

第一课 上云第一课 冬季实战营第一期

Linux
操作系统实战入门

云起实验室冬季实战营

云起实验室冬季实战营

云起实验室冬季实战营





前往云起实验室，基于实际场景
深度学习并体验云产品



阿里云开发者“藏经阁”
海量电子手册免费下载

目录

动手实战--初识上云基础，动手实操 ECS 云服务器新手上路	4
动手实战--Linux 系统管理入门深入解析动手实操	36
动手实战--Linux 磁盘管理入门深入解析动手实操	55
动手实战--Linux 文件与权限入门深入解析动手实操	64
动手实战--Linux 文件管理入门深入解析动手实操	80
用户反馈	89

动手实战--初识上云基础，动手实操 ECS 云服务器新手上路

体验简介

本场景将提供一台配置了 Aliyun Linux 2 的 ECS 实例（云服务器）。您可以参考本教程学习 Linux 系统中的文本编辑工具 Vim 以及文本处理命令。

背景知识

云服务器 ECS

云服务器（Elastic Compute Service，简称 ECS）是阿里云提供的性能卓越、稳定可靠、弹性扩展的 IaaS（Infrastructure as a Service）级别云计算服务。云服务器 ECS 免去了您采购 IT 硬件的前期准备，让您像使用水、电、天然气等公共资源一样便捷、高效地使用服务器，实现计算资源的即开即用和弹性伸缩。阿里云 ECS 持续提供创新型服务器，解决多种业务需求，助力您的业务发展。

Aliyun Linux 2

Aliyun Linux 2 是阿里云推出的下一代 Linux 发行版，它为云上应用程序环境提供 Linux 社区的最新增强功能，在提供云上最佳用户体验的同时，也针对阿里云基础设施做了深度的优化。Aliyun Linux 2 OS 镜像可以运行在阿里云全规格系列 VM 实例上，包括弹性裸金属服务器（神龙）。

Vim

Vim 是从 vi 发展出来的一个文本编辑器。代码补全、编译及错误跳转等方便编程的功能特别丰富，在程序员中被广泛使用，和 Emacs 并列成为类 Unix 系统用户最喜欢的文本编辑器。Vim 的设计理念是命令的组合。用户学习了各种各样的文本间移动、跳转的命令和其他的普通模式

的编辑命令，并且能够灵活组合使用的话，能够比那些没有模式的编辑器更加高效的进行文本编辑。同时 Vim 与很多快捷键设置和正则表达式类似,可以辅助记忆。并且 Vim 针对程序员做了优化。

目录一：文本编辑工具 Vim

文本编辑工具 Vim

vim 的三种操作模式

vim 有三种操作模式，分别是命令模式（Command mode）、输入模式（Insert mode）和底线命令模式（Last line mode）。

三种模式切换快捷键：

模式	快捷键
命令模式	ESC
输入模式	i 或 a
底线命令模式	:

1、命令模式

在命令模式中控制光标移动和输入命令，可对文本进行复制、粘贴、删除和查找等工作。

使用命令 `vim filename` 后进入编辑器视图后，默认模式就是命令模式，此时敲击键盘字母会被识别为一个命令，例如在键盘上连续敲击两次 `d`，就会删除光标所在行。

以下是在命令模式中常用的快捷操作：

操作	快捷键
光标左移	h
光标右移	l (小写 L)
光标上移	k
光标下移	j
光标移动到下一个单词	w
光标移动到上一个单词	b
移动游标到第 n 行	nG
移动游标到第一行	gg
移动游标到最后一行	G
快速回到上一次光标所在位置	Ctrl+o
删除当前字符	x
删除前一个字符	X
删除整行	dd
删除一个单词	dw 或 daw
删除至行尾	d\$或 D
删除至行首	d^
删除到文档末尾	dG
删除至文档首部	d1G
删除 n 行	ndd
删除 n 个连续字符	nx

操作	快捷键
将光标所在位置字母变成大写或小写	~
复制光标所在的整行	yy (3yy 表示复制 3 行)
粘贴至光标后 (下)	p
粘贴至光标前 (上)	P
剪切	dd
交换上下行	ddp
替换整行，即删除光标所在行并进入插入模式	cc
撤销一次或 n 次操作	u{n}
撤销当前行的所有修改	U
恢复撤销操作	Ctrl+r
整行将向右缩进	>>
整行将向左退回	<<
若档案没有更动，则不储存离开，若档案已经被更动过，则储存后离开	ZZ

2、输入模式

在命令模式下按 i 或 a 键就进入了输入模式，在输入模式下，您可以正常的使用键盘按键对文本进行插入和删除等操作。

3、底线命令模式

在命令模式下按:键就进入了底线命令模式，在底线命令模式中 can 输入单个或多个字符的命令。

以下是底线命令模式中常用的快捷操作：

操作	命令
保存	:w
退出	:q
保存并退出	:wq (:wq!表示强制保存退出)
将文件另存为其他文件名	:w new_filename
显示行号	:set nu
取消行号	:set nonu
使本行内容居中	:ce
使本行文本靠右	:ri
使本行内容靠左	:le
向光标之下寻找一个名称为 word 的字符串	:/word
向光标之上寻找一个字符串名称为 word 的字符串	:?word
重复前一个搜寻的动作	:n
从第一行到最后一行寻找 word1 字符串, 并将该字符串取代为 word2	:1,\$s/word1/word2/g 或 :%s/word1/word2/g

使用示例

1、List item

新建一个文件并进入 vim 命令模式。


```
vim 静夜思.txt
```



2、按下 i 进入输入模式，输入《静夜思》的诗名。



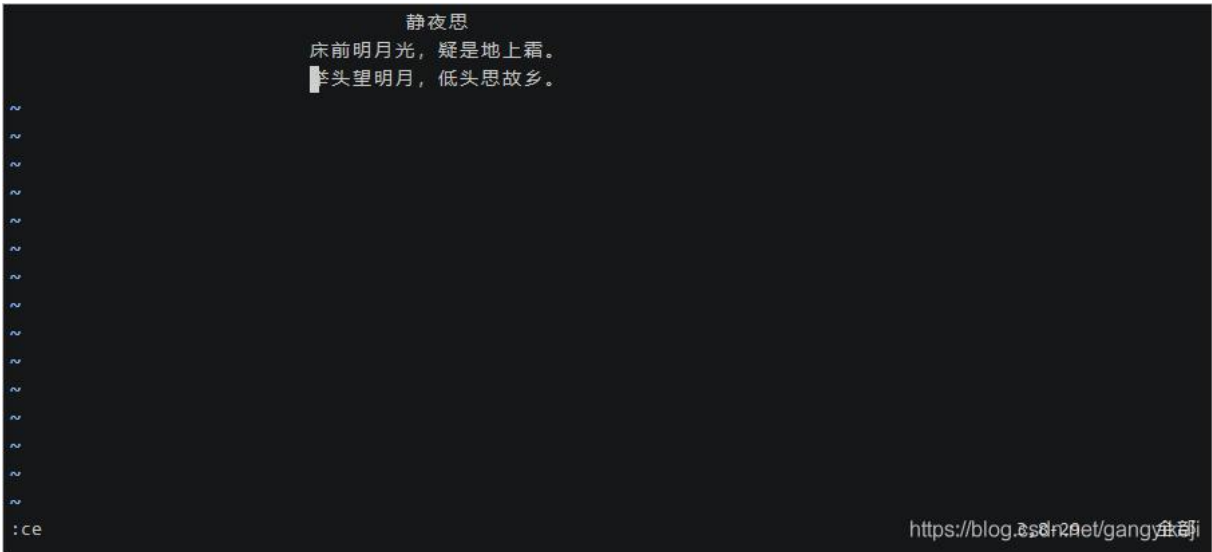
3、按下 ESC 键回到命令模式，并输入底线命令:ce，使诗名居中。



6、按下 **o** 键换行并进入输入模式，输入第二行诗。



7、按下 **ECS** 键回到命令模式，并输入底线命令:ce，使第二行诗居中。



8、在命令模式中执行底线命令:wq 离开 vim。

目录二：文本文件查看命令

cat

命令描述：cat 命令用于查看内容较少的纯文本文件。

命令格式：cat [选项] [文件]。

命令参数说明：

参数	说明
-n 或--number	显示行号
-b 或--number-nonblank	显示行号，但是不对空白行进行编号
-s 或--squeeze-blank	当遇到有连续两行以上的空白行，只显示一行的空白行

命令使用示例：

1、将一个自增序列写入 test.txt 文件中。

```
for i in $(seq 1 10); do echo $i >> test.txt ; done
```

2、查看文件内容。

```
cat test.txt
```

命令输出结果：

```
[root@myhost ~]# cat test.txt
1
2
3
4
5
6
7
8
9
10
[root@myhost ~]#
```

<https://blog.csdn.net/gangyikeji>

3、将文件内容清空。

```
cat /dev/null > test.txt
```

4、再次检查文件内容。

```
cat test.txt
```

命令输出结果：

```
[root@myhost ~]# cat /dev/null > test.txt
[root@myhost ~]# cat test.txt
[root@myhost ~]#
```

more

命令描述：more 命令从前向后分页显示文件内容。

常用操作命令：

操作	作用
Enter	向下 n 行，n 需要定义，默认为 1 行
Ctrl+F 或空格键（Space）	向下滚动一页
Ctrl+B	向上滚动一页
=	输出当前行的行号
!命令	调用 Shell 执行命令
q	退出 more

命令使用示例：

从第 20 行开始分页查看系统日志文件/var/log/messages。

```
more +20 /var/log/messages
```

命令输出结果：

```

Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped Apply the settings specified in cloud-config.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped target Cloud-config availability.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopping Dynamic System Tuning Daemon...
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopping LSB: aegis update....
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopping Authorization Manager...
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopping aliyun-assist...
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopping Postfix Mail Transport Agent...
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Removed slice system-selinux\x2dpolicy\x2dmigrate\x2dlocal\x2dchanges.slice.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped NTP client/server.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped OpenSSH server daemon.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped Job spooling tools.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped Command Scheduler.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped Getty on tty1.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped Serial Getty on ttyS0.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped Authorization Manager.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped aliyun-assist.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped Session 1 of user root.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Removed slice User Slice of root.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopping Login Service...
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Removed slice system-serial\x2dgetty.slice.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Starting Show Plymouth Power Off Screen...
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Removed slice system-getty.slice.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopping Permit User Sessions...
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped Permit User Sessions.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd-udevd: Network interface NamePolicy= disabled on kernel command line, ignoring.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped Login Service.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ aegis: Aegis stopped
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped LSB: aegis update..
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped target Remote File Systems.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped target Network is Online.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Started Show Plymouth Power Off Screen.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped Dynamic System Tuning Daemon.
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopping D-Bus System Message Bus...
Mar 29 12:18:56 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped D-Bus System Message Bus.
Mar 29 12:18:57 i2bp1iqhhs6ve1zk2706jsZ systemd: postfix.service: control process exited, code=exited status=1
Mar 29 12:18:57 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped Postfix Mail Transport Agent.
Mar 29 12:18:57 i2bp1iqhhs6ve1zk2706jsZ systemd: Unit postfix.service entered failed state.
Mar 29 12:18:57 i2bp1iqhhs6ve1zk2706jsZ systemd: postfix.service failed.
Mar 29 12:18:57 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped target Network.
Mar 29 12:18:57 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopping LSB: Bring up/down networking...
Mar 29 12:18:57 i2bp1iqhhs6ve1zk2706jsZ network: Shutting down interface eth0: [ OK ]
Mar 29 12:18:58 i2bp1iqhhs6ve1zk2706jsZ network: Shutting down loopback interface: [ OK ]
Mar 29 12:18:58 i2bp1iqhhs6ve1zk2706jsZ systemd: Stopped LSB: Bring up/down networking.
--More-- (2%)

