git

git log 显示最近到最远的提交日志

HEAD -> 表示当前版本

HEAD^ 上一版本

HEAD^^ 上上版本

指向版本

git reset --hard commit\_id

git reset --hard HEAD^ 退回到上一版本

查看提交历史,以便回到哪个版本

git log

要回到最新的版本就要指向最新的commit ID

git reflog 查看命令历史

工作区就是电脑上的目录，如learning目录，.git目录是版本库,暂存区就是index或stage.

步聚

git add 把文件添加到暂存区

git commit 把文件从暂存区提交到当前分支(git创建版本库时自动创建了一个master分支)

可以理解为工作区文件修改通通添加到暂存区，然后一次性提交所有修改到分支

git跟踪并管理的是修改

git commit 只负责把暂存区的修改提交到分区

git checkout -- file 把文件在工作区的修改全部撤消

file修改没添加到暂存区,checkout后file恢复成与版本库一样的状态

file修改后已添加到暂存区,又对了file做修改, checkout后file恢得到添加到暂存区后的状态

暂存区的修改撤消

假如已经把readme.txt add到暂存区了

git reset HEAD readme.txt #丢弃暂存区readme的修改

然后就可以继续丢弃工作区的readme.txt

删除文件

本地删除test.txt,远程仓库删除文件 git rm test.txt,并且要commit

如果误删除本地test.txt,可以git checkout -- test.txt恢复

本地库关联远程仓库

git remote add origin git@server-name:path/repo-name.git

第一次推送

git push -u origin master #之后推送不用-u

本地克隆远程仓库

git clone git@egegegegeeg.git

clone支持https和ssh协议,ssh速度，https每次推送需要账密

分支

查看分支 git branch

创建分支 git branch <name> 或 git switch <name>

切换分支 git checkout <name> 或 git switch <name>

创建加切换分支 git checkout <name> 或 git switch -c <name>

合并分支到当前分支 git merge <name>

如当前分支为master,合并dev到当前分支

git merge dev

删除分支 git branch -d <dev>

冲突

GIT无法自动合并分支时，必须先解决冲突文件(手动修改文件内容一样)。解决冲突后，先提交后再合并.

git log --graph 查看分支合并图

linux

目录权限

rwx对于目录，内容是文件名。

r 读到目录的文件名

w 修改文件名(文件或目录),建立新的文件与目录，移动或更名

x cd 能进入到目录里面

20200421

普通合并，不使用fast forward合并

master为主分支,dev分支有作了修改，切换回master后

git merge --no-ff -m "修改信息" dev

查看历史合并信息

git log --graph --pretty=oneline --abbrev-commit

#普通合并能看出分支合并信息, fast forward不行

保存现场

如在dev分支上正在进行代码工作，此时master要改临时的bug，所以先要保存dev的工作现场(dev已添加修改到暂存区，不添加保存不了现场)

git stash

Saved working directory and index state WIP on dev: f52c633 add merge

保存后就切换到master分支然后创建临时分支issue-1,在issu-1分支上修改完后再切回master分支普通合并issue分支

然后再切换到dev分支

git stash list 查看保存的工作现场

恢复工作现场

git stash pop #恢复的同时把stash内容也删除

最后因为dev分支是从master出来的,dev分支上也同样有bug，此时需要把master修复的提效复制到dev

git cherry-pick <commit ID>

强行删除没有合并的分支

git branch -D commit\_ID #开发一个新的feature(功能)最好新建一个分支.若新分支修改完成后没有被合并到分支(如dev)就要删除，要加上 -D

