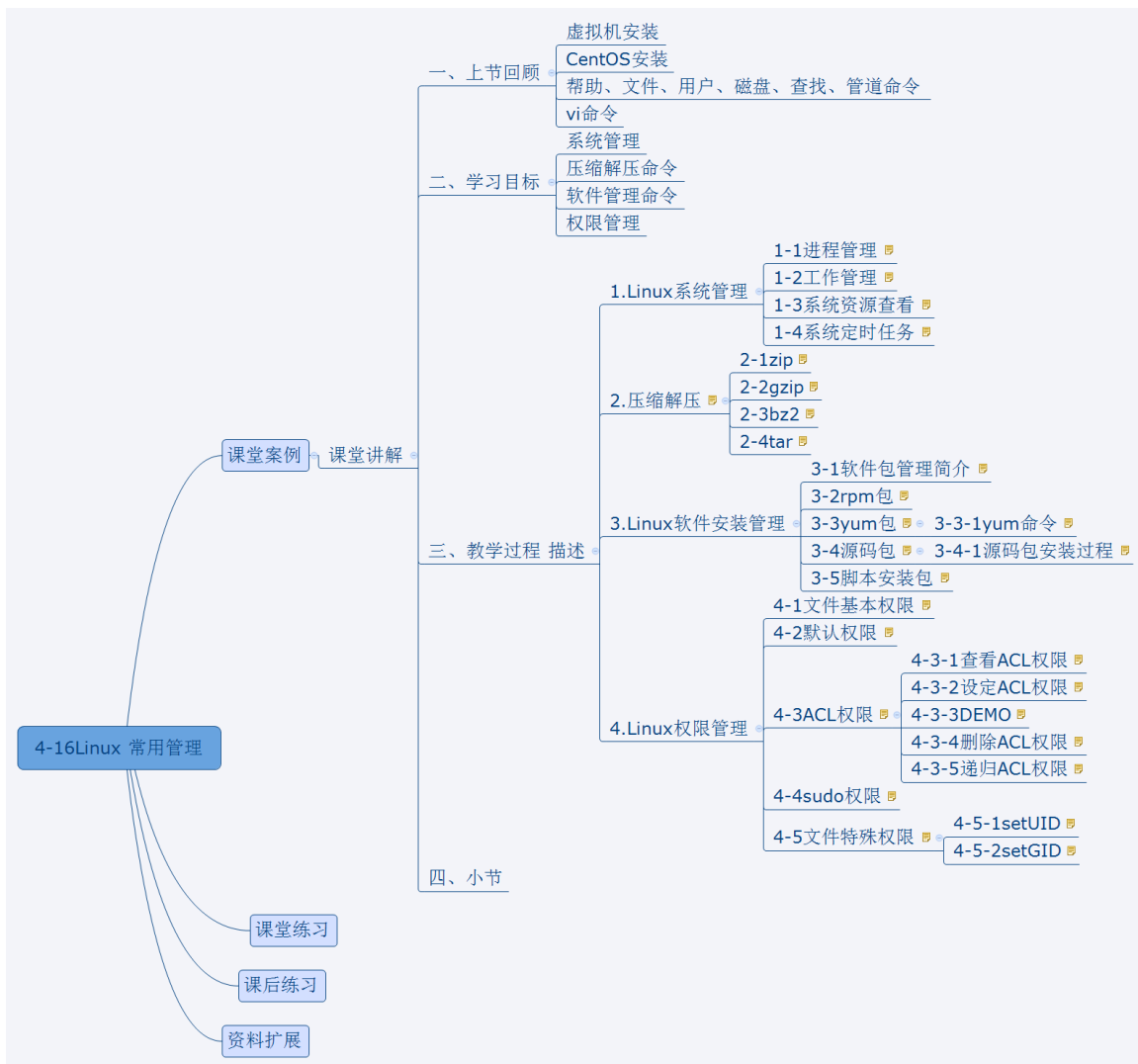


4-16Linux 常用管理

4-16Linux 常用管理.....	1
1. 课堂案例	2
课堂讲解	2
一、上节回顾	2
虚拟机安装	2
CentOS安装	2
帮助、文件、用户、磁盘、查找、管道命令	2
vi命令	3
二、学习目标	3
系统管理	3
压缩解压命令	3
软件管理命令	3
权限管理	3
三、教学过程 描述	3
1.Linux系统管理.....	3
2.压缩解压	17
3.Linux软件安装管理.....	24
4.Linux权限管理.....	32
四、小节	41
2. 课堂练习	41
3. 课后练习	41
4. 资料扩展	41



1. 课堂案例

课堂讲解

一、上节回顾

虚拟机安装

CentOS安装

帮助、文件、用户、磁盘、查找、管道命令

vi命令

二、学习目标

系统管理

压缩解压命令

软件管理命令

权限管理

三、教学过程 描述

1.Linux系统管理

1-1进程管理

什么是进程？

进程就是程序在内存中的表现, 比如一个命令(ls,cd,cp等)一个服务比如(apach,mysql)就是一个进程

为什么要对进程进行管理？

判断服务器健康状态

查看系统中所有的进程

杀死进程

ps 命令

有两个常用组合

- **ps aux**
#查看系统中所有进程，使用BSD操作系统格式
- **ps -le**
#查看系统中所有进程，使用Linux标准命令格式
- **选项**
 - **a**：显示一个终端的所有进程，除了会话引线
 - **u**：显示进程的归属用户及内存的使用情况
 - **x**：显示没有控制终端的进程
 - **-l**：长格式显示。显示更加详细的信息
 - **-e**：显示所有进程，和-A作用一致

ps命令的输出

- ✧ **USER**：该进程是由哪个用户产生的；
- ✧ **PID**：进程的ID号；
- ✧ **%CPU**：该进程占用CPU资源的百分比，占用越高，进程越耗费资源；
- ✧ **%MEM**：该进程占用物理内存的百分比，占用越高，进程越耗费资源；
- ✧ **VSZ**：该进程占用虚拟内存的大小，单位KB；
- ✧ **RSS**：该进程占用实际物理内存的大小，单位KB；
- ✧ **TTY**：该进程是在哪个终端中运行的。其中**tty1-tty7**代表本地控制台终端，**tty1-tty6**是本地的字符界面终端，**tty7**是图形终端。**pts/0-255**代表虚拟终端。

✧ **STAT**：进程状态。常见的状态有：

- **R**：运行
- **S**：睡眠
- **T**：停止状态
- **s**：包含子进程
- **+**：位于后台

✧ **START**：该进程的启动时间

✧ **TIME**：该进程占用**CPU**的运算时间，注意不是系统时间

✧ **COMMAND**：产生此进程的命令名

pstree 命令

选项：

-p：显示进程的PID

-u：显示进程的所属用户

top 命令 查看系统的健康状态

[root@localhost ~]# top [选项]