```

<https://blog.csdn.net/gangyikeji>

less

命令描述: less 命令可以对文件或其它输出进行分页显示, 与 more 命令相似, 但使用 less 可以随意浏览文件, 而 more 仅能向前移动, 却不能向后移动。

命令格式: less [参数] 文件。

命令参数说明:

参数	说明
-e	当文件显示结束后, 自动离开
-m	显示类似 more 命令的百分比
-N	显示每行的行号
-S	显示连续空行为一行

命令常用操作：

快捷键	说明
/字符串	向下搜索字符串
?字符串	向上搜索字符串
n	重复前一个搜索
N	反向重复前一个搜索
b 或 pageup 键	向上翻一页
空格键或 pagedown 键	向下翻一页
u	向前翻半页
d	向后翻半页
y	向前滚动一行
回车键	向后滚动一行
q	退出 less 命令

命令使用示例：

查看命令历史使用记录并通过 less 分页显示。

```
history | less
```

head

命令描述：head 命令用于查看文件开头指定行数的内容。

命令格式：head [参数] [文件]。

命令参数说明：

参数	说明
-n [行数]	显示开头指定行的文件内容，默认为 10
-c [字符数]	显示开头指定个数的字符数
-q	不显示文件名字信息，适用于多个文件，多文件时默认会显示文件名

命令使用示例：

查看/etc/passwd 文件的前 5 行内容。

```
head -5 /etc/passwd
```

命令输出结果：

```
[root@myhost ~]# head -5 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
[root@myhost ~]#
```

tail

命令描述：tail 命令用于查看文档的后 N 行或持续刷新内容。

命令格式：tail [参数] [文件]。

命令参数说明：

参数	说明
-f	显示文件最新追加的内容
-q	当有多个文件参数时，不输出各个文件名
-v	当有多个文件参数时，总是输出各个文件名
-c [字节数]	显示文件的尾部 n 个字节内容
-n [行数]	显示文件的尾部 n 行内容

命令使用示例：

查看/var/log/messages 系统日志文件的最新 10 行，并保持实时刷新。

```
tail -f -n 10 /var/log/messages
```

```
[root@myhost ~]# tail -f -n 10 /var/log/messages
May 12 14:04:13 chronyd[792]: System clock wrong by -0.976247 seconds, adjustment started
May 12 14:10:01 systemd: Started Session 36 of user root.
May 12 14:14:47 auditd[555]: Audit daemon rotating log files
May 12 14:20:01 systemd: Started Session 37 of user root.
May 12 14:30:01 systemd: Started Session 38 of user root.
May 12 14:37:36 systemd-logind: New session 39 of user root.
May 12 14:37:36 systemd: Started Session 39 of user root.
May 12 14:40:01 systemd: Started Session 40 of user root.
May 12 14:40:25 systemd-logind: Removed session 39.
May 12 14:50:01 systemd: Started Session 41 of user root.
https://blog.csdn.net/gangyikeji
```

按 ctrl+c 键退出文本实时查看界面。

stat

命令描述：用来显示文件的详细信息，包括 inode、atime、mtime、ctime 等。

命令使用示例：

查看/etc/passwd 文件的详细信息。

```
stat /etc/passwd
```

命令输出结果：

```
[root@myhost ~]# stat /etc/passwd
 文件："/etc/passwd"
 大小：956           块：8           IO 块：4096   普通文件
设备：fd01h/64769d   Inode：1054963   硬链接：1
权限：(0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
最近访问：2020-05-12 13:47:33.302835606 +0800
最近更改：2020-05-12 13:47:33.293835311 +0800
最近改动：2020-05-12 13:47:33.294835344 +0800
创建时间：-
```

WC

命令描述：wc 命令用于统计指定文本的行数、字数、字节数。

命令格式：wc [参数] [文件]。

命令参数说明：

参数	说明
-l	只显示行数
-w	只显示单词数
-C	只显示字节数

命令使用示例：

统计/etc/passwd 文件的行数。

```
wc -l /etc/passwd
```

命令输出结果：

```
[root@myhost ~]# wc -l /etc/passwd
22 /etc/passwd
```

file

命令描述：file 命令用于辨识文件类型。

命令格式：file [参数] [文件]。

命令参数说明：

参数	说明
-b	列出辨识结果时，不显示文件名称
-c	详细显示指令执行过程，便于排错或分析程序执行的情形
-f [文件]	指定名称文件，其内容有一个或多个文件名称时，让 file 依序辨识这些文件，格式为每列一个文件名称
-L	直接显示符号连接所指向的文件类别

命令使用示例：

查看/var/log/messages 文件的文件类型。

```
file /var/log/messages
```

命令输出结果：

```
[root@myhost ~]# file /var/log/messages
/var/log/messages: UTF-8 Unicode text, with very long lines
```

diff

命令描述: diff 命令用于比较文件的差异。

命令使用示例:

1、构造两个相似的文件

```
echo -e '第一行\n 第二行\n 我是 log1 第 3 行\n 第四行\n 第五行\n 第六行' > 1.log  
echo -e '第一行\n 第二行\n 我是 log2 第 3 行\n 第四行' > 2.log
```

2、分别查看两个文件

```
[root@myhost ~]# echo -e '第一行\n第二行\n我是log1第3行\n第四行\n第五行\n第六行' > 1.log  
[root@myhost ~]# echo -e '第一行\n第二行\n我是log2第3行\n第四行' > 2.log  
[root@myhost ~]# cat 1.log  
第一行  
第二行  
我是log1第3行  
第四行  
第五行  
第六行  
[root@myhost ~]# cat 2.log  
第一行  
第二行  
我是log2第3行  
第四行  
[root@myhost ~]#
```

<https://blog.csdn.net/gangyikeji>

3、使用 diff 查看两个文件的差异

```
[root@myhost ~]# diff 1.log 2.log  
3c3  
< 我是log1第3行  
---  
> 我是log2第3行  
5,6d4  
< 第五行  
< 第六行  
[root@myhost ~]#
```

对比结果中的 3c3 表示两个文件在第 3 行有不同，5,6d4 表示 2.log 文件相比 1.log 文件在第 4 行处开始少了 1.log 文件的第 5 和第 6 行。

文本文件处理命令

grep

命令描述：grep 命令用于查找文件里符合条件的字符串。

grep 全称是 Global Regular Expression Print，表示全局正则表达式版本，它能使用正则表达式搜索文本，并把匹配的行打印出来。

在 Shell 脚本中，grep 通过返回一个状态值来表示搜索的状态：

- 0：匹配成功。
- 1：匹配失败。
- 2：搜索的文件不存在。
- 命令格式：grep [参数] [正则表达式] [文件]。

命令常用参数说明：

参数	说明
-c 或 --count	计算符合样式的列数
-d recurse 或 -r	指定要查找的是目录而非文件
-e [范本样式]	指定字符串做为查找文件内容的样式
-E 或 --extended-regexp	将样式为延伸的正则表达式来使用
-G 或 --basic-regexp	将样式视为普通的表示法来使用
-i 或 --ignore-case	忽略字符大小写的差别

参数	说明
-n 或 --line-number	在显示符合样式的那一行之前，标示出该行的列数编号
-v 或 --revert-match	显示不包含匹配文本的所有行

命令使用示例：

查看 sshd 服务配置文件中监听端口配置所在行编号。

```
grep -n Port /etc/ssh/ssh_config
```

命令输出结果：

```
[root@myhost ~]# grep -n Port /etc/ssh/ssh_config
41:#    Port 22
```

查询字符串在文本中出现的行数。

```
grep -c localhost /etc/hosts
```

命令输出结果：

```
[root@myhost ~]# cat /etc/hosts
::1    localhost        localhost.localdomain  localhost6        localhost6.localdomain6
127.0.0.1    localhost        localhost.localdomain  localhost4        localhost4.localdomain4

172.24.79.251    izf8zgikm9wkbgdcm1t5zgZ izf8zgikm9wkbgdcm1t5zgZ

[root@myhost ~]# grep -c localhost /etc/hosts
2
```

反向查找，不显示符合条件的行。

```
ps -ef | grep sshd
ps -ef | grep -v grep | grep sshd
```

命令输出结果：

```
[root@myhost ~]# ps -ef | grep sshd
root      1239      1  0 09:11 ?        00:00:00 /usr/sbin/sshd -D
root      1661 18445  0 15:34 pts/0    00:00:00 grep --color=auto sshd
root      18443  1239  0 11:29 ?        00:00:33 sshd: root@pts/0,pts/1
[root@myhost ~]# ps -ef | grep -v grep | grep sshd
root      1239      1  0 09:11 ?        00:00:00 /usr/sbin/sshd -D
root      18443  1239  0 11:29 ?        00:00:33 sshd: root@pts/0,pts/1
```

以递归的方式查找目录下含有关键字的文件。

```
grep -r *.sh /etc
```

命令输出结果：

使用正则表达式匹配 httpd 配置文件中异常状态码响应的相关配置。

```
grep 'ntp[0-9].aliyun.com' /etc/ntp.conf
```

命令输出结果：

```
[root@myhost ~]# grep -r *.sh /etc
/etc/bash_completion.d/git:      create-ignore,--*|propget,--*|proplist,--*|show-ignore,--*|\
匹配到二进制文件 /etc/udev/hwdb.bin
/etc/NetworkManager/dispatcher.d/11-dhclient:  for f in $ETCDIR/dhclient.d/*.sh; do
/etc/bashrc:  for i in /etc/profile.d/*.sh; do
/etc/profile:for i in /etc/profile.d/*.sh /etc/profile.d/sh.local ; do
```

sed

命令描述：sed 是一种流编辑器，它是文本处理中非常中的工具，能够完美的配合正则表达式使用。

- 1、处理时，把当前处理的行存储在临时缓冲区中，称为模式空间（pattern space）。
- 2、接着用 sed 命令处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。
- 3、接着处理下一行，这样不断重复，直到文件末尾。

注意：

- sed 命令不会修改原文件，例如删除命令只表示某些行不打印输出，而不是从原文件中删去。
- 如果要改变源文件，需要使用-i 选项。
- 命令格式：sed [参数] [动作] [文件]。

参数说明：

参数	说明
-e [script]	执行多个 script
-f [script 文件]	执行指定 script 文件
-n	仅显示 script 处理后的结果
-i	输出到原文件，静默执行（修改原文件）

动作说明：

动作	说明
a	在行后面增加内容
c	替换行
d	删除行
i	在行前面插入
p	打印相关的行
s	替换内容

命令使用示例：

删除第 3 行到最后一行内容。

```
sed '3,$d' /etc/passwd
```

命令输出结果：

```
[root@myhost ~]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
postfix:x:89:89:/:/var/spool/postfix:/sbin/nologin
chrony:x:998:996:/:/var/lib/chrony:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
tcpdump:x:72:72:/:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
[root@myhost ~]# sed '3,$d' /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
[root@myhost ~]#
```

<https://blog.csdn.net/gangyikeji>

在最后一行新增行。

```
sed '$a admin:x:1000:1000:admin:/home/admin:/bin/bash' /etc/passwd
```

命令输出结果:

```
[root@myhost ~]# sed '$a admin:x:1000:1000:admin:/home/admin:/bin/bash' /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
postfix:x:89:89:/:/var/spool/postfix:/sbin/nologin
chrony:x:998:996:/:/var/lib/chrony:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
tcpdump:x:72:72:/:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
admin:x:1000:1000:admin:/home/admin:/bin/bash
[root@myhost ~]#
```

<https://blog.csdn.net/gangyikeji>

替换内容。

```
sed 's/SELINUX=disabled/SELINUX=enforcing/' /etc/selinux/config
```

命令输出结果:

```
[root@myhost ~]# cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three values:
#   targeted - Targeted processes are protected,
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted

[root@myhost ~]# sed 's/SELINUX=disabled/SELINUX=enforcing/' /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three values:
#   targeted - Targeted processes are protected,
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted

[root@myhost ~]#
```

