

psReference

1. 《Unix/Linux/Osx 中的Shell 的编程》
2. 《Linux Shell 命令行及脚本编程实例详解》
- 3.

H1 Unix/Linux

linux中一切皆文件

ref:[common unix shell commands](#) & [wiki_unix](#)

1. Practice:

<https://wiki.vip.corp.ebay.com/display/DW/Unix>

2. 文件特点可以不指定 文件格式，不同的软件 (**notepad++,word,记事本 etc**) 都可以打开

H1 Linux

H3 结构框架

通过终端 (**putty or mutputty**) 写**shell** 指令，去访问内核



- 内核系统: centos & ubuntu(鸟班图)

一般来说著名的linux系统基本上分两大类:

- RedHat系列: Redhat、Centos、Fedora等
- Debian系列: Debian、Ubuntu等

RedHat 系列

1. 常见的安装包格式 rpm包, 安装rpm包的命令是 "rpm -参数"
2. 包管理工具 yum
3. 支持tar包

Debian系列

- 1. 常见的安装包格式 deb包, 安装deb包的命令是 "dpkg -参数"
- 2. 包管理工具 apt-get
- 3. 支持tar包

yum可以用于运作rpm包, 例如在Fedora系统上对某个软件的管理:

1. 安装: yum install
2. 卸载: yum remove
3. 更新: yum update

apt-get可以用于运作deb包, 例如在Ubuntu系统上对某个软件的管理:

1. 安装: apt-get install
2. 卸载: apt-get remove

H4 内核cli

1. service --status-all :

```
--full-restart, the script is run twice, first with the stop command, then with the start command.  
service --status-all runs all init scripts, in alphabetical order, with the status command. The status is [ + ] for running services, [ - ] for stopped services and [ ? ] for services without a status command. This option only calls status for sysvinit jobs.  
ONES
```

H3 background

• window vs linus vs unix

1. window 内核是nt(new technology) ,linus 内核是shell ,后者开源;
2. unix 是linus的父亲, 其中linus是开源, 是类unix操作系统

• slash 正斜杠(/),back-slash 反斜杠(\)

• 虚拟机和操作系统

• 指令+ --help , 可以查看其他选项命令含义:

mkdir --help

wc --help

-->拓展: man(manual) 指令

```
man wc  
-l :line  
-w :words  
-c :bytes;  
-m :chars;
```

man mkdir man rename :与上面等价

-

H3 File attributes

W3CSCHOOL

- **ls -l** : 查看文件属性以及文件所属的用户和组别

```
$ ls -l
total 8
drwxr-xr-x 1 wenbluo 1049089 0 Mar 19 09:35 123/
-rw-r--r-- 1 wenbluo 1049089 389 Mar 18 14:41 test_1.sh
-rw-r--r-- 1 wenbluo 1049089 324 Mar 18 14:15 test_1.txt
-rw-r--r-- 1 wenbluo 1049089 400 Mar 18 14:56 test_2.sh
-rw-r--r-- 1 wenbluo 1049089 165 Mar 18 15:05 test_3.sh
-rw-r--r-- 1 wenbluo 1049089 101 Mar 18 17:17 test_4.sh
-rw-r--r-- 1 wenbluo 1049089 194 Mar 18 17:42 test_5.sh
-rw-r--r-- 1 wenbluo 1049089 22 Mar 19 09:43 test_6.sh
-rw-r--r-- 1 wenbluo 1049089 43 Mar 19 09:48 test_7.sh
```

备注:

- **ls** 是简单有什么文件: (list simple)
- **ll** 是列出文件具体信息 (d l - owner size date)
- 其中可以一定规则输出结果:

ls命令指令

后面的 **a l lt ltr** : 都是 '指定选项' : 通过 “-” letter 操作; 命令选项

For list_details

ls -a	输出所有文件包含隐藏文件
ls -l	输出明细信息
ls -lt	以时间顺序输出 (默认倒叙, 从大到小)
ls -ltr	以时间顺序倒序 (从小到大)
ls etl_home	可以直接指定目录, 而无需先cd etl_home 再ls
ls -i	返回节点node
ls -h	<pre>[HERC]~/export/home/b_clsfd/wenbluo/ad_cube/ > ls -lh total 45M 1125950461 drwxrwxrwx 2 b_clsfd hdmi-technology 4.0K Aug 15 04:18 123/ 1125950463 -rw-rw-rw- 1 b_clsfd hdmi-technology 14K Aug 15 04:14 ad_cube_history.spark.sql 1124553973 -rw-rw-rw- 1 b_clsfd hdmi-technology 568K Aug 18 19:38 tmp_log_2017_01_010.log 1124553974 -rw-rw-rw- 1 b_clsfd hdmi-technology 609K Aug 18 19:38 tmp_log_2017_01_011.log 1124553975 -rw-rw-rw- 1 b_clsfd hdmi-technology 144K Aug 14 08:53 tmp_log_2017_01_012.log 1124553976 -rw-rw-rw- 1 b_clsfd hdmi-technology 515K Aug 18 19:38 tmp_log_2017_01_013.log 1124553977 -rw-rw-rw- 1 b_clsfd hdmi-technology 621K Aug 18 19:38 tmp_log_2017_01_014.log 1125950456 -rw-rw-rw- 1 b_clsfd hdmi-technology 16M Aug 18 19:38 tmp_log_2017_01_015.log 1125950457 -rw-rw-rw- 1 b_clsfd hdmi-technology 504K Aug 18 19:38 tmp_log_2017_01_016.log</pre> <p>h: human</p>

标框部分显示为 隐藏文件: 文件前面有一个.

```
[wenbluo@phxdpeet1019 ~]$ ls -a
. .bash_history .bash_profile .etlenv .kshrc .logins .ssh
.. .bash_logout .bashrc etl_home .lessht .profile test_Crontab.out
[wenbluo@phxdpeet1019 ~]$ ls -i
total 30
2593918133 -rwxrwxr-x 1 wenbluo wenbluo 19909 Mar 27 22:58 clsfd_daily_mkt_sc_bak.sql
2593918131 -rw-rw-r-- 1 wenbluo wenbluo 0 Mar 28 19:00 crontab_chewu
2593918135 drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 3 22:25 huangxiaoxia
2593918136 drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 2 21:56 test
2593918129 drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 3 22:28 wbluo
2593918135 drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 4 02:04 WbLuo
[wenbluo@phxdpeet1019 sql_test]$ ls -i
2593918133 -clsfd_daily_mkt_sc_bak.sql 2593918131 crontab_chewu 2593918125 huangxiaoxia
```

d:表示目录， **-**:表示文件, **l**:表示链接文档

1. 在linux中每个文件有所有者、所在组、其他组的概念

- 所有者: 文件创建者
- 当某个用户创建一个文件之后, 这个文件的所在组就是该用户所在的组

ll test_5.sh -->查看文件

```
wenbluo@L-SHC-16505239 MINGW64 ~/desktop/wenbluo/111
$ ll test_4.sh
-rw-r--r-- 1 wenbluo 1049089 101 Mar 18 17:17 test_4.sh

wenbluo@L-SHC-16505239 MINGW64 ~/desktop/wenbluo/111
$
```

[rwx]:表示read ,write ,execute

每个文件的属性由左边第一部分的10个字符来确定 (如下图)。

文件类型	属主权限			属组权限			其他用户权限		
0	1	2	3	4	5	6	7	8	9
d 目录文件	r	w	X	r	-	X	r	-	X
	读	写	执行	读	写	执行	读	写	执行

[1-3]:user : U ,[4-6] group : G ,[7-9] others :O

3.1.1 第一个字符表示文件的类型:

- [d] 目录 (directory)
- [-] 文件 (file, f)
- [l] 符号链接文件 (Symbolic links file)
- [b] 块设备文件 (block)
- [c] 字符设备文件 (character)
- [p] 命名管道 (pipe)
- [s] 套接字 (socket)

--> 拓展如何ls 只看某文件类型

```
[wenbluo@phxdpeet1019 sql_test]$ ls -l
total 148
-rwxrwxr-x 1 wenbluo wenbluo 19909 Mar 27 22:58
clsf_d_daily_mkt_sc_bak.sql
-rw-rw-r-- 1 wenbluo wenbluo 0 Mar 28 19:00 crontab_chewu
-rw-rw-r-- 1 wenbluo wenbluo 0 Apr 12 00:52 CURRENT_DATE
-rwxrwxr-x 1 wenbluo wenbluo 112148 Apr 15 23:36
dw_clsf_d.job_runner_v2.ksh
drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 12 01:07 huangxiaoxia
drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 24 01:44 test
```

```
drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 16 00:13 wbluo
###只看文件
[wenbluo@phxdpeetl019 sql_test]$ ls -l |grep ^-
-rwxrwxr-x 1 wenbluo wenbluo 19909 Mar 27 22:58
clsfd_daily_mkt_sc_bak.sql
-rw-rw-r-- 1 wenbluo wenbluo 0 Mar 28 19:00 crontab_chewu
-rw-rw-r-- 1 wenbluo wenbluo 0 Apr 12 00:52 CURRENT_DATE
-rwxrwxr-x 1 wenbluo wenbluo 112148 Apr 15 23:36
dw_clsfid.job_runner_v2.ksh
```

####只看目录

```
[wenbluo@phxdpeetl019 sql_test]$ ls -l |grep ^d
drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 12 01:07 huangxiaoxia
drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 24 01:44 test
drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 16 00:13 wbluo
```

-->拓展：

- 字符设备和块设备的区别
- 块设备：硬盘（有固定的chunks），可以随意访问
- 字符设备：键盘，提供字符流，不能随意访问

第一组rwx：文件所有者的权限是读、写和执行

- 第二组rw-：与文件所有者同一组的用户的权限是读、写但不能执行

- 第三组r--：不与文件所有者同组的其他用户的权限是读不能写和执行

也可用数字表示为：r=4, w=2, x=1 因此rwx=4+2+1=7

- 1 表示连接的文件数

wenbluo :表示文件所有者

1049089 : 表示用户所在的组

- (3) 对此文件加入可执行权限
- chmod +x ./hello.sh

```
yuxiaomi@yxm:~$ ll hello.sh
-rwxrwxr-x 1 yuxiaomi yuxiaomi 33 7月 27 17:36 hello.sh*
```

- **/tmp** 文件：

是Unix系统专门用于放置临时文件的目录，每次系统重启时候，其中的内容都会被清空。

- **/dev/null** :

是Unix系统一个特殊文件，任何人都可以读取或写入。向该文件写入的任何东西都会消失，就像一个巨大的黑洞；

```

[bash: cd: dev/null: not a directory
[wenbluo@phxpeetl019 ~] $ ll dev/null
crw-rw-rw- 1 root root 1, 3 Mar 28 2018 dev/null
[wenbluo@phxpeetl019 ~] $ pwd
/

```

- **ksh .sh what's difference?**

1. In Linux extensions generally have no purpose like they do in DOS/Windows so aren't required. If someone puts an extension on a file they're just trying to let people know what it is without having to open it or run the "file" command on it.
2. ksh (kore shell) is a superset of sh with a lot of compatibility

Wikipedia also contains a lot of info about the shell.
For shell scripts, use sh or POSIX shell if you need maximum compatibility, and bash if not. Bash has many features that make it easier to write and read scripts. For example, this original Bourne shell code:

```

Code:
if [ $a + $b -gt 13 ]
then ...

```

can be written as follows in bash:

```

Code:
if (( a+b>13 ))
then ...

```

On many (most? all?) distros /bin/sh is a symbolic link to /bin/bash; you need to check if the Bourne shell exists at all.

```

ksh target_table_load_handler.ksh
dw_clsfid.clsfd_daily_mkt_sc_wenbluo td2
dw_clsfid.clsfd_daily_mkt_sc_wbluo.sql  start_date=2018-01-01
end_date=2019-03-27

```

-

H3 Linux远程登陆

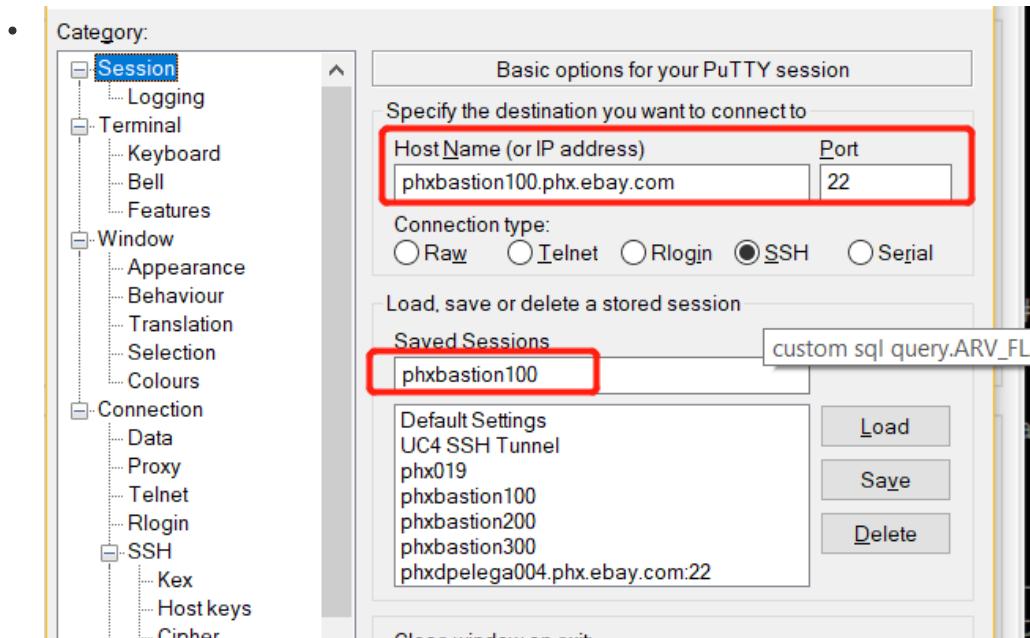
1. putty远程登陆linux服务器
2. [putty tutorial & CSDN](#)

Linux在服务器端应用的普及，Linux系统管理越来越依赖于远程，在各种远程登录工具中，Putty是出色的工具之一，然后Putty是最常见的一个SSH工具

H4 [putty connect](#)