推送分支

git push origin master #推送本地master分支到远程origin库

git push origin dev #推送本地开发分支到远程origin库

本地创建和远程分支对应的分支，如dev分支#默认clone时只创建master对应

创建dev对应

git checkout -b dev origin/dev

多人协作

1 首先，先试图用git push origin 分支到远程仓库

2 如果推送失败，则是因为远程仓库分比本地要新，则先git pull试图将远程库分支合并到本地

3 如果合并有冲突，先解决冲突后在本地commit提交，最后就能push到远程分支

如果git pull 提示no tracking information,则说明本地分支与远程会支的链路接关系没有创建。

用命令git branch --set-upstream-to=origin/dev dev

dev为分支

标签

git tag <tagname> 用于新建标签 ，默认最新HEAD，也可以指定COMMIT\_ID

git tag -a <tagname> -m "信息" 指定标签信息

git tag #查看所有标签

git show <tagname> #查看标签信息

标签与commit\_id挂钩,如果这个commit有master与dev分支，则两个分支都可以看到标签

推送一个本地tag到远程

git push origin <tagname>

推送全部未推送的标签到远程

git push origin --tags

本地删除一个标签

git tag -d <tagname>

删除远程标签

先在本地删除

git push origin :refs/tags/<tagname>

分支与标签很像，但分支可以移动，标签不行

.gitignore

忽回文件原则

1 忽略系统自动生成的文件，比如缩略图等

2 忽略包含敏感信息文件，比如方账密或数据库文件

3 忽略编译产生的中间文件，可执行文件

如果你确实想添加该文件，可以用-f强制添加到Git：

$ git add -f App.class

或者你发现，可能是.gitignore写得有问题，需要找出来到底哪个规则写错了，可以用git check-ignore命令检查：

$ git check-ignore -v App.class

.gitignore:3:\*.class App.class

git reset --hard commitid 指向某一版本

git checkout -- file

20200420

监控主机看要被监控主机的地域来开设机房的机子，传输稳定减少变数

被监控机子要提供验收范围(port与服务)

windows设置远程桌面白名单

入站规则先找到对应的PORT,然后

在“作用域”中的“远程IP地址”，点击“添加”按钮，添加白名单ip，点击确定

20200425

mysql all数据库备份与还原

备份

mysqldump -uroot -pabc123 --all-databases > all.sql

还原

mysql -uroot -pabc123 < all.sql

bash

\rm -r 目录 #\忽略掉alias的指定选项

cp 链接文件 文件 #命令指行后的文件不是链路接文件而是被链路接指向的源文件

cp -d 链接文件 文件 # 得出的文件也是链路接文件

cp -a #将文件的所有属性一起复制

20200427

数据库用户，如果网站不是太重要，或是内部服务器，数据库的用户可以是 root

修改时区命令

timedatectl

touch #创建空文件或修改文件时间

touch 文件 #修改时间与读取时间与状态时间都会变成当前时间

touch -d "2 days ago" 文件 #文件的读取与修改时间会变成2天前,状太时间为当时间

20200429

mysqldump -uroot -p -B database > database.sql

#使用-B的好处就是导出的数据库文件中已经存在创建库和使用库的语句，不需要手动在要还原的机子上创建库的操作，可直接还原。

还原

mysq -u root -p '123456' < database.sql #还原时不需要指定恢复的数据库

linux

查找etc目录下容量大于50K且小于60K的文件并ls -l查看详细信息

find /etc -size +50k -a -size -60k -exec ls -l {} \;

查找etc目录下容量大于1500K以及容量等于0的文件

find /etc -size +1500k -o -size 0 -exec ls -l {} \;

# -a 且 , -o 或

查看文件的隐藏属性

lsattr file

设置隐藏属性

chattr [+aiS]

locate 文件 #locate通过数据库/var/lib/mlocate查找文件,不用找硬盘速度快

20200430

SUID

文件具有suid权限时，代表当用户执行此二进制文件时会暂时具有文件拥有者的权限

SGID

目录具有sgid时,代表用户在此目录下新建的文件的用户组会与该目录的组名相同

SBIT

目录具有sbit时，代表该目录下用户建立的文件只有自己与root能删除

设置

SUID 4

SGID 2

SBIT 1

如 4775 #4为设置suid

7775 #设置 suid,sgid,sbit

计算机安全，通信

截获

被动攻击，如cain软件利用ARP欺骗把自身的mac地址(假网关地址)发给同网段各主机,这样同网段各主机发送报文都能截获，如user/pass

篡改

主动攻击,如cain软件利用arp欺骗把自身的mac地址(假网关地址)发给网段，各主机访问外网时cain提供假域名给各主机,cain主要去访问真域名，再把信息转给各主机，这是一种DNS劫持。

中断

后端主机控制大量肉鸡发送大量无用的数据包给服务器，占用下载带宽，使其正常服务不能使用.DDOS攻击

伪造

同一网段ip邦定mac来上网，若这一台机子关机，另一台机子可以有机会使用相网的ip上网，达到伪造的效果

磁盘/硬盘

磁片

主轴马达

机械手臂与磁头

文件系统

通常是一个分区对应一个文件系统，现在多数是一个挂载点对应一个文件系统

一个文件有文件实际内容与文件的权限属性等相关的参数

inode

记录文件的权限与属性相关的参数，一个文件站用一个inode，同时记录该文件的实际数据所在的区块号码，有编号

数据区块

实际记录文件的内容，若文件太大会占用多个区块，有编号

超级区块

记录文件系统的整体信息，包括Inode与数据区块的总容量，使用量，剩余量，以及文件系统的格式与相关信息

索引式文件系统

通过Indode可以一口气读取所有的数据区块,读写性能较好

FAT(u盘类)是非索引式文件系统，没有inode,每个区块的号码都记录在前一个区块中,不能一次性读取。所谓的碎片整理就是把太过于离散的区块整理集中。

ext2文件系统

文件系统一开始就会将inode和数据区块规划好，除非重新格式化，否则inode和数据区块是不会变的

如果文件系统高达数百G，将所有的inode和数据区块放在一起不易管理，所以会有多个区块组(block group)