<https://blog.csdn.net/gangyikeji>

替换行。

```
sed '1c abcdefg' /etc/passwd
```

命令输出结果：

```
[root@myhost ~]# sed '1c abcdefg' /etc/passwd
abcdefg
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:999:998:User for polkitd:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
postfix:x:89:89:/:/var/spool/postfix:/sbin/nologin
chrony:x:998:996:/:/var/lib/chrony:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
tcpdump:x:72:72:/:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
[root@myhost ~]#
```

<https://blog.csdn.net/gangyikeji>

awk

命令描述: 和 sed 命令类似, awk 命令也是逐行扫描文件 (从第 1 行到最后一行), 寻找含有目标文本的行, 如果匹配成功, 则会在该行上执行用户想要的操作; 反之, 则不对行做任何处理。

命令格式: awk [参数] [脚本] [文件]。

参数说明:

参数	说明
-F fs	指定以 fs 作为输入行的分隔符，awk 命令默认分隔符为空格或制表符
-f file	读取 awk 脚本
-v val=val	在执行处理过程之前，设置一个变量 var，并给其设置初始值为 val

内置变量：

变量	用途
FS	字段分隔符
\$n	指定分隔的第 n 个字段，如\$1、\$3 分别表示第 1、第三列
\$0	当前读入的整行文本内容
NF	记录当前处理行的字段个数（列数）
NR	记录当前已读入的行数
FNR	当前行在源文件中的行号

awk 中还可以指定脚本命令的运行时机。默认情况下，awk 会从输入中读取一行文本，然后针对该行的数据执行程序脚本，但有时可能需要在处理数据前运行一些脚本命令，这就需要使用 BEGIN 关键字，BEGIN 会在 awk 读取数据前强制执行该关键字后指定的脚本命令。

和 BEGIN 关键字相对应，END 关键字允许我们指定一些脚本命令，awk 会在读完数据后执行它们。

命令使用示例：

查看本机 IP 地址。

```
ifconfig eth0 |awk '/inet/{print $2}'
```


命令输出结果:

```
[root@myhost ~]# ifconfig eth0 |awk '/inet/{print $2}'  
172.24.79.251  
[root@myhost ~]#
```

查看本机剩余磁盘容量。

```
df -h |awk '/\$/ {print $4}'
```

命令输出结果:

```
[root@myhost ~]# df -h |awk '/\$/ {print $4}'  
36G  
[root@myhost ~]#
```

统计系统用户个数。

```
awk -F: '$3<1000{x++} END{print x}' /etc/passwd
```

命令输出结果:

```
[root@myhost ~]# awk -F: '$3<1000{x++} END{print x}' /etc/passwd  
22  
[root@myhost ~]#
```

输出其中登录 Shell 不以 nologin 结尾 (对第 7 个字段做!~反向匹配) 的用户名、登录 Shell 信息。

```
awk -F: '$7!~/nologin$/{print $1,$7}' /etc/passwd
```

命令输出结果：

```
[root@myhost ~]# awk -F: '$7!~/nologin$/{print $1,$7}' /etc/passwd
root /bin/bash
sync /bin/sync
shutdown /sbin/shutdown
halt /sbin/halt
[root@myhost ~]#
```

输出/etc/passwd 文件中前三行记录的用户名和用户 uid。

```
head -3 /etc/passwd | awk 'BEGIN{FS=":";print "name\tuid"}{print $1,"\t"$3}END{print "sum lines "NR}'
```

命令输出结果：

```
[root@myhost ~]# head -3 /etc/passwd | awk 'BEGIN{FS=":";print "name\tuid"}{print $1,"\t"$3}END{print "sum lines "NR}'
name    uid
root    0
bin     1
daemon  2
sum lines 3
[root@myhost ~]#
```

查看 tcp 连接数。

```
netstat -na | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a}]'
```

命令输出结果：

```
[root@myhost ~]# netstat -na | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a}]'
LISTEN 1
ESTABLISHED 2
[root@myhost ~]#
```

关闭指定服务的所有的进程。

```
ps -ef | grep httpd | awk '{print $2}' | xargs kill -9
```


cut

命令描述：cut 命令主要用来切割字符串，可以对输入的数据进行切割然后输出。

命令格式：cut [参数] [文件]。

参数说明：

参数	说明
-b	以字节为单位进行分割
-c	以字符为单位进行分割
-d	自定义分隔符，默认为制表符

命令使用示例：

按字节进行切割。

```
[root@myhost ~]# echo "hello world" | cut -b 1,3
hl
[root@myhost ~]# echo "hello world" | cut -b 1-3
hel
[root@myhost ~]#
```

按字符进行切割。

```
[root@myhost ~]# echo "hello world" | cut -c 1,3
hl
[root@myhost ~]# echo "hello world" | cut -c 1-3
hel
[root@myhost ~]# echo "h和o" | cut -c 2
和
[root@myhost ~]#
```

按指定字符进行切割。

```
[root@myhost ~]# echo "hello,world,ok" | cut -d , -f 1,3
hello,ok
[root@myhost ~]# echo "hello,world,ok" | cut -d , -f 2-3
world,ok
[root@myhost ~]#
```

tr

命令描述：tr 命令用于对来自标准输入的字符进行替换、压缩和删除。

命令格式：tr [参数] [文本]。

参数说明：

参数	说明
-c	反选指定字符
-d	删除指定字符
-s	将重复的字符缩减成一个字符
-t [第一字符集] [第二字符集]	删除第一字符集较第二字符集多出的字符，使两个字符集长度相等

命令使用示例：

将输入字符由大写转换为小写。

```
echo "HELLO WORLD" | tr 'A-Z' 'a-z'
```

命令输出结果：

```
[root@myhost ~]# echo "HELLO WORLD" | tr 'A-Z' 'a-z'
hello world
[root@myhost ~]#
```

删除字符。

```
echo "hello 123 world 456" | tr -d '0-9'
```

命令输出结果：

```
[root@myhost ~]# echo "hello 123 world 456" | tr -d '0-9'
hello world
[root@myhost ~]#
```

压缩字符。

```
echo "thissss is      a text linnnnnnne." | tr -s ' sn'
```

命令输出结果：

```
[root@myhost ~]# echo "thissss is      a text linnnnnnne." | tr -s ' sn'
this is a text line.
[root@myhost ~]#
```

产生随机密码。

```
cat /dev/urandom | tr -dc a-zA-Z0-9 | head -c 13
```

命令输出结果：

```
[root@myhost ~]# cat /dev/urandom | tr -dc a-zA-Z0-9 | head -c 13
debVQ8z8zCW2x[root@myhost ~]#
[root@myhost ~]#
```

动手实战--Linux 系统管理入门深入解析动手实操

体验简介

本场景将提供一台配置了 Aliyun Linux 2 的 ECS 实例（云服务器）。您可以参考本教程学习 Linux 系统中常用的系统工作命令以及查看系统状态的命令。

背景知识

本场景主要涉及以下云产品和服务：

云服务器 ECS

云服务器（Elastic Compute Service，简称 ECS）是阿里云提供的性能卓越、稳定可靠、弹性扩展的 IaaS（Infrastructure as a Service）级别云计算服务。云服务器 ECS 免去了您采购 IT 硬件的前期准备，让您像使用水、电、天然气等公共资源一样便捷、高效地使用服务器，实现计算资源的即开即用和弹性伸缩。阿里云 ECS 持续提供创新型服务器，解决多种业务需求，助力您的业务发展。

Aliyun Linux 2

Aliyun Linux 2 是阿里云推出的下一代 Linux 发行版，它为云上应用程序环境提供 Linux 社区的最新增强功能，在提供云上最佳用户体验的同时，也针对阿里云基础设施做了深度的优化。Aliyun Linux 2 OS 镜像可以运行在阿里云全规格系列 VM 实例上，包括弹性裸金属服务器（神龙）。

目录一：常用系统工作命令

echo

命令描述：echo 命令用于在终端输出字符串或变量提取后的值。

命令格式：echo [字符串 | \$变量]。

命令用法示例：

显示普通字符串

```
echo "Hello World"
```

显示变量

首先在 shell 环境中定义一个临时变量 name。

```
export name="Tom"
```

使用 echo 命令将变量 name 的值显示到终端。

```
echo $name
```

输出结果：

```
[root@myhost ~]# export name="Tom"
[root@myhost ~]# echo $name
Tom
[root@myhost ~]#
```

显示结果定向至文件

以下命令会将文本 This is a test text.输出重定向到文件 test.txt 中，如果文件已存在，将会覆盖文件内容，如果不存在则创建。其中>符号表示输出重定向。

```
echo "This is a test text." > test.txt
```

如果您希望将文本追加到文件内容最后，而不是覆盖它，请使用>>输出追加重定向符号。

显示命令执行结果

以下命令将会在终端显示当前的工作路径。

```
echo `pwd`
```

注意：pwd 命令是用一对反引号（` `）包裹，而不是一对单引号（' '）。

使用\$(command)形式可以达到相同效果。

```
echo $(pwd)
```

输出结果：

```
[root@myhost ~]# echo pwd
pwd
[root@myhost ~]# echo `pwd`
/root
[root@myhost ~]# echo $(pwd)
/root
[root@myhost ~]#
```

date

命令描述：date 命令用于显示和设置系统的时间和日期。

命令格式：date [选项] [+格式]。

其中，时间格式的部分控制字符解释如下：

字符	说明
%a	当地时间的星期名缩写（例如： 日，代表星期日）
%A	当地时间的星期名全称（例如：星期日）
%b	当地时间的月名缩写（例如：一，代表一月）
%B	当地时间的月名全称（例如：一月）
%c	当地时间的日期和时间（例如：2005 年 3 月 3 日 星期四 23:05:25）
%C	世纪；比如 %Y，通常为省略当前年份的后两位数字（例如：20）
%d	按月计的日期（例如：01）
%D	按月计的日期；等于%m/%d/%y
%F	完整日期格式，等价于 %Y-%m-%d
%j	按年计的日期（001-366）
%p	按年计的日期（001-366）
%r	当地时间下的 12 小时时钟时间（例如：11:11:04 下午）
%R	24 小时时间的时和分，等价于 %H:%M
%s	自 UTC 时间 1970-01-01 00:00:00 以来所经过的秒数
%T	时间，等于%H:%M:%S
%U	一年中的第几周，以周日为每星期第一天（00-53）
%x	当地时间下的日期描述（例如：12/31/99）

字符	说明
%X	当地时间下的时间描述 （例如：23:13:48）
%w	一星期中的第几日（0-6），0 代表周一
%W	一年中的第几周，以周一为每星期第一天（00-53）

命令用法示例：

按照默认格式查看当前系统时间

```
date
```

输出结果：

```
[root@myhost ~]# date
2020年 05月 12日 星期二 11:42:04 CST
```

按照指定格式查看当前系统时间

```
date "+%Y-%m-%d %H:%M:%S"
```

输出结果：

```
[root@myhost ~]# date "+%Y-%m-%d %H:%M:%S"
2020-05-12 11:42:23
```

查看今天是当年的第几天

```
date "+%j"
```

输出结果：


```
[root@myhost ~]# date "+%j"
133
```

将系统的当前时间设置为 2020 年 02 月 20 日 20 点 20 分 20 秒

```
date -s "20200220 20:20:20"
```

输出结果：

```
[root@myhost ~]# date -s "20200220 20:20:20"
2020年 02月 20日 星期四 20:20:20 CST
```

校正系统时间，与网络时间同步

A、安装 ntp 校时工具

```
yum -y install ntp
```

B、用 ntpdate 从时间服务器更新时间

```
ntpdate time.nist.gov
```

输出结果：

```
[root@myhost ~]# ntpdate time.nist.gov
12 May 14:03:13 ntpdate[6990]: step time server 132.163.97.6 offset 7061707.178529 sec
```

wget

命令描述：在终端中下载文件。

命令格式：wget [参数] 下载地址。

参数说明：

参数	作用
-b	后台下载
-P	下载到指定目录
-t	最大重试次数
-c	断点续传
-p	下载页面内所有资源，包括图片、视频等
-r	递归下载

命令使用示例：

下载一张图片到路径/root/static/img/中，-p 参数默认值为当前路径，如果指定路径不存在会自动创建。

```
wget -P /root/static/img/
http://img.alicdn.com/tfs/TB1.R_t7L0gK0jSZFxXXXWHVXa-2666-1500.png
```