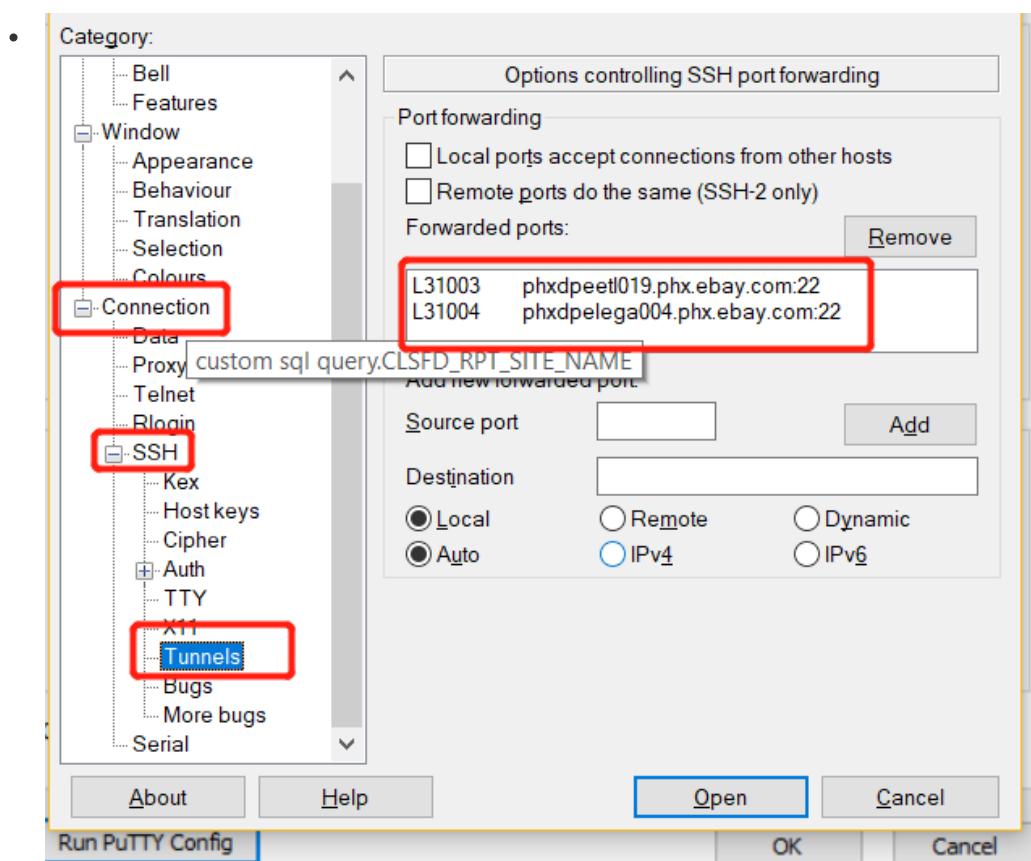
H5 Tunnel

可以[download MTputty](#) (可以一个窗口 管理多个putty)



SSH默认端口号 : 22

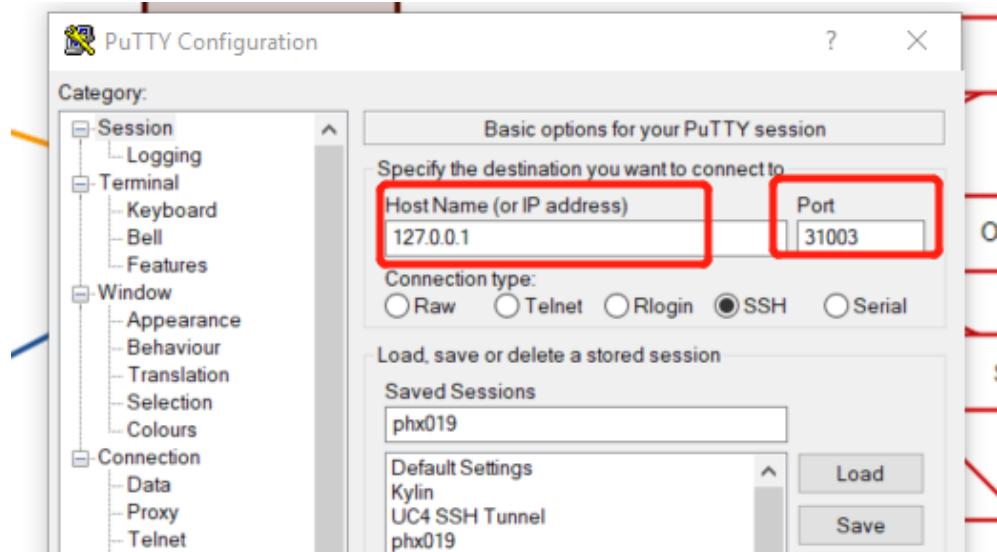
```
ssh: Could not resolve hostname ssh: Name or service not known
[wenbluo@phxdpeetl019 ~]$ ssh apollo-rno-devours.vip.hadoop.ebay.com
ssh: connect to host apollo-rno-devours.vip.hadoop.ebay.com port 22: Connection timed out
[wenbluo@phxdpeetl019 ~]$ ssh apollo-rno-devours.vip.hadoop.ebay.com
```



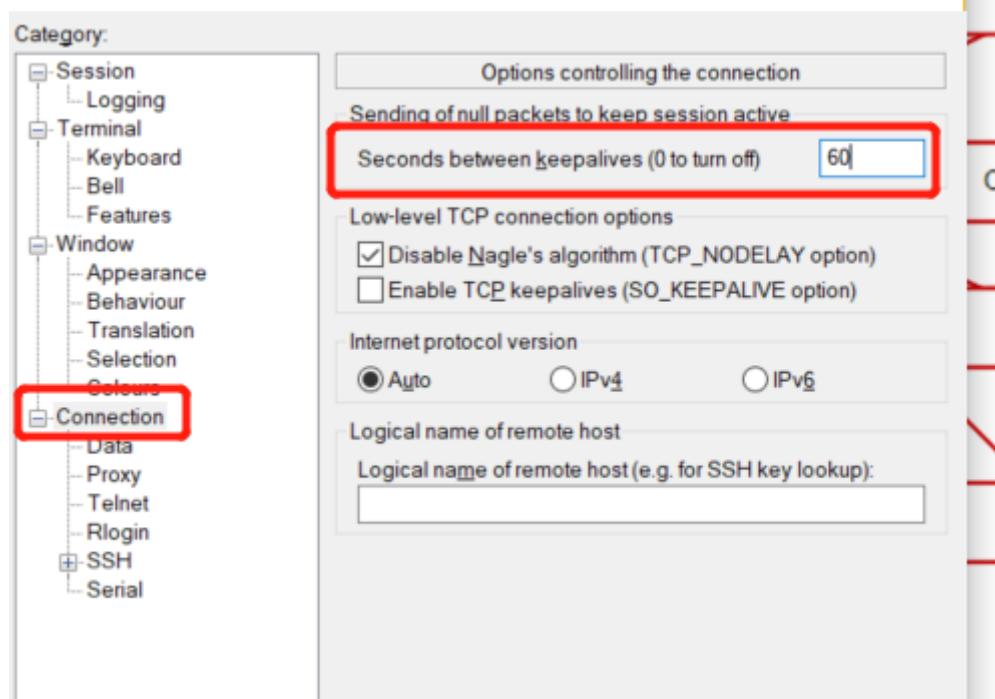
31003:SOURCE Port : 表示隧道入口

Phxdpeetl019.phx.ebay.com.22:表示隧道出口

然后Tunnels:作用是远程主机有firewall时候，可以创建tunnel形式连接到主机上。



127.0.0.1:表示localhost 不需要联网，都是本机访问.



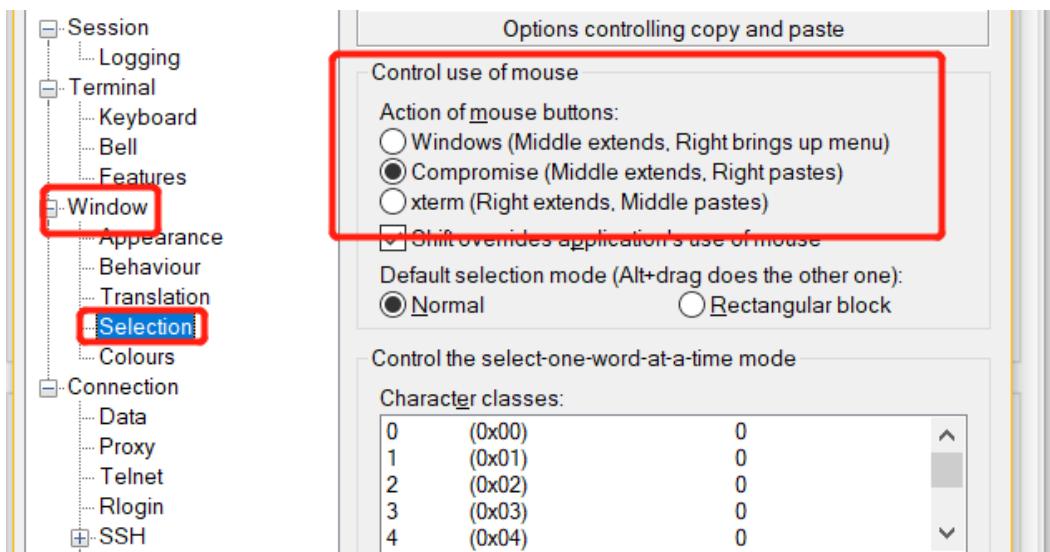
s 设置每隔60s发送一次数据包，保持连接

以及在.ssh目录下，添加文件 conf ServerAliveInterval 2

```
ssh-keygen -R "you server hostname or ip"
```

--to generate the new/valid key

H5 Copy and paste



Compromise: 左键选择，右键粘贴

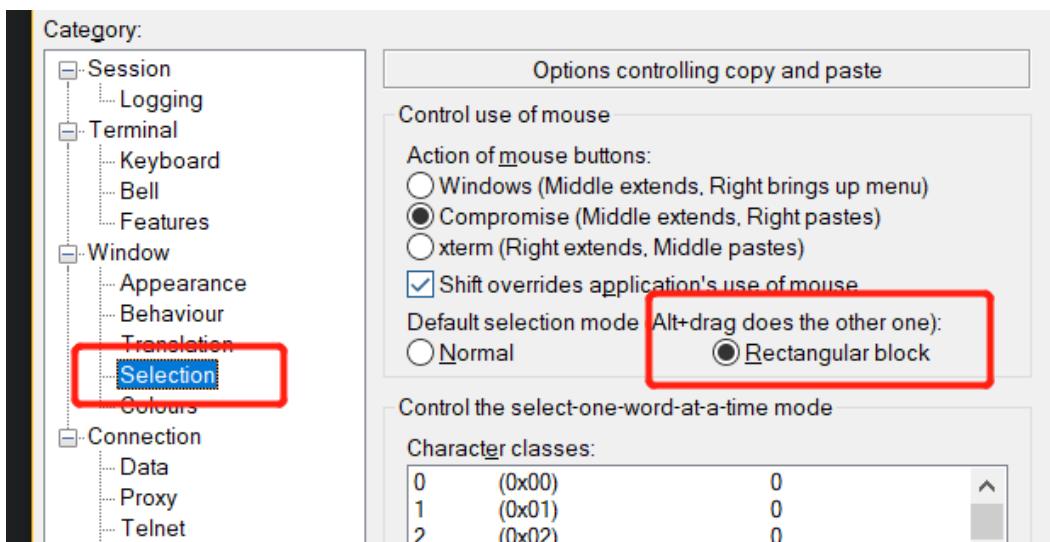
```
#####
# This system is for the use of authorized users only.
# Individuals using this computer system without
# authority, or in excess of their authority, are
# subject to having all of their activities on this
# system monitored and recorded by system personnel.
#
# In the course of monitoring individuals improperly
#
```

如何实现复制的时候 只复制有数据的，而不是跟图一一样，整行复制

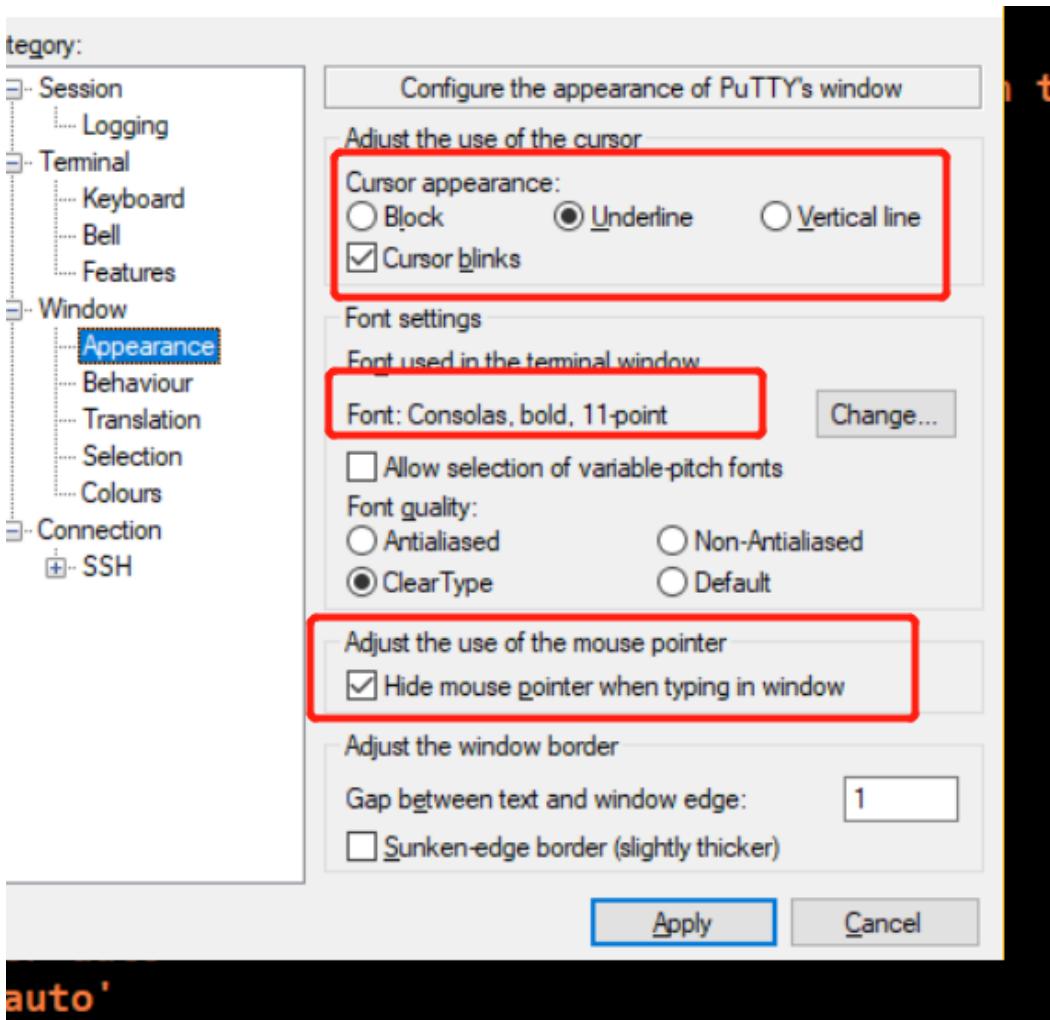
```

[wenbluo@ares-devour.vip.ebay.com ~] $ cat /etc/motd
wenbluo@ares-devour.vip.ebay.com's password:
Last login: Thu May 30 21:06:12 2019 from lvslb43-3-vlan422-floatingip3.lvs.ebay.com
#####
# This system is for the use of authorized users only.
# Individuals using this computer system without
# authority, or in excess of their authority, are
# subject to having all of their activities on this
# system monitored and recorded by system personnel.
#
# In the course of monitoring individuals improperly
# using this system, or in the course of system
# maintenance, the activities of authorized users
#