文件系统最前面有一个启动扇区(boot sector),可以安装引导程序，我们可以安装不同的启动引导到别的的文件系统的的boot sector，这样就不用覆盖磁盘的唯一MBR，实现多引导

区块组

每个区块组都有独立的超级区块，Inode,数据区块

数据区块

inode table

inode,记录区块号码可以有12个直接指向区块的号码，

一个间接

如果文件太,再拿一个区块来当作记录区块号码的记录区

一个双间接，一个三间接

inode对应表

记录空白的与使用的In(空区块)ode,哪些可以使用哪些不可以使用

区块对应表

记录可使用的区块(空区块)，不可使用的区块

文件系统描述

超级区块

事实上除了第一扇区会有超级区块外，后续的区块组不一定含有超级区块。若含有则是主要为第一个超级区块做备份

目录

linux下新建一个目录时，会分配一个Inode与至少一块区块给目录，inode记录目录相关的权限与属性，并可记录分配到哪块区块号码，而区块则是记录目录下的文件名与该文件名占用的Inode号码数据

文件

新建一个文件时，会分配一个inode与相对该文件大小的区块数量给文件

/

由于目录树是由根开始，因此系统通过挂载的信息可以找到挂载点的Inode号码，就能找到根目录的inode内容，再通过该inode读取根目录区块内的文件名数据，再一层一层往下读到正确的文件名

日志文件系统

inode对照表，数据块对照表，超级区块经常变动，称为元数据

inode tabes，数据块区块称为数据存放区域

万一如果出现停电或发生错误导致系统中断，数据存放区域与元数据产生不一致的情况，那么系统重新启动时就要通过超级区块和文件系统状态来进行数据一致性检查，浪费大量时间，所以就有日志文件系统的出现。

日志文件系统，系统中规划了一个区块专门记录文件写入与修改的步聚，当数据出现不一致性时，只需检查该区块即可，不用针对整个文件系统检查。

异步处理

cpu处理文件时，文件要先放到内存中，如果编辑文件太大，因为内存比磁盘速度快，所以等待读写入磁盘的时间太久，没效率。

异步处理就是当系统加载一个文件到内存后，如果该文件没有被修改，则内存区段的数据就设置为clean,若修改则设置为dirty,此时操作都还在内存中进行，并没有写入磁盘。系统会不定时的把内存中dirty的数据写入磁盘，保持数据一致性。也可以手动sync命令

系统会将常用的数据放置到内存缓冲，加快文件系统的读写操作，所以Linux的物理内存都会被用光，正常现象，加速系统性能。

20200501

linux的内核通过VFS来管理读取文件系统

xfs

centos7.x默认的文件系统

数据三个部分

数据区(data section)

与ext2的区块群组一样，不同的是xfs的inode与区块都是系统需要用到时动态配置产生，所以格式化操作超快

文件系统登录区

有点日志区块，还可以指定外部磁盘来作为xfs文件系统的日志区块

实时运行区

当有文件新建立时，xfs会在这个区段里面找一个到数个的extent区，将文件放置在这个区块内，等分配完毕后再写入到data section的inodae与区块中

20200504

硬链接

不能跨文件系统

不能跨接目录

如果链接的比如是etc目录,那目录下的所有文件都要硬链路接，工作环境太过复杂

硬链接就是多个文件名指向同一个Inode，文件系统的inode与数据区块一般是不会变化的

创建硬链接

ln /etc/crontab . #创建硬链接到当前目录

[root@localhost ~]# ls -il crontab /etc/crontab

4372557 -rw-r--r--. 2 root root 451 Jun 10 2014 crontab

4372557 -rw-r--r--. 2 root root 451 Jun 10 2014 /etc/crontab

第3个字段2，代表是有多少个文件名链路接到这个Inode

软链接（快捷方程式)

符号链接就是建立一个独立的文件，而这个文件会让数据的读取指向它链接的那个文件的文件名

创建软链

ln -s /etc/crontab crontab2

[root@localhost ~]# ll -i crontab2

8414548 lrwxrwxrwx. 1 root root 12 Apr 30 00:48 crontab2 -> /etc/crontab

当源文件被删除后，符号链接文件会打不开，实际上是找不到原始文件名而已

符号链接所建立的文件为一个独立的新文件，所以会占用inode与区块

目录链接数

当我们建立一个新目录时，新的目录链接数为2，上层目录的链接数增加1

磁盘分区

lsblk #列出系统上的所有磁盘列表

blkid #列出设备的uuid参数， uuid全局唯一标识符，linux会将系统内所有的设备给于一个标识符，标识符可以拿来挂载或使用

parted /dev/sda print #列出磁盘的分区表类型与分区信息

gdisk/fdisk /dev/sdb #对/dev/sdb磁盘进行分区管理(增删等)，q退出不保存，注意不要去处理一个正在使用的分区

一般先用lsblk找到磁盘，然后用parted找出分区格式(gpt or MBR), 再利用fdisk分区(MBR格式的话)