选项：

- **-d 秒数**：指定top命令每隔几秒更新。默认是3秒
- **-b**：使用批处理模式输出。一般和“-n”选项合用
- **-n 次数**：指定top命令执行的次数。一般和“-b”选项合用

在top命令的交互模式当中可以执行的命令：

- **? 或h**：显示交互模式的帮助
- **P**：以CPU使用率排序，默认就是此项
- **M**：以内存的使用率排序
- **N**：以PID排序
- **q**：退出top

如：top -b -n 1 > top.log 进行进程信息写到top.log 日志中查看

```
top - 08:00:10 up 15 min, 1 user, load average: 0.00, 0.00, 0.00
Tasks: 100 total, 1 running, 99 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.5%us, 1.8%sy, 0.0%ni, 97.4%id, 0.2%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1004768k total, 206472k used, 798296k free, 19160k buffers
Swap: 2047996k total, 0k used, 2047996k free, 64120k cached
```

最关键的5行信息

第一行信息为任务队列信息

内容	说明
12:26:46	系统当前时间
up 1 day, 13:32	系统的运行时间，本机已经运行1天13小时32分钟
2 users	当前登录了两个用户
load average: 0.00, 0.00, 0.00	系统在之前1分钟，5分钟，15分钟的平均负载。一般认为小于1时，负载较小。如果大于1，系统已经超出负荷。

第二行为进程信息

内容	说明
Tasks: 95 total	系统中的进程总数
1 running	正在运行的进程数
94 sleeping	睡眠的进程
0 stopped	正在停止的进程
0 zombie	僵尸进程。如果不是0，需要手工检查僵尸进程

第三行为CPU信息

内容	说明
Cpu(s): 0.1%us	用户模式占用的CPU百分比
0.1%sy	系统模式占用的CPU百分比
0.0%ni	改变过优先级的用户进程占用的CPU百分比
99.7%id	空闲CPU的CPU百分比
0.1%wa	等待输入/输出的进程的占用CPU百分比
0.0%hi	硬中断请求服务占用的CPU百分比
0.1%si	软中断请求服务占用的CPU百分比
0.0%st	st (Steal time) 虚拟时间百分比。就是当有虚拟机时，虚拟CPU等待实际CPU的时间百分比。

第四行为物理内存信息

内容	说明
Mem: 625344k total	物理内存的总量，单位KB
571504k used	已经使用的物理内存数量
53840k free	空闲的物理内存数量，我们使用的是虚拟机，总共只分配了628MB内存，所以只有53MB的空闲内存了
65800k buffers	作为缓冲的内存数量

第五行为交换分区（swap）信息

内容	说明
Swap: 524280k total	交换分区（虚拟内存）的总大小
0k used	已经使用的交互分区的大小
524280k free	空闲交换分区的大小
409280k cached	作为缓存的交互分区的大小

杀死进程 kill 命令

kill -l 查看可用的进程信号

kill -l 2235 重启进程

kill -9 2237 强制杀死进程

信号代号	信号名称	说明
1	<u>SIGHUP</u>	该信号让进程立即关闭，然后重新读取配置文件之后重启。
2	SIGINT	程序终止信号，用于终止前台进程。相当于输出ctrl+c快捷键。
8	SIGFPE	在发生致命的算术运算错误时发出，不仅包括浮点运算错误，还包括溢出及除数为0等其它所有的算术的错误。
9	SIGKILL	用来立即结束程序的运行，本信号不能被阻塞、处理和忽略。一般用于强制终止进程。
14	SIGALRM	时钟定时信号，计算的是实际的时间或时钟时间。alarm函数使用该信号。
15	SIGTERM	正常结束进程的信号，kill命令的默认信号。有时如果进程已经发生问题，这个信号是无法正常终止进程的，我们才会尝试SIGKILL信号，也就是信号9。
18	SIGCONT	该信号可以让暂停的进程恢复执行，本信号不能被阻断。
19	SIGSTOP	该信号可以暂停前台进程，相当于输入ctrl+z快捷键。本信号不能被阻断。

killall [选项][信号] 进程名 按照进程名杀死进程

选项：

-i: 交互式，询问是否要杀死某个进程

-I:忽略进程名的大小写

pskill [选项][信号] 进程名 按照进程名终止进程

选项:

-t 终端号: 按照终端号踢出用户

如:pskill -9 -t tty1

kill 是操作一个进程, killall 与 pskill 操作多进程

个性进程优先级

Linux操作系统是一个多用户, 多任务的操作系统, Linux

系统中通知运行着的非常多的进程, 但是CPU在同一个时钟内只能运算一个指令, 进程优先级决定了每个进程处理的先后顺序。

```
[echo@bogon ~]$ ps -el
F S  UID  PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   0    1    0  0  80   0 -  4841 poll_s ?           00:00:02 init
1 S   0    2    0  0  80   0 -    0 kthrea ?           00:00:00 kthreadd
1 S   0    3    2  0 -40   - -    0 migrat ?           00:00:00 migration/0
1 S   0    4    2  0  80   0 -    0 ksofti ?           00:00:00 ksoftirqd/0
1 S   0    5    2  0 -40   - -    0 cpu_st ?           00:00:00 stopper/0
```

PRI 代表Priority NI代表Nice,这两个值都是优先级,数字越小代表该进程优先级越高

修改NI值时有几个注意事项

- NI的值的范围是-20到19；
- 普通用户调整NI值的范围是0到19，而且只能调整自己的进程
- 普通用户只能调高NI值，而不能降低，如原本NI值为0，则只能调整为大于0；
- root用户才能设定进程NI值为负值，而且可以调整任何用户的进程。
- $PRI(最终值) = PRI(原始值) + NI$
- 用户只能修改NI的值，不能直接修改PRI

nice命令

- nice [选项] 命令

#nice命令⁺可以给新执行的命令直接赋予NI值，但是不能修改已经存在进程的NI值

- 选项：

➤ -n NI值：给命令赋予NI值。

- 例如

- nice -n -5 service httpd start

renice命令

- **renice [优先级] PID**
- ***#renice命令是修改已经存在进程的NI值的命令***
- **例如**
- **renice -10 2125**

1-2工作管理

什么是工作管理

工作管理指的是在单个登录终端中(也就是登录的shell界面中)同时管理多个工作的行为

注意事项:

1. 当前的登录终端, 只能管理当前终端的工作, 而不能管理其他登录终端的工作
2. 放入后台的命令必须可以持续运行一段时间, 这样我们才能捕捉操作这个工作
3. 放入后台执行的命令不能和前台用户有交互或需要前台输入, 否则放入后台只能暂停, 而不能执行

管理方法

放入后台方法: 命令后加 **&** 或 **ctrl+z**(放入后台后暂停)

jobs [-l]

选项:

