
    - -I 设置密码过期到用户锁定的天数
    - -m 设置修改密码之间间隔多少天
    - -W 设置密码过期前多久出现提示
- chfn 修改用户备注信息
- chsh 修改用户默认登陆的shell

> 这里的时间可以用YYYY-MM-DD来设置

组

`/etc/group` 中显示组信息

- 组名
- 组密码
- 组Id
- 属于该组的用户列表
- `groupadd` 来创建新组

理解文件权限

`drwxrwxr-x` 解释

- 第一个字符
    -
        - 文件
    - d 目录
    - l 连接
    - c 字符型设备
    - b 块设备
    - n 网络设备
- 接下来是三组字符码，每一组都是三位，每一组都是表示三重访问权限
    - 对象属性
    - 对象组属性
    - 系统其他用户
- umask是默认权限值的掩码
    - 如果umask = 002

> 默认文件最大权限是666（读写），掩码是002，666-002=664 -> rw-rw-r–
>
> ```
> gouchao@gouchao:~$ umask
> 0002
> gouchao@gouchao:~$ touch a
> gouchao@gouchao:~$ ls -l a
> -rw-rw-r-- 1 gouchao gouchao 0 5月 9 12:02 a
> gouchao@gouchao:~$
> ```

- 更改权限 `chmod`
    - `chmod 777 a`
    - `chmod a+x a`
        - a+x

        - 第一位
            - u 用户
            - g 组
            - o 其他
            - a 全部
        - 第二位
            - + 添加
            - - 删除
            - = 赋值
        - 第三位
            - 权限
- 更改文件所属关系
    - chown
        - `chowm gouchao a`
        - `sudo chown gouchao.igouc a` 同时改变owner与group
    - chgrp
        - `chgrp igouc a`

安装软件包

dpkg命令是Debian系PMS工具的核心，包含在这个PMS中的其他工具有：

- dpkg
    - `dpkg -L package //列出跟package相关的所有的文件`
- apt-get
- apt-cache
- aptitude //带界面的管理软件的工具
    - aptitude show package //显示是否安装了package
    - aptitude search package 查询package
        - 在显示内容中前 `i` 表示已经安装了， `p` 表示没有安装
    - aptitude install package
    - aptitude remove package 删除包且不删除文件
    - aptitude purge package 删除相关文件和配置文件

Shell 编程

- ``

```
gc@gouchao:~/shell$ bash ./b.sh
2016年 05月 11日 星期三 16:57:41 CST
gc@gouchao:~/shell$ cat b.sh
#!/bin/bash
gcTest=`date`
echo $gcTest
```

- 重定向输出
  - `>` 覆盖
  - `>>` 追加
- 重定向输入
  - `<`
- 算数表达式
  - `[ 1 + 2 ]`

    > shell最大的限制是没有提供完整的浮点算术运算，这是一个巨大的限制

解决方案:

- bash计算机: bc
  - bc识别
    - 数字
    - 变量
    - 注释 `#` OR `/* */`
    - 表达式
    - 变成语句
    - 函数
  - 浮点运算

```
gc@gouchao:~/shell$ bc -q  //-q 参数是让系统省去长欢迎词
10 / 3
3
scale=4 //设置小数点位数，默认0
10/3
3.3333
quit //退出bc
gc@gouchao:~/shell$ echo '1+1' | bc
2
---------------------------------------
#!/bin/bash
echo `echo "scale=4;10/3" | bc`
gc@gouchao:~/shell$ bash ./textbc.sh
3.3333
---------------------------------------
gc@gouchao:~/shell$ cat textbc.sh
#!/bin/bash
echo `bc << EOF //内联重定向 EOF标记了数据的开始和结尾
scale=4 //换行或者；分开表达式
a=1+2
b=a+3
c=b/a
a+b+c
EOF`
gc@gouchao:~/shell$ bash ./textbc.sh
11.0000
```

- 退出码
    - 0~255 范围，超出范围取余
    - Linux提供了 $? 代表上个shell的退出码
    - 退出码可用参考
        - 0 成功执行
        - 1 通用未知错误
        - 2 误用shell
        - 126 命令不可执行 / 权限不足
        - 127 没有找到命令
        - 128 无效退出参数
        - 130 ctrl+c退出
        - 255 退出状态码越界

```
gc@gouchao:~/shell$ bash ./textbc.sh
11.0000
gc@gouchao:~/shell$ echo $?
44
gc@gouchao:~/shell$ cat textbc.sh
#!/bin/bash
echo `bc << EOF
scale=4;a=1+2;
b=a+3
c=b/a
a+b+c
EOF`
exit 300 //退出程序，返回状态码
```