```



H5 Appearance

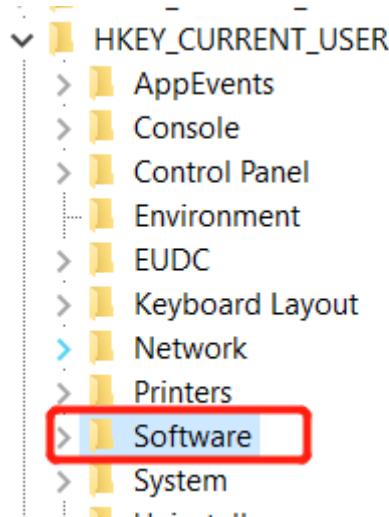


auto'

设置字体大小以及输入提示符

H5 Back 备份

win+run :regedit

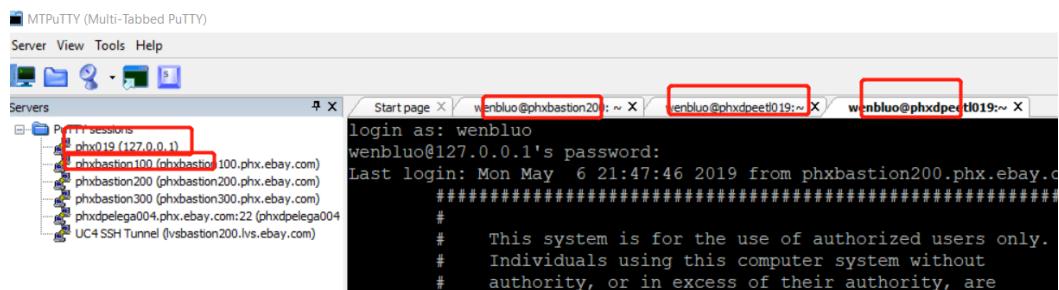


选择putty -->export ,

保存注册表

#####

- 通过跳板机baston 再远程SSH 到 phx019 服务器



1. [wiki_putty_SSH_Tunnels](#)

2. SSH(security shell)

[ssh](#)

- 远程连接主机的用于应用层和传输层的安全协议

3.

H3 公钥和私钥

H4 SSH 通信

winscp 本机传送文件 互信机制，集群内部之间是通过 (ssh) , cli 与集群之间是

ssh-keygen 用于为

生成、管理和转换认证密钥，包括 RSA 和 DSA 两种密钥。

密钥类型可以用 -t 选项指定。如果没有指定则默认生成用于SSH-2的RSA密钥。

ssh-keygen 还可以用来产生 Diffie-Hellman group exchange (DH-GEX) 中使用的素数模数。

参见 [模数和生成小节](#)。

一般说来，如果用户希望使用RSA或DSA认证，那么至少应该运行一次这个程序，

在 `~/.ssh/identity`, `~/.ssh/id_dsa` 或 `~/.ssh/id_rsa` 文件中创建认证所需的密钥。

另外，系统管理员还可以用它产生主机密钥。

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora/.git (GIT_DIR!)
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/users/wenbluo/.ssh/id_rsa): |
```

实际上，OpenSSH 7.0 及以上版本默认禁用了 ssh-dss (DSA) 公钥算法，都才作用 RSA 方法

1. RSA & DSA

- key

```
ssh-keygen ##生成公钥和密钥 ## key generate
cd /home/wenbluo/.ssh/
[wenbluo@phxdpeetl019 .ssh]$ ll
total 12
-rw----- 1 wenbluo wenbluo 1675 Mar 21 00:34 id_rsa#密钥
-rw-r--r-- 1 wenbluo wenbluo 415 Mar 21 00:34 id_rsa.pub ##公
钥
-rw-r--r-- 1 wenbluo wenbluo 2477 May 8 02:33 known_hosts

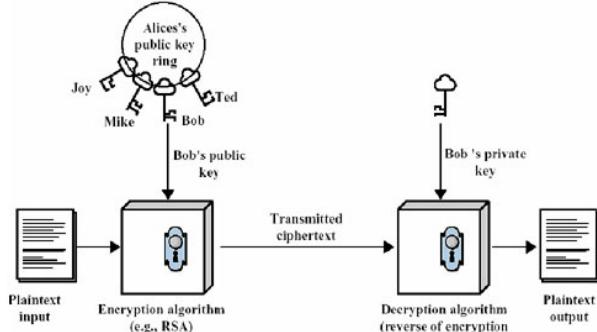
##打开服务器B，将刚刚在服务器A内复制的内容（公钥）追加
到/root/.ssh/authorized_keys
##这样ssh 到服务器B 时候就无需再次输入密码；

ssh-keyscan github.corp.ebay.com
ssh [-T/t] git@github.com : 查看是否能ssh 通
ssh -v git@github.com : 查看具体过程
ssh-agent : 打开代理
ssh-add /c/Users/wenbluo/.ssh/id_rsa : 将密钥添加到代理中
```

summary:

1. 公钥密钥成对出现，互相解密
2. 公钥加密，私钥解密

上面的过程可以用下图表示，Alice 使用 Bob 的公钥进行加密，Bob 使用自己的私钥进行解密。



例子和图出自《网络安全基础 应用与标准第二版》

3. 公钥认证，私钥加密（数字签证）

-->2 & 3 作用可以解释：RSA 为非对称加密原因是：公钥加密，公钥认证

ssh -vT git@github.com : trace

You can also check that the key is being used by trying to connect to `git@github.com`:

```
$ ssh -vT git@github.com
> ...
> debug1: identity file /Users/you/.ssh/id_rsa type -1
> debug1: identity file /Users/you/.ssh/id_rsa-cert type -1
> debug1: identity file /Users/you/.ssh/id_dsa type -1
> debug1: identity file /Users/you/.ssh/id_dsa-cert type -1
> ...
> debug1: Authentications that can continue: publickey
> debug1: Next authentication method: publickey
> debug1: Trying private key: /Users/you/.ssh/id_rsa
> debug1: Trying private key: /Users/you/.ssh/id_dsa
> debug1: No more authentication methods to try.
> Permission denied (publickey).
```

In that example, we did not have any keys for SSH to use. The "-1" at the end of the "identity file" lines means SSH couldn't find a file to use. Later on, the "Trying private key" lines also indicate that no file was found. If a file existed, those lines would be "1" and "Offering public key", respectively:

```
$ ssh -vT git@github.com
> ...
> debug1: identity file /Users/you/.ssh/id_rsa type 1
> ...
> debug1: Authentications that can continue: publickey
> debug1: Next authentication method: publickey
> debug1: Offering RSA public key: /Users/you/.ssh/id_rsa
```

ssh-agent

eval \$(ssh-agent) :开启agent

background: 管理密钥，当需要验证时候，自动将public key 与private key 匹对，减少重复输入密码。

ssh-add /c/Users/wenbluo/.ssh/id_rsa :添加密钥 给agent、

```
debug1: Authentications that can continue: publickey
debug1: No more authentication methods to try.
wenbluo@github.corp.ebay.com: Permission denied (publickey).

[venbluo@L-SHC-16505239 MINGW64 ~/Desktop/work/kylin_cube/New folder/ad_cube (ad_cube_09_07_26)
$ ssh -vT wenbluo@github.corp.ebay.com
```

- ssh 之 kown_hosts 的作用

```
[21:08:58]wenbluo@lvspeeti001 ~/.ssh cbob
warning: Permanently added 'lvspeeti001.lvs.ebay.com,10.102.21.101' (RSA) to the list of known hosts.
Last login: Wed Sep 11 20:40:21 2019 from localhost.localdomain
#####
#
```

每次ssh 到另一台机子时候，系统会默认添加到远程机子 到 kown_hosts 文件中。

1. **ps:** 可以删除该文件，只要密钥没有变动就行。
 - 2.
- 免密登陆

ssh 之authorized_keys

-->这个是免密登陆重要文件

master 's operation

1. **cp id_rsa.pub authorized_keys**
2. **chmod 644 authorized_keys**
3. **chmod 700 .ssh**
4. **scp .ssh/authorized_keys slave :xx/.ssh**

4. ssh slave

- alias 别名SSH

1. alias che_wu=ssh chewu@phxdpeetl019.phx.ebay.com
2. ssh -p 22 sk@server.example.com

##远程连接到sk 用户主机 ,端口号22

3. • 前提是设置免密登陆
 - config
 - 关键字:

```
1. Host cctl_server
   #HostName rnoetl-prod019-tess0025.stratus.rno.ebay.com
   HostName rnoetl-prod019-tess0025.stratus.rno.ebay.com
   #PreferredAuthentications publickey
   IdentityFile ~/.ssh/id_rsa

   Host chercules
   HostName hercules-lvs-cli.vip.ebay.com
   IdentityFile ~/.ssh/id_rsa

   Host stm2rno
   #HostName rnoetl-preprod004-tess0025.stratus.rno.ebay.com
   HostName rnoetl-preprod004-tess0025.stratus.rno.ebay.com
   IdentityFile ~/.ssh/id_rsa
```

2. host :alias , 后面直接运行指令 : ssh cctl_server
3. hostname : 远程ssh ip
4. user
5. IdentityFile: 私钥 : 简写 : -i

man ssh : show us more details args

```
-i IdentityFile
  Selects a file from which the identity (private key) for public key authentication is read. The default is ~/.ssh/identity for protocol version 1, and ~/.ssh/id_dsa, ~/.ssh/id_ecdsa, ~/.ssh/id_ed25519 and ~/.ssh/id_rsa for protocol version 2. Identity files may also be specified on a per-host basis in the configuration file. It is possible to have multiple -i options (and multiple identities specified in configuration files). ssh will also try to load certificate information from the filename obtained by appending .cert.pub to identity filenames.
```

ssh -i ./id_rsa -p 22 yuebli@lvsdpeetl001.lvs.ebay.com

ssh -i ~/.ssh/id_rsa_ceph 10.148.187.208

-o ServerAliveInterval=100 -o ServerAliveCountMax=3

:ServerAliveInterval 每隔100s 发送一次请求， 保持连接。

:ServerAliveCountMax:如果发送请求后没有response.则将在3*100后关闭连接

```

ServerAliveCountMax
    Sets the number of server alive messages (see below) which may be sent without ssh(1) receiving any messages back from the server. If this threshold is reached while server alive messages are being sent, ssh will disconnect from the server, terminating the session. It is important to note that the use of server alive messages is very different from TCPKeepAlive (below). The server alive messages are sent through the encrypted channel and therefore will not be spoofable. The TCP keepalive option enabled by TCPKeepAlive is spoofable. The server alive mechanism is valuable when the client or server depend on knowing when a connection has become inactive.

    The default value is 3. If, for example, ServerAliveInterval (see below) is set to 15 and ServerAliveCountMax is left at the default, if the server becomes unresponsive, ssh will disconnect after approximately 45 seconds. This option applies to protocol version 2 only.

ServerAliveInterval
    Sets a timeout interval in seconds after which if no data has been received from the server, ssh(1) will send a message through the encrypted channel to request a response from the server. The default is 0, indicating that these messages will not be sent to the server. This option applies to protocol version 2 only.

```

6.

- permission deny with publickey

```

debug1: Authentication attempts remaining: 0
debug1: No more authentication methods to try.
wenbluo@github.corp.ebay.com: Permission denied (publickey).

wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/work/kylin_cube/New folder/ad_cube (ad_cube_09_07_26)
$ ssh -T wenbluo@github.corp.ebay.com

```

始终使用 "git" 用户

所有连接（包括远程 URL 的连接）必须以 "git" 用户进行。如果尝试以 GitHub 用户名连接，将会失败：

```

$ ssh -T GITHUB-USERNAME@github.com
> Permission denied (publickey).

```

如果连接失败且您通过 GitHub 用户名使用远程 URL，可以[更改远程 URL 以使用 "git" 用户](#)。

应键入以下命令来验证连接：

```

$ ssh -T git@github.com
> Hi username! You've successfully authenticated...

```

```

ssh -L 9000:kylin.rno.corp.ebay.com:80
liapan@phxbastion100.phx.ebay.com

```

- 退出远程连接：exit
- ssh -4 -L 9000:kylin.rno.corp.ebay.com:443
wenbluo@phxbastion300.phx.ebay.com
- 如何让其他人可以免密ssh登陆我的机子
- 如何生成多个ssh-keygen ?

1. 多个git账号

2.

ssh -T git@github.com

ssh -T git@github.corp.ebay.com

3. 一个git 多个仓库

4. 最终clone 也可以采用 https or ssh

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (master)
$ git clone https://github.com/privatewenbluo/excel.git
Cloning into 'excel'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 18 (delta 5), reused 18 (delta 5), pack-reused 0
Unpacking objects: 100% (18/18), done.

wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (master)
$ |
```

ssh 通过http 代理

H5 hyperlink

1. ssh -t

2. ssh_hive

H3 Unix 创建软链

soft_symbolic_link

1. l : 表示 link 类似 window 的快捷方式

创建连接方式：

连接到 file1 并命名为 link1

\$ ln -s file1 link1

bin : 表示 link 1 (别名)

--> 表示 bin 实际对接路径 (file1)

```
[wenbluo@phxdpeet1019 etl_home]$ ls -ll
total 16
-rwxrwxr-x 1 wenbluo wenbluo 8986 Mar 25 03:12 abinitio.setup
lwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:05 bin -> /home/chewu/etl_home/bin
lwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:06 cfg -> /home/chewu/etl_home/cfg
lwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:06 dat -> /home/chewu/etl_home/dat
lwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:07 dbc -> /home/chewu/etl_homedbc
lwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:07 dml -> /home/chewu/etl_home/dml
lwxrwxrwx 1 wenbluo wenbluo 23 Mar 25 03:07 in -> /home/chewu/etl_home/in
lwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:07 lib -> /home/chewu/etl_home/lib
lwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:07 log -> /home/chewu/etl_home/log
lwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:07 out -> /home/chewu/etl_home/out
drwxrwxr-x 2 wenbluo wenbluo 4096 Mar 25 03:26 sql
lwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:08 tmp -> /home/chewu/etl_home/tmp
lwxrwxrwx 1 wenbluo wenbluo 26 Mar 25 03:08 watch -> /home/chewu/etl_home/watch
lwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:08 xfr -> /home/chewu/etl_home/xfr
```

2.

3.

