

1.用户管理命令

/etc/passwd:存放用户信息的配置文件

用户登陆时，系统查询这个文件，确定用户的UID并验证用户口令

```
1  ubuntu:x:1000:1000:ubuntu,,,:/home/ubuntu:/bin/bash
2  用户名：经过加密的密码：UID：默认GID：用户信息：用户主目录：当前用户使用的shell
```

/etc/group：存放用户组的配置信息

每一行代表一个组，包括4个字段：

```
1  ubuntu:x:1000:
2  组名：口令：组ID：成员列表
```

用户和用户组的关系：

- 1.当创建一个新用户时，系统会默认创建一个同名的用户组
- 2.当把一个用户添加到一个用户组时，用户也就具备了该用户组在系统中的权限
- 3.一个用户可以添加在多个用户组里，一个用户组里可以添加多个用户

添加用户命令

adduser

```
1  语法：    adduser  <username>
2
```

实例：

```
3  adduser  newuser
```

添加用户名为newuser的新用户

修改用户信息命令

usermod

```
1  usermod [-u uid [-o]] [-g group] [-G group, ...]
2
3  [-d home [-m]] [-s shell] [-c comment]
4
5  [-l new_name] [-f inactive] [-e expire]
6
7  [-p passwd] [-L|-U] name
```

```
6      usermod -p 123456 username :修改密码

7

8      1、将 newuser2 添加到组 staff 中
9      # usermod -G staff newuser2
10     2、修改 newuser 的用户名为 newuser1
11     # usermod -l newuser1 newuser
12     3、锁定账号 newuser1
13     # usermod -L newuser1
14     4、解除对 newuser1 的锁定
15     # usermod -U newuser1
```

删除用户

deluser

```
1  、语法：    deluser <username>
2
      使用方法：
3
      deluser  --remove-home user1
4
      删除用户user1的同时删除用户的工作目录
```