-l: 显示工作的PID

备注: “+”号代表最近一个放入后台的工作, 也是工作恢复时默认恢复的工作, “-

”号代表倒数第二个放入后台的工作

将后台暂停工作恢复到前台执行：

`fg %工作号`

参数：

`-%工作号` %号可以省略，但是注意工作号和PID的区别

把后台暂停的工作恢复到后台执行

`bg %工作号`

注意：后台恢复执行的命令，是不能和前台有交互的，否则不能恢复到后台执行

后台命令脱离登录终端

前面我们讲过了，工作都是跟终端关联的，即终端关闭后，程序也会关闭，但我们总能发现某些进程，即使终端关闭，程序也不会关闭，比如 mysql,apach等，这些守护进程，我们这种行为叫脱离终端。

非守护进程，脱离终端的方式：

1. 第一种方法，是把需要后台执行的命令加入 `/etc/rc.local` 文件
2. 第二种方法是，使用系统定时任务，让系统在指定的时间执行某个后台命令
3. 第三种方法是使用 `nohup`命令

`nohup top &`

1-3系统资源查看

`vmstat [刷新延时 刷新次数]` 监控系统资源

如：`vmstat 1 3`

```
[echo@bogon ~]$ vmstat 1 3
procs -----memory----- ---swap-- ----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo    in   cs  us  sy  id  wa  st
 1  0     0 345548  63376 424412    0    0   84   60   33   38   0   1  98   0   0
 0  0     0 345500  63376 424412    0    0    0    0   27   17   0   1  99   0   0
 0  0     0 345500  63376 424412    0    0    0    0   13   16   0   0 100   0   0
```

- **procs : 进程信息字段 :**
 - **r :** 等待运行的进程数，数量越大，系统越繁忙。
 - **b :** 不可被唤醒的进程数量，数量越大，系统越繁忙。
- **memory : 内存信息字段 :**
 - **swpd :** 虚拟内存的使用情况，单位KB。
 - **free :** 空闲的内存容量，单位KB。
 - **buff :** 缓冲的内存容量，单位KB。
 - **cache :** 缓存的内存容量，单位KB。
- **system : 系统信息字段 :**
 - **in :** 每秒被中断的进程次数。
 - **cs :** 每秒钟进行的事件切换次数。此两个数越大，代表系统与接口设备的通信非常繁忙。
- **CPU : CPU信息字段 :**
 - **us :** 非内核进程消耗CPU运算时间的百分比。
 - **sy :** 内核进程消耗CPU运算时间的百分比。
 - **id :** 空闲CPU的百分比。
 - **wa :** 等待I/O所消耗的CPU百分比。
 - **st :** 被虚拟机所盗用的CPU占比。
- **system : 系统信息字段 :**
 - **in :** 每秒被中断的进程次数。
 - **cs :** 每秒钟进行的事件切换次数。此两个数越大，代表系统与接口设备的通信非常繁忙。
- **CPU : CPU信息字段 :**
 - **us :** 非内核进程消耗CPU运算时间的百分比。
 - **sy :** 内核进程消耗CPU运算时间的百分比。
 - **id :** 空闲CPU的百分比。
 - **wa :** 等待I/O所消耗的CPU百分比。
 - **st :** 被虚拟机所盗用的CPU占比。

dmesg 判断硬件状态

free [-b|-k|-m|-g] 查看内存使用状态

选项:

-b: 以字节为单位显示

-k: 以KB为单位显示, 默认是KB为单位显示

-m: 以MB 为单位

-g: 以GB为单位

查看CPU信息:

cat /proc/cpuinfo

uptime top命令第一行

uname [选项] 查看系统内核相关信息

-a 查看系统所有相关信息

-r 查看内核版本

-s: 查看内核名称

uname -a

lsuf [选项] 列出进程打开或使用的文件信息

-c 字符串: 只列出以字符串开头的进程打开的文件

-u 用户名: 只列出某个用户进程打开的文件

-p pid: 列出某个PID进程打开的文件

如:

lsdf | more 查询系统中所有进程调用的文件
lsdf /sbin/init 查询某个文件被哪些进程调用
lsdf -c httpd 查看httpd进程调用了哪些文件
lsdf -u root 按照用户名, 查询某用户的进程调用的文件名

1-4系统定时任务

什么是定时任务?

定时任务类似我们手机中的闹钟, 到点就响, linux
中的定时任务就是到点就执行一定的命令。比如一个shell 脚本

crontab常用操作

列出crontab文件: crontab -l 显示当前的crontab 文件(默认编写的crontab文件会保存在
(/var/spool/cron/用户名 例如: /var/spool/cron/roger)

编辑crontab文件: crontab -e

删除crontab文件: crontab -r (谨慎使用此方法, 因为将会把所有的计划任务全部删除)

查看/etc/crontab文件

crontab 的文件格式:

minutes hour day-of-month month-of-year day-of-week commands

上面的值 可以使用 * 或 / 或 - 或 逗号

星号(*)表示所有可用的值

整数间的连字号(-)表示整数列, 例如1-4意思是整数1,2,3,4

符号"/"指定步进设置

如: */2 12-14 * 3-6,9-12 1-5 bash_scripts

表示: 每年的3-6月份, 以及9-12月份的周一到周五的下午12-14点, 每隔两分钟执行一个脚本

crontab服务的启动关闭

/sbin/service crond start //启动服务

```
/sbin/service crond stop //关闭服务
/sbin/service crond restart //重启服务
/sbin/service crond reload //重新载入配置
/sbin/service crond status //查看服务状态
```

crontab服务会每分钟检查一次/etc/crontab、/etc/cron.d、/var/spool/cron文件下的变更。如果发现变化,就会下载到存储器中。因此,即使crontab文件改变了,程序也不需要重新启动。推荐自定义的任务使用crontab -e命令添加,退出后用/etc/init.d/crond restart命令重启crond进程,官方文件说不用重启进程,但我遇到不重启无法运行任务的情况。

crontab命令使用权限

crontab权限问题到/var/adm/cron/下一看,文件cron.allow和cron.deny是否存在
用法如下:

- 1、如果两个文件都不存在,则只有root用户才能使用crontab命令。
- 2、如果cron.allow存在但cron.deny不存在,则只有列在cron.allow文件里的用户才能使用crontab命令,如果root用户也不在里面,则root用户也不能使用crontab。
- 3、如果cron.allow不存在, cron.deny存在,则只有列在cron.deny文件里面的用户不能使用crontab命令,其它用户都能使用。
- 4、如果两个文件都存在,则列在cron.allow文件中而且没有列在cron.deny中的用户可以使用crontab,如果两个文件中都有同一个用户,以cron.allow文件里面是否有该用户为准,如果cron.allow中有该用户,则可以使用crontab命令。

排错: 如果发现crontab不执行的问题,首先可以去看/var/spool/mail/用户名这个Log,以确定具体的失败原因

at:

at 命令被用来在指定时间内调度一次性的任务。

at [-mldv] TIME

选项与参数:

-m :当at的任务完成后,即使没有输出信息,也以 email 通知给使用者

-l :列出目前系统上面的所有该使用者的at任务(同atq)

-d :可以取消一个在 at 任务(同atrm)

-v : 可以使用较明显的时间格式列出 at 任务

-c : 可以列出后面接的该项任务的内容

at命令的时间格式:

now + 时间 : 时间以 minutes、hours、days、或 weeks 为单位

HH:MM : 24小时制度, 如果时间已过, 就会在第二天的这一时间执行

midnight : 表示00:00

noon : 表示12:00

teatime : 表示16:00

如:

```
[root@rhel6 ~]# at 13:10 //定义一个at任务在13:10执行
```

```
at> date >> /tmp/at //将当前时间输入/tmp/at文件
```

```
at> echo "at command test" >> /tmp/at
```

```
at> uname -r >> /tmp/at
```

```
at> <EOT> //另选一行按Ctrl+D退出at命令模式
```

```
job 1 at 2012-12-28 13:10
```

2. 压缩解压

源码安装包, 一般都是压缩包

我们先来了解压缩与解压

常见的压缩包格式:

zip

gz

bz

bz2

tar

2-1zip

压缩 语法:zip [选项] 压缩后文件 被压缩文件

解压 语法:unzip [选项] zip文件 [解压到的目录]

选项:

主要参数

-c:将解压缩的结果

-l:显示压缩文件内所包含的文件

-p:与-c参数类似,会将解压缩的结果显示到屏幕上,但不会执行任何的转换

-t:检查压缩文件是否正确

-u:与-f参数类似,但是除了更新现有的文件外,也会将压缩文件中的其它文件解压缩到目录中

-v:执行是时显示详细的信息

-z:仅显示压缩文件的备注文字

-a:对文本文件进行必要的字符转换

-b:不要对文本文件进行字符转换

-C:压缩文件中的文件名称区分大小写

-j:不处理压缩文件中原有的目录路径

-L:将压缩文件中的全部文件名改为小写

-M:将输出结果送到more程序处理

-n:解压缩时不要覆盖原有的文件

-o:不必先询问用户, unzip执行后覆盖原有文件

-P:使用zip的密码选项

-q:执行时不显示任何信息

-s:将文件名中的空白字符转换为底线字符

-V:保留VMS的文件版本信息

-X:解压缩时同时回存文件原来的UID/GID

例:

- 1、把/home目录下面的mydata目录压缩为mydata.zip
zip -r mydata.zip mydata #压缩mydata目录
- 2、把/home目录下面的mydata.zip解压到mydatabak目录里面
unzip mydata.zip -d mydatabak
- 3、把/home目录下面的abc文件夹和123.txt压缩成为abc123.zip
zip -r abc123.zip abc 123.txt
- 4、把/home目录下面的wwwroot.zip直接解压到/home目录里面
unzip wwwroot.zip
- 5、把/home目录下面的abc12.zip、abc23.zip、abc34.zip同时解压到/home目录里面
unzip abc*.zip
- 6、查看把/home目录下面的wwwroot.zip里面的内容
unzip -v wwwroot.zip
- 7、验证/home目录下面的wwwroot.zip是否完整
unzip -t wwwroot.zip
- 8、把/home目录下面wwwroot.zip里面的所有文件解压到第一级目录
unzip -j wwwroot.zip

2-2gzip

语□□法: gzip [-acfhLnNqrtvV][☐-s <压缩字尾字符串>][文件...]
或 gzip [-acfhLnNqrtvV][☐-s <压缩字尾字符串>][目录]

补充说明: gzip是个使用广泛的解压缩程序, 它用于解开被gzip压缩过的文件, 这些压缩文件预设最后的扩展名为".gz"。事实上gzip就是gzip的硬连接, 因此不论是压缩或解压缩, 都可通过gzip指令单独完成。

参□□数:

- a或--ascii ☐使用ASCII文字模式。
 - c或--stdout或--to-stdout ☐把解压后的文件输出到标准输出设备。
 - f或--force ☐强行解开压缩文件, 不理睬文件名称或硬连接是否存在以及该文件是否为符号连接。
 - h或--help ☐在线帮助。
 - l或--list ☐列出压缩文件的相关信息。
 - L或--license ☐显示版本与版权信息。
 - n或--no-name ☐解压缩时, 若压缩文件内含有远来的文件名称及时间戳记, 则将其忽略不予处理。
 - N或--name ☐
- 解压缩时, 若压缩文件内含有原来的文件名称及时间戳记, 则将其回存到解开的文件上。
- q或--quiet ☐不显示警告信息。
 - r或--recursive ☐递归处理, 将指定目录下的所有文件及子目录一并处理。
 - S<压缩字尾字符串>或--suffix<压缩字尾字符串> ☐更改压缩字尾字符串。

-t或--test □测试压缩文件是否正确无误。

-v或--verbose □显示指令执行过程。

-V或--version 显示版本信息。

2-3bz2

语法: bzip2(选项)(参数)

选项:

-c或——stdout: 将压缩与解压缩的结果送到标准输出;

-d或——decompress: 执行解压缩;

-f或-force: bzip2在压缩或解压缩时, 若输出文件与现有文件同名, 预设不会覆盖现有文件。

若要覆盖。请使用此参数;

-h或——help: 在线帮助;

-k或——keep: bzip2在压缩或解压缩后, 会删除原始文件。若要保留原始文件, 请使用此参数;

-s或——small: 降低程序执行时内存的使用量;

-t或——test: 测试.bz2压缩文件的完整性;

-v或——verbose: 压缩或解压缩文件时, 显示详细的信息;

-z或——compress: 强制执行压缩;

-V或——version: 显示版本信息;

--repetitive-best: 若文件中有重复出现的资料时, 可利用此参数提高压缩效果;

--repetitive-fast: 若文件中有重复出现的资料时, 可利用此参数加快执行效果。

实例

压缩指定文件filename:

bzip2 filename

或

bzip2 -z filename

这里, 压缩的时候不会输出, 会将原来的文件filename给删除,

替换成filename.bz2. 如果以前有filename.bz2则不会替换并提示错误(如果想要替换则指定-

f选项, 例如bzip2

-f

filename; 如果filename是目录则也提醒错误不做任何操作; 如果filename已经是压过的了有bz2后缀就提醒一下, 不再压缩, 没有bz2后缀会再次压缩。

解压指定的文件filename.bz2:

```
bzip2 -d filename.bz2
```

或

```
bunzip2 filename.bz2
```

这里, 解压的时候没标准输出, 会将原来的文件filename.bz2给替换成filename。如果以前有filename则不会替换并提示错误(如果想要替换则指定-f选项, 例如bzip2 -df filename.bz2。

压缩解压的时候将结果也输出:

```
$bzip2 -v filename 输入之后,
```

输出如下: filename: 0.119:1, 67.200 bits/byte, -740.00% saved, 5 in, 42 out.