H1 Shell

H4 其他快捷方式:

1. 查看时间 : date

--> 拓展: **date +‘format’** 以特定格式输出

```
date +'%Y%m%d-%H%M%S'
```

```
20190415-215810
```

```
[wenbluo@phxdpeet1019 ~]$ date +'%Y%m%d-%H%M%S'
```

```
20190415-Apr1924
```

```
[wenbluo@phxdpeet1019 ~]$ date '+%Y%m%d-%H%M%S'
```

```
20190415-Apr1935
```

###两者等价，一般后者最好 这样git 上也能使用；

-->**shell** 日期区分大小写:

%Y	年份(以四位数来表示)
%y	年份(以00-99来表示)
%d	以0-31表示
%D	含年月日
%m	月份(以01-12来表示)
%M	分钟(以00-59来表示)
%H	小时(00-23来表示)

2. 查看谁登陆 : who

3. 查看登陆用户名: whoami

H4 Alias

####

1. 临时设置，重启后无效

```
[wenbluo@phxdpeet1019 cfg]$ alias cbin='cd /home/wenbluo/etl_home/bin'  
[wenbluo@phxdpeet1019 cfg]$ cbin  
[wenbluo@phxdpeet1019 bin]$ pwd  
/home/wenbluo/etl_home/bin  
[wenbluo@phxdpeet1019 bin]$ alias  
-bash: alias: command not found  
[wenbluo@phxdpeet1019 bin]$ alias  
alias cbin='cd /home/wenbluo/etl_home/bin'  
alias ccfg='cd /home/wenbluo/etl_home/cfg'  
alias cfg_2=' cd /home/${whoami}/etl_home/cfg'  
alias csqI=' cd /home/wenbluo/etl_home/sql_test'  
alias l..is d .. --color=auto  
alias ll='ls -l --color=auto'  
alias ls='ls --color=auto'  
alias vi='vim'  
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'  
[wenbluo@phxdpeet1019 bin]$
```

定义方式

2. 如何长期有效？因为上面用alias方法，当再次重启时候就没有了；

通过编辑 /home/wenbluo 目录下的隐藏文件： vi .bashrc

```

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
alias ccfg='cd /home/wenbluo/etl_home/cfg'
alias csq1='cd /home/wenbluo/etl_home/sql_test'
alias cb1n='cd /home/wenbluo/etl_home/bin'
# User specific aliases and functions

```

然后重启，就能访问这些快捷键；

或者采用 . \$HOME/.bashrc

```

alias capollo='ssh Apollo-devour.vip.ebay.com'
alias cares='ssh ares-devour.vip.ebay.com'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
wenbluo@phxbastion100:~$ . $Home/.bashrc
-bash: /bashrc: No such file or directory
wenbluo@phxbastion100:~$ . $HOME/.bashrc
wenbluo@phxbastion100:~$ alias
alias alert='notify-send --urgency=low -i "$( [ $? = 0 ] \
    error)" "$(history|tail -n1|sed -e '\''s/^\\s*[0-9]\\+\\s*\'')"'
alias capollo='ssh Apollo-devour.vip.ebay.com'
alias cares='ssh ares-devour.vip.ebay.com'
alias csq1=' cd ~/etl_home '
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
wenbluo@phxbastion100:~$ 

```

3. **unalias ccfg** 可以删除 别名；但是系统重启之后仍旧还有ccfg ,可以从bashrc根本删除
4. **alias** 可以查看有那些别名
5. **env** 可以查看所有的系统变量

H2 .Profile

1. 环境变量\$HOME \$PATH \$PS1\$PS2 \$CPATH

以bash (bourn shell 为例子): 修改.bashrc 文件 修改命令行提示符 (cli command line : Ps1)

```

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
alias ccfg='cd /home/wenbluo/etl_home/cfg'
alias csq1='cd /home/wenbluo/etl_home/sql_test'
alias cb1n='cd /home/wenbluo/etl_home/bin'
alias clog='cd /home/wenbluo/etl_home/log/secondary/dw_clsf1'
alias ctbl='cd /home/wenbluo/etl_home/tmp/secondary/dw_clsf1'
alias cjob='cd /dw/etl/home/prod/land/dw_clsf1/job_runner'
alias cares='ssh ares-devour.vip.ebay.com'
alias capollo='ssh Apollo-devour.vip.ebay.com'
PS1="wenbluo "
# User specific aliases and functions

```

```
[wenbluo@phxdpeet1019 ~] $ . .bashrc  
wenbluo hellp
```

此时提示符 变成wenbluo

H5 FUN:

这个怎么设置？开机登陆



2. 开机读取文件: .profile .bash_profile .bashrc
3. 当没有该文件时候，可以自己vi 创建属于自己的文件
- 4.

H4 拓展

- 1.

what's difference between **.profile ,.bash_profile,.bashrc**

- .bash_profile or .profile is read by login shell , along with .bashrc
 - >登录shell 以及.bashrc 读取.bash_profile , .profile
 - >子shell 只读取.bashrc
 - >

`.bash_profile` and `.`.bashrc` are specific to `bash`,
whereas `.`.profile` is read by many shells in the
absence of their own shell-specific config files.`

经过测试：文件夹下文件都有的时候，系统默认（本机是bash）,先读**.bash_profile**

```
[wenbluo@lvsdpeet1001 ~]$ ls -al
total 88
drwx----- 3 wenbluo wenbluo 4096 Jun 12 21:57 .
drwxr-xr-x 560 root root 49152 Jun 12 20:49 ..
-rw----- 1 wenbluo wenbluo 162 Jun 12 21:57 .bash_history
-rw----- 1 wenbluo wenbluo 18 Jun 12 20:49 .bash_logout
-rw----- 1 wenbluo wenbluo 176 Jun 12 20:49 .bash_profile
-rw----- 1 wenbluo wenbluo 1422 Jun 12 20:56 .bashrc
-rw----- 1 wenbluo wenbluo 500 Jun 12 20:49 .emacs
-rw----- 1 wenbluo wenbluo 121 Jun 12 20:49 .kshrc
drwx----- 2 wenbluo wenbluo 4096 Jun 12 21:16 .ssh
-rw----- 1 wenbluo wenbluo 675 Jun 12 20:56 viminfo
-rw----- 1 wenbluo wenbluo 658 Jun 12 20:49 .zshrc
[wenbluo@lvsdpeet1001 ~]$ cat .bash_logout
# ~/.bash_logout

rw----- 1 wenbluo wenbluo 121 Mar 21 00:32 .kshrc
rw----- 1 wenbluo wenbluo 18 Mar 21 00:32 .bash_logout
rw----- 1 wenbluo wenbluo 218 Apr 11 19:44 .bash_profile
rw----- 1 wenbluo wenbluo 23946 Apr 11 19:42 .bash_history
rw----- 1 wenbluo wenbluo 531 Apr 11 02:47 .bashrc
rw----- 1 wenbluo wenbluo 658 Mar 21 00:32 .zsnrc
rw----- 1 wenbluo wenbluo 90 Apr 11 18:37 .lesshist
rw-r--r-- 1 wenbluo wenbluo 0 Apr 8 00:00 date.txt
rw-r--r-- 1 wenbluo wenbluo 17460 Apr 8 23:59 world.txt
rw-rw-r-- 1 wenbluo wenbluo 0 Apr 2 23:50 .bashrx
rw-rw-r-- 1 wenbluo wenbluo 1676 Apr 11 02:47 nfs0000000009ad910b
rw-rw-r-- 1 wenbluo wenbluo 1676 Apr 11 19:42 .profile
rw-rw-r-- 1 wenbluo wenbluo 3144 Apr 9 18:58 2.txt

# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin

export PATH

alias ccfg='cd $HOME/etl_homedbc'
alias cf='cd /home/wenbluo/etl_homecfg'
[bash_profile (END)]
```

-->拓展：

- 在.bashrc 中设置了命名：

```
alias ccfg='cd /home/wenbluo/etl_homecfg'
```

在.bash_profile中设置了命名：

```
alias ccfg='cd $HOME/etl_homedbc'
```

我们会发现开启shell 时候，ccfg 是 /dbc

--> **bash_profile 包含 .bashrc**

-->说明每次开启shell ,系统都会读取.bashrc 以及 .bash_profile

-->如果先写ccfg 再调用.bashrc文件会怎样？

```

# .bash_profile
alias ccfg='cd $HOME/etl_home/dbc'
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

[wenbluo@phxdpeet1019 ~]$ alias
alias capollo='ssh Apollo-devour.vip.ebay.com'
alias cares='ssh ares-devour.vip.ebay.com'
alias cbin='cd /home/wenbluo/etl_home/bin'
alias ccfg='cd /home/wenbluo/etl_home/cfg'
alias cf='cd /home/wenbluo/etl_home/cfg'

```

通过测试可以看到：

ccfg 最后被.bashrc的alias 覆盖

```

export SCREENDIR=$HOME/.screen

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin directories
PATH="$HOME/bin:$HOME/.local/bin:$PATH"

> [.profile (END)]

```

--->拓展：

How to check which shell am i using

1. bash ,ksh ,csh ,tsh ...

ALL shell scripts (Bourne, BASH, Korn, Posix sh)

- bash is standard linux shell

基本已经取代sh (bourne shell)

```

[wenbluo@phxdpeet1019 bin]$ pwd
/bin
[wenbluo@phxdpeet1019 bin]$ ll sh
lrwxrwxrwx 1 root root 4 Mar 28 2018 sh ->
bash

```

- bourne is standard unix shell
- korn add numerous features to sh

广泛应用于所有*nix系统中，只要能访问命令行，就能使用

ksh

```
[wenbluo@phxdpeetl019 bin]$ ls -l | grep -i sh
-rwxr-xr-x 1 root root 906568 Feb 15 2017 bash
-rw-rxr-xr-x 1 root root 15952 Sep 21 2017 cgsnapshot
lrwxrwxrwx 1 root root 4 Mar 28 2018 csh -> tcsh
-rwxr-xr-x 1 root root 106216 Sep 21 2012 dash
lrwxrwxrwx 1 root root 21 Nov 13 2011 ksh -> /etc/alternatives/ksh
-rwxr-xr-x 1 root root 1357672 Apr 11 2012 ksh93
lrwxrwxrwx 1 root root 4 Mar 28 2018 sh -> bash
-rwxr-xr-x 1 root root 384488 Nov 24 2016 tcsh
-rwxr-xr-x 1 root root 707280 Mar 11 2016 zsh
```

there is some difference between ksh and bash/sh

Korn shell 集合了C shell & Bourne Shell 的优点

2. echo \$0 or echo \$SHELL

```
[wenbluo@phxdpeetl019 ~]$ echo $0
-bash
[wenbluo@phxdpeetl019 ~]$ echo $SHELL
/bin/bash
```

也可以通过 cat /etc/shells : 查看支持的shell 版本

```
[root@lvsdpeetl001 shell_book_practice]$ cat /etc/shells
/bin/sh
/bin/bash
/sbin/nologin
/bin/tcsh
/bin/csh
/bin/ksh
/bin/zsh
/bin/dash
```

H2 文件操作符号

H3 Test 指令

-f file	普通文件
-e file	文件存在
-x file	可执行文件
-s file	不是空文件
-r file	可读文件
-d file	是一个目录

```
they are different
[wenbluo@lvsdpeetl001 test]$ if test $num == $num2; then echo 'they are same'; else echo "they are different"; fi
they are different
[wenbluo@lvsdpeetl001 test]$ if test -e xv_test.sh ; then echo "the file $_ exists" ;else echo "the file $_ vanish ";fi
the file xv_test.sh exists
[wenbluo@lvsdpeetl001 test]$ _
```

H3 字符串操作符

-z	为空则为真
-n	不为空则为真 : not null
str1=str2	字符串相同
str1<str2	字符串str1字段顺序排在str2之前 (ascii排序)

ps:作比较的时候，< 需要转义 "\<"

```
-bash
[wenbluo@lvsdpeetl001 test]$ [ str1 \> str2 ]; echo $?
1
[wenbluo@lvsdpeetl001 test]$ test str1 \> str2 ; echo $?
1
[wenbluo@lvsdpeetl001 test]$ [[ str1 > str2 ]] ; echo $?
1
[wenbluo@lvsdpeetl001 test]$
```

[[[]]:这个高阶版本，仅在**bash****korn shell** 使用。[] 任何**shell** 都可以使用；

H3 算术操作符

-eq	相等
-ne	不相等
-lt、 -le	小于\小于等于
-gt、 -ge	大于\大于等于

H3 逻辑操作符/布尔操作符

&&	[[\$num -ge 90 && \$num -le 100]] : [90,100] :逻辑与
	[[\$num -ge 90 \$num -le 60]] : >=90 or <=60 :逻辑或
!	if [! -d /etc/home]逻辑非

H2 Configuration

文本编辑器+脚本解释器(命令解析器)

Linux 的 Shell 种类众多，常见的有：

- Bourne Shell (/usr/bin/sh或/bin/sh)
- Bourne Again Shell (/bin/bash)
- C Shell (/usr/bin/csh)
- K Shell (/usr/bin/ksh)
- Shell for Root (/sbin/sh)

- Bsh:由贝尔实验室编写，Bsh是产生较早的UNIX Shell程序，实现了最基本的命令解释器的功能，同时也可作为脚本编程语言
- Csh:是因使用C语言的语法风格而得名，在用户的命令行交互界面上进行了很多改进，并增加了历史，别名，文件名替换，作业掏等功能，相比Bsh，C操作
- Ksh:在Bsh和Csh之后出现的，结合了两者的功能优势，兼具Bsh的语法和Csh的交互特性。
- Bash:从名称可以看出是Bsh的升级版本，是著名的开源软件项目，目前大多数的Linux版本（包括Red Hat公司的Linux系统）都使用Bash作为默认时，实际运行的是Bash程序
- Zsh:更多地基于交互式操作考虑进行设计的Shell程序，集成了Bash,Ksh等多种Shell程序的优点

- 1. 直接在电脑上打开那个文件夹，然后在文件夹空白处右键选择Git Bash here

2.

```
wenbluo@L-SHC-16505239 MINGW64 ~
$ cd ~/Desktop/wbluo/git
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/wbluo/git
$ cd wenluo/Desktop/wbluo/git
```

3. 文件夹处理：

- ls -l :查看目录下的文件： list files/subfolders
- ls (list simple) 简单输出有哪些文件和目录
- ll : 详细文件信息 (d, -, l , RWX , time owner)等等
- 快捷方式： ll

```
wenbluo@L-SHC-16505239 MINGW64 ~/desktop/wbluo/111
$ ls -l
total 8
drwxr-xr-x 1 wenbluo 1049089 0 Mar 19 09:35 123/
-rw-r--r-- 1 wenbluo 1049089 389 Mar 18 14:41 test_1.sh
-rw-r--r-- 1 wenbluo 1049089 324 Mar 18 14:15 test_1.txt
-rw-r--r-- 1 wenbluo 1049089 400 Mar 18 14:56 test_2.sh
-rw-r--r-- 1 wenbluo 1049089 165 Mar 18 15:05 test_3.sh
-rw-r--r-- 1 wenbluo 1049089 101 Mar 18 17:17 test_4.sh
-rw-r--r-- 1 wenbluo 1049089 194 Mar 18 17:42 test_5.sh
-rw-r--r-- 1 wenbluo 1049089 22 Mar 20 19:09 test_6.sh
-rw-r--r-- 1 wenbluo 1049089 43 Mar 19 09:48 test_7.sh
```

wenbluo :表示文件所有者

1049089 : 表示用户所在的组

389:表示文件大小

H4 拓展：

图片中的total 为 8 是什么意思呢？

- 该目录占用block块数（sql server 是以8k），

blocksize

```
[wenbluo@phxdpeetl019 sql_test]$ ll
total 36
-rwxrwxr-x 1 wenbluo wenbluo 19909 Mar 27 22:58 clsfid_daily_mkt_sc_bak.sql
-rw-rw-r-- 1 wenbluo wenbluo 0 Mar 28 19:00 crontab_chewu
drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 1 01:05 huangxiaoxia
drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 2 21:56 test
drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 1 01:19 wbluo
drwxrwxr-x 2 wenbluo wenbluo 4096 Apr 3 01:27 Wbluo
[wenbluo@phxdpeetl019 sql_test]$ getconf PAGESIZE
4096
[wenbluo@phxdpeetl019 sql_test]$
```

通过指令 getconf pagesize : 得到一个block size 为4k 4096byte

$(19909+4096*4)/4096=9$ 个blocks，所以占据空间为36 kb

- 退出文件夹： cd .. -->返回上一层文件夹（类似html5,返回上一层）

1. cd: change directory

2. cd ~ : 返回到用户 :\$(whoami) ##wenbluo 注意是括号并不是花括号

cd :返回用户目录(主目录 \$HOME) : /home/wenbluo 与 cd ~等价

```
[wenbluo@phxdpeet1019 cfg]$ whoami  
wenbluo  
[wenbluo@phxdpeet1019 cfg]$ ${whoami}  
[wenbluo@phxdpeet1019 cfg]$ echo $whoami  
  
[wenbluo@phxdpeet1019 cfg]$ echo ${whoami}  
  
[wenbluo@phxdpeet1019 cfg]$ whoami  
wenbluo  
[wenbluo@phxdpeet1019 cfg]$
```

3. cd .. :返回上一级

cd desktop/wbluo/111 :直接跳转到该文件夹下

ps: 如果通过指令 cd wbluo 想返回上一级，报错；只能 cd ..