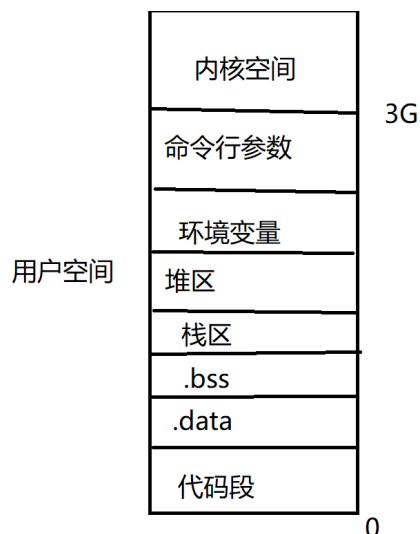
删除用户组

delgroup

```
1  delgro <groupname>
```

2.进程管理命令

程序的一次执行就是一个进程



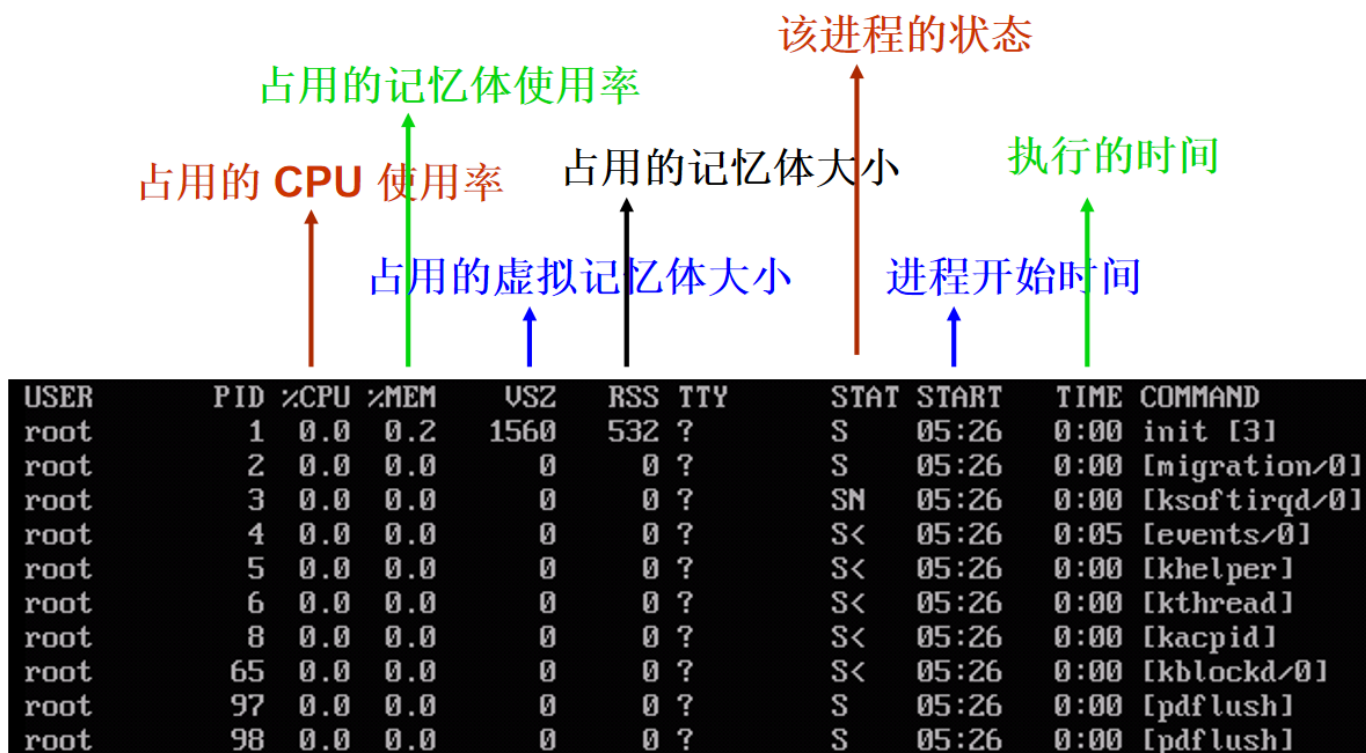
PS

语法:

常见的参数:

-w 显示加宽可以显示较多的资讯

-aux 显示所有包含其他使用者的行程



占用的记忆体使用率

占用的记忆体大小

该进程的状态

执行的时间

占用的虚拟记忆体大小

进程开始时间

1	D:	不可中断的静止
2		
	R:	正在执行中
3		
	S:	阻塞状态
4		
	T:	暂停执行
5		
	Z:	不存在但暂时无法消除
6	W:	没有足够的内存分页可分配
7		
	<:	高优先级的进程
8		
	N:	低优先级的进程
9		
	L:	有内存分页分配并锁在内存中