这里, 加上-v选项就会输出了, 只用压缩举例了, 解压的时候同理bzip2 -dv filename.bz2不再举例了。

模拟解压实际并不解压: bzip2 -tv filename.bz2

输入之后, 输出如下:

```
filename.bz2:                                ok                                这里, -
t指定要进行模拟解压, 不实际生成结果, 也就是说类似检查文件, 当然就算目录下面有filename也不会有什么错误输出了, 因为它根本不会真的解压文件。为了在屏幕上输出, 这里加上-v选项了, 如果是真的解压bzip2 -dv filename.bz2则输出的是把"ok"替换成了"done"。
```

压缩解压的时候, 除了生成结果文件, 将原来的文件也保存:

```
bzip2 -k filename
```

这里, 加上-

k就保存原始的文件了, 否则原始文件会被结果文件替代。只用压缩举例了, 解压的时候同理\$bzip2 -dk filename.bz2不再举例了。

解压到标准输出: bzip2 -dc filename.bz2

输入之后, 输出如下:

hahahhaahaha

这里, 使用-c指定到标准输出, 输出的是文件filename的内容, 不会将filename.bz2删除。

压缩到标准输出:

```
bzip2 -c filename
```

```
bzip2: I won't write compressed data to a terminal.
```

```
bzip2: For help, type: `bzip2 --help'.
```

这里, 使用-c指定压缩到标准输出不删除原有文件, 不同的是, 压缩后的文件无法输出到标准输出。

使用bzip2的时候将所有后面的看作文件(即使文件名以'-'开头):

```
bzip2 -- -myfilename
```

2-4tar

tar 是一个打包命令, 但一般我们跟是打包跟压缩一起执行

Linux中很多压缩程序只能针对一个文件进行压缩, 这样当你想要压缩一大堆文件时, 你得先将这一大堆文件先打成一个包(tar命令), 然后再用压缩程序进行压缩(gzip bzip2命令)。

语法: tar【选项】参数

选项:

-A或--catenate: 新增文件到以存在的备份文件;

-B: 设置区块大小;

-c或--create: 建立新的备份文件;

-C <目录>: 这个选项用在解压缩, 若要在特定目录解压缩, 可以使用这个选项。

-d: 记录文件的差别;

-x或--extract或--get: 从备份文件中还原文件;
-t或--list: 列出备份文件的内容;
-z或--gzip或--ungzip: 通过gzip指令处理备份文件;
-Z或--compress或--uncompress: 通过compress指令处理备份文件;
-f<备份文件>或--file=<备份文件>: 指定备份文件;
-v或--verbose: 显示指令执行过程;
-r: 添加文件到已经压缩的文件;
-u: 添加改变了和现有的文件到已经存在的压缩文件;
-j: 支持bzip2解压文件;
-v: 显示操作过程;
-l: 文件系统边界设置;
-k: 保留原有文件不覆盖;
-m: 保留文件不被覆盖;
-w: 确认压缩文件的正确性;
-p或--same-permissions: 用原来的文件权限还原文件;
-P或--absolute-names: 文件名使用绝对名称, 不移除文件名称前的“/”号;
-N <日期格式> 或 --newer=<日期时间>: 只将较指定日期更新的文件保存到备份文件里;

案例:

将文件全部打包成tar包:

```
tar -cvf log.tar log2012.log 仅打包, 不压缩!  
tar -zcvf log.tar.gz log2012.log 打包后, 以 gzip 压缩  
tar -jcvf log.tar.bz2 log2012.log 打包后, 以 bzip2 压缩
```

在选项f之后的文件档名是自己取的, 我们习惯上都用 .tar 来作为辨识。
如果加z选项, 则以.tar.gz或.tgz来代表gzip压缩过的tar包; 如果加j选项, 则以.tar.bz2来作为tar包名。

查阅上述tar包内有哪些文件:

```
tar -ztvf log.tar.gz
```

由于我们使用 gzip 压缩的log.tar.gz, 所以要查阅log.tar.gz包内的文件时, 就得要加上z这个选项了。

将tar包解压缩:

```
tar -zxvf /opt/soft/test/log.tar.gz
```

在预设的情况下, 我们可以将压缩档在任何地方解开的

只将tar内的部分文件解压出来:

```
tar -zxvf /opt/soft/test/log30.tar.gz log2013.log
```

我可以透过tar tar -ztvf来查阅

包内的文件名称, 如果单只要一个文件, 就可以透过这个方式来解压部分文件!

文件备份下来, 并且保存其权限:

```
tar -zcvpf log31.tar.gz log2014.log log2015.log log2016.log
```

这个-p的属性是很重要的, 尤其是当您要保留原本文件的属性时。

在文件夹当中, 比某个日期新的文件才备份:

```
tar -N "2012/11/13" -zcvf log17.tar.gz test
```

备份文件夹内容是排除部分文件:

```
tar --exclude scf/service -zcvf scf.tar.gz scf/*
```

其实最简单的使用 tar 就只要记忆底下的方式即可:

压缩: tar -jcv -f filename.tar.bz2 要被压缩的文件或目录名称

查询: tar -jtv -f filename.tar.bz2

解压缩: tar -jxv -f filename.tar.bz2 -C 欲解压缩的目录

3.Linux软件安装管理

3-1软件包管理简介

linux 是一个底层系统,好多功能只能通过其它的软件来实现,就像window一样,我们需要安装各种各样的软件,如office,QQ等,才能满足我们的日常工作

linux 下的软件安装包,有两种:

第一种是源码包 (不适合初学者)

将原码编译成二进制包(RPM包, 系统默认包)

优点: 开源, 可以自由选择所需功能, 更加稳定高效, 卸载方便

缺点: 安装复杂, 容易出错, 且新手很难排查, 安装时间长

第二种是rpm包(二进制包), window 下的.exe或.msi 软件包其实就是二进制包

优点: 包管理简单, 几个命令就搞定, 安装速度快

缺点: 看不到源代码, 功能不如源码灵活, 依赖问题

软件安装方法:

rpm 命令

yum 命令

编译安装命令

脚本安装包 (其实安装的还是源码包, 或是 二进制包)

3-2rpm包

RPM 在哪, 我们的安装光盘中就有

RPM 命名规则:

httpd-2.2.15-15.el6.centos.1.i686.rpm

- httpd软件包名
- 2.2.15软件版本
- 15软件发布的次数
- el6.centos适合的Linux平台
- i686适合的硬件平台
- rpmrpm包扩展名

RPM包依赖性

树形依赖: a->b->c

环形依赖: a->b->c->a

模块依赖: 查询网站: www.rpmfind.net 最难的就是这种情况

由于依赖性问题, 因此又出了一个yum安装包, 自动安装依赖

安装之前, 有一个概念先讲:

包全名(安装/更新) 全新安装一个软件使用包全名

包名(卸载/查询) 已经安装好的包可以使用包名, 其是就是搜索 /var/lib/rpm 中的数据库

RPM 命令:

安装

rpm -ivh 包全名

选项:

- i: (install) 安装
- v: (verbose) 显示详细信息
- h: (hash) 显示进度
- nodeps 不检测依赖性

安装之前, 要先进入安装包目录, 或使用绝对路径

升级

rpm -Uvh 包全名

-U: (upgrade)升级

卸载

rpm -e 包名

卸载也有依赖性, 要按顺序卸载

查询

rpm -q 包名

rpm -qi 包名 查询包的详细信息

-i 查询软件信息(information)

-p 查询未安装包信息(package)

rpm -qa 查询全部已安装的包

rpm -p 查询未安装包

rpm -ql 包名 查询包中文件安装位置

-l: 列表(list)

-p: 查询未安装包信息 (package)

如: rpm -qa | grep httpd

RPM包默认安装路径	
/etc/	配置文件安装目录
/usr/bin/	可执行的命令安装目录
/usr/lib/	程序所使用的函数库保存位置
/usr/share/doc/	基本的软件使用手册保存位置
/usr/share/man/	帮助文件保存位置

rpm -qf 文件名 查看文件属于哪个包

3-3yum包

yum 在线安装

自动解决依赖

redhat 的yum 在线安装需要付费

yum 源文件 ls /etc/yum.repos.d/ 目录下有如下几个文件, 默认用的是红线文件配置源

```
-rw-r--r--. 1 root root 1991 8月 4 2015 CentOS-Base.repo
-rw-r--r--. 1 root root 647 8月 4 2015 CentOS-Debuginfo.repo
-rw-r--r--. 1 root root 289 8月 4 2015 CentOS-fasttrack.repo
-rw-r--r--. 1 root root 630 8月 4 2015 CentOS-Media.repo
-rw-r--r--. 1 root root 6259 8月 4 2015 CentOS-Vault.repo
```

打开 vi /etc/yum.repos.d/CentOS-Base.repo 可以看到如下的代码:

```
[base]
name=CentOS-$releasever - Base
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=os&infra=$infra
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
```

vi /etc/yum.repos.d/CentOS-Base.repo

- ◆ [base] 容器名称, 一定要放在[]中
- ◆ name 容器说明, 可以自己随便写
- ◆ mirrorlist 镜像站点, 这个可以注释掉
- ◆ baseurl 我们的yum源服务器的地址。默认是CentOS官方的yum源服务器, 是可以使用的, 如果你觉得慢可以改成你喜欢的yum源地址
- ◆ enabled 此容器是否生效, 如果不写或写成enable=1都是生效, 写成enable=0就是不生效
- ◆ gpgcheck 如果是1是指RPM的数字证书生效, 如果是0则不生效
- ◆ gpgkey 数字证书的公钥文件保存位置。不用修改

光盘搭建yum源 无法上网时使用光盘

步骤:

1. 挂载光盘:

```
mkdir /mnt/cdrom      创建挂载点
mount /dev/cdrom /mnt/cdrom  挂载光盘
```

2. 修网络yum源失效

将 /etc/yum.repos.d/CentOS-Base.repos 改名, 就失效

3. 修改 /etc/yum.repos.d/CentOS-Media.repo 文件

```
name=CentOS-$releasever - Media
baseurl=file:///media/CentOS/
file:///media/cdrom/
file:///media/cdrecorder/
gpgcheck=1
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-6
```

3-3-1yum命令

yum list 查询所有可用安装包

yum search 关键字 搜索服务器上所有和关键字相关的包

yum -y install 包名 安装命令

选项:

install 安装
-y 自动回答 yes

yum -y update 包名 升级命令

选项:

update 升级
-y 自动回答yes

yum -y remove 包名 卸载

选项:

```
remove 卸载  
-y 自动回答yes
```

备注:

最小化安装, 要什么装什么, 尽量不要卸载, 原因, 有可能把依赖卸载后, 把其它软件需要依赖的包也卸载了.. 包括系统依赖的包
很可能把系统搞奔溃了...

yum 软件组管理命令

yum group list 列出所有可用的软件组列表

yum groupinstall 软件组名 安装指定软件组, 组名可以由grouplist 查询出来

yum groupremove 软件组名 卸载指定软件组

```
LANG=en_US      改为英文  
LANG=zh_CN.utf8 改为中文
```

3-4源码包

源码包与RPM很重要的一个区别是, 安装位置不一样, 源码包是自己做主, RPM包是软件作者决定的

源码包不指定安装位置, 会使软件装得到处都是, 没办法卸载, 因为源码包没有卸载命令, 而是直接删除文件来卸载。而且可以卸载得很干净, 不会有任何垃圾文件

安装位置不同, 造成启动方式不同, 默认不能使用service 命令来设置启动方式

3-4-1源码包安装过程

安装准备:

1. 安装C语言编译器
2. 下载源码包, 一般去官方网站下载源码包

安装注意事项:

1. 源码保存位置: /usr/local/src/
2. 软件安装位置: /usr/local/
3. 如何确定安装过程报错:

安装过程停止

并出现error,warning 或 no 的提示

安装过程:

1. 首先下载安装包, 一般是tar.gz 后缀的源码包

比如: `wget http://mirror.bit.edu.cn/apache/httpd/httpd-2.2.32.tar.gz`

2. 解压源码包:

如: `tar -zxvf /usr/local/src/httpd-2.2.32.tar.gz`

3. 进入解压后的目录

如: `cd /usr/local/src/httpd-2.2.32`

4. `./configure` 软件配置与检查

定义需要的功能选项

检测系统环境是否符合安装要求

把定义好的功能选项和检测系统环境的信息都写入Makefile 文件, 用于后续的编译

5. `make` 编译

`make clean` 清除编译缓存(一般报错是使用, 清除后再次安装)

6. `make install` 编译安装

3-5脚本安装包

我们通过脚本安装包, 来安装nginx

准备工作

1. 关闭RPM包安装的httpd 和 MySQL
2. 保证yum源正常使用
3. 关闭SELinux 和 防火墙

我们实现一个demo: lnmp 的一键安装包:脚本下载地址:
<https://lnmp.org>

4.Linux权限管理

4-1文件基本权限

我们在文件管理时, 讲解的 `chmod` 就是对文件基本权限修改命令, 本节课做一些补充:

通过 `ls -l`

```
[echo@localhost ~]$ ll
总用量 36
drwxr-xr-x. 2 echo echo 4096 6月 30 22:02 a.text
drwxr-xr-x. 2 echo echo 4096 6月 29 23:42 公共的
drwxr-xr-x. 2 echo echo 4096 6月 29 23:42 模板
drwxr-xr-x. 2 echo echo 4096 6月 29 23:42 视频
drwxr-xr-x. 2 echo echo 4096 6月 29 23:42 图片
drwxr-xr-x. 2 echo echo 4096 6月 29 23:42 文档
```