用fdisk分好区并w保存后，分区表并没有立即更新，需要

partprobe -s #更新linux内核的分区信息

查看内核分区的信息

lsblk

cat /proc/partitions

磁盘格式化(创建文件系统)

mkfs.xfs /dev/sdb6 #一般使用默认参数就行，除非有其它额外的要求，如-f, 强制格式化,如果原来已有文件系统

mkfs.ext4

文件系统检验

xfs\_repair /dev/sdb4 #恢复命令,万一服务器停电或硬件软件出错导致文件系统发生错乱

通常文件系统出问题时才使用(单人模式下)，正常状况使用此命令可能会造成系统损坏，且被检验的硬盘分区不能挂载，要在卸载状态下使用

20200505

挂载与卸载

单一文件系统对应单一目录(挂载点)，目录应该是空目录才行，如果不是那原先存在目录下的文件在该目录被挂截之后会暂时先隐藏起来(并不是消失)，当目卸载之后才会看到

mount #什么都不接就是列出目前挂载信息

mount -a #依照配置/etc/fstab的数据把所有没挂载的磁盘都挂载上来

mount -t #指定挂载的文件系统种类。ceoto 7 会自动分析文件系统种类来尝试挂载设备(根据文件系统的超级区块与linux自己的驱动程序去测试)，测试成功则自动挂载.

主要参考这两个文件来

/etc/filesystems 系统指定的测试挂载文件系统类型的优先级

/proc/filesystems linux系统已加载的文件系统类型

如何知道系统有没有相关的文件系统的驱动程序呢

/lib/modules/$(uname -r)/kernel/fs/

mount 文件系统 挂载点

如 mount /dev/sdb6 /newfile

mount UUID /newfile

挂载cd/dvd, u盘(vfat格式)

先blkid查看设备名称,再mount挂载,一般cd/dvd挂载后都是只能只读，使用率为100%，u盘最后是vfat格式,linux本身不支持ntfs格式，如果有中文文件名的话，可以指定挂载参数语系

重新挂载或挂载不特定目录

如当/为只读状态，可以重启系统或重新挂载

mount -o remount,rw,auto /

可以将某个目录挂载到另一个目录，有点类似软链，特殊时才会用

mount /var /test/var #将/var这个目录暂时挂载到/test/var下面

卸载

umount 文件系统设备名/挂载点

umount /dev/sdb6

umount /test

磁盘/文件系统自定义参数

mknod #设置如/dev/sda 文件名，一般系统会自动帮设置好

xfs\_admin #设置lable name与uuid

20200507

启动挂载

/etc/fstab #开机就启动的文件挂载文件，注意 / 必须是要先挂载的且必须挂载

六字段意义

[设备/uuid] [挂载点] [文件系统类型如xfs] [文件系统参数] [dump] [fsck为]

设备/uuid

可以是设备名如/dev/sda2,也可以是uuid,也可以是label名称

挂载点

一般是目录

文件系统类型

xfs, ext4，nfs等

文件系统参数

rw,auto,exec等相关参数

dump

备份方案太多一般不用,为0即可

fsck

检验扇区,早期的系统使用,xfs不使用,xfs会自己检验,为0即可

/etc/fstab修改好后，一定要测试语法正确性, mount -a 检验成功与否

/etc/fstab是启动配置文件，不过实际挂载信息是记录到/etc/mtab与/proc/mounts这两个文件，万一如果/etc/fstab写入的数据出错导致无法顺利启动而进入单人维护模式,就得重新挂载 /, 此时/为只读mtab这两个文件不能写。

mount -n -o remount,rw /

特殊设备loop挂载(镜像文件不刻录就挂载使用)

cd/dvd镜像文件

mkdir -p /data/centos\_dvd

mount -o loop /tmp/Centos-7.0.XXXDVD.iso /data/ceotos\_dvd #这就就能够不需要将iso刻录成dvd或光盘就读取到内部的数据了，并且可以修改里面的文件，这就了为什么镜像文件会提供md5验证码给用户确认该镜像没有问题