top

监视进程

通常会全屏显示，而且会随着进程状态的变化不断更新

整个系统的信息也会显示，为查找问题提供了便利

可以显示系统总共有多少CPU和内存资源以及负载平衡等信息。

shift+'>':向下翻页

shift+'<':向上翻页

q:退出

kill

杀死一个进程

kill [-signal] PID

signal是信号，PID是进程号

kill 命令向指定的进程发出一个信号signal，在默认的情况下，kill 命令向指定进程发出信号15，正常情况下，将杀死那些不捕捉或不忽略这个信号的进程

1	kill -l:列出可用信号				
2	1) SIGHUP	2) SIGINT	3) SIGQUIT	4) SIGILL	5) SIGTRAP
3	6) SIGABRT	7) SIGBUS	8) SIGFPE	9) SIGKILL	10) SIGUSR1
4	11) SIGSEGV	12) SIGUSR2	13) SIGPIPE	14) SIGALRM	15) SIGTERM
5	16) SIGSTKFLT	17) SIGCHLD	18) SIGCONT	19) SIGSTOP	20) SIGTSTP

```
6 21) SIGTTIN      22) SIGTTOU      23) SIGURG      24) SIGXCPU      25) SIGXFSZ
7 26) SIGVTALRM    27) SIGPROF      28) SIGWINCH     29) SIGIO        30) SIGPWR
8 31) SIGSYS       34) SIGRTMIN     35) SIGRTMIN+1  36) SIGRTMIN+2   37) SIGRTMIN+3
9 38) SIGRTMIN+4   39) SIGRTMIN+5   40) SIGRTMIN+6   41) SIGRTMIN+7   42) SIGRTMIN+8
10 43) SIGRTMIN+9   44) SIGRTMIN+10  45) SIGRTMIN+11  46) SIGRTMIN+12  47) SIGRTMIN+13
11 48) SIGRTMIN+14  49) SIGRTMIN+15  50) SIGRTMAX-14  51) SIGRTMAX-13  52) SIGRTMAX-12
12 53) SIGRTMAX-11  54) SIGRTMAX-10  55) SIGRTMAX-9   56) SIGRTMAX-8   57) SIGRTMAX-7
13 58) SIGRTMAX-6   59) SIGRTMAX-5   60) SIGRTMAX-4   61) SIGRTMAX-3   62) SIGRTMAX-2
14 63) SIGRTMAX-1   64) SIGRTMAX
15
16 2) SIGINT
17 3) SIGQUIT: ctrl+'\'
18 9) SIGKILL:ctrl+c
19 19) SIGSTOP:ctrl+z
```

3.文件管理命令

在Windows下，目录结构属于分区；在Linux下，分区属于目录结构。

Linux文件系统就是一个树形的分层组织结构。将根 (/) 作为整个文件系统的惟一起点，其他所有目录都从该点出发。将Linux的全部文件按照一定的用途归类，合理地挂载到这颗“大树”的“树枝”或“树叶”上

文件查看命令

1.ls 查看目录文件下的信息

2.cat 查看指定文件的内容

cat -n:查看指定文件中的内容，带行号

3.vi:使用vi编辑器查看文件

```
1 ~/.vim/snippets/c.snippets :c文件补全设置的配置文件
2 hdr:文件介绍补全
3 funh:函数注释补全
4 main:main函数框架补全
5 Inc头文件补全
```

4.head:查看文件前N行内容

- 1 `head -N` 文件名
- 2 `head` 文件名: 查看文件前10行内容

5.tail:查看文件最后N行内容

- 1 `tail -N` 文件名
- 2 `tail` 文件名: 查看文件后10行内容

6.file 文件名: 查看文件的具体信息

文件创建命令

- 1 `touch` 文件名: 创建普通文件
- 2 `mkdir` 目录名 : 创建目录文件
- 3 `mkdir -p`: 可以进行递归检查创建目录文件

文件删除命令

- 1 `rm` 文件名
- 2 `rmdir` 文件夹名: 删除空文件夹
- 3 `rm -r`: 删除文件夹
- 4 `rm -rf`: 彻底删除文件

文件的移动

- 1 `mv` 目标文件 目标路径
- 2 `mv` 给文件改名: `mv` 文件1 文件2

文件的复制

- 1 `cp` 目标文件 目标路径
- 2 `cp -r` 目标文件 目标路径: 将目标目录文件移动到指定的路径下

文件的统计

- 1 `wc` 文件名: 统计文件中的行数、单词个数、字符个数
- 2 `-l`: 统计行号
- 3 `-w`: 统计单词
- 4 `-c`: 统计字符

文件的查找

- 1 格式: `find 路径 -name 文件名`
- 2 作用: 查找指定路径下出现的所有文件
- 3 案例:
- 4 `find ./ -name 1.c`
- 5 查找当前路径下的目录中出现的所有`1.c`的位置

文件中单词的查找

- 1 格式: `grep "目标单词" 查找的路径 参数`
- 2 参数:
- 3 `-n`: 查找时显示行号
- 4 `-R`: 递归查询
- 5 `-w`: 搜索的目标为单词
- 6 `-i`: 查找时忽略大小写
- 7
- 8 案例:
- 9 `grep "hello" 1.c -n`: 在`1.c`中查找单词`hello`, 并显示行号
- 10 `grep "printk" ./ -R`: 在当前路径下递归查询所有出现的`printk`
- 11