被分为8个部份

第一部份: 文件类型: 常见3种类型:

d 目录

- 文件

l 连接

第二部份: 用户、组、其它 三种身份的权限 (r表示读, w表示写,x表示执行)

第三部份: 表示文件连接数(包括硬连接和软连接)

第四部份：文件所属用户

第五部份：文件所属组

第六部份：文件大小

第七部份：最后修改时间

第八部份：文件名称

权限对文件的作用

- **r**：读取文件内容 (`cat more head tail`)
- **w**：编辑、新增、修改文件内容 (`vi echo`)
 - 但是不包含删除文件
- **x**：可执行

权限对目录的作用

- **r**：可以查询目录下文件名 (`ls`)
- **w**：具有修改目录结构的权限。如新建文件和目录，删除此目录下文件和目录，重命名此目录下文件和目录，剪切 (`touch rm mv cp`)
- **x**：可以进入目录 (`cd`)

4-2默认权限

修改默认权限:

umask 查看默认权限

0022

每一位0, 是特殊权限

文件的默认的权限:

1. 文件默认不能建立为执行文件, 必须手工加执行权限
2. 所以文件默认权限最大为666
3. 默认权限需要换算成字符再相减
4. 建立文件之后的默认权限, 为666减去umask值

如:

文件默认最大权限666 umask值022

-rw-rw-rw- 减去 -----w--w- 等于 -rw-r--r--

文件默认最大权限666 umask值033

-rw-rw-rw- 减去 -----wx-wx 等于 -rw-r--r--

目录的默认权限最大为777

默认权限需要换算成字母再相减

建立文件之后的默认权限, 为777减去umask值

如:

目录最大权限为 777, umask值为 022

-rwxrwxrwx 减去 -----w--w- 等于 -rwxr-xr-x

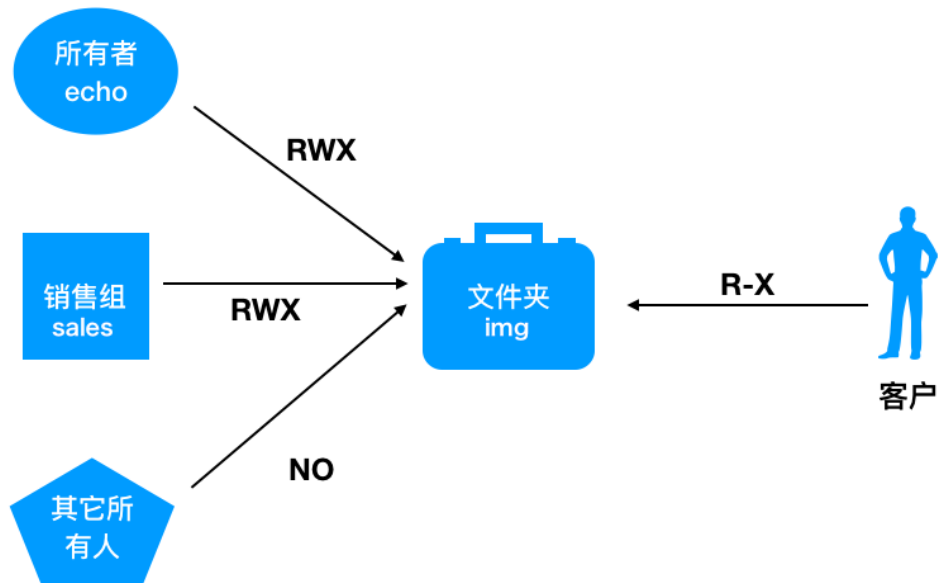
umask临时修改: umask 0002

umask永久个性: vi /etc/profile

4-3ACL权限

ACL权限: 解决 用户身份不主足的情况:

如下案例:



要求: 文件夹对于 echo 拥有读写执行, 销售部拥有: 读写执行, 其它所有人没有任何权限, 现在我想让客户: 有查看文件夹的内容, 即读和执行权限

查看分区ACL权限是否开启:

dumpe2fs -h /dev/sda5 查看指定分区详细文件系统信息的命令

-h 仅显示超级块中信息, 而不显示磁盘块组的详细信息

步骤: 我们看下我们目录有哪些分区: df

```
[echo@bogon ~]$ df
Filesystem            1K-blocks    Used Available Use% Mounted on
/dev/mapper/VolGroup-root 16273832 3012160 12428344 20% / 根分区
tmpfs                  502384      0      502384  0% /dev/shm
/dev/sda1              194241     36315   147686  20% /boot
/dev/mapper/VolGroup-home 1983056     3860   1876796  1% /home
```

```
[echo@bogon ~]$ sudo dumpe2fs -h /dev/mapper/VolGroup-root
dumpe2fs 1.41.12 (17-May-2010)
Filesystem volume name: <none>
Last mounted on: /
Filesystem UUID: d766fd7d-beed-4c53-9128-6ca3c09dde87
Filesystem magic number: 0xEF53
Filesystem revision #: 1 (dynamic)
Filesystem features: has_journal ext_attr resize_inode dir_index filetype needs_recovery ex
e_file uninit_bg dir_nlink extra_isize
Filesystem flags: signed_directory_hash
Default mount options: user_xattr acl
Filesystem state: clean
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: 1042432
Block count: 4166656
Reserved block count: 208332
Free blocks: 3315472
Free inodes: 951054
```

可以看到我们的根分区已经开启了 **acl** 权限

如果上面的分区不支持ACL,

临时开启: `mount -o remount,acl /`

永久开启分区ACL权限: `vi /etc/fstab`

`UUID=c2ca6f57-b15c-43ea-bca0-f239083d8bd2 / ext4 defaults,acl 1 1`

`mount -o remount /` #重新挂载文件系统或重启系统, 使修改生效

4-3-1查看ACL权限

`getfacl 文件名` #查看acl权限

4-3-2设定ACL权限

`setfacl [选项] 文件名`

选项:

-m 设定ACL权限

-x 删除指定的ACL权限

-b 删除所有ACL权限

-d 设定默认ACL权限

-k 删除默认ACL权限

-R 递归设定ACL权限

4-3-3DEMO

`groupadd sales` #添加销售组

```
useradd customer #添加客户用户
passwd customer #设置customer密码为 qaz1234qaz
mkdir /home/img #添加 img 文件夹
```

```
chown echo:sales /home/img #将img 所有者设置为echo 所有组设置为 sales
chmod 770 /home/img
```