- if-then 结构化

```
gc@gouchao:~/shell$ bash ./ifThenTest
a.sh  b.sh  ifThenTest  ls  textbc.sh
gc@gouchao:~/shell$ cat ifThenTest
#!/bin/bash
if ls  //if后面必须+命令，当返回码=0，则执行then
then
    sl
fi
gc@gouchao:~/shell$ bash ./ifThenTest
a.sh  b.sh  ifThenTest  ls  textbc.sh
gc@gouchao:~/shell$ cat ifThenTest
#!/bin/bash
ls
if [ `whoami` == "gc"  ]
then
    sl
fi
```

- if-then-else

```
gc@gouchao:~/shell$ su gouchao
密码：
gouchao@gouchao:/home/gc/shell$ bash ./ifThenTest
a.sh  b.sh  ifThenTest  ls  textbc.sh
gouchao@gouchao:/home/gc/shell$ exit
exit
gc@gouchao:~/shell$ bash ./ifThenTest
a.sh  b.sh  ifThenTest  ls  textbc.sh
 _______
< hello >
 -------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
gc@gouchao:~/shell$ cat ifThenTest
#!/bin/bash
ls
if [ `whoami` == "gouchao"  ]
then
    sl
else
    cowsay "hello"
fi
```

- 嵌套if

```
#!/bin/bash
ls
if [ `whoami` == "gouchao"  ]
then
    sl
elif [ `whoami` == "gc" ]
then
    cowsay "hello"
else
    cmatrix
fi
```

- test 命令
    - test 命令是用来检验表达式是否成立的

```
gc@gouchao:~/shell$ test 1 -eq 1
gc@gouchao:~/shell$ echo $?
0
```

- test 命令可以换成 `[ ]`

```
gc@gouchao:~/shell$ [ 1 -eq 1 ]
gc@gouchao:~/shell$ echo $?
0
```

- `[ 1 -eq 1 ]` 在[ ] 中，左括号后面必须跟空格，右括号前面必须跟空格
- 数值比较
  - `n1 -eq n2` equal
  - `n1 -ge n2` greater and equal
  - `n1 -gt n2` greater then
  - `n1 -le n2` lower and equal
  - `n1 -lt n2` lower then
  - `n1 -ne n2` not equal
- 字符串比较
  - =
  - ! =
  - \<
  - \>
  - -n not zero
  - -z zero
- 文件比较
  - -d
  - -e
  - -f
  - -r
  - -w
  - -x
  - -s 判断file是否存在且非空
  - -O 检查是否属于当前用户
  - -G 默认组与当前用户相同
  - file1 -nt file2 new then
  - file1 -ot file2 old then
- 复合运算
  - &&
  - ||

```
gc@gouchao:~/shell$ bash ./ifThenTest
a.sh  b.sh  ifThenTest  ls  textbc.sh
gc@gouchao:~/shell$ cat ifThenTest
#!/bin/bash
ls
if [ `whoami` == "gouchao"  ] || [ `whoami` = "gc" ]
then
    sl
else
    cmatrix
fi
```

- 使用 `(( ))` 进行数字运算
  - val++
  - val–
  - –val
  - ++val
  - !
  - ~
  - ** 幂运算
  - <<
  - >>
  - &
  - &&
  - |
  - ||
- `[[ ]]` 进行字符串运算
  - `[[ $USER == r* ]]` 可以在括号里进行匹配

- `case`

```
gouchao@gouchao:/home/gc/shell$ cat ./caseTest.sh
#!/bin/bash
case $USER in
gc) echo "hello $USER";;
gouchao | root) echo "hello yours";;
*) echo "default";;
esac
gc@gouchao:~/shell$ bash ./caseTest.sh
hello gc
gc@gouchao:~/shell$ su gouchao
密码：
gouchao@gouchao:/home/gc/shell$ bash ./caseTest.sh
hello yours
```

- for
  - 格式

```
for var in list
do
    command
done
```

  - 例子

```
#!/bin/bash
filepath="/home/gouchao"
for file in `ls $filepath`
do
    echo $file
done
```

  - 字段分隔符
    - 默认情况下，bash shell 会将下列字符视为字段分隔符
      - 空格
      - 换行
      - 制表符
    - 显示分隔符
      - `IFS=$'\n'` 只将换行符作为字段分隔符

```
#!/bin/bash
#修改IFS之后
filepath="/home/gouchao"
IFS=$'\n'
for file in `cat /home/gc/shell/forTest.sh`
do
    echo $file
done
----------------------------------------
#!/bin/bash
#修改IFS之前
filepath="/home/gouchao"
#IFS=$'\n'
for
file
in
`cat
/home/gc/shell/forTest.sh`
do
echo
$file
done
```

- 使用通配符做for-in

```
gouchao@gouchao:/home/gc/shell$ bash ./forTest.sh
/home/gouchao/sublime_imfix.c
gouchao@gouchao:/home/gc/shell$ cat forTest.sh
#!/bin/bash
filepath="/home/gouchao"
#IFS=$'\n'
for file in /home/gouchao/*.c
do
    echo $file
done
```

> bash shell 能处理的数仅仅限于整数
> 当一个变量没有定义时，长度为空