文件权限的修改

文件类型: 7种

b: block块设备文件, 一般是磁盘文件

c: char字符设备文件, 鼠标、键盘。。。。

d: 目录 文件

-: 普通文件, 可以进行编辑或者执行的文件

l: link链接文件

s:socket套接字文件，和网络属性相关的文件

p:管道文件

文件权限：

w:写权限

r:读权限

x:执行权限

文件权限的修改

命令：chmod

```
1 格式：
2  chmod 权限 文件名
3  一个权限是一个二进制数，有权限就写1，没权限写0，最后转化成8进制形式
4  r w x
5  1 1 0->6
6  chmod 777 1.c//给1.c权限修改为当前用户权限、用户组权限，其他用户权限均为读写执行
7
8  单独给某一组权限进行修改的方式：
9  chmod 目标+/-权限
10 目标：
11    u:当前用户  g:用户组  o:其他用户  a:所有用户
12  +: 赋权限
13  -: 撤销权限
14 权限：r w x
15
16 例子：chmod o+w 2.c
17 把2.c文件其他用户权限加上可写
```

链接文件的创建

分类：软链接、硬链接

软链接：

基于源文件创建的一个链接文件，软链接文件类型为l

```
1  创建的格式：ln -s 源文件名 链接文件名
```

性质

- 1.软链接类似于windows下面的快捷方式
- 2.软链接文件类型为l
- 3.如果软链接对应的源文件路径发生改变或者被删除，链接文件不会被删除，但是不能起作用了，除非路径下有新的以原来源文件为名的文件出现
- 4.软链接可以链接普通文件，也可以链接目录文件
- 5.一个源文件可以有多个链接文件
- 6.当一个文件被软链接后，硬链接个数不会增加

硬链接

类似于给源文件起了一个别名，不会再额外创建一个链接文件

- 1 格式：
- 2 `ln 源文件名 硬链接文件名`
- 3 `ln 1.c 6.c`//给1.c创建一个名字为6.c的硬链接

性质

- 1.硬链接文件相当于给源文件起别名，不会在磁盘占用额外的空间
- 2.当源文件或者硬链接文件被删除时，不会影响文件在磁盘上的位置，直到源文件和所有的链接文件都被删除，源文件才会在磁盘上被删除
- 3.对文件进行一次硬链接，文件属性里硬链接个数+1
- 4.硬链接即使源文件消失，硬链接也能正常操作
- 5.硬链接文件类型和源文件一致
- 6.硬链接不可以作用于目录文件（不可以为目录文件创建硬链接）

文件的压缩和解压

ubuntu下可以把文件压缩成三种不同的后缀的压缩文件：.gz .bz2 .xz

三种后缀的压缩方式依次：压缩率越来越高，压缩效率越来越低

1. gz后缀

- 1 压缩：
- 2 `gzip 文件名`
- 3 解压：
- 4 `gunzip 压缩文件名`

2. bz2后缀

- 1 压缩:
- 2 `bzip2` 文件名
- 3 解压:
- 4 `bunzip2` 压缩文件名

3. xz后缀

- 1 压缩:
- 2 `xz` 文件名
- 3 解压:
- 4 `unxz` 压缩文件名

目录文件的归档和释放

使用tar命令可以实现对于目录文件的归档以及归档之后的释放

tar命令的参数:

- 1 `-c`: 创建一个归档文件, 目录归档时使用
- 2 `-x`: 释放归档后的目录文件, 压缩文件释放时使用
- 3 `-z`: 将目录文件归档成`.gz`格式的压缩文件
- 4 `-j`: 将目录文件归档成`.bz2`格式的压缩文件
- 5 `-J`: 将目录文件归档成`.xz`格式的压缩文件
- 6 `-v`: 显示整个归档和释放的过程
- 7 `-f`: 后面跟文件名字

归档案例:

- 1 `tar -cvf 3.tar 3`: 把3归档生成压缩文件`3.tar`
- 2 `tar -czvf 3.tar.gz 3`: 把3归档生成压缩文件`3.tar.gz`

释放案例:

- 1 `tar -xvf 3.tar`: 将`3.tar`释放
- 2 `tar -xzvf 3.tar.gz`: 将`3.tar.gz`释放

万能拆包书写格式

- 1 `tar -xvf` 压缩包名

将释放后的目录文件存放在指定的路径下：

```
1 tar -xvf 目标压缩文件 -C 路径
```

shell脚本

定义

一个或者多个shell命令通过指定的逻辑或者规则编写在一个文件中，可以按照指定的顺序去执行，这个文件就被称为shell脚本文件，shell脚本文件的后缀是sh

第一个shell脚本程序

在shell脚本里面注释用#

```
1 #!/bin/bash
2 #第一行用来告诉系统当前写的shell脚本使用的bash
3 echo hello world
```

shell脚本的编程步骤

- 1.创建shell脚本文件，编写脚本逻辑
- 2.给脚本文件赋执行权限，chmod u+x 脚本文件名
- 3.执行脚本文件：./脚本文件名字

注意事项：

- 1.执行脚本文件时如果使用./的形式执行，是在别的终端执行，然后把结果反馈到当前终端上，如果想要直接在当前终端执行，可以使用source 脚本文件名的形式
- 2.一个命令结束了一定要以空格或者;隔开

shell脚本中的变量

```
1 #!/bin/bash
2
3 var=hello
4 echo $var
5 #echo ${var}
```

1.变量的性质

shell脚本中的变量不需要大家定义，直接进行赋值即可

shell脚本中的变量没有数据类型，只有字符串类型

变量赋值用=， '=' 两边不要加空格

shell脚本认为所有变量的数值都是字符串类型

shell脚本变量的数值的访问方式 \$变量名或者\${变量名}

2.变量的赋值

```
1  #!/bin/bash
2
3  var=1234 #把字符串1234赋值给变量var
4  var1='hello world' #如果赋值的字符串有空格则需要双引号或者单引号
5  var2=$var #把变量的数值赋值给另外一个变量
6  var3="$var $var1 $var2" #把多个变量的数值赋值给一个变量用""
7  echo ${var}
8  echo ${var1}
9  echo ${var2}
10 echo ${var3}
```

3.shell脚本中的命令行传参

shell里面我们使用位置变量来接收命令行终端的参数：

\$0:终端传来的第一个参数，也就是脚本名

\$1:终端传来的第二个参数

...

\$n:终端传来的第n+1个参数

```
1  #!/bin/bash
2
3  echo $0 #命令行第一个参数
4  echo $1 #命令行第二个参数
5  echo $2
6  echo $3
7
```

如果想要输出命令行传来的所有参数，可以使用\$@或者\$*,不包括执行的脚本名

```
1  #!/bin/bash
2
```

```
3 echo $@
4 echo $*
5
6 输出结果:
7 ubuntu@ubuntu:~/3$ ./hello.sh hah nihao aaaaa
8 hah nihao aaaaa
9 hah nihao aaaaa
10
```

使用\$#来表示命令行终端传来的参数的个数

```
1 #!/bin/bash
2
3 echo $#
4 echo $*
5 结果:
6 ubuntu@ubuntu:~/3$ ./hello.sh hah nihao aaaaa
7 3
8 hah nihao aaaaa
9
```

补充：当一个字符串比较长的时候，我们可以在在字符串中间+ ‘\n’ ，将后面的内容换行显示

```
1 #!/bin/bash
2
3 var="hello world"
4 var1="你好 世界"
5 var2="$var1 \n$var"
6 echo -e $var2 # -e参数表示可以识别输出内容中的换行符
```

4.清除变量的数值

unset 变量名

```
1 #!/bin/bash
2
3 var="hello world"
4 echo $var
5 unset var #把var的值清空
6 echo $var #没有输出内容
```

5.把变量设置成只读变量

readonly 变量名

```
1  #!/bin/bash
2
3  var="hello world"
4  echo $var
5  var="你好"
6  echo $var
7  readonly var
8  var=hahha
9  echo $var
10 输出结果:
11      hello world
12  你好
13  ./hello.sh: 行 8: var: 只读变量
14  你好
15
```

任务

1.复习今日内容

2.通过命令行传参给shell脚本文件传入两个字符串，把数值分别赋值给两个变量，实现两个变量的数值交换，然后输出两个变量的数值

3.通过命令行传一个文件路径~/1

在这个路径下创建两个文件：1.txt 2.txt,在1.txt里面放一个字符串“hello world”，在2.txt里面存放“你好世界”