总结：

1. ~/.. : 先执行返回用户目录，再返回根目录

```
[wenbluo@phxdpeet1019 etl_home]$ pwd  
/home/wenbluo/etl_home  
[wenbluo@phxdpeet1019 etl_home]$ cd ~  
[wenbluo@phxdpeet1019 ~]$ pwd  
/home/wenbluo  
[wenbluo@phxdpeet1019 ~]$ cd ~/..  
[wenbluo@phxdpeet1019 home]$ pwd  
/home  
[wenbluo@phxdpeet1019 home]$
```

4. pwd: 返回完整路径

```
wenbluo@L-SHC-16505239 MINGW64 ~/desktop/wbluo  
$ cd ~  
  
wenbluo@L-SHC-16505239 MINGW64 ~  
$ pwd  
/c/Users/wenbluo  
  
wenbluo@L-SHC-16505239 MINGW64 ~  
$ |
```

pwd(print working directory) 命令是用于显示当前的目录。

-->R语言中的 getwd

- 新建文件夹 mkdir 111

```

E73-8@E73-8-PC ~ (master)
$ cd D:                                指定磁盘
E73-8@E73-8-PC /d
$ cd www                                指定文件夹
E73-8@E73-8-PC /d/www
$ mkdir testgit                            创建文件夹
E73-8@E73-8-PC /d/www
$ cd testgit
E73-8@E73-8-PC /d/www/testgit
$ pwd
/d/www/testgit
E73-8@E73-8-PC /d/www/testgit
$
```

-->拓展删除文件夹

rmdir 123 124 125 :删除多个文件夹

```

mkdir -p dir_1/dir_2
[wenbluo@phxdpeet1019 test]$ cp ./while_test
./dir_1/dir_2
[wenbluo@phxdpeet1019 test]$ rm dir_1
rm: cannot remove `dir_1': Is a directory
###因为dir_1 目录不为空

###方法一： rm -r dir_1 :则会删除整个目录
###方法二：
rmdir dir_1
rmdir: failed to remove `dir_1': Directory not empty

###既然还没有该功能，看来只能rm -r 了；
rmdir -r dir_1
rmdir: invalid option -- 'r'
Try `rmdir --help' for more information.
```

-->当文件夹不为空的时候，可以采用 **rm -r** (recursion)

注意文件路径要正确！！，而且不要轻易使用该指令

```

[wenbluo@phxdpeet1019 etl_home]$ rmdir test_dir
rmdir: failed to remove `test_dir': Directory not empty
[wenbluo@phxdpeet1019 etl_home]$ rm -r test_dir
[wenbluo@phxdpeet1019 etl_home]$ ll
total 16
-rw-rw-r-- 1 wenbluo wenbluo 8986 Mar 25 03:12 abinitsc.setup
```

-->创建目录A下，再创建一个目录B

mkdir -p dat/secondary dat/extract dat/primary dat/td2 dat/td1

- **touch** 创建文件：

1. **touch test_8.sh / test_9.xlsx / test_10.r**

2. 还可以通过 **echo 赋值方式** :

```

-rw-rw-r-- 1 wenbluo wenbluo    22 Mar 25 03:23 test.sql
[wenbluo@phxdpeet1019 sql_test]$ echo 20 > break_point.seq
[wenbluo@phxdpeet1019 sql_test]$ ll
total 20
-rw-rw-r-- 1 wenbluo wenbluo     3 Mar 28 04:33 break_point.seq
-rwxrwxr-x 1 wenbluo wenbluo 19909 Mar 27 22:58 clsfd_daily_mkt_sc.sql
-rw-rw-r-- 1 wenbluo wenbluo    22 Mar 25 03:25 test.sql
[wenbluo@phxdpeet1019 sql_test]$
```

表示传参

当break_point.seq文件不存在时候，创建文件，并赋值；

```
[wenbluo@phxdpeet1019 ~]$ cd ..
[wenbluo@phxdpeet1019 ~]$ echo 'test' >connect_text
[wenbluo@phxdpeet1019 ~]$ echo 'test' >/~connect_text
[wenbluo@phxdpeet1019 ~]$ cd ~
[wenbluo@phxdpeet1019 ~]$ ls
connect_text  etl_home  test_Crontab.out  test.out
[wenbluo@phxdpeet1019 ~]$ cat connect_text
test
[wenbluo@phxdpeet1019 ~]$ ls
```

- wc:word count

```
wenbluo@L-SHC-16505239 MINGW64 ~/desktop/wbluo/111
$ wc test_8.txt
2 23 106 test_8.txt

wenbluo@L-SHC-16505239 MINGW64 ~/desktop/wbluo/111
$ |
```

1. 2: 表示文本共两行数据 -l (lines)
2. 23 : 表示文本有23个单词 -w (words)
3. 106: 表示字节bytes -c
4. test_8.txt ,表示文件名称
5. wc --help 查看含义
6. "#": 表示注释

H3 vim文本编辑器：

- [vim](#)

- vi /vim 有三种模式

1. 命令模式 输入： 进入命令模式

- 可以gg (返回首行)
- shift+g (返回末行)
- u : 返回上一步
- ctrl+r :撤销

2. 输入模式

- 按 i 字母 进入编辑模式

3. 末行模式

- 按 : 字符
- 骚操作

1. 查找： / work

--> 如何按 n 可以下翻找下一个work，按 N 查找上一个 work

2. 替换：

- :s /u/you/g : 替换鼠标所在行 所有的u-->you ##其中/u slash block 一定要
- :%s /u/you : 替换整篇的u 为 you

-->

如果想控制替换，可以添加一个指令/c

:%s /u/you/c , 则会强制每个替换需要用户进行确认

- 恢复文件

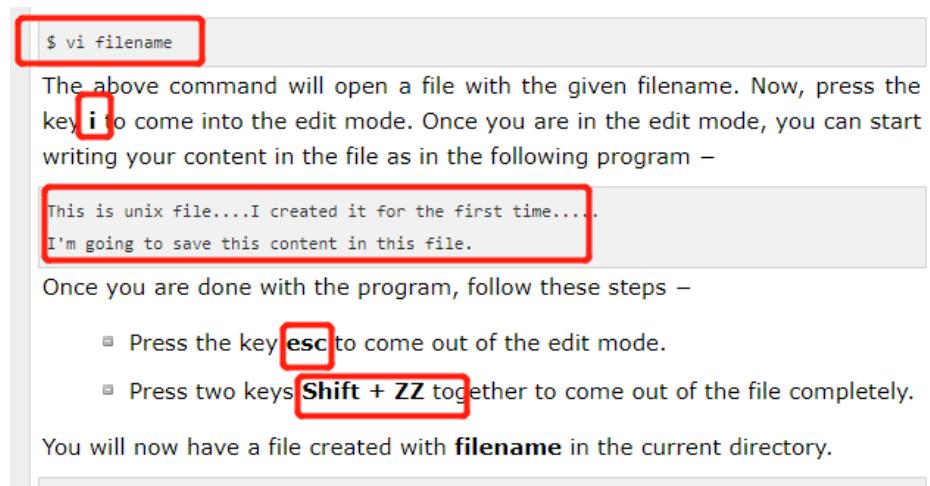
vi在编辑某一个文件时，会生成一个临时文件，这个文件以.**开头并以.swp结尾**。正常退出该文件自动删除，如果意外退出例如忽然断电，该文件不会删除，我们在下次编辑(vi file ,并不是vi .swp)时可以选择一下命令处理：

O只读打开，不改变文件内容 E继续编辑文件，不恢复.swp文件保存的内容
R将恢复上次编辑以后未保存文件内容 Q退出vi D删除.swp文件 或者使用**vi -r** 文件名来恢复未保存的内容

•

1. :quit /:exit -->**规范退出vim 编辑器**

2. **vi 创建文件并且编辑文件**



1. vi test_8.txt

2. i: 表示insert

3. esc :跳出插入

4. Shift+ZZ: 退出 文本操作 :wq (**windows .quit**) 保存文件并退出

有时候也会通过 q! 退出; -->**并不会保存文件**

5. 通过： 可以查找文件 :/ (模糊查找，区分大小写)

6. 快捷键:

- dd 删除整行数据
- ctrl+f : front 翻页（下一页）

- **ctrl+b:back** 翻页（上一页）

- **shift+g** :快速到达末行

-->**如果向快速定位都某一行？**

17 + shift +g

其实就是 17G

- **gg**:到达首行

- **u** 撤销上一步的操作

- **Ctrl+r** 恢复上一步被撤销的操作

- 全部复制: 按`esc`键后, 先按`gg`, 然后`ggyG` (Yank:拉取 Global)

- 全选高亮显示: 按`esc`键后, 先按`gg`, 然后`ggVG`或者`ggVG`

- **^**:当前行首行

- **\$**:当前行末尾

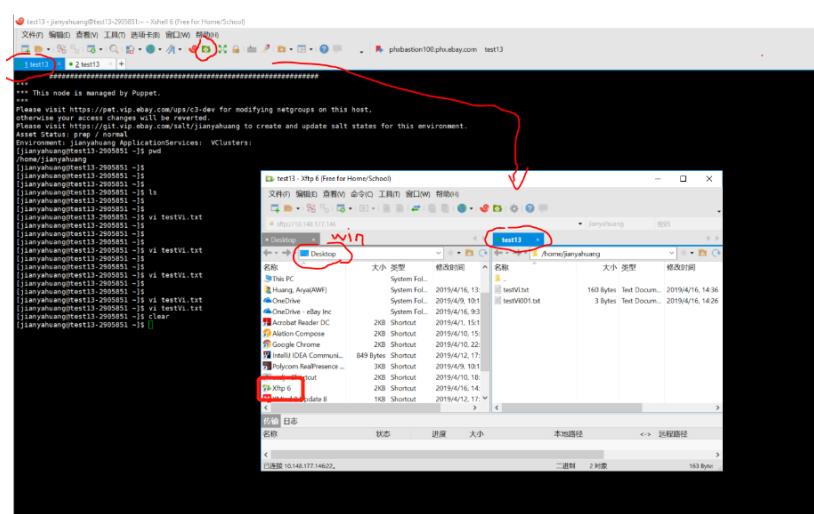
- **r**:表示替换`replace`

- **y**:表示复制(yank) : **注意是复制到vi的缓冲区**

-->要想copy 到本机,只能`cat` 文件, 然后复制

- **p**:表示粘贴(paste) -->这些快捷键只能使用与`vim`, 没法复制到记事本上;

-->可以采用`xftp6`将unix 文件上传到windows



-->如何解决indent 问题, 因为paste 到remote terminal 结果文件全部错乱

vi indent

1. = :表示indent

- `gg=G` :所有行缩进

- `=g`:当前行下所有行都缩进

- `==`: 当前行缩进

2. 为了避免paste 时候出现乱行

`:set paste`

This will turn off the auto-indent, line-wrap, etc. features
that are messing up your paste.