使用sales登录, 检查当前情况下:img 文件夹权限

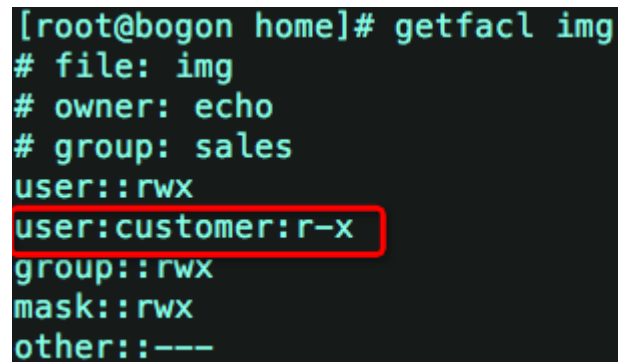
```
su customer
ls /home/img
cd /home/img
```

可以发现, 对 img 无任何权限

```
su root #切换为root用户
```

```
setfacl -m u:customer:rx /home/img #给用户 sale 添加acl 权限为 读和执行权限
```

```
getfacl img
```



```
[root@bogon home]# getfacl img
# file: img
# owner: echo
# group: sales
user::rwx
user:customer:r-x
group::rwx
mask::rwx
other::---
```

4-3-4删除ACL权限

```
setfacl -x u:用户名 文件名 #删除指定用户的ACL权限
```

```
setfacl -x g:组名 文件名 #删除指定用户组的ACL权限
```

setfacl -b 文件名 #删除文件的所有ACL权限

4-3-5递归ACL权限

setfacl -m u:用户名:权限 -R 文件名 #设置/home/img下面所有文件都有读和执行权限, 即递归设置
如:

setfacl -m u:echo:rx -R /home/img #设置/home/img下面所有文件都有读和执行权限, 即递归设置

4-4sudo权限

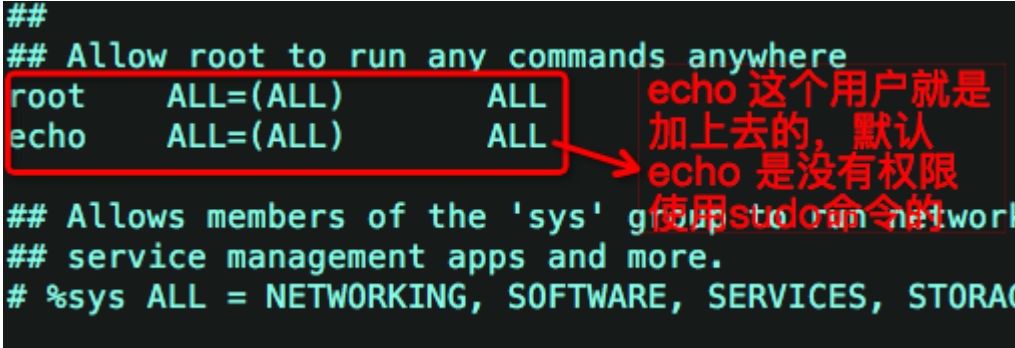
sudo权限概念:

就是把本来只能超级用户执行的命令赋予普通用户执行

sudo 的操作对象是系统命令

不是所有的用户都能使用sudo 命令, 是否能够使用sudo命令, 得看是否在配置文件: /etc/sudoers
文件中, 配置了使用

打开 /etc/sudoers 或使用 visodu 命令打开



Allow root to run any commands anywhere
root ALL=(ALL) ALL
echo ALL=(ALL) ALL
Allows members of the 'sys' group to run network
service management apps and more.
%sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE

echo 这个用户就是
加上去的, 默认
echo 是没有权限
使用sudo命令的

找到里面:

root ALL=(ALL) ALL

#用户名 被管理主机的地址=(可使用的身份)授权命令(绝对路径)

%wheel ALL=(ALL) ALL

##组名 被管理主机的地址=(可使用的身份)授权命令(绝对路径)

查看 sudo 能执行的命令: `sudo -l`

备注: `echo` `All=(ALL)` `ALL` 这样的配置是很危险的, 这样意为着, `echo` 有跟root用户一样的权限, 因此我们一般会设置为具体的命令:
如:

```
echo ALL=/usr/sbin/useradd
echo ALL=/usr/bin/passwd [A-Za-z]*, !/usr/bin/passwd "", !/usr/bin/pass root
```

4-5 文件特殊权限

1. setUID
2. setGID

4-5-1 setUID

SetUID是一个危险的命令, 最好不要轻易使用此命令:

只有可以执行的二进制程序才能设定SetUID权限

命令执行者要对该程序拥有x(执行)权限

命令执行者在执行该程序时获得该程序文件属主的身份(在执行程序的过程中灵魂附体为文件的属主)

SetUID 权限只在该程序执行过程中有效, 也就是说身份改变只在程序执行过程中有效

如: `/usr/bin/passwd`

```
[echo@bogon ~]$ ll /usr/bin/passwd
-rwsr-xr-x. 1 root root 30768 2月 22 2012 /usr/bin/passwd
[echo@bogon ~]$
```

`passwd` 有特殊权限, s

```
[echo@bogon ~]$ ll /etc/shadow
-----. 1 root root 1237 7月 1 08:34 /etc/shadow
```

上图可以看出：`shadow` 是除了root
以为，其它用户都没有权限修改，但为什么我们以其它用户登录时，可以修改自己密码呢??

其实就是因为 `/usr/bin/passwd` 这个命令有 `s` 权限 他在运行时会获得 `/usr/bin/passwd`
这个文件所有者的权限，而这个文件的所有者是root，因此在执行 `/usr/bin/passwd`
文件时，就相当于有了root权限，因此就能修改密码成功了

设置SetUID 权限命令：

`chmod 4775 文件名`

或

`chmod u+s 文件名`

取消SetUID

`chmod 0775 文件名`

或

`chmod u-s 文件名`

4-5-2setGID

SetGID

普通用户必须对此目录原有r和x权限，才能进入目录

普通用户在此目录中的有效组会变成此目录的属组

若普通用户对此目录有w权限时，新建的文件默认属组是这个目录的属组

SetGID权限同样只是在该程序执行过程中有效，也就是说，组身份改变只在执行过程中有效

如：`locate` 命令 其实是搜索 `/var/lib/mlocate/mlocate.db` 文件来搜索文件


```
[echo@bogon ~]$ sudo ls -l /var/lib/mlocate/mlocate.db
-rw-r-----. 1 root slocate 2208536 7月  1 04:17 /var/lib/mlocate/mlocate.db
[echo@bogon ~]$
```

普通用户没有权限查看 mlocate.db文件
但我们却可以使用locate 命令，即可以来查看 mlocate.db文件

```
[echo@bogon ~]$ sudo ls -l /usr/bin/locate
-rwx--s--x. 1 root slocate 38464 3月  12 2015 /usr/bin/locate
[echo@bogon ~]$
```

拥有SetGID权限

四、小节

2. 课堂练习
3. 课后练习
4. 资料扩展