建立大文件以制作loop设备文件

比如一开始分区不合理，只有/目录有多余的容量，可以在/里面制作出一个大文件用来挂载，就相当于多了一个分区

dd if=/dev/zero of=/srv/loopdev bs=1M count=512

#if input file,输入文件,/dev/zero是会一直输出0的装备

#of outputfile 将一堆0写入到后面的文件中，

block 相当于文件系统的区块

count 总共有多少bs

mkfs.xfs -f /srv/loopdev

mount -o loop /srv/loopdev /data/file

交换分区

交换分区就是如果内存不足时，会把内存中不常用的数据或程序暂时放到磁盘上的交换分区里，以空出内存给后续程序或数据使用

两种创建交换分区的方式

使用磁盘分区来格式化一个swap文件系统

先用分区出来一块如1G的分区

mkswap /dev/sdb6 #格式他生成一个是swap文件系统

swapon /devsdb6 #启动交换分区

swapon -s #查看交换分区或使用free 查看

开机启动vi /etc/fstab

/dev/sdb6 swap swap defaults 0 0 #第二栏没有挂载点

使用大文件来格式化生成一个swap文件系统

dd if=/dev/sero of=/srv/swap\_dev bs=1M count=1024 #生成大文件

mkswap /srv/swap\_dev #格式化swap文件

swapon /srv/swap\_dev #启动

开机启动vi /etc/fstab

/srv/swap\_dev swap swap defaults 0 0 # 第一栏必须是设备名不能是uuid,因为系统只会查询设备的uuid

关闭swap

swapoff /dev/sdb6

swapoff /srv/swap\_dev

ls -l 显示的total数值

ll -sh,目录下的total值是文件数据区块数量\*区块大小值

root@localhost ~]# ll -sh

total 10M

4.0K -rw-r--r--. 1 root root 120 Apr 19 11:23 1.txt.tar.gz

4.0K -rw-------. 1 root root 1.3K Aug 11 2019 anaconda-ks.cfg

4.0K -rw-r--r--. 2 root root 451 Jun 10 2014 crontab

crontab的实际大小是451bytes，但这个文件占用一个区块(4k大小)

20200509

文件压缩

压缩技技原理

如数字1用1个字节表示,事实计算机的最小计算单位是1bit，所以数字1就是00000001,利用一些特殊的算法通过把前7个0"丢掉“就达到了节省空间的目的

如1111111111111111111111110，前面有十几个1甚至更多,可以用15\*1来表示存储之类达到节省空间

gzip

替代compress,最常用的压缩命令,gzip压缩后的文件可以被windows的winrar和7.zip解压

gzip 1.txt # 会自动生成1.txt.gz，.gz后缀名文件,注意源文件1.txt会不再存在

gzip -d 1.txt.gz #解压缩得到1.txt, gz文件会不再存在

gzip -v 1.txt #压缩的同时显示压缩比

zcat/zmore/zless/zgrep 可以用来查看或查找被压缩的文件

gzip -9 -c service > service.gz #-c将原本要压缩的内容变成文字类型从屏幕输出，然后利用>输出到service.gz(手动建立)，这样原文件与压缩文件都同时存在

bzip2 与gzip用法一样，压缩比更好，后缀名bz2

xz用法与gzip用法一样，压缩比更好，后缀名 xz

xz -l abc.xz #显示压缩前后容量对比

xz -k abc.txt #压宿文件同时保留源文件

tar

gzip等压缩软件只能对单一文件解压缩,tar是将多个文件或目录进行打包并能结合gzip解压缩命令

tar [-jzJ]cv -f 打包的文件名 要打包的文件 #打包并压缩

tar [-jzJ]tv -f 压缩的文件 #查看文件

tar [-jzJ]xv -f 压缩的文件 [-C 目录] #解压文件

-p #保留备份数据的原始权限与属性

-P #保留绝对路径,即允许备份数据中含有根目录,慎用

--exclude=文件 #不包含某个文件

备份/etc

tar -zpcvf /root/etc.tar.gz /etc

查看tar内部文件

tar -ztvf /root/etc.tar.gz

解压.tar.gz 到/tmp目录下

tar -zxvf /root/etc.tar.gz -C /tmp

打包某目录但不包含特定目录

tar -zcvf /root/system.tar.gz --exclude=/root/etc\* \

--exclude=/root/system.tar.gz /etc /root #打包etc和root目录但不包含root目录下的etc相关文件与自己本身,exclude最好放在tar.gz后面