3. :set number (可以查看行数)

```
1 #! /bin/bash
2 if [ $# -le 0 ];then
3 echo "usage bash $0 arg1 arg2 .."
4 exit 0
5 fi
6
7
8 while [[ $# != 0 ]];do
9 echo "第一个参数 $1, 参数个数$#"
10 shift
11 done
12
13
14
~
```

wc -l word.txt test_8.txt : 统计两个文件行数

- mv 移动文件夹/重命名

mv a.txt /b/c.txt : 将a文本移到b文件夹下，并重命名为c文本

mv a.txt c.txt : 将a 文件重命名为 c文件

---拓展 rename :

- unix系统有两种rename编程语言 一个是C，一个是perl
- 通过man rename command 查看type (rename --help) rename -V
- c 语言下：

[c_rename_tutorial](#)

[per_rename_tutorial](#)

rename old-c new-c filename

```
[wenbluo@phxdpeet1019 test]$ ll
total 0
-rw-rw-r-- 1 wenbluo wenbluo 56 Apr  2 21:55 wbluo_1.txt
-rw-rw-r-- 1 wenbluo wenbluo 48 Apr  2 01:32 wbluo_2.txt
-rw-rw-r-- 1 wenbluo wenbluo 48 Apr  2 01:36 wbluo_3.txt
-rw-rw-r-- 1 wenbluo wenbluo 14 Apr  2 01:37 wbluo_4.txt
[wenbluo@phxdpeet1019 test]$ rename wbluo f *
##将所有wbluo 都特换成 f
[wenbluo@phxdpeet1019 test]$ ll
total 0
-rw-rw-r-- 1 wenbluo wenbluo 56 Apr  2 21:55 f_1.txt
-rw-rw-r-- 1 wenbluo wenbluo 48 Apr  2 01:32 f_2.txt
-rw-rw-r-- 1 wenbluo wenbluo 48 Apr  2 01:36 f_3.txt
-rw-rw-r-- 1 wenbluo wenbluo 14 Apr  2 01:37 f_4.txt
```

困惑： rename是否支持正则表达式：

```
[wenbluo@phxdpeet1019 test]$ rename 'f|w' wbluo *
[wenbluo@phxdpeet1019 test]$ ll
total 0
-rw-rw-r-- 1 wenbluo wenbluo 14 Apr  2 01:37 F_4.txt
-rw-rw-r-- 1 wenbluo wenbluo 48 Apr  2 01:36 w_3.txt
-rw-rw-r-- 1 wenbluo wenbluo 56 Apr  2 21:55 wbluo_1.txt
-rw-rw-r-- 1 wenbluo wenbluo 48 Apr  2 01:32 wbluo_2.txt
### 结论是不支持!
```

---拓展：如何大小写替换：

sed (stream editor)

基本格式： sed pattern filename filename2

- 仅仅是在输出替换，并没有底层更改名称
- 大写转小写: echo "ABCDS" | sed 's/[A-Z]/\l/g'
- 小写转大写: echo "abcds" | sed 's/[a-z]/\u/g'
 - 取出固定位置数值

```
[wenbluo@phxdpeet1019 test]$ date '+%Y%m%d' | sed
's/(\....)\(..)\(..)/\1 \2 \3/g'
2019 04 16
###把模式放入括号内会将其匹配的内容保存到寄存器中，在这里寄
存器是1, 2, 3，可以采用\1 \2 \3来引用
###类似hive regexp_extract(columnname, '(\d+)
(12)',1)

$ echo '2019-01-02' | sed 's/(\....)\(..)\(..)/
(..)/\1 \2 \3/g' | read YEAR_EXP MONTH_EXP
DAY_EXP
$ echo $YEAR_EXP
2019
```

```
sed 's/[ir]/11/g' wbluo_1.txt
echo '11 am 1lobe1lt'
[wenbluo@phxdpeet1019 test]$ cat wbluo_1.txt |
sed 's/[ir]/11/g'
echo '11 am 1lobe1lt'
[wenbluo@phxdpeet1019 test]$ cat wbluo_1.txt
echo 'i am robert'
```

```
sed 's/[ir]/11/g' wbluo_1.txt  wbluo_2.txt
echo '11 am 1lobe1lt'
echo '11 am Robe1lt'
```

•

tr (translate:switch)

- cat t.txt | tr 'abc' 'xyz'
 - cat file| tr [a-z] [A-Z] ###小写转换为大写
 - cat file |tr -d 'snail' ##删掉所有**snail** 字符串
 - cat file | tr -s ":" '\n' ###冒号替换成换行符
-

- 查看文件内容: \$ **cat** 文件名.文件类型 -->catenate :表示连接文件并输出到前端;
- **cat** 一次输出多个文件

```
case_test.txt child.txt explicit log.txt parent.txt s
[wenbluo@lvsdpeetl001 test]$ cat jobs_2 jobs_1;
#!/bin/bash
echo 'hello robert ,nice to see U'
read -p 'input your number :' -t 5 num
echo $num
export ${num}
bash
echo hello world ,hello robert;
echo i am glad to see u;
echo todday we will learn how to run shell command.
[wenbluo@lvsdpeetl001 test]$ cat jobs_2
#!/bin/bash
echo 'hello robert ,nice to see U'
read -p 'input your number :' -t 5 num
echo $num
export ${num}
bash
[wenbluo@lvsdpeetl001 test]$ cat jobs_1
echo hello world ,hello robert;
echo i am glad to see u;
echo todday we will learn how to run shell command.
[wenbluo@lvsdpeetl001 test]$
```

拓展: [文件查看指令](#)

1. less test /// 退出形式按: q

特点:

- 不像cat test , 全部输出

- 而是像vi一样,可以翻页 (**page up\page dn**),便与查看, 但又不会修改文件
- 也可以模糊匹配: /clsfd_site_id

2. head

head -20 dw_clsfid.step2_clsfid_daily_sc_mkt_mtcr.sql 返回前20行数据

3. tail

tail -20 dw_clsfid.step2_clsfid_daily_sc_mkt_mtcr.sql 返回后20行数据

4. more

显示所有信息, 但是可以通过输入enter键, 翻页

ps -aux | more 优于 **ps -aux**

--> 也可以通过 **d(down)/ b (back)**: 快捷键 快速翻页

5. top 5 or tail 5:

- 查看文件前几行:

```
head -n 10 file.txt #head 默认值为10
```

--> head file.txt ## head -2 file.txt 不写 -n : number也行;

```
tail -n 10 file.txt
```

- 查看目录前几行:

```
ll -l | head
```

6. column : 格式化输出

```
[wenbluo@lvsdpeet1005 test]$ cat dd.txt
aaaa  bbbb
cc  dd
号号号  好好号
[wenbluo@lvsdpeet1005 test]$ cat dd.txt | column -t
aaaa  bbbb
cc  dd
号号号  好好号
[wenbluo@lvsdpeet1005 test]$ ...
```

- 其他参数 -c : controls table's column width

```
twenty
[wenbluo@lvsdpeet1005 test]$ column -c 60 number
one          eight      fifteen
two          nine       sixteen    宽度为60
three        ten        seventeen
four         eleven     eighteen
five         twelve    nineteen
six          thirteen   twenty
seven        fourteen
[wenbluo@lvsdpeet1005 test]$ column -c 40 number
one          eleven
two          twelve
three        thirteen
four         fourteen
five         fifteen
six          sixteen
seven        seventeen
eight        eighteen
nine         nineteen
ten          twenty
[wenbluo@lvsdpeet1005 test]$ ...
```

•

总结: less>>head==tail>>more >>cat

- 删除目录下的文件; \$ rm 文件名.文件类型 remove file

- 运行shell文件: \$ sh 文件名.sh

```
wenbluo@lvsdpeet1005 MINGW64 ~/desktop/wenbluo/111
$ sh test_1.sh
a + b : 30
a - b : -10
a * b : 200
b / a : 2
b % a : 0
a == b
```

- 改变权限的命令:

chmod 755 abc.sh :表示rwxr-xr-x

chmod a+r abc.sh :表示所有用户添加读的权限

-->也可以 chmod +r abc.sh

-->撤销读权限 chmod -r abc.sh

只能分批次赋予用户权限 (U,G,O)

```
[wenbluo@phxdpeet1019 test_dir]$ chmod u+x o+x test.sql
chmod: cannot access `o+x': No such file or directory
[wenbluo@phxdpeet1019 test_dir]$ chmod u+x test.sql
[wenbluo@phxdpeet1019 test_dir]$ ll
total 0
-rwxrw-r-- 1 wenbluo wenbluo 22 Mar 27 21:33 test.sql
[wenbluo@phxdpeet1019 test_dir]$ chmod o+x test.sql
[wenbluo@phxdpeet1019 test_dir]$ ll
total 0
-rwxr-wr-x 1 wenbluo wenbluo 22 Mar 27 21:33 test.sql
[wenbluo@phxdpeet1019 test_dir]$
```

chmod 711 abc :更改目录用户权限

给所有人添加 执行权限:

```
[wenbluo@phxdpeet1019 test_dir]$ chmod +x test.sql
[wenbluo@phxdpeet1019 test_dir]$ ll
total 0
-rwxrwxr-x 1 wenbluo wenbluo 22 Mar 27 21:33 test.sql
[wenbluo@phxdpeet1019 test_dir]$ chmod -x test.sql
[wenbluo@phxdpeet1019 test_dir]$ ll
total 0
-rw-rw-r-- 1 wenbluo wenbluo 22 Mar 27 21:33 test.sql
[wenbluo@phxdpeet1019 test_dir]$
```

1. 通配符 (metacharacter)

*:任意字符 ?:表示单字符

```
莹控 wenbluo@L-SHC-16505239 MINGW64 ~/desktop/wbluo/111
$ ls test*.sh
test_1.sh test_2.sh test_3.sh test_4.sh test_5.sh test_6.sh test_7.sh
wenbluo@L-SHC-16505239 MINGW64 ~/desktop/wbluo/111
$ |
```

ll -tr *dw_clsfld.arg_start_endtime_wenbluo*

等价于:

ll -tr | grep *dw_clsfld.arg_start_endtime_wenbluo*

2. Pipes and Filters

符号是 : |

实例:

```
1 [chengmo@centos5 shell]$ cat test.sh | grep -n 'echo'
2 5: echo "very good!";
3 7: echo "good!";
4 9: echo "pass!";
5 11: echo "no pass!";
6 #读出test.sh文件内容，通过管道转发给grep 作为输入内容
7
```

3. cp: copy 复制文件

```

cp: cannot stat '/test_dir': No such file or directory
[wenbluo@phxdpeet1019 etl_home]$ cp test_dir ~/.../xiaoxyang/etl_home/
cp: omitting directory 'test_dir'
[wenbluo@phxdpeet1019 etl_home]$ cp -r test_dir ~/.../xiaoxyang/etl_home/
cp: cannot create directory '/home/wenbluo/.../xiaoxyang/etl_home/test_dir': Permission denied
[wenbluo@phxdpeet1019 etl_home]$

```

因为 **test_dir** 目录下有 **test.sql** 文件，所以仅仅是 **copy** 目录到 **xiaoxyang** 目录下不行；

那可以调用 **-r** 指令，将 **test_dir** 下所有文件都复制到 **xiaoxyang** 路径下：

```

- cp ~/etl_home/sql/dw_clsfad.step2_clsfad_daily_sc_mkt_mttrc.sql
~/etl_home/sql_test/wuxia

```

会自动在 **wuxia** 目录下生成同样名称的文件；

```
dw_clsfad.step2_clsfad_daily_sc_mkt_mttrc.sql
```

```

- <span style='background-color:lightblue'>cp 会完全复制文件属性
</span> :X R W ;

```

- **cp** 文件夹：

```

```shell
mkdir cp_test
[wenbluo@phxdpeet1019 sql_test]$ cp -r wbluo cp_test
[wenbluo@phxdpeet1019 sql_test]$ cd cp_test
```

```

```

• cp: cube_daily/clsfad_ad_cube_daily/': No such file or directory
Command failed with exit code = 1
every returned error message is shown: null
(wenbluo@phxdpeet1019 ~)$ dtr -v /views://apollo-rno/sys/eds/working/clsfad/clsfad_working/clsfad/clsfad_ad_cube_daily_views://apollo-rno/user/hive/warehouse/wenbluo/ad_cube/clsfad_ad_cube_daily/clsfad_ad_cube_daily/clsfad_site_id=3001/clsfad_nm_dt=2018-09-01/part_00000_feactk85-5d05-4072-bfc1-16eenc0df143.c000
File exists
(wenbluo@phxdpeet1019 ~)$ dtr -v /views://apollo-rno/user/hive/warehouse/wenbluo/ad_cube/clsfad_ad_cube_daily/clsfad_ad_cube_daily/clsfad_nm_dt=2018-09-01/part_00001_feactk85-5d05-4072-bfc1-16eenc0df143.c000
File exists
(wenbluo@phxdpeet1019 ~)$ dtr -v /views://apollo-rno/user/hive/warehouse/wenbluo/ad_cube/clsfad_ad_cube_daily/clsfad_ad_cube_daily/clsfad_nm_dt=2018-09-01/part_00002_feactk85-5d05-4072-bfc1-16eenc0df143.c000
File exists
(wenbluo@phxdpeet1019 ~)$

```

当文件已经存在时候，并不在 **cp**，并不会 **overwrite**，或者跟 **window** 一样，生成 **back or override**

- 拓展远程 **scp (secure copy)**

我在工作中经常要将一些文件传输到另外一个服务器上，而且都是 **Linux** 的命令行环境，那么对于我来讲 **scp** 就是最直接有效的方法了，其他诸如 **FTP**、**SMB** 以及 **Winscp** 这些有界面的文件传输工具到反而有些多余了。

```

scp ./while_test
wenbluo@phxdpeet1019.phx.ebay.com:/home/wenbluo/etl_home/sql_test/
test/while_test
wenbluo@phxdpeet1019.phx.ebay.com's password:

```

```

scp -r .ssh
wenbluo@lvspdeet1001.lvs.ebay.com:/home/wenbluo/
    ### 将 .ssh 目录赋值到从 phx -->lvs 目录下
    ### 其中 :/home/wenbluo/ 中的 : 不可以少!

```

- SFTP**
- wget**

Linux wget 是一个下载文件的工具！--> 类似 **windows** 迅雷

4. 覆盖文件内容！————》

```
[wenbluo@phxdpeet1019 wbluo]$ echo "hello world" > test  
[wenbluo@phxdpeet1019 wbluo]$ ll  
total 52  
-rw-rw-r-- 1 wenbluo wenbluo 19908 Mar 28 04:37 dw_clsfld.step2_clsfld_daily_sc_mkt_mt  
-rw-rw-r-- 1 wenbluo wenbluo 25973 Apr 1 01:09 dw_clsfld.step2_clsfld_weekly_sc_mkt_m  
-rw-rw-r-- 1 wenbluo wenbluo 12 Apr 1 01:21 test  
[wenbluo@phxdpeet1019 wbluo]$ cp crontab_chewu wbluo/test  
cp: cannot stat `crontab_chewu': No such file or directory  
[wenbluo@phxdpeet1019 wbluo]$ cp ./crontab_chewu wbluo/test  
cp: cannot create regular file `wbluo/test': No such file or directory  
[wenbluo@phxdpeet1019 wbluo]$ cp ./crontab_chewu test  
[wenbluo@phxdpeet1019 wbluo]$ ll  
total 52  
-rw-rw-r-- 1 wenbluo wenbluo 19908 Mar 28 04:37 dw_clsfld.step2_clsfld_daily_sc_mkt_mt  
-rw-rw-r-- 1 wenbluo wenbluo 25973 Apr 1 01:09 dw_clsfld.step2_clsfld_weekly_sc_mkt_m  
-rw-rw-r-- 1 wenbluo wenbluo 0 Apr 1 01:21 test  
[wenbluo@phxdpeet1019 wbluo]$ ll
```

5. sudo 指令

6. 通配符 : * and ?

```
[wenbluo@phxdpeet1019 bin]$ ll *table*handler*  
-rwxrwxrwx 1 chewu chewu 22947 Jan 20 2016 single_table_extract_handler.ksh  
-rwxrwxrwx 1 chewu chewu 23333 Mar 27 2016 single_table_extract_handler.ksh_bk  
-rwxrwxrwx 1 chewu chewu 6984 Dec 29 2014 single_table_load_handler.ksh  
-rwxrwxrwx 1 chewu chewu 9045 Nov 20 02:04 target_table_load_handler_bk.ksh  
-rwxrwxrwx 1 chewu chewu 9098 Mar 21 03:22 target_table_load_handler.ksh  
-rwxrwxrwx 1 chewu chewu 8907 Jun 21 2018 target_table_load_handler.ksh.passOnC3  
-rwxrwxrwx 1 chewu chewu 9057 Nov 20 01:24 target_table_load_handler_test.ksh
```

H5 QA

1. 如何排序

2.