输出结果：

```
[root@myhost ~]# wget -P /root/static/img/ http://img.alicdn.com/tfs/TB1.R_t7L0gK0jSZFxXXXWHVXa-2666-1500.png
--2020-05-12 14:04:57-- http://img.alicdn.com/tfs/TB1.R_t7L0gK0jSZFxXXXWHVXa-2666-1500.png
正在解析主机 img.alicdn.com (img.alicdn.com)... 106.15.218.207, 106.15.218.254
正在连接 img.alicdn.com (img.alicdn.com)[106.15.218.207]:80... 已连接。
已发出 HTTP 请求，正在等待响应... 200 OK
长度：5483908 (5.2M) [image/png]
正在保存至: "/root/static/img/TB1.R_t7L0gK0jSZFxXXXWHVXa-2666-1500.png"

100%[*****] 5,483,908 12.3MB/s 用时 0.4s

2020-05-12 14:04:57 (12.3 MB/s) - 已保存 "/root/static/img/TB1.R_t7L0gK0jSZFxXXXWHVXa-2666-1500.png" [5483908/5483908]
https://blog.csdn.net/gangyikeji
```

ps

命令描述：ps 命令用于查看系统中的进程状态。

命令格式：ps [参数]。

命令参数说明：

参数	作用
-a	显示现行终端机下的所有程序，包括其他用户的程序
-u	以用户为主的格式来显示程序状况
-x	显示没有控制终端的进程，同时显示各个命令的具体路径
-e	列出程序时，显示每个程序所使用的环境变量
-f	显示当前所有的进程
-t	指定终端机编号，并列出于属于该终端机的程序的状况

命令使用示例：

```
ps -ef | grep sshd
```

输出结果：

```
[root@myhost ~]# ps -ef | grep sshd
root      1239      1  0 09:11 ?          00:00:00 /usr/sbin/sshhd -D
root     10688 18445  0 14:06 pts/0    00:00:00 grep --color=auto sshd
root     18443  1239  0 11:29 ?          00:00:20 sshd: root@pts/0,pts/1
```

top

命令描述：top 命令动态地监视进程活动与系统负载等信息。

命令使用示例：

```
top
```

输出结果：

```
top - 14:08:26 up 4:57, 2 users, load average: 0.01, 0.05, 0.05
Tasks: 82 total, 1 running, 81 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.5 us, 0.8 sy, 0.0 ni, 98.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 7862388 total, 6489292 free, 217880 used, 1155216 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 7383900 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
6610	root	10	-10	134392	22184	9084	S	1.0	0.3	2:57.80	AliYunDun
18534	root	20	0	162652	2924	1596	S	0.7	0.0	0:06.55	top
18443	root	20	0	157640	6324	4632	S	0.3	0.1	0:20.86	sshd
1	root	20	0	125496	3984	2616	S	0.0	0.1	0:01.07	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
4	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0H
5	root	20	0	0	0	0	S	0.0	0.0	0:00.16	kworker/u4:0
6	root	20	0	0	0	0	S	0.0	0.0	0:00.26	ksoftirqd/0
7	root	rt	0	0	0	0	S	0.0	0.0	0:00.32	migrate/0

命令输出参数解释：

以上命令输出视图中分为两个区域，一个统计信息区，一个进程信息区。

1、统计信息区

- 第一行信息依次为：系统时间、运行时间、登录终端数、系统负载（三个数值分别为 1 分钟、5 分钟、15 分钟内的平均值，数值越小意味着负载越低）。
- 第二行信息依次为：进程总数、运行中的进程数、睡眠中的进程数、停止的进程数、僵死的进程数。
- 第三行信息依次为：用户占用资源百分比、系统内核占用资源百分比、改变过优先级的进程资源百分比、空闲的资源百分比等。
- 第四行信息依次为：物理内存总量、内存使用量、内存空闲量、作为内核缓存的内存量。
- 第五行信息依次为：虚拟内存总量、虚拟内存使用量、虚拟内存空闲量、预加载内存量。

2、进程信息区

列名	含义
PID	进程 ID
USER	进程所有者的用户名
PR	进程优先级
NI	nice 值。负值表示高优先级，正值表示低优先级
VIRT	进程使用的虚拟内存总量，单位 kb
RES	进程使用的、未被换出的物理内存大小，单位 kb
SHR	共享内存大小，单位 kb
S	进程状态 D：不可中断的睡眠状态 R：正在运行 S：睡眠 T：停止 Z：僵尸进程
%CPU	上次更新到现在的 CPU 时间占用百分比
%MEM	进程使用的物理内存百分比
TIME+	进程使用的 CPU 时间总计，单位 1/100 秒
COMMAND	命令名

按 q 键退出监控页面。

pidof

命令描述：pidof 命令用于查询指定服务进程的 PID 值。

命令格式：pidof [服务名称]。

命令参数说明：

参数	说明
-s	仅返回一个进程号
-c	只显示运行在 root 目录下的进程，这个选项只对 root 用户有效
-o	忽略指定进程号的进程
-x	显示由脚本开启的进程

命令使用示例：

查询出 crond 服务下的所有进程 ID。

```
pidof crond
```

输出结果：

```
[root@myhost ~]# pidof crond
1247
```

kill

命令描述：kill 命令用于终止指定 PID 的服务进程。

kill 可将指定的信息送至程序。预设的信息为 SIGTERM(15)，可将指定程序终止。若仍无法终止该程序，可使用 SIGKILL(9)信息尝试强制删除程序。

命令格式：kill [参数] [进程 PID]。

命令使用示例：

删除 pid 为 1247 的进程。

```
kill -9 1247
```

killall

命令描述：killall 命令用于终止指定名称的服务对应的全部进程。

命令格式：killall [进程名称]。

命令使用示例：

删除 crond 服务下的所有进程。

```
killall crond
```

reboot

命令描述：reboot 命令用来重启系统。

命令格式：reboot [-n] [-w] [-d] [-f] [-i]。

命令参数说明：

- -n：保存数据后再重新启动系统。
- -w：仅做测试，并不是真的将系统重新开机，只会把重新开机的数据写入记录文件/var/log/wtmp。
- -d：重新启动时不把数据写入记录文件/var/tmp/wtmp。
- -f：强制重新开机，不调用 shutdown 指令的功能。
- -i：关闭网络设置之后再重新启动系统。

命令使用示例：

```
reboot
```

poweroff

命令描述：poweroff 命令用来关闭系统。

命令使用示例：

```
poweroff
```

目录二：系统状态检测命令

ifconfig

命令描述：ifconfig 命令用于获取网卡配置与网络状态等信息。

命令示例：

```
[root@myhost ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.24.79.251 netmask 255.255.240.0 broadcast 172.24.79.255
    ether 00:16:3e:03:69:bb txqueuelen 1000 (Ethernet)
    RX packets 457509 bytes 207193769 (197.5 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 471479 bytes 65932688 (62.8 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
https://blog.csdn.net/gangyikeji
```

命令输出说明：

- 第一部分的第一行显示网卡状态信息。
 - eth0 表示第一块网卡。
 - UP 代表网卡开启状态。
 - RUNNING 代表网卡的网线被接上。
 - MULTICAST 表示支持组播。
- 第二行显示网卡的网络信息。
 - inet (IP 地址) : 172.16.132.195。
 - broadcast (广播地址) : 172.16.143.255。
 - netmask (掩码地址) : 255.255.240.0。
 - RX 表示接收数据包的情况, TX 表示发送数据包的情况。
 - lo 表示主机的回环网卡, 是一种特殊的网络接口, 不与任何实际设备连接, 而是完全由软件实现。与回环地址 (127.0.0.0/8 或::1/128) 不同, 回环网卡对系统显示为一块硬件。任何发送到该网卡上的数据都将立刻被同一网卡接收到。

uname

命令描述: uname 命令用于查看系统内核与系统版本等信息。

命令语法: `uname [-amnrsv][--help][--version]`。

命令使用示例:

显示系统信息。

```
uname -a
```

命令输出结果:

```
[root@myhost ~]# uname -a
Linux myhost 3.10.0-1062.18.1.el7.x86_64 #1 SMP Tue Mar 17 23:49:17 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

显示当前系统的硬件架构。

```
uname -i
```

命令输出结果：

```
[root@myhost ~]# uname -i  
x86_64
```

显示操作系统发行编号。

```
uname -r
```

命令输出结果：

```
[root@myhost ~]# uname -r  
3.10.0-1062.18.1.el7.x86_64
```

显示操作系统名称。

```
uname -s
```

命令输出结果：

```
[root@myhost ~]# uname -s  
Linux
```

显示主机名称。

```
uname -n
```

命令输出结果：

```
[root@myhost ~]# uname -n  
myhost
```

uptime

命令描述：uptime 用于查看系统的负载信息。

命令使用示例：

```
[root@myhost ~]# uptime
14:29:18 up 5:17, 2 users, load average: 0.27, 0.10, 0.07
```

命令输出说明：

负载信息	命令输出值
当前服务器时间	14:20:27
当前服务器运行时长	2 min
当前用户数	2 users
当前负载情况	load average: 0.03, 0.04, 0.02（分别取 1min，5min，15min 的均值）

free

命令描述：free 用于显示当前系统中内存的使用量信息。

命令语法：free [-bkmotV][-s <间隔秒数>]。

命令参数说明：

参数	说明
-b	以 Byte 为单位显示内存使用情况
-k	以 KB 为单位显示内存使用情况
-m	以 MB 为单位显示内存使用情况
-h	以合适的单位显示内存使用情况，最大为三位数，自动计算对应的单位值。

命令使用示例：

```
[root@myhost ~]# free -h
              total        used        free      shared  buff/cache   available
Mem:           7.5G          212M          6.1G          476K          1.2G          7.0G
Swap:           0B           0B           0B
```

命令输出说明：

参数	说明
total	物理内存总数
used	已经使用的内存数
free	空间的内存数
share	多个进程共享的内存总额
buff/cache	应用使用内存数
available	可用的内存数
Swap	虚拟内存（阿里云 ECS 服务器默认不开启虚拟内存）

who

命令描述：who 命令显示关于当前在本地系统上的所有用户的信息。

命令使用示例：

- 显示当前登录系统的用户

```
[root@myhost ~]# who
root    pts/0      2020-05-12 11:29 (42.███.███.157)
root    pts/1      2020-05-12 11:29 (42.███.███.157)
```

- 显示用户登录来源

```
[root@myhost ~]# who -l -H
名称  线路      时间          空闲  进程号  备注
登录  ttyS0      2020-05-12 09:11          1257 id=tyS0
登录  tty1       2020-05-12 09:11          1258 id=tty1
```

- 只显示当前用户

```
[root@myhost ~]# who -m -H
名称  线路      时间          备注
root  pts/0      2020-05-12 11:29 (42.███.███.157)
```

- 精简模式显示

```
[root@myhost ~]# who -q
root root
# 用户数=2
```

last

命令描述： last 命令用于显示用户最近登录信息。

命令使用示例：

```
[root@myhost ~]# last
root    pts/3      42.█.█.█.157    Tue May 12 14:37 - 14:40  (00:02)
root    pts/2      42.█.█.█.157    Tue May 12 14:37 - 14:40  (00:02)
root    pts/1      42.█.█.█.157    Tue May 12 11:29      still logged in
root    pts/0      42.█.█.█.157    Tue May 12 11:29      still logged in
root    pts/1      42.█.█.█.157    Tue May 12 09:12 - 11:29  (02:17)
root    pts/0      42.█.█.█.157    Tue May 12 09:12 - 11:29  (02:17)
reboot  system boot  3.10.0-1062.18.1 Tue May 12 17:11 - 14:41  (-2:-30)

wtm begins Sun Apr 26 16:02:02 2020      https://blog.csdn.net/gangyikeji
```