备份比某个时刻要新的文件

[root@localhost ~]# ll /etc/passwd

-rw-r--r--. 1 root root 1498 Apr 29 05:09 /etc/passwd #日期为20200429

tar -zcvf /root/etc.newer.then.passwd.tar.gz --newer-mtime="20200429" /etc/\* #打包比20200429要新的mtime的文件

tar -ztvf /root/etc.newer.than.passwd.tar.gz | grep -v "/$" # 调用grep找出非/结尾的文件就是我们要的

tar -cvf file.tar # 仅是打包文件称为tarfile

tar -zcvf file.tar.gz #有压缩的支持称为tarball

tar可以将文件打包到某些特定的设备中，如磁带tab,磁带是一次性读取/写入设备，不能用cp来复制,如将/home /root /etc备份到磁带/dev/st0

tar -cvf /dev/st0 /home /root /etc

xfs文件系统的备份与恢复

xfsdump #备份，可以完整备份，增量备份，注意只能备份已挂载的文件系经

xfsrestore #还原，可以还原守整备份，增量备份(要安顺序level0->level1---),可以还原指定的文件, -i交互模式

光盘写入工具

先将所需要备份的数据创建成为一个镜像文件iso

将该镜像文件刻录到cd或dvd中

一般用图形界面软件来操作即可，无需用命令行工具

20200511

dd

制做文件或备份功能(基本直接读取扇区)

dd if="input file" of="output\_file" bs="block\_size" count="number"

if 就是Inputfile ,可以是设备

of 主是outputfile, 可以是设备

bs 设置一个block大小默认512byte(扇区大小)

count 多少个bs

复制/etc/passwd到/tmp/passwd.back中

dd if=/etc/passwd of=/tmp/passwd.back #

dd if=/dev/sr0 of=/tmp/system.iso #复制刻录好的光盘的内容备份成为镜像文件

把上面的iso刻录到U盘，假设/dev/sda为u盘

dd if=/tmp/system.iso of=/dev/sda #可以linux镜像文件写样做u盘就具有可启动功能了，然后可以安装Linux系统

将/boot文件系统通过dd备份下来，假设/boot挂载设备为/dev/vda2

dd if=/dev/vda2 of=/tmp/vda2.img #文件大小与磁盘大小一样，哪怕磁盘只使用50%

将/dev/vda2完整地复制到另一个硬盘分区上，假设我们已经分区好一个比vda2大的分区sda1

不需要格式化sda1

dd if=/dev/vda2 of=/dev/sda1

xfs\_repair -L /dev/sda1 #先清楚一堆log

#] uuidgen

948gjei-egeg-48g05656-56

xfs\_admin -U 948gjei-egeg-48g05656-56 /dev/sda1 #这两行是用于创建新的uuid，因为dd复制时连同uuid都复制过来了

mount /dev/sda1 /mnt #挂载发现/mnt与/boot一模一样

xfs\_growfs /mnt #系统放大空间

dd是将原本旧的硬盘分区上面的扇区数据整个复制过来，连同超级区块，启动扇区元数据等，所以不用格式化

如果想创建两块一模一样的磁盘，只要执行类似 dd if=/dev/sda of=/dev/sdb, sdb不用分区与格式化，因为命令包括MBR与分区表都复制过sdb了