7. 以文件形式，保存输出结果,输出重定向

符号 >

```
[wenbluo@phxdpeet1019 ~]$ cd etl_home  
[wenbluo@phxdpeet1019 etl_home]$ ls > ls.txt  
[wenbluo@phxdpeet1019 etl_home]$ ll  
total 20  
-rwxrwxr-x 1 wenbluo wenbluo 8986 Mar 25 03:12 abinitio.setup  
lrxwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:05 bin -> /home/chewu/etl_home/bin  
lrxwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:06 cfg -> /home/chewu/etl_home/cfg  
lrxwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:06 dat -> /home/chewu/etl_home/dat  
lrxwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:07 dbc -> /home/chewu/etl_homedbc  
lrxwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:07 dml -> /home/chewu/etl_home/dml  
lrxwxrwxrwx 1 wenbluo wenbluo 23 Mar 25 03:07 in -> /home/chewu/etl_home/in  
lrxwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:07 lib -> /home/chewu/etl_home/lib  
lrxwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:07 log -> /home/chewu/etl_home/log  
-rw-rw-r-- 1 wenbluo wenbluo 75 Mar 27 22:31 ls.txt  
lrxwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:07 out -> /home/chewu/etl_home/out  
drwxrwxr-x 2 wenbluo wenbluo 4096 Mar 25 03:26 sql  
lrxwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:08 tmp -> /home/chewu/etl_home/tmp  
lrxwxrwxrwx 1 wenbluo wenbluo 26 Mar 25 03:08 watch -> /home/chewu/etl_home/watch  
lrxwxrwxrwx 1 wenbluo wenbluo 24 Mar 25 03:08 xfr -> /home/chewu/etl_home/xfr
```

```
[wenbluo@phxdpeet1019 ~]$ ls  
etl_home test_Crontab.out test.out  
[wenbluo@phxdpeet1019 ~]$ crontab -l  
30 20 * * * /home/chewu/etl_home/bin/cront  
t  
31 20 * * * echo test > ~/test.out  
35 20 * * * /home/chewu/etl_home/bin/cron  
t > ~/test_Crontab.out 2>&1  
[wenbluo@phxdpeet1019 ~]$ ll
```

8. 拓展 : >>

在已有的文件下，新增内容！

H4 Grep

(global regular expression)

unix 常见命令大全

- 查找所有txt文件中，含有wenbluo字样的文件

```
[wenbluo@phxdpeetl019 test]$ grep 'wenbluo' *.txt
f_1.txt:wenbluo is very working hard
f_1.txt:wenbluo is very outgoing
f_1.txt:wenbluo is very sensitive
f_2.txt:who am i ? i am wenbluo ,you can call me robert
f_3.txt:hi robert ,wenbluo ,welcome to the shell world!
```

- 删除指令 grep -v : 只读不含有outgoing 的行信息，然后存放到临时文件夹，(invert)

并替换原始文件

```
[wenbluo@phxdpeetl019 test]$ cat f_1.txt
wenbluo is very working hard
wenbluo is very outgoing
wenbluo is very sensitive

[wenbluo@phxdpeetl019 test]$ grep -v 'outgoing' f_1.txt >/tmp/f_1.txt
[wenbluo@phxdpeetl019 test]$ mv /tmp/f_1.txt f_1.txt
[wenbluo@phxdpeetl019 test]$ cat f_1.txt
wenbluo is very working hard
wenbluo is very sensitive
```

```
ll -tr | grep wbluo
-rw-rw-r-- 1 wenbluo wenbluo 20191 Mar 31 19:41 clsf_daily_mkt_sc_wbluo.bt.dw_clsf.daily_mkt_sc_wbluo.sql.tmp
-rw-rw-r-- 1 wenbluo wenbluo 20193 Apr 1 00:10 clsf_daily_mkt_sc_wbluo.bt.dw_clsf.clsfd.daily_mkt_sc_wbluo.sql.tmp
-rw-rw-r-- 1 wenbluo wenbluo 26239 Apr 1 01:15 clsf_weekly_mkt_sc_wbluo.bt.dw_clsf.step2_clsf_weekly_sc_mkt_mtrc_wbluo.sql.tmp
-rw-rw-r-- 1 wenbluo wenbluo 21050 Apr 3 05:09 clsf_daily_adrev_sc_wbluo.bt.dw_clsf.clsfd.weekly_adrev_sc_wbluo.sql.tmp
-rw-rw-r-- 1 wenbluo wenbluo 0 Apr 4 00:06 clsf_daily_adrev_sc_wbluo.bteq_btclsf.clsfd.daily_adrev_sc_wbluo.complete
-rw-rw-r-- 1 wenbluo wenbluo 10586 Apr 4 00:06 clsf_daily_adrev_sc_wbluo.bt.dw_clsf.clsfd.daily_adrev_sc_wbluo.sql.tmp

ll *arg_start_endtime_wenbluo*
-rw-rw-r-- 1 wenbluo wenbluo 325 Apr 4 01:35 arg_start_endtime_wenbluo.bt.dw_clsf.arg_start_endtime_wenbluo.sql.tmp
-rw-rw-r-- 1 wenbluo wenbluo 0 Apr 4 01:35 arg_start_endtime_wenbluo.bteq_btclsf.arg_start_endtime_wenbluo.complete

ll -tr | grep wbluo *
arg_start_endtime_wenbluo.bt.dw_clsf.arg_start_endtime_wenbluo.sql.tmp:insert into p_chewu_t_wbluo_2019_04_04 values(2034/08/03,1994/03/30);
clsf_daily_adrev_sc_wbluo.bt.dw_clsf.clsfd.daily_adrev_sc_wbluo.sql.tmp:SET QUERY_BAND = 'SA=du_clsf:TBID=clsf.daily_adrev_sc_wbluo;SCRIPTNAME=dw_clsf_daily_mkt_sc_wbluo.bt.dw_clsf.clsfd.daily_mkt_sc_wbluo.sql.tmp:SET QUERY_BAND = 'SA=du_clsf:TBID=clsf.daily_mkt_sc_wbluo;SCRIPTNAME=dw_clsf_weekly_adrev_sc_wbluo.bt.dw_clsf.clsfd.weekly_adrev_sc_wbluo.sql.tmp:SET QUERY_BAND = 'SA=du_clsf:TBID=clsf_weekly_adrev_sc_wbluo;SCRIPTNAME=dw_clsf_weekly_mkt_sc_wbluo.bt.dw_clsf.step2_clsf_weekly_sc_mkt_mtrc_wbluo.sql.tmp:SET QUERY_BAND = 'SA=du_clsf:TBID=clsf_weekly_mkt_sc_wbluo'
```

ll -tr | grep wbluo * 实质就是 grep wbluo * 查看所有包含wbluo 的文件

```
[wenbluo@phxdpeetl019 test]$ grep wenbluo *
f_1.txt:wenbluo is very working hard
f_1.txt:wenbluo is very sensitive
f_2.txt:who am i ? i am wenbluo ,you can call me robert
f_3.txt:hi robert ,wenbluo ,welcome to the shell world!
[wenbluo@phxdpeetl019 test]$ grep -n wenbluo *
f_1.txt:1:wenbluo is very working hard
f_1.txt:2:wenbluo is very sensitive
f_2.txt:1:who am i ? i am wenbluo ,you can call me robert
f_3.txt:1:hi robert ,wenbluo ,welcome to the shell world!
[wenbluo@phxdpeetl019 test]$ cat f_1
cat: f_1: No such file or directory
[wenbluo@phxdpeetl019 test]$ cat f_1.txt
wenbluo is very working hard
wenbluo is very sensitive
```

grep -n : 告知在文件第几行出现；

```
[wenbluo@phxdpeetl019 test]$ grep -in robert jobs_2 jobs_1  
jobs_2:1:echo 'hello robert ,nice to see U'  
###在jobs_2:第一行出现  
jobs_1:1:echo hello world ,hello robert;  
###在jobs_1:第一行出现  
  
###作用可以结合 vi 中的 rownumber + SHIFT+ G 快速定位
```

```
grep -il  
CLSF_DTRFC_SUM_RPT_MKT|CLSF_DAILY_MKT_SC|CLSF_WEEKLY_  
MKT_SC * : 查找含有xxx忽略大小写的文件:  
--> ll l:list 简单列表出来
```

```
f_1.txt  
f_2.txt  
f_3.txt  
[wenbluo@phxdpeetl019 test]$ grep -il wenbluo *  
f_1.txt  
f_2.txt  
f_3.txt  
[wenbluo@phxdpeetl019 test]$
```

- \$ grep -i robert jobs_1 jobs_2
jobs_1:echo hello world ,hello robert;
jobs_2:echo 'hello robert ,nice to see U'

####grep 格式 : partern file1 file2
####可以在多个文件中查找

```
#### : partern *  
[wenbluo@phxdpeetl019 test]$ grep -il robert jobs_2 jobs_1  
jobs_2  
jobs_1
```

```
connected_ads_ended  
[wenbluo@lvsdpeetl001 test]$ grep -in 'connected ads' grep_test  
2:connected ads  
[wenbluo@lvsdpeetl001 test]$ grep -in 'connected ads replies' grep_test  
[wenbluo@lvsdpeetl001 test]$ egrep -in connect|replies grep_test  
-bash: replies: command not found  
^C  
[wenbluo@lvsdpeetl001 test]$ egrep -in 'connect|replies' grep_test  
2:connected ads  
4:email_replies  
5:connected_ads  
6:connected_ads_ended  
[wenbluo@lvsdpeetl001 test]$
```

当有空格时候，采用引号，进行模糊查找

```
[wenbluo@lvsdpeetl001 test]$ vi grep_test  
[wenbluo@lvsdpeetl001 test]$ grep -in 'connect ads' grep_test  
2:connect ads  
3:%connect ads  
6:% connect ads  
[wenbluo@lvsdpeetl001 test]$ _ 模糊匹配
```

H4 Summary:

1. shell是使用空白字符来分隔 命令参数。

2.

- **grep -e :匹配多个模型**

```
grep -ie 'robert' -ie 'wenbluo' * | wc -l
```

###等价于

```
egrep -i 'robert|wenbluo' * | wc -l  
11
```

- ```
cat dw_clsfld.step2_bsc_reply.sql | grep -i
p_clsfld_t.clsfd_reply_mth_sc | egrep -ni "create|delete|insert"
2:DELETE FROM P_CLSFD_T.CLSFD_REPLY_MTH_SC
3:INSERT INTO P_CLSFD_T.CLSFD_REPLY_MTH_SC
```

```
cat dw_clsfld.step2_bsc_reply.sql | grep -ni
p_clsfld_t.clsfd_reply_mth_sc | egrep -i "create|delete|insert"
1168:DELETE FROM P_CLSFD_T.CLSFD_REPLY_MTH_SC
1173:INSERT INTO P_CLSFD_T.CLSFD_REPLY_MTH_SC
```

##-ni 位置不要弄错了

- 如何通过grep 覆盖文件?

#### 删除failed 集, 只保留success

```
[wenbluo@phxdpeet1019 ~]$ cat weekly_sc
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190107-233002,20190108-011745,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190114-233003,20190115-011953,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190121-233002,20190122-003644,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190128-233005,20190129-000303 failed
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190129-003838,20190129-003840,failed
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190129-003945,20190129-011610,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190204-233002,20190205-002649,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190211-233002,20190212-004234,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190218-233002,20190219-003222,success
```

错误做法一

```
grep -v failed weekly_sc >weekly_sc
grep: input file 'weekly_sc' is also the output
[wenbluo@phxdpeet1019 ~]$ cat weekly_sc
[wenbluo@phxdpeet1019 ~]$ ps -u
##weekly_sc文件消失
```

正确做法一：放到临时文件夹中/tmp

```
[wenbluo@phxdpeet1019 ~]$ grep -v failed weekly_sc_bak >/tmp/weekly_sc_bak
[wenbluo@phxdpeet1019 ~]$ mv /tmp/weekly_sc_bak ./weekly_sc
[wenbluo@phxdpeet1019 ~]$ cat weekly_sc
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190107-233002,20190108-011745,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190114-233003,20190115-011953,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190121-233002,20190122-003644,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190129-003945,20190129-011610,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190204-233002,20190205-002649,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190211-233002,20190212-004234,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190218-233002,20190219-003222,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190225-233001,20190226-002321,success
dw_clsfld.job_runner_v2.ksh[space]weekly_sc,20190304-233002,20190305-022127,success
```

最安全做法：

```
[wenbluo@phxdpeet1019 ~]$ grep -v failed weekly_sc_bak >/tmp/weekly_sc_bak$$
[wenbluo@phxdpeet1019 ~]$ mv /tmp/weekly_sc_bak$$./weekly_sc
[wenbluo@phxdpeet1019 ~]$ cat weekly_sc
```

- **grep -r "printf" \*** 在当前目录及所有子目录下递归查找调用了printf函数的行，并显示行号。

•

#### H4 重定向

##### 输入输出重定向

##### H5 输出重定向

结果将不再在终端显示，而是存放到文件中

type	description
>	输出重定向
<	输入重定向
>>	输出追加 (insert)
/dev/null	特殊文件
0--> <	stdin (标准输入) :文件夹: /dev/stdin
1--> 1> 等价与 >	stdout (标准输出) :文件夹 : /dev/stdout
2---> 2>	stderr (错误输出) :/dev/stderr
>& (中间不能有空格)	输出重定向到指定文件描述符中
&> (与上面等价)	输出重定向到指定文件描述符中
文件描述符	0 1 2

```
[wenbluo@phxdpeet1019 dev]$ ll *std*
total 0
lrwxrwxrwx 1 root root 15 Apr 2 22:52 stderr -> /proc/self/fd/2
lrwxrwxrwx 1 root root 15 Apr 2 22:52 stdin -> /proc/self/fd/0
lrwxrwxrwx 1 root root 15 Apr 2 22:52 stdout -> /proc/self/fd/1
[wenbluo@phxdpeet1019 dev]$
```

实例：

- [wenbluo@phxdpeet1019 ~]\$ ls 2.txt 33  
`ls: cannot access 33: No such file or directory`  
`2.txt`  
[wenbluo@phxdpeet1019 ~]\$ ls 2.txt 33 >4.txt 2>&1  
##终端不显示任何提示信息，不管任何信息（错误输出or标准输出）都放到4.txt文件中  
[wenbluo@phxdpeet1019 ~]\$ cat 4.txt  
`ls: cannot access 33: No such file or directory`  
`2.txt`

- [wenbluo@phxdpeet1019 test]\$ ll  
total 0  
-rw-rw-r-- 1 wenbluo wenbluo 14 Apr 2 01:37 F\_4.txt  
-rw-rw-r-- 1 wenbluo wenbluo 48 Apr 2 01:36 w\_3.txt  
-rw-rw-r-- 1 wenbluo wenbluo 56 Apr 2 21:55 wbluo\_1.txt  
-rw-rw-r-- 1 wenbluo wenbluo 48 Apr 2 01:32 wbluo\_2.txt  
[wenbluo@phxdpeet1019 test]\$ ll wbluo\_1.txt f\_2.txt 2>f\_6.txt  
-rw-rw-r-- 1 wenbluo wenbluo 56 Apr 2 21:55 wbluo\_1.txt  
###2>仅仅是错误输出重定向，正缺输出还是在终端显示  
[wenbluo@phxdpeet1019 test]\$ cat f\_6.txt  
ls: cannot access f\_2.txt: No such file or directory