由于这些信息都是以日志文件的形式保存在系统中，黑客可以很容易地对内容进行篡改，所以该命令输出的信息并不能作为服务器是否被入侵的依据。

history

命令描述：history 命令用于显示历史执行过的命令。

bash 默认记录 1000 条执行过的历史命令，被记录在 ~/.bash_history 文件中。

命令使用示例：

- 显示最新 10 条执行过的命令。

```
[root@myhost ~]# history 10
 6  uname -n
 7  uptime
 8  free -h
 9  top
10  who
11  who -l -H
12  who -m -H
13  who -q
14  last
15  history 10

https://blog.csdn.net/gangyikeji
```

清除历史记录。

```
history -c
```

动手实战--Linux 磁盘管理入门深入解析动手实操

体验简介

本场景将提供一台配置了 Aliyun Linux 2 操作系统的 ECS 实例（云服务器）。通过本教程的操作，您可以学习 Linux 系统中常用的磁盘管理命令。

体验此场景后，可以掌握的知识有：Linux 基本操作。

磁盘管理命令

1、df 命令

df 命令描述：该命令检查文件系统的磁盘空间占用情况。可以利用该命令来获取硬盘被占用了多少空间，目前还剩下多少空间等信息。

df 命令语法：df [参数] [目录或文件名]。

-a

列出所有的文件系统，包括系统特有的 /proc 等文件系统。

-k

以 KBytes 为单位，返回各文件系统容量。

-m

以 MBytes 为单位，返回各文件系统容量。

-h

以 GBytes、MBytes、KBytes 为单位，返回各文件系统容量。

-H

以 M=1000K 取代 M=1024K 的进位方式显示各文件系统容量。

-T

显示文件系统类型。

- i

显示 inode 信息。

df 命令使用示例：

示例一：显示磁盘使用情况。

执行如下命令，显示磁盘使用情况。

df

返回结果如下所示。

```
[root@izuf6060000 ~]# df
```

文件系统	1K-块	已用	可用	已用%	挂载点
devtmpfs	496920	0	496920	0%	/dev
tmpfs	507280	0	507280	0%	/dev/shm
tmpfs	507280	492	506788	1%	/run
tmpfs	507280	0	507280	0%	/sys/fs/cgroup
/dev/vda1	41152812	2044148	37204956	6%	/
tmpfs	101456	0	101456	0%	/run/user/0

示例二：以 inode 模式来显示磁盘使用情况。

执行如下命令，以 inode 模式来显示磁盘使用情况。

df -i

返回结果如下所示。


```
[root@i ~]# df -i
文件系统      Inode 已用(I) 可用(I) 已用(I)% 挂载点
devtmpfs      124230    327  123903      1% /dev
tmpfs         126820      2  126818      1% /dev/shm
tmpfs         126820    393  126427      1% /run
tmpfs         126820     16  126804      1% /sys/fs/cgroup
/dev/vda1     2621440  60204 2561236      3% /
tmpfs         126820      1  126819      1% /run/user/0
```

示例三：显示系统内的所有特殊文件格式、名称及磁盘使用情况。

执行如下命令，显示系统内的所有特殊文件格式、名称及磁盘使用情况。

```
df -aT
```

返回结果如下所示。

```
[root@i ~]# df -aT
文件系统      类型      1K-块    已用    可用 已用% 挂载点
sysfs         sysfs          0      0      0    - /sys
proc          proc          0      0      0    - /proc
devtmpfs      devtmpfs    496920      0  496920    0% /dev
securityfs    securityfs    0      0      0    - /sys/kernel/security
tmpfs         tmpfs     507280      0  507280    0% /dev/shm
devpts        devpts        0      0      0    - /dev/pts
tmpfs         tmpfs     507280    460  506820    1% /run
tmpfs         tmpfs     507280      0  507280    0% /sys/fs/cgroup
cgroup        cgroup          0      0      0    - /sys/fs/cgroup/systemd
pstore        pstore          0      0      0    - /sys/fs/pstore
cgroup        cgroup          0      0      0    - /sys/fs/cgroup/cpu,cpuacct
cgroup        cgroup          0      0      0    - /sys/fs/cgroup/freezer
cgroup        cgroup          0      0      0    - /sys/fs/cgroup/perf_event
cgroup        cgroup          0      0      0    - /sys/fs/cgroup/blkio
cgroup        cgroup          0      0      0    - /sys/fs/cgroup/net_cls,net_prio
cgroup        cgroup          0      0      0    - /sys/fs/cgroup/hugetlb
cgroup        cgroup          0      0      0    - /sys/fs/cgroup/memory
cgroup        cgroup          0      0      0    - /sys/fs/cgroup/devices
cgroup        cgroup          0      0      0    - /sys/fs/cgroup/pids
cgroup        cgroup          0      0      0    - /sys/fs/cgroup/cpuset
configfs      configfs        0      0      0    - /sys/kernel/config
/dev/vda1     ext4    41152812 2042876 37206228    6% /
systemd-1     -            -      -      -    - /proc/sys/fs/binfmt_misc
mqueue        mqueue          0      0      0    - /dev/mqueue
debugfs       debugfs          0      0      0    - /sys/kernel/debug
hugetlbfs     hugetlbfs        0      0      0    - /dev/hugepages
tmpfs         tmpfs     101456      0  101456    0% /run/user/0
binfmt_misc   binfmt_misc      0      0      0    - /proc/sys/fs/binfmt_misc
```

示例四：以 GBytes、MBytes、KBytes 等格式显示各文件系统容量。

执行如下命令，以 GBytes、MBytes、KBytes 等格式显示各文件系统容量。

```
df -h
```

返回结果如下所示。

```
[root@i Z ~]# df -h
文件系统      容量  已用  可用  已用% 挂载点
devtmpfs      486M    0  486M    0% /dev
tmpfs         496M    0  496M    0% /dev/shm
tmpfs         496M  460K  495M    1% /run
tmpfs         496M    0  496M    0% /sys/fs/cgroup
/dev/vda1      40G   2.0G   36G    6% /
tmpfs         100M    0  100M    0% /run/user/0
```

2、du 命令

du 命令描述：查看磁盘使用空间。du 与 df 命令不同点在于，du 命令用于查看文件和目录磁盘的使用空间。

du 命令语法：du [参数] [文件或目录名称]。

参数说明：

-a

列出所有的文件与目录容量。

-h

以 G、M、K 为单位，返回容量。

-s

列出总量。

-S

列出不包括子目录下的总量。

-k

以 KBytes 为单位，返回容量。

-m

以 MBytes 为单位，返回容量。

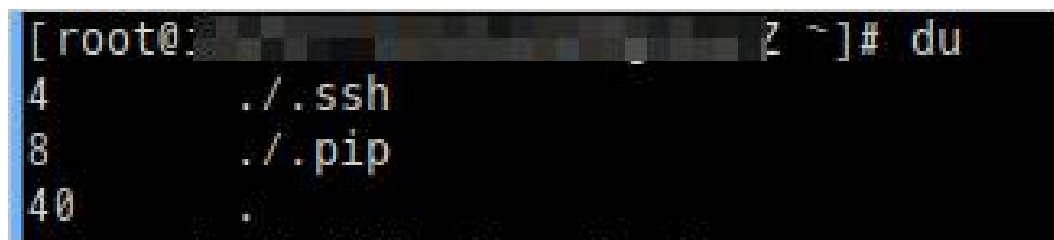
du 命令使用示例：

示例一：列出当前目录下的所有文件夹的容量。

执行如下命令，列出当前目录下的所有文件夹的容量。

```
du
```

返回结果如下所示。



```
[root@: ~]# du
4      ./ssh
8      ./pip
40     .
```

示例二：列出当前目录下的所有文件夹和文件的容量。

执行如下命令，列出当前目录下的所有文件夹和文件的容量。

```
du -a
```

返回结果如下所示。

```
[root@iZ...~]# du -a
0      ../ssh/authorized_keys
4      ../ssh
4      ../pydistutils.cfg
4      ../bash_logout
4      ../tcshrc
4      ../bash_profile
4      ../bashrc
4      ../pip/pip.conf
8      ../pip
4      ../cshrc
40     .
```

<https://blog.csdn.net/gangyikeji>

示例三：列出当前目录下的所有文件夹和文件的容量，并以 G、M、K 格式显示容量。

执行如下命令，列出当前目录下的所有文件夹和文件的容量。

```
du -ah
```

返回结果如下所示。

```
[root@...~]# du -sm /*
0      /bin
146    /boot
0      /dev
34     /etc
1      /home
0      /lib
0      /lib64
1      /lost+found
1      /media
1      /mnt
1      /opt
du: 无法访问 "/proc/3683/task/3683/fd/4": 没有那个文件或目录
du: 无法访问 "/proc/3683/task/3683/fdinfo/4": 没有那个文件或目录
du: 无法访问 "/proc/3683/fd/4": 没有那个文件或目录
du: 无法访问 "/proc/3683/fdinfo/4": 没有那个文件或目录
0      /proc
1      /root
1      /run
0      /sbin
1      /srv
0      /sys
77     /tmp
```

<https://blog.csdn.net/gangyikeji>

fdisk 命令描述：该命令用于磁盘分区。

fdisk 命令语法: fdisk [-l] 装置名称。

-1

输出后面装置名称的所有的分区内容。若仅有 `fdisk -l` 时， 则系统将会把整个系统内能够搜寻到的装置的分区均列出来。

fdisk 命令使用示例:

示例一：列出系统所有装置的分区信息。

执行如下命令，列出系统所有装置的分区信息。

```
fdisk -l
```

返回结果如下所示。

```
[root@iZuf6t12b1kxarimangz6ae2 ~]# fdisk -l
```

磁盘 /dev/vda: 42.9 GB, 42949672960 字节, 83886080 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理): 512 字节 / 512 字节
I/O 大小(最小/最佳): 512 字节 / 512 字节
磁盘标签类型: dos
磁盘标识符: 0x0002b49c

设备	Boot	Start	End	Blocks	Id	System
/dev/vda1	*	2048	83886046	41941999+	83	Linux

示例二：列出系统中的根目录所在磁盘，并查阅该硬盘内的相关信息。

A、执行如下命令，找出根目录所在磁盘名。

```
df /
```

返回结果如下所示。

```
[root@iZuf1-...~]# df /  
文件系统      1K-块    已用    可用  已用% 挂载点  
/dev/vda1    41152812 2041132 37207972    6% /
```

B、执行如下命令，对磁盘/dev/vda 进行分区操作。

```
fdisk /dev/vda
```

注意：对磁盘进行分区操作时，磁盘名不包含数字。

返回结果如下所示。

```
[root@iZuf1-...~]# fdisk /dev/vda  
欢迎使用 fdisk (util-linux 2.23.2).  
  
更改将停留在内存中，直到您决定将更改写入磁盘。  
使用写入命令前请三思。  
  
命令(输入 m 获取帮助)：
```

C、执行如下命令，获取帮助。

```
m
```

返回结果如下所示。


```
命令(输入 m 获取帮助): m
命令操作
a  toggle a bootable flag
b  edit bsd disklabel
c  toggle the dos compatibility flag
d  delete a partition
g  create a new empty GPT partition table
G  create an IRIX (SGI) partition table
l  list known partition types
m  print this menu
n  add a new partition
o  create a new empty DOS partition table
p  print the partition table
q  quit without saving changes
s  create a new empty Sun disklabel
t  change a partition's system id
u  change display/entry units
v  verify the partition table
w  write table to disk and exit
x  extra functionality (expert mode)
https://blog.csdn.net/gangyikeji
```

执行如下命令，查看磁盘状态。

p

返回结果如下所示，您可以查看到磁盘的相关状态。

```
命令(输入 m 获取帮助): p

磁盘 /dev/vda: 42.9 GB, 42949672960 字节, 83886080 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理): 512 字节 / 512 字节
I/O 大小(最小/最佳): 512 字节 / 512 字节
磁盘标签类型: dos
磁盘标识符: 0x0002b49c

 设备 Boot      Start          End      Blocks   Id  System
/dev/vda1  *         2048        83886046     41941999+  83  Linux
https://blog.csdn.net/gangyikeji
```