cpio

可以备份任何东西，包括设备文件，要结合find与管道 |

也可以将系统数据完整的备份到磁带

备/boot下的所有文件到/tmp

cd /

find boot | cpio -ocvB > /tmp/boot.cpio #需要去掉根目录，与tar 的-P一个道理

20200512

vim

一般命令模式

ctrl + f 向下移动一页

ctrl + b 向上移动一页

0或Home键，移动到一行的最前面的字符处

$ 或END， 移动到一行的最后面的字符处

G移动到文件的最后一行

gg移动到文件的第一行

/word 向光标之下寻找一个名为word的字符串

?word 向光标之上寻找word

n 重复前一个查找的操作,如/word继续查找

N 与n相反，反向查找

:n1,n2s/word1/word2/g 替换n1到n2行的word1变成word2

:1,$s/word1/word2/g 从第一行到最后一行全都替换在word2

:1,$s/word1/word2/gc 从第一行到最后一行都替换，但替换前让客户确认

x, X x向后删除一个字符, X向前删除一个字符

dd 删除光标所在的那一行

ndd n为数字,向下删除n行

yy 复制光标所在那一行

nyy 复制光标以下的n行

p与P p为将已复制的内容在光标下一行粘巾,P为上一行

u 恢复前一个 ctrf + r 重做上一个操作

. 重复前一个操作

一般模式切换到编辑模式

i与I， i在光标处插入, I在本行的第一个非空字符处插入

a与A a在光标下一个字符出插入, A在光标所在行的最后一个字符处插入

o与O o在光标所在的下一行插入新的一行,O在光标所在处的上一行插入新的一行

r和R r替光标所在的字符一次, R一直替换直到esc

编辑模式

命令行模式

:w 保存

:w! 强制保存

:q 退出

:q! 强制退出

:wq 退出后保存

:w filename 将编辑的数据保存为另一个文件

:! command 暂时退出vim执行下命令

:set nu 显示行号

:set nonu 取消行号

vim缓存，恢复与打开警告

当修改还没来得及保存退出时，再重新vim时会出现警告,缓存文件为.file.swp。或多人打开并编辑了file

如果是其他人在编辑，可以选 择O模式只读

如果是未来得及保存则可以

选R，使用swp来恢复文件再决定要不要保存，这样可以救回来之前没保存的修改，退出后记得删除file.swp

或者不要恢复修改直接删除.swp文件

可视区块

Ctrl + v 选中之后可以d删除或y复制后再p粘贴

多文件

多窗口

在用vim打开一个文件后，在命令行输入

:sp filename , 打开一个新窗口，如果有file表示打开一个新文件，没有file则打开同一个文件

ctrl + w + 下 先按住ctrl再按w,然后放开所有的健, 再下 下 则光标移动到下方的窗口

ctrl + w + 上 移动到上方的窗口

:q 退出窗口

命令补全

20200514

利用iconv可以进行文件编码的转换

dos2unix, unix2dos可以变更文件的每一行的换行符,如windows与linux文件的换行符不一样

bash

控制计算机硬件的是操作系统的内核,我们用过shell输入命令与内核沟通，让内核来操作硬件准确工作，bash属于壳程序(shell),其它应用程序也称为壳程序

type 命令可以查看命令是外部命令还是bash内置的命令,后接执行命令，不能是文件

[root@localhost ~]# type cd

cd is a shell builtin

# cp /test/1.txt /12/2.txt /3/3.txt \

>/5/5/txt /root #复制4个文件到/root目录下, \转义回车键，\后面要紧接着ENTER,不能多空格,因为\仅转义下一个字符

提示符命令行下

ctrl + u 或 ctrl + k 从光标处向前删除命令或向后删除命令

ctrl + a 或 ctrl +e 从光标处移动到整个命令的最前面或最后面

一般登录后,linux会根据/etc/passwd设置一个shell，默认bash

环境变量通常以大写字符表示

变量，类型默认字符串

变量的设置，变量与变量内容以一个=号来连接

name=VBird

等号两边不能接空格,变量内容有空格需用双引号或单引号结合起来

name = VBird name=Vbird good #错误

变量只能是英文与数字且不能数字开头

2myname=VBird #错误

双引号内的特殊字符可以保持原本的属性

var="lang is $LANG" -> echo $var -> lang is zh\_CN.UTF8

单引号内的特殊字符为一般字符

var=‘lang is $LANG' -> echo $var -> lang is $LANG

可用\转义符将特殊字符([ENter], \ , ', 空格等)变成一般字符

name=Vbird\'s\ good -> echo $name -> Vbird's good

在一串命令中需要借由其它命令提供信息时,可以使用反单引号`或$()

version=$(uname -r)

变量扩增内容可以用${}字符

PATH=${PATH}:/home/hello

若变量需要在子进程中执行，

export 变量 #使其在为环境变量

取消变量

unset 变量

设置变量代替工作目录

work="/home/good" #不用""也行

env 显示所有的环境变量

RANDOM 生成随机数的变量, 0~32767

set 观察显示所有的环境变量与用户自定义变量

PS1='[\u@\h \W]\$ ' 命令提示符就是 [root@www~]#

$ shell的PID

echo $$ 2787

? 上个执行命令的返回值

ls -h -> echo $? -> 0 #命令能成功执行就返回0，否则就返回非0

export 自定变量转成环境变量

export myname #使myname变成环境变量，这样myname就可以在子进程中使用，export后没接变量时，会列出所有的环境变量

父进程与子进程

Linux登录之后得到一个BASH(父进程),在这Bash下面所执行的任何命令(子进程)都是由这个BASH所衍生出来的.如在父进程的bash下面执行bash进入到另一个界面(子进程)，原本的Bash就会暂时sleep，当子进程bash exit退出后才回到原本的BASH。

子进程仅继承父进程的环境变量,不会继承自定义变量

declare 将环境变量转成自定义变量

为什么环境变量的数据可以被字进程引用?