- /dev/null 2>&1 --> /dev/null 表示特殊文件，（黑洞），并不会往终端输出结果

```
[wenbluo@phxdpeet1019 ~]$ ls 2.txt 33 >/dev/null

ls: cannot access 33: No such file or directory

###错误信息依旧显示在终端

[wenbluo@phxdpeet1019 ~]$ ls 2.txt 33 >/dev/null 2>&1

##stdout/stderr都重定向到/dev/null文件中，终端不显示任何信息
```

### 拓展可以清空数据

cat /dev/null > data.log ##清空data.log 文件

-->2019/09/04: 清空文件的新做法, >data.log

### 拓展与IF 的结合操作:

```
$ who | grep -n wenbluo

23:wenbluo pts/28 2019-04-18 23:39

(phxbastion100.phx.ebay.com)

29:wenbluo pts/35 2019-04-18 23:17

(phxbastion100.phx.ebay.com)

31:wenbluo pts/38 2019-04-19 00:06

(phxbastion100.phx.ebay.com)

[wenbluo@phxdpeet1019 test]$ who | grep -n wenbluo

>/dev/null

[wenbluo@phxdpeet1019 test]$ echo $?

0

[wenbluo@phxdpeet1019 test]$ if who | grep -n wenbluo

>/dev/null

> then

> echo 'wenbluo is log in'

> else

> echo 'user is not log in'

> fi

wenbluo is log in
```

- [wenbluo@phxdpeetl019 ~]\$ ls 2.txt 33 2>3.txt  
2.txt  
###错误信息重定向到3.txt 文件  
[wenbluo@phxdpeetl019 ~]\$ ll  
total 76  
-rw-rw-r-- 1 wenbluo wenbluo 3144 Apr 9 18:58 2.txt  
-rw-rw-r-- 1 wenbluo wenbluo 48 Apr 9 20:04 3.txt

- 2019/04/30

```
ll vartest4 vartest6 > /dev/null 1>&2
ls: cannot access vartest6: No such file or directory
-rwxrwxr-x 1 wenbluo wenbluo 31 Apr 30 00:55 vartest4
[wenbluo@phxdpeetl019 test]$ ll vartest4 vartest6 > /dev/null
2>&1
```

##为什么 1/2位置不能互换？ 文件描述符号？

2>&1 :表示标准错误输出重定向到标准输出，而标准输出是重定向到 /dev/null文件，

所以所有输出都重定向到 /dev/null中

**1/2位置不可以换**

## H5 输入重定向

- cat > reorient.txt <while\_test  
###cat 从while\_test 获取数据并输出到reorient.txt 文件中  
cat | wc -l <reorient.txt  
###cat reorient.txt 文件并 统计行数  
17

## H2 引用quote

- 单引号('): 括起来的字符作为普通字符
- 双引号("):括起来的字符，除 "\$", "\", ``和``''保留其特殊功能外，其余仍作为普通字符

\$:表示变量替换

\:表示转义符

::表示shell 命令

- 反引号 (`) :括起来的字串被解释为命令， shell首先执行该命令，并将他的标准输出结果例如：取代整个反引号部分

## H3 summary

- [00:31:20]wenbluo@lvsdpeetl001 ~/etl\_home/sql\_test/test` text='i am robert ,my name is wenbluo'  
[01:31:58]wenbluo@lvsdpeetl001 ~/etl\_home/sql\_test/test` echo \$text  
i am robert ,my name is wenbluo  
[01:32:02]wenbluo@lvsdpeetl001 ~/etl\_home/sql\_test/test` \_

shell 程序并不会讲 引号传入程序中；

- 1. 单引号和双引号的区别:

```
[wenbluo@phxdpeetl019 ~]$ date='2019-01-01'
[wenbluo@phxdpeetl019 ~]$ echo '$date'
$date
[wenbluo@phxdpeetl019 ~]$ echo $date
2019-01-01
[wenbluo@phxdpeetl019 ~]$ echo "'$date'"
2019-01
[wenbluo@phxdpeetl019 ~]$ echo "$date"
2019-01-01
[wenbluo@phxdpeetl019 ~]$
```

- 2.

```
[wenbluo@lvsdpeetl001 shell_book_practice]$ awk -F '-' '{ print $2 }' < info.txt
SystemAdminin
Oracle , MySql etc
FireWall ,Network ,Online Security etc.
Website
Net app ,Disk

[wenbluo@lvsdpeetl001 shell_book_practice]$ awk -F '-' "{ print $2 }" < info.txt
Linux - SystemAdminin
Database - Oracle , MySql etc
Security -FireWall ,Network ,Online Security etc.
Cool - Website
Storage -Net app ,Disk

[wenbluo@lvsdpeetl001 shell_book_practice]$
```

- 3.

```
hello robert
[07:31:25]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice str2='hello world,"hello robert"'
[07:32:14]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice echo $str2
hello world,"hello robert"
[07:32:18]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice str3="hello world,'hello robert'"
[07:32:43]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice echo $str3
hello world,'hello robert'
[07:32:46]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice _
```

单引号中可以有双引号，反之亦可

### H3 \:反斜线

- 1. 作为续航符，主要针对长code时候，放在末行

```
[07:40:22]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice str="the shell treats a backslash that's the \
>> last character of the line of input as a line \
>> continuation. it removes the newline too"
[07:40:32]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice echo $str
the shell treats a backslash that's the > last character of the line of input as a line > continuation. it removes the newline too
[07:40:37]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice _
```

- 2.

## H2 Smart key

- **ctrl+c** : 退出指令 --暂时不用
- **clear** :清屏操作
- **smart key**
- **ctrl+U** :清空当前行
- **tab** : 自动补全
- 

## H2 Concepts

### H4 变量

- 1. 采用echo 输出变量

两种输出方式

```
echo $your_name/ ${your_name}
```

```
your_name="qinjx"
echo $your_name
echo ${your_name}
```

变量名外面的花括号是可选的，加不加都行，加花括号是为了帮助解释器识别变量的边界 比如下面这种情况：

```
for skill in Ada Coffe Action Java; do
 echo "I am good at ${skill}Script"
done
```

```
wenbluo@L-SHC-16505239 MINGW64 ~/desktop/wbbluo/111
$ echo '$a'
10
wenbluo@L-SHC-16505239 MINGW64 ~/desktop/wbbluo/111
$ echo ' $a'
$a
wenbluo@L-SHC-16505239 MINGW64 ~/desktop/wbbluo/111
$ echo $a
10
{y
nj>
me wenbluo@L-SHC-16505239 MINGW64 ~/desktop/wbbluo/111
$ echo ${a}
10
$
```

不要用单引号 ‘ ’

2. 转义符"\\"、
3. read命令行：接收键盘输入

```
[wenbluo@lvsdpeet1001 ~]$ read name
23
###直接read +参数名字
[wenbluo@lvsdpeet1001 ~]$ echo $name
23
[wenbluo@lvsdpeet1001 ~]$ read -p "how old are you ?"
age_value
how old are you ?2222
###增强一种人机对话
[wenbluo@lvsdpeet1001 ~]$ echo $age_value
2222
```

option		
read - p	#延迟五秒，没有输入将自动退出 read -p "Input a number:" -t 5 pwd #read [-p "提示信息"] 变量名	echo \$pwd
read - n	read -p "Input a word:" -n 5 Word #限制输入长度为5	echo \$Word

option		
read -t	wait times	
read -d	read -dp -p "Input some words end with q:" word	直到输入q, 将自动退出

#### 4. 系统变量:

- \$HOME \$PATH \$who \$CPATH

```
echo $HOME
/home/chewu
```

#### whereis my Path ?

1. Your computer is comprise of files ;of which two types . **data files and executable files**

when you use command i.e ls , are just executable files .

2. PATH is a global variabel that contains **a string of different paths separated by a :**

your computer then uses this variable to understand what directoried it shoud look in to find the executable you're requesting .

3. if you want to add new directory into path

- export PATH='/my/directory/bin:\$PATH'
- export PATH='\$PATH:/my/directory/bin'
- -->**elimeter is " : "**

4. if we want the changes to our PATH to persist,so we could edit file :

**.bashrc and .bash\_profile**

summary :

跟windows系统一样，添加applicaiton 到环境变量中：

- \$(hostname)

```
echo $(hostname)
phxdpeet1019.phx.ebay.com
```

```
[wenbluo@phxdpeet1019 ~]$ echo `hostname`
phxdpeet1019.phx.ebay.com
```

- whereis bash

##查看bash 位置

```
[ARES]/etc/hadoop > whereis bash
bash: /bin/bash /usr/share/man/man1/bash.1.gz
```

whereis 是一个命令符

- alias
- date ###当前时间

--> 拓展: **date +‘format’** 以特定格式输出

```
date +'Y%m%d-%H%M%S'
```

```
20190415-215810
```

```
[wenbluo@phxdpeet1019 ~] $ date +'Y%m%d-%H%M%S'
20190415-Apr1924
[wenbluo@phxdpeet1019 ~] $ date '+%Y%m%d-%H%M%S'
20190415-Apr1935
```

###两者等价，一般后者最好 这样git 上也能使用；

- hostname ##服务器名称

```
$ hostname
phxdpeet1019.phx.ebay.com
```

- \$SHELL

```
$ echo $SHELL
/bin/bash
###返回shell 哪个版本?
```

- \$(whoami)

```
echo $(whoami)
wenbluo
[wenbluo@phxdpeet1019 ~] $ echo `whoami`
wenbluo
```

- who am i

```
who am i
wenbluo pts/27 2019-05-04 23:38
(phxbastion100.phx.ebay.com)
```

- whereis

```
-rw-r--r-- 1 hadoop hadoop 15028 Oct 18 2018 truststore
[ARES]/etc/hadoop > whereis bash
bash: /bin/bash /usr/share/man/man1/bash.1.gz
[ARES]/etc/hadoop > whereis ksh
ksh: /bin/ksh /usr/bin/ksh
[ARES]/etc/hadoop > whereis hive
hive:
[ARES]/etc/hadoop >
```

- which hive which hadoop

```
[ARES]/> which hadoop /apache/hadoop/bin/hadoop
```

- 

5.

6.

#### H4 运算符

1.

```
$ echo `expr 2 + 2`
4

wenbluo@L-SHC-16505239 MINGW64
$ echo 2+2
2+2

wenbluo@L-SHC-16505239 MINGW64
$ |
```

```
a=10
```

```
((a=a+3)) #a加3
```

```
[wenbluo@phxdpeet1019 test]$ expr 2 * 2
expr: syntax error
[wenbluo@phxdpeet1019 test]$ expr 2 + 2
4
[wenbluo@phxdpeet1019 test]$ expr 2 ^ 2
expr: syntax error
```

2. expr

background :**Expr** 与 **let** 一样，并且还可以进行字符串运算

```
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu]expr 1 + 1
2
```

```
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu]expr 1.1 + 1
expr: non-numeric argument
```

##expr只能输出不含有空格的字符串

```
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu]str2='iamrobert'
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu]expr length $str2
9
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu]str3='i am
robert'
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu]expr length $str3
expr: syntax error
```

####通过\${#str3}

```
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu]echo ${#str3}
11
```

###截取字段

```
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu]expr substr $str2
1 5
iamro
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu]expr substr $str3
1 5
expr: syntax error
```

###

```
echo ${str3:1:5}
am r
```

## Expr expression1 操作符 expression2 --format 注意间隔

```
val=`expr 2 + 2`
echo "两数之和为 : $val"
```

运行实例 »

执行脚本，输出结果如下所示：

```
两数之和为 : 4
```

两点注意：

- 表达式和运算符之间要有空格，例如 2+2 是不对的，必须写成 2 + 2，这与我们熟悉的大多数编程语言不一样。
- 完整的表达式要被 `` 包含，注意这个字符不是常用的单引号，在 Esc 键下边。反引号

-->拓展

- let 指令

background : Let命令让**BASH shell**执行算数运算的操作

- let 变量名 = 变量1 运算符 变量2 --format

```
num1=106
```

```

num2=23
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu] let
add=$num1+$num2
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu] echo $add
129
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu] let
div=$num1/$num2
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu] echo $div
4
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu] let
mod=$num1%$num2
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu] echo $mod
14
####let 不适用与小数运算
let 1+1
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu] let 1.1+1
-bash: let: 1.1+1: syntax error: invalid arithmetic
operator (error token is ".1+1")

```

3. 注意：条件表达式要放在方括号之间，并且要有空格，例如：[\$a==\$b] 是错误的，必须写成 [ \$a == \$b ]。

#### 4. 关系运算符

-eq	检测两个数是否相等，相等返回 true。	[ \$a -eq \$b ] 返回 false。
-ne	检测两个数是否不相等，不相等返回 true。	[ \$a -ne \$b ] 返回 true。
-gt	检测左边的数是否大于右边的，如果是，则返回 true。	[ \$a -gt \$b ] 返回 false。
-lt	检测左边的数是否小于右边的，如果是，则返回 true。	[ \$a -lt \$b ] 返回 true。
-ge	检测左边的数是否大于等于右边的，如果是，则返回 true。	[ \$a -ge \$b ] 返回 false。
-le	检测左边的数是否小于等于右边的，如果是，则返回 true。	[ \$a -le \$b ] 返回 true。

echo \$? 表示对刚刚执行的表达式进行判断

0表示成立，1表示不成立

```

a=10
b=13
[$a -gt $b]
echo $?
1
0 表示TRUE 1表示False

```

-->拓展：不支持文本操作？

```

name='wbluo'
[wenbluo@phxdpeet1019 test]$ [$name -eq wbluo]
bash: [: wbluo: integer expression expected

```

## 5. 布尔运算符

运算符	说明	举例
!	非运算，表达式为 true 则返回 false，否则返回 true。	[ ! false ] 返回 true。
-o	或运算，有一个表达式为 true 则返回 true。	[ \$a -lt 20 -o \$b -gt 100 ] 返回 true。
-a	与运算，两个表达式都为 true 才返回 true。	[ \$a -lt 20 -a \$b -gt 100 ] 返回 false。

## 6. 逻辑运算符

运算符	说明	举例
&&	逻辑的 AND	[ [ \$a -lt 100 && \$b -gt 100 ] ] 返回 false
	逻辑的 OR	[ [ \$a -lt 100    \$b -gt 100 ] ] 返回 true

## H2 Dot 命令

```
[wenbluo@phxdpeet1019 test]$. parent.sh #####与parent.sh文件之间要有空格！
```

```
[wenbluo@phxdpeet1019 test]$ echo $a