D、执行如下命令，不存储任何操作并离开。

q

动手实战--Linux 文件与权限入门深入解析动手实操

场景体验

本场景将提供一台配置了 Aliyun Linux 2 的 ECS 实例（云服务器）。您可以参考本教程学习 Linux 系统中常用的文件目录管理与文件权限管理命令。

背景知识

云服务器 ECS

云服务器（Elastic Compute Service，简称 ECS）是阿里云提供的性能卓越、稳定可靠、弹性扩展的 IaaS（Infrastructure as a Service）级别云计算服务。云服务器 ECS 免去了您采购 IT 硬件的前期准备，让您像使用水、电、天然气等公共资源一样便捷、高效地使用服务器，实现计算资源的即开即用和弹性伸缩。阿里云 ECS 持续提供创新型服务器，解决多种业务需求，助力您的业务发展。

Alibaba Cloud Linux

Aliyun Linux 2 是阿里云推出的下一代 Linux 发行版，它为云上应用程序环境提供 Linux 社区的最新增强功能，在提供云上最佳用户体验的同时，也针对阿里云基础设施做了深度的优化。Aliyun Linux 2 OS 镜像可以运行在阿里云全规格系列 VM 实例上，包括弹性裸金属服务器（神龙）。

目录一：文件目录管理命令

tree

命令描述：tree 命令用于以树状图列出目录的内容。

tree 命令没有内置在系统中，使用 tree 命令需要执行以下命令来安装：

```
yum install -y tree
```

命令使用示例：

```
tree /usr/share/wallpapers/
```

命令输出结果：

```
[root@myhost ~]# tree /usr/share/wallpapers/
/usr/share/wallpapers/
├── backgrounds -> /usr/share/backgrounds
└── CentOS7
    ├── contents
    │   └── images
    │       └── 2560x1600.jpg -> /usr/share/backgrounds/day.jpg
    └── metadata.desktop

4 directories, 2 files
[root@myhost ~]#
```

<https://blog.csdn.net/gangyikeji>

ls

命令描述：ls 命令用于显示指定工作目录下的内容。

命令格式：ls [参数] [目录名]。

参数	说明
-a	显示所有文件及目录（包括隐藏文件）
-l	将文件的权限、拥有者、文件大小等详细信息列出（ll 等同于 ls -l）
-r	将文件反序列出（默认按英文字母正序）
-t	将文件按创建时间正序列出
-R	递归遍历目录下文件

命令使用示例：

查看当前目录下的所有文件（包括隐藏文件）。

```
ll -a
```

命令输出结果：

```
[root@myhost ~]# ll -a
总用量 60
dr-xr-x---.  7 root root 4096 5月  12 16:43 .
dr-xr-xr-x. 18 root root 4096 5月  12 09:11 ..
-rw-----  1 root root  886 5月  12 14:40 .bash_history
-rw-r--r--.  1 root root   18 12月 29 2013 .bash_logout
-rw-r--r--.  1 root root  176 12月 29 2013 .bash_profile
-rw-r--r--.  1 root root  176 12月 29 2013 .bashrc
drwxr-xr-x   4 root root 4096 5月  12 09:13 .config
-rw-r--r--.  1 root root  100 12月 29 2013 .cshrc
drwxr-xr-x   2 root root 4096 5月  12 09:14 .fcli
drwxr-xr-x   2 root root 4096 4月  26 15:58 .pip
drwxr-----  3 root root 4096 5月  12 09:12 .pki
-rw-r--r--  1 root root  206 5月  12 09:11 .pydistutils.cfg
drwx-----  2 root root 4096 4月  26 07:59 .ssh
-rw-r--r--.  1 root root  129 12月 29 2013 .tcshrc
-rw-----  1 root root 1120 5月  12 15:11 .viminfo
[root@myhost ~]#
```

<https://blog.csdn.net/gangyikeji>

pwd

命令描述：获取当前工作目录的绝对路径。

命令使用示例：

```
[root@myhost ~]# pwd
/root
[root@myhost ~]#
```

cd

命令描述：cd 命令用于切换工作目录。

命令使用示例：

```
[root@myhost ~]# cd /usr/local/etc
[root@myhost etc]# pwd
/usr/local/etc
[root@myhost etc]# cd ../../
[root@myhost usr]# pwd
/usr
[root@myhost usr]# cd
[root@myhost ~]# pwd
/root
[root@myhost ~]#
```

<https://blog.csdn.net/gangyikeji>

在路径表示中：

- 一个半角句号 (.) 表示当前目录，例如路径./app/log 等同于 app/log。
- 两个半角句号 (..) 表示上级目录，例如路径/usr/local/./src 等同于/usr/src，其中 local 和 src 目录同级。

cd 命令的默认参数为~, 符号 ~ 表示当前用户的家目录, 即在 root 用户登录时, 命令 cd、cd ~和 cd /root 执行效果相同。

touch

命令描述: touch 命令用于修改文件或者目录的时间属性, 包括存取时间和更改时间。若文件不存在, 系统会建立一个新的文件。

命令格式: touch [参数] [文件]。

参数说明:

参数	说明
-c	如果指定文件不存在, 不会建立新文件
-r	使用参考文件的时间记录
-t	设置文件的时间记录

命令使用示例:

创建两个空文件。

```
[root@myhost ~]# touch demo1.txt demo2.txt
[root@myhost ~]# ll
总用量 0
-rw-r--r-- 1 root root 0 5月 12 16:51 demo1.txt
-rw-r--r-- 1 root root 0 5月 12 16:51 demo2.txt
[root@myhost ~]#
```

修改 demo1.txt 的时间记录为当前系统时间。

```
[root@myhost ~]# touch demo1.txt
[root@myhost ~]# ll
总用量 0
-rw-r--r-- 1 root root 0 5月 12 16:53 demo1.txt
-rw-r--r-- 1 root root 0 5月 12 16:51 demo2.txt
[root@myhost ~]#
```

更新 demo2.txt 的时间记录，使其和 demo1.txt 的时间记录相同。

```
[root@myhost ~]# touch -r demo1.txt demo2.txt
[root@myhost ~]# ll
总用量 0
-rw-r--r-- 1 root root 0 5月 12 16:53 demo1.txt
-rw-r--r-- 1 root root 0 5月 12 16:53 demo2.txt
[root@myhost ~]#
```

mkdir

命令描述：mkdir 命令用于新建子目录。-p 参数确保目录名称存在，不存在的就新建一个。

命令使用示例：

新建目录 a/b/c/d，并使用 tree 命令查看创建后的目录结构。

```
[root@myhost ~]# mkdir -p a/b/c/d
[root@myhost ~]# tree
.
├── a
│   ├── b
│   │   ├── c
│   │   └── d
├── demo1.txt
└── demo2.txt

4 directories, 2 files
[root@myhost ~]#
```

<https://blog.csdn.net/gangyikeji>

rm

命令描述：rm 命令用于删除一个文件或者目录。

命令格式：rm [参数] [文件]。

参数说明：

参数	说明
-i	删除前逐一询问确认
-f	无需确认，直接删除
-r	删除目录下所有文件

命令使用示例：

无需确认直接删除文件。

```
[root@myhost ~]# ll
总用量 4
drwxr-xr-x 3 root root 4096 5月 12 16:57 a
-rw-r--r-- 1 root root    0 5月 12 16:53 demo1.txt
-rw-r--r-- 1 root root    0 5月 12 16:53 demo2.txt
[root@myhost ~]# rm -rf demo*
[root@myhost ~]# ll
总用量 4
drwxr-xr-x 3 root root 4096 5月 12 16:57 a
[root@myhost ~]#
```

<https://blog.csdn.net/gangyikeji>

无需确认直接删除目录 a 及其目录下所有子目录和文件。

```
[root@myhost ~]# rm -rf a
[root@myhost ~]# ll
总用量 0
[root@myhost ~]#
```

cp

命令描述： cp 命令主要用于复制文件或目录。

命令格式： cp [参数] [源文件] [目标文件]。

参数说明：

参数	说明
-d	复制时保留链接
-f	覆盖已经存在的目标文件而不给出提示
-i	覆盖前询问
-p	除复制文件的内容外，还把修改时间和访问权限也复制到新文件中
-r	复制目录及目录内的所有项目

命令使用示例：

将目录 c/d 中的所有内容复制到目录 a/b 下。

```
[root@myhost ~]# mkdir -p a/b
[root@myhost ~]# mkdir -p c/d
[root@myhost ~]# tree
.
├── a
│   └── b
└── c
    └── d

4 directories, 0 files
[root@myhost ~]# cp -r c a/b/
[root@myhost ~]# tree
.
├── a
│   ├── b
│   │   └── c
│   │       └── d
└── c
    └── d

6 directories, 0 files
[root@myhost ~]#
```

<https://blog.csdn.net/gangyikeji>

mv

命令描述： mv 命令用来为文件或目录改名、或将文件或目录移入其它位置。

命令格式： mv [参数] [源文件] [目标文件]。

参数说明：

参数	说明
-i	若指定目录已有同名文件，则先询问是否覆盖旧文件
-f	如果目标文件已经存在，不会询问而直接覆盖

命令使用示例：

将文件名 a.txt 改为 b.txt。


```
[root@myhost ~]# touch a.txt
[root@myhost ~]# ls
a  a.txt  c
[root@myhost ~]# mv a.txt b.txt
[root@myhost ~]# ls
a  b.txt  c
[root@myhost ~]#
```

将 c 目录移动到 a/b/c/d/下。

```
[root@myhost ~]# mv c a/b/c/d/
[root@myhost ~]# tree
.
├── a
│   ├── b
│   │   ├── c
│   │   │   ├── d
│   │   │   │   ├── c
│   │   │   │   └── d
│   └── b.txt
└──
```

6 directories, 1 file

[root@myhost ~]# <https://blog.csdn.net/gangyikeji>

将当前目录内容全部移动到/tmp 目录中。

```
mv ./* /tmp
```

rename

命令描述：rename 命令用字符串替换的方式批量改变文件名。rename 命令有 C 语言和 Perl 语言两个版本，这里介绍 C 语言版本的 rename 命令，不支持正则表达式。

命令使用示例：

- 将当前目录下所有文件名中的字符串 demo 改为大写的字符串 DEMO。

```
[root@myhost ~]# touch demo1.txt demo2.txt
[root@myhost ~]# ls
demo1.txt  demo2.txt
[root@myhost ~]# rename demo DEMO *
[root@myhost ~]# ls
DEMO1.txt  DEMO2.txt
[root@myhost ~]#
```

将当前目录下所有.txt 文件后缀都改为 text。

```
[root@myhost ~]# rename .txt .text *
[root@myhost ~]# ls
DEMO1.text  DEMO2.text
[root@myhost ~]#
```