因为内存配置关系，启动一个shell时，操作系统会给这个SHELL在内存中分配一个区域，些内存中的变量就可以让子进程使用。若父进程用export功能，就可以让自定义的变量的内容写到上述内存区域中(环境变量)，当加载另一个子进程时(离开父进程),子shell可以调用父进程的环境变量所在内存区域到自己的环境变量内存区域中

语系 locale

查看语系

locale

查看支持的所有语系

locale -a #没有的话可以安装语言包

修时修改语系

LANG=en\_US.utf8

永处修改要写到配置文件去

cat /etc/locale.conf

LANG=en\_US.utf8

read 读取来自键盘输入的变量内容

-p 后面接提示字符

-t 接等待的秒数

read -p "please enter your name: " -t 30 name -> hello -> echo $name -> hello

declare 声明变量的类型

-a 将后面的变量定义为数组(array)类型

-i 将后面的变量定义为整数类型

-x 用法与export一样，将后面的变量变成环境变量

-r 交后面的变量设置成为readonly，不可更改内容与不能unset

-P 单独列出变量的类型

declare -i sum=100+200+300 -> echo $sum -> 600 #变量类型默认为字符串所以要声明为整数类型,BASH的数值运算中,默认最多仅能到达整数形态，所以1/3结果是0

数组

设置数组 var[index]=content

var[1]="small min"

var[2]="big min"

var[3]="nice min"

echo ${var[1]} -> "small min"

echo "${var[1]}, ${var[2]}, ${var[3]}"

ulimit

可以限制用户某些资源，如可以开启的文件数量,可以使用的cpu时间，使用的内存总量等

-a 列出所有的限制额度

-f 此shell可以建立的最大文件容量，单位为kbytes

变量的删除与替换

[root@localhost ~]# echo ${path}

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin

删除/usr/local/sbin 第一个目录

[root@localhost ~]# echo ${path#/\*local/sbin:}

/usr/local/bin:/usr/sbin:/usr/bin:/root/bin

删除前面所有的目录只保留 /root/bin

[root@localhost ~]# echo ${path##/\*bin:}

/root/bin

# 删除符合文字的最短那一个

## 删除符合文字的最长那一个

删除最后一个目录

[root@localhost ~]# echo ${path%:\*bin}

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin

只保留第一个目录

[root@localhost ~]# echo ${path%%:\*bin}

/usr/local/sbin

% %% 与# ##类似，但是从后面匹配

${变量/旧字符/新字符} 替换符合的第一个旧字符

￥{变量//旧字符/新字符} 全部替换

20200519

prtg添加https域名监控,add sensor要选择HTTP Advanced类型，然后域名填http://abc.com就行,Require Keyword要选择Set sensor to error if keyword is missing,对应的字符串可以填要监控的字符串(检测有没有劫持),也可以随便填比如(abc.com)

变量的测试与内容转换

root@localhost ~]# echo $username

username变量没有设置时

[root@localhost ~]# username=${username-root}

[root@localhost ~]# echo $username

root

变量为空时

[root@localhost ~]# username=""

[root@localhost ~]# username=${username-root}

[root@localhost ~]# echo $username

[root@localhost ~]# username=${username:-root}

[root@localhost ~]# echo $username

root

其它先跳过

命令别名

alias, 可以将常用或长的命令创建成一个新的命令

alias lm='ls -al|mort' #

也可以替换命令

alias rm='rm -i' #这样rm就变成了新的有-i参数的命令

alias 直接列出所有别名

unalias 取消别名

历史命令

history

-n 列出最近n条命令

-c 将目前的shell中的所有history内容全部清除

-w 将目前的history记录到histfiles中

当以bash登录后,系统会主动同~/.bash\_history读取以前执行

过的命令。~/.bash\_history能记录的数量与HISTFILESIZE变量有关.假设这次登录后执行了100条命令,那当注销时,系统会将101~1100更新到~/.bash\_history中，也就是说注销时会把最近的HISTFILESIZE记录到记录文件中,也可以用history -w强制写入

多用户登录命令写入为最后注销那个

bash shell的操作环境

路径与命令查找顺序

1, 相对/绝对路径执行命令, 如/bin/ls

2, 由alias找到该命令执行

3, 由bahs内置的(builtin)命令来执行

4, 通过$PATH变的的顺序找到的第一个变量来执行

可以通过type -a ls 查看顺序

bash登录与欢迎信息

/etc/issue #登录就看到的欢迎信息如操作系统与硬件等级，可以修改

/etc/motd #可以设置让用户登录就看到的信息,如某时间点系统维护

bash的环境配置文件

一进入Bash就会取得一些有用的变量是因为系统有一些环境配置文件，让bash启动时就直接读取这些文件，这些环境配置文件又分为全局系统配置文件与文人偏好配置文件，alias与自定义的变量在bash注销时会失效，要想保留就得写得配置文件中

login shell 与non-login shell

区别在于有没有登录

login shell

取得bash时需要完整的登录流程,如tty1~tty6登录需要账密,些时取得的bash称为login shell

non-login shell

如在X的图形界面中启动终端或在原本的bash环境下再次输入bash命令,都是不需要账密的

login shell与non-loginshell读取的配置文件不一样