目录二：文件权限管理

文件权限管理

ls 命令可以查看 Linux 系统上的文件、目录和设备的权限。

```
ls -l /boot/
```

```
[root@myhost ~]# ls -l /boot/
总用量 140804
-rw-r--r-- 1 root root 153187 3月 18 07:53 config-3.10.0-1062.18.1.el7.x86_64
-rw-r--r-- 1 root root 152976 8月 8 2019 config-3.10.0-1062.el7.x86_64
drwxr-xr-x 3 root root 4096 4月 26 15:48 efi
drwxr-xr-x 2 root root 4096 4月 26 15:49 grub
drwx----- 5 root root 4096 4月 26 15:58 grub2
-rw----- 1 root root 57931787 4月 26 15:52 initramfs-0-rescue-20200426154603174201708213343640.img
-rw----- 1 root root 18197454 4月 26 16:01 initramfs-3.10.0-1062.18.1.el7.x86_64.img
-rw----- 1 root root 10734218 4月 26 15:56 initramfs-3.10.0-1062.18.1.el7.x86_64kdump.img
-rw----- 1 root root 18198855 4月 26 15:56 initramfs-3.10.0-1062.el7.x86_64.img
-rw----- 1 root root 10732204 4月 26 15:54 initramfs-3.10.0-1062.el7.x86_64kdump.img
-rw-r--r-- 1 root root 318991 3月 18 07:53 symvers-3.10.0-1062.18.1.el7.x86_64.gz
-rw-r--r-- 1 root root 318717 8月 8 2019 symvers-3.10.0-1062.el7.x86_64.gz
-rw----- 1 root root 3600165 3月 18 07:53 System.map-3.10.0-1062.18.1.el7.x86_64
-rw----- 1 root root 3594971 8月 8 2019 System.map-3.10.0-1062.el7.x86_64
-rwxr-xr-x 1 root root 6734016 4月 26 15:52 vmlinuz-0-rescue-20200426154603174201708213343640
-rwxr-xr-x 1 root root 6738112 3月 18 07:53 vmlinuz-3.10.0-1062.18.1.el7.x86_64
-rwxr-xr-x 1 root root 6734016 8月 8 2019 vmlinuz-3.10.0-1062.el7.x86_64
[root@myhost ~]#
```

<https://blog.csdn.net/gangyikeji>

上述 `ls -l` 命令中显示的第一列就是文件权限信息，共 11 位字符，分 5 部分。

- 第 1 位表示存档类型，d 表示目录，-表示一般文件。
- 第 2~4 位表示当前用户的权限（属主权限）。
- 第 5~7 位表示同用户组的用户权限（属组权限）。
- 第 8~10 位表示不同用户组的用户权限（其他用户权限）。
- 第 11 位是一个半角句号.，表示 SELinux 安全标签。

用户权限每组三位，`rwX` 分别表示读、写、执行权限，对应八进制表示为 4、2、1。

例如 `efi` 目录的 `root` 用户权限为 `drwxr-xr-x`。

- 该目录对 `root` 用户具有读写和执行所有权限。
- 该目录对 `root` 组其他用户有读和执行权限。
- 该目录对其他用户有读和执行权限。

所以该权限表示对应八进制权限表示为：

- 属主权限：4+2+1=7。
- 属组权限：4+1=5。
- 其他用户权限：4+1=5。

即 755。

chmod

chmod 命令用于修改文件权限 mode，-R 参数以递归方式对子目录和文件进行修改。

命令使用示例：

1、新建名为 hello.sh 的 Shell 脚本，该脚本将会输出 Hello World。用 ll 命令可以看到新建的脚本没有执行权限，其权限用八进制表示为 644。

```
echo "echo 'Hello World'" > hello.sh
ll
```

```
[root@myhost ~]# echo "echo 'Hello World'" > hello.sh
[root@myhost ~]# ll
总用量 4
-rw-r--r-- 1 root root 19 5月 12 17:17 hello.sh
[root@myhost ~]#
```

2、将 hello.sh 文件增加属主的执行权限。

```
chmod u+x hello.sh
ll
```

```
[root@myhost ~]# chmod u+x hello.sh
[root@myhost ~]# ll
总用量 4
-rwxr--r-- 1 root root 19 5月 12 17:17 hello.sh
[root@myhost ~]#
```

3、将 hello.sh 文件撤销属主的执行权限。

```
chmod u-x hello.sh  
ll
```

```
[root@myhost ~]# chmod u-x hello.sh  
[root@myhost ~]# ll  
总用量 4  
-rw-r--r-- 1 root root 19 5月 12 17:17 hello.sh  
[root@myhost ~]#
```

4、将 hello.sh 文件权限修改为八进制表示的 744 权限。

```
chmod 744 hello.sh  
ll
```

```
[root@myhost ~]# chmod 744 hello.sh  
[root@myhost ~]# ll  
总用量 4  
-rwxr--r-- 1 root root 19 5月 12 17:17 hello.sh  
[root@myhost ~]#
```

5、使用 bash 命令解释器执行 hello.sh 脚本文件。

```
/bin/bash hello.sh
```

```
[root@myhost ~]# /bin/bash hello.sh  
Hello World  
[root@myhost ~]#
```

其中，u+x 表示增加属主的执行权限，u 表示属主，g 表示属组，o 表示其他，a 表示所有用户。

chown

chown 命令修改文件的属主和属组；-R 参数以递归方式对子目录和文件进行修改；ls -l 命令显示的第三列和第四列就是文件的属主和属组信息。

命令使用示例：

1、新建一个文本文件 test.txt，用 ll 命令可以看到该文件的属主和属组是 root。whoami 命令可以查看当前 Shell 环境登录的用户名。

```
whoami
```

```
touch test.txt
```

```
ll
```

```
[root@myhost ~]# whoami
root
[root@myhost ~]# touch test.txt
[root@myhost ~]# ll
总用量 0
-rw-r--r-- 1 root root 0 5月 12 17:25 test.txt
[root@myhost ~]#
```

2、创建两个用户。

```
useradd test
```

```
useradd admin
```

3、修改 test.txt 文件的属主用户为 test。

```
chown test test.txt
```

```
ll
```

```
[root@myhost ~]# chown test test.txt
[root@myhost ~]# ll
总用量 0
-rw-r--r-- 1 test root 0 5月 12 17:25 test.txt
[root@myhost ~]#
```

4、修改 test.txt 文件的属主和属组为 admin。

```
chown admin:admin test.txt
ll
```

chgrp

chgrp 命令用于修改文件的属组。

命令使用示例：

将 test.txt 文件的属组改为 root。

```
chgrp root test.txt
ll
```

```
[root@myhost ~]# chgrp root test.txt
[root@myhost ~]# ll
总用量 0
-rw-r--r-- 1 admin root 0 5月 12 17:25 test.txt
[root@myhost ~]#
```

动手实战--Linux 文件管理入门深入解析动手实操

体验简介

场景将提供一台配置了 Aliyun Linux 2 操作系统的 ECS 实例（云服务器）。通过本教程的操作，您可以学习 Linux 系统中常用的文件管理命令。

磁盘管理命令

1、cat 命令

cat 命令描述：该命令用于连接文件并打印到标准输出设备上。

cat 命令语法：cat [参数] [文件名]。

参数说明：

参数	说明
-n	由 1 开始对所有输出的行数进行编号。
-b	由 1 开始对所有输出的行数进行编号，对于空白行不编号。
-s	当遇到有连续两行以上的空白行，就替换为一行的空白行。
-E	在每行结束处显示\$。
-T	将 TAB 字符显示为^I。

cat 命令使用示例：

A、执行如下命令，将一个自增序列写入 test1.txt 文件中。

```
for i in $(seq 1 10); do echo $i >> test1.txt ; done
```

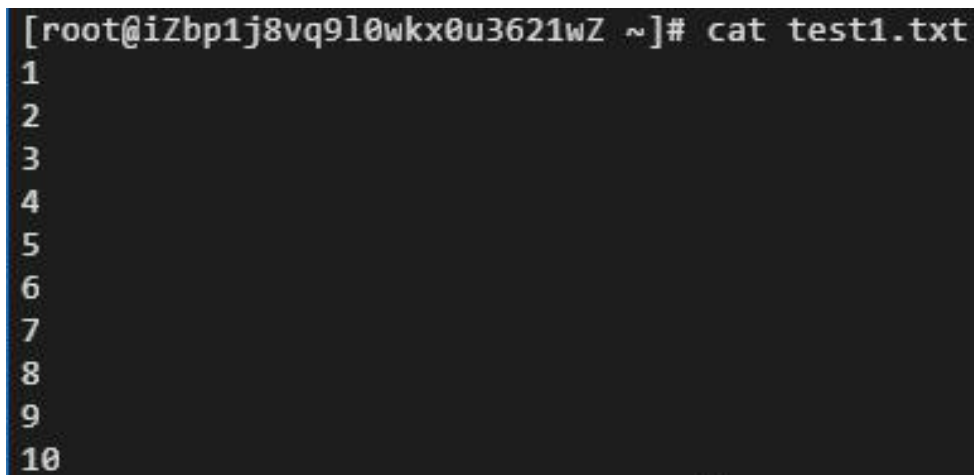


```
[root@iZbp1j8vq9l0wkx0u3621wZ ~]# for i in $(seq 1 10); do echo $i >> test1.txt ; done
```

B、执行如下命令，查看文件 test1.txt 内容。

```
cat test1.txt
```

返回结果如下所示。



```
[root@iZbp1j8vq9l0wkx0u3621wZ ~]# cat test1.txt
1
2
3
4
5
6
7
8
9
10
```

C、执行如下命令，将 test1.txt 的文件内容加上行号后输入到 test2.txt 文件。

```
cat -n test1.txt > test2.txt
```



```
[root@iZbp1j8vq9l0wkx0u3621wZ ~]# cat -n test1.txt > test2.txt
```

D、执行如下命令，查看文件 test2.txt 内容。

```
cat test2.txt
```

返回结果如下所示。

```
[root@iZ...Z ~]# cat test2.txt
1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
10 10
```

E、执行如下命令，将 test1.txt 文件内容清空。

```
cat /dev/null > test1.txt
```

```
[root@iZ...Z ~]# cat /dev/null > test1.txt
```

F、执行如下命令，查看文件 test1.txt 内容。

```
cat test1.txt
```

返回结果如下所示，您可以看到 test1.txt 文件没有任何内容。

```
[root@iZ...Z ~]# cat test1.txt
[root@iZ...Z ~]#
```

2、cmp 命令

cmp 命令描述：该命令用于比较两个文件是否有差异。当相互比较的两个文件完全一样时，该指令不会显示任何信息。否则会标示出第一个不同之处的字符和列数编号。当不指定任何文件名称，或文件名为"-"，则 cmp 指令会从标准输入设备读取数据。

cmp 命令语法：cmp [-clsv][*-i* <字符数目>][*--help*][第一个文件][第二个文件]。


参数说明：

参数	说明
-c	除了标明差异处的十进制字码之外，一并显示该字符所对应字符。
-i <字符数目>	指定一个数目。
-l	标示出所有不一样的地方。
-s	不显示错误信息。
-v	显示版本信息。
--help	在线帮助。

cmp 命令使用示例：

A、执行如下命令，将一个自增序列 1-5 写入 test1.txt 文件中。

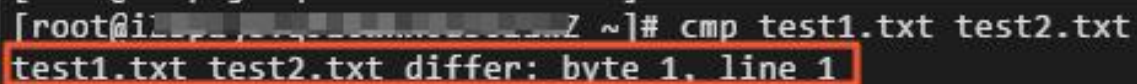
```
for i in $(seq 1 5); do echo $i >> test1.txt ; done
```



B、执行如下命令，比较 test1.txt 文件和 test2.txt 文件是否相同。

```
cmp test1.txt test2.txt
```

返回结果如下所示，您可以看到 test1.txt 文件和 test2.txt 文件第一行就有不同之处。



3、diff 命令

diff 命令描述：该命令用于比较文件的差异。diff 命令以逐行的方式，比较文本文件的异同处。如果指定要比较目录，则 diff 会比较目录中相同文件名的文件，但不会比较其中子目录。

diff 命令语法：diff [参数] [文件或目录 1] [文件或目录 2]。

参数说明：

参数	说明
-<行数>	指定要显示多少行的文本。此参数必须与-c 或-u 参数一并使用。
-c	显示全部内文，并标出不同之处。
-u	以合并的方式来显示文件内容的不同。
-a	diff 预设只会逐行比较文本文件。
-b	不检查空格字符的不同。
-d	使用不同的演算法，以较小的单位来做比较。
-i	不检查大小写的不同。
-y	以并列的方式显示文件的异同之处。
-W<宽度>	在使用-y 参数时，指定栏宽。

diff 命令使用示例：

执行如下命令，比较 test1.txt 文件和 test2.txt 文件，以并排格式输出。

```
diff test1.txt test2.txt -y -W 50
```

```
[root@~]# diff test1.txt test2.txt -y -W 50
```

返回结果如下所示，您可以看到 test1.txt 文件和 test2.txt 文件的不同之处。

```
[root@localhost ~]# diff test1.txt test2.txt -y -W 50
1      |      1  1
2      |      2  2
3      |      3  3
4      |      4  4
5      |      5  5
      >      6  6
      >      7  7
      >      8  8
      >      9  9
      >     10 10
```

4、file 命令

file 命令描述：该命令用于辨识文件类型。

file 命令语法：file [参数] [文件]。

参数说明：

参数	说明
-b	列出辨识结果时，不显示文件名称。
-c	详细显示指令执行过程，便于排错或分析程序执行的情形。
-f<名称文件>	指定名称文件，其内容有一个或多个文件名称时，让 file 依序辨识这些文件，格式为每列一个文件名称。
-L	直接显示符号连接所指向的文件的类别。
-v	显示版本信息。
-z	解读压缩文件的内容。

file 命令使用示例：

A、执行如下命令，显示 test1.txt 文件类型。

```
file test1.txt
```

```
[root@i77777777777777777777 ~]# file test1.txt
```

返回结果如下所示，您可以看到 test1.txt 文件类型是 ASCII text。

```
[root@i77777777777777777777 ~]# file test1.txt  
test1.txt: ASCII text
```

B、执行如下命令，显示 test2.txt 文件类型并不显示文件名称。

```
file -b test2.txt
```

```
[root@i77777777777777777777 ? ~]# file -b test2.txt
```

5、find 命令

find 命令描述：该命令用来在指定目录下查找文件。任何位于参数之前的字符串都将被视为欲查找的目录名。如果使用该命令时，不设置任何参数，则 find 命令将在当前目录下查找子目录与文件。并且将查找到的子目录和文件全部进行显示。

find 命令语法：find [参数] [文件]。

参数说明：

参数	说明
-mount	只检查和指定目录在同一个文件系统下的文件，避免列出其它文件系统中的文件。
-amin n	在过去 n 分钟内被读取过文件。
-type c	文件类型是 c 的文件。
-cmin n	在过去 n 分钟内被修改过。
-name name	查找文件名称为 name 的文件。

find 命令使用示例：

A、执行如下命令，将当前目录及其子目录下所有文件后缀为.txt 的文件列出来。

```
find . -name "*.txt"
```

返回结果如下所示。

```
[root@iZbp1-j0q-p10mxa0c321wZ ~]# find . -name "*.txt"
./test1.txt
./test2.txt
```

B、执行如下命令，查找系统中所有文件长度为 0 的普通文件，并列出它们的完整路径。

```
find / -type f -size 0 -exec ls -l {} \;
```

```
[root@iZbp1-j0q-p10mxa0c321wZ ~]# find / -type f -size 0 -exec ls -l {} \;
```

返回结果如下所示。

```
-rw-r--r-- 1 root root 0 Apr 3 2013 /usr/share/doc/json-c-0.11/NEWS
-rw-r--r-- 1 root root 0 Mar 8 2013 /usr/share/doc/python-slip-dbus-0.4.0/example/import_marker.py
-rw-r--r-- 1 root root 0 Jan 2 2019 /usr/share/lsb/4.1/submodules/core-4.1-noarch
-rw-r--r-- 1 root root 0 Jan 2 2019 /usr/share/lsb/4.1/submodules/security-4.1-noarch
-rw-r--r-- 1 root root 0 Jan 2 2019 /usr/share/lsb/4.1/submodules/core-4.1-amd64
-rw-r--r-- 1 root root 0 Jan 2 2019 /usr/share/lsb/4.1/submodules/security-4.1-amd64
-rw-r--r-- 1 root root 0 Mar 25 11:46 /usr/share/mime/icons
-rw-r--r-- 1 root root 0 Nov 26 2020 /usr/lib64/python2.7/email/mime/_init_.py
-rw-r--r-- 1 root root 0 Nov 26 2020 /usr/lib64/python2.7/pydoc_data/_init_.py
-rw-r--r-- 1 root root 0 Mar 25 03:53 /usr/lib64/python2.7/site-packages/PyYAML-5.4.1.dist-info/REQUESTED
-rw-r--r-- 1 root root 0 Nov 11 2020 /usr/lib64/python3.6/email/mime/_init_.py
-rw-r--r-- 1 root root 0 Nov 11 2020 /usr/lib64/python3.6/pydoc_data/_init_.py
-rw-r--r-- 1 root root 0 Nov 11 2020 /usr/lib64/python3.6/urllib/_init_.py
-rw-r--r-- 1 root root 0 Jun 28 15:33 /usr/local/share/assist-daemon/aa2d0258-ffe9-11e7-ba89-0ed5f89f718b-assist_daemon
-rw-r--r-- 1 root root 0 Mar 25 03:53 /usr/local/aegis/aegis_update/aegis_singleapp_update
-rw-r--r-- 1 root root 0 Nov 16 2016 /usr/local/aegis/PythonLoader/lib/python2.7/bsddb/test/_init_.py
-rw-r--r-- 1 root root 0 Nov 16 2016 /usr/local/aegis/PythonLoader/lib/python2.7/email/test/_init_.py
-rw-r--r-- 1 root root 0 Nov 16 2016 /usr/local/aegis/PythonLoader/lib/python2.7/email/mime/_init_.py
-rw-r--r-- 1 root root 0 Nov 16 2016 /usr/local/aegis/PythonLoader/lib/python2.7/pydoc_data/_init_.py
-rw-r--r-- 1 root root 0 Nov 16 2016 /usr/local/aegis/PythonLoader/lib/python2.7/lib-tk/test/test_ttk/_init_.py
-rw-r--r-- 1 root root 0 Nov 16 2016 /usr/local/aegis/PythonLoader/lib/python2.7/lib-tk/test/test_tkinter/_init_.py
-rw-r--r-- 1 root root 0 Nov 16 2016 /usr/local/aegis/PythonLoader/lib/python2.7/test/nullcert.pem
-rw-r--r-- 1 root root 0 Nov 16 2016 /usr/local/aegis/PythonLoader/lib/python2.7/lib2to3/tests/data/fixers/myfixes/_init_.py
-rw-r--r-- 1 root root 0 Nov 16 2016 /usr/local/aegis/PythonLoader/lib/python2.7/sqlite3/test/_init_.py
-rw-r--r-- 1 root root 0 Jan 4 2017 /usr/local/aegis/PythonLoader/third_party/requests/packages/urllib3/packages/backports/_init_.py
-rw-r--r-- 1 root root 0 Jan 4 2017 /usr/local/aegis/PythonLoader/third_party/requests/packages/urllib3/contrib/_init_.py
-rw-r--r-- 1 root root 0 May 6 2020 /usr/local/aegis/PythonLoader/third_party/kazoo/tests/_init_.py
-rw-r--r-- 1 root root 0 May 6 2020 /usr/local/aegis/PythonLoader/third_party/couchdb/tests/_init_.py
-rw-r--r-- 1 root root 0 Oct 8 2012 /usr/local/aegis/PythonLoader/third_party/lxml/includes/_init_.py
-rw-r--r-- 1 root root 0 Apr 28 2020 /usr/local/aegis/PythonLoader/third_party/crossplane/ext/_init_.py
-rw-r--r-- 1 root root 0 Apr 14 17:39 /usr/local/aegis/PythonLoader/third_party/google/protobuf/compiler/_init_.py
-rw-r--r-- 1 root root 0 Apr 14 17:39 /usr/local/aegis/PythonLoader/third_party/google/protobuf/pyext/_init_.py
-rw-r--r-- 1 root root 0 Apr 14 17:39 /usr/local/aegis/PythonLoader/third_party/google/protobuf/util/_init_.py
-rw-r--r-- 1 root root 0 Mar 3 2020 /usr/local/aegis/PythonLoader/third_party/bsddb3/test/_init_.py
-rw-r--r-- 1 root root 0 Mar 18 2017 /usr/local/aegis/PythonLoader/third_party/pymysql/constants/_init_.py
-rw-r--r-- 1 root root 0 Mar 25 03:53 /usr/local/aegis/aegis_client/aegis_10_85/singleApp_aegisInner
-rw-r--r-- 1 root root 0 Jun 28 15:10 /usr/local/aegis/aegis_client/aegis_10_85/data/web_path
-rw-r--r-- 1 root root 0 Jun 28 15:11 /usr/local/aegis/aegis_client/aegis_10_95/singleApp_aegisInner
-rw-r--r-- 1 root root 0 Jun 28 15:32 /usr/local/aegis/aegis_client/aegis_10_95/data/web_path
```

<https://blog.csdn.net/gangyikeji>

用户反馈

mattpower: 通过本次活动了解并使用了 Linux 的入门基本操作。对后续的学习有了一定基础。课程讲解十分详细，并能通过免费创建的实验室资源直接进行练习，十分方便，省出了自己搭建环境的时间，并且在网页上同步展示，能够方便查看和学习！十分感谢阿里云提供的这次学习机会！

Fano: 第二期的内容都是对 Linux 环境下的系统命令进行学习，涉及了许多常用命令。此外还学习了 Vim 编辑器。如果能够熟练掌握本期训练营的全部内容，在没有图形界面的时候就可以不那么捉急了。这期内容，对 Linux 小白来讲真是特别友好。如果有朋友刚接触 Linux，我想我会十分乐意把这期的这几个实验推荐给他。

机器猫: 此次学习，主要讲述的是面对不同要求所用到的相关操作命令。所牵涉的基本都是需要自己背诵记忆的，建议在实验中一边操作一边记忆命令，实际与理论相结合，效率相对单纯的背诵肯定要高很多，也可以在交流群里交流各自的记忆方法。

第一帅帅: 从实战中，对 linux 命令有了更好的掌握，之前都是从网上看命令，解释的基本不全面，没有服务器的情况下，只能看看无法实战。感谢这次活动主办方，能够让我们广大开发者能够参与学习和实战，对基本入门命令有了全面的系统的学习和掌握。

溜爷: Linux 的话，日常工作中接触的非常的多，但很多命令我使用的还不是很熟悉，主要日常都是使用宝塔面板了。

通过本期课程的学习，让我对 Linux 的一些命令有了新的认识，虽然现在让我直接上手，不一定都能全部搞定，毕竟接触的少，比较生疏。但是通过本次课程，让我对后续工作有了新的想法，而且直接通过 Linux 执行操作的话，不仅感觉高大上，也能减少系统资源的浪费。所以，后续如果有机会的话，我觉得有必要多熟悉命令。

古城小白：在参加实战训练营的时候，感觉到自己对 Linux 的入门都算不上，更别说熟练使用 Linux 命令了。这也使自己在参加训练营后收获颇丰。

参加训练营时，对训练营提供的体验手册与云产品资源体验感觉很好。之前学习 Linux 命令需要自己搭建环境，这让原本想学习 Linux 的命令的我几度有了放弃的想法。每次想敲命令充实一下自己。但是因为没有环境而让自己有了放弃的理由。人们的内心本来对自己的非舒适区就有一定的抵触，在走向自己核心目的道路上出现些许的阻碍或者岔路。80%的人往往先选择放弃。训练营不仅提供的云产品资源可以让我们一键创建学习环境，还提供了详细的体验手册让我们直接可以通过手册进行命令的学习。这样大大减少了学习的路上的障碍。使我们可以专心去学习。

高晓波：学习并进行是 linux 命令的实战操作，能够了解命令的实际功能，同时也提高了自己在 linux 领域打下了基础，后续会继续学习阿里云开发者社区的其他教程,希望以后自己能够学有所用,并且能够在开发者领域有更大的提升。

汪啾啾：认真的做了这几个实验，发现受益匪浅。阿里云的这几个实验把基本的 linux 命令讲解的很透彻，比如 vim 命令和参数，文本处理的参数，mkdir 创建目录，-p 就可以递归创建目录，以及磁盘分区命令：fdisk。让我对这些基础的命令复习的同时又学习到了新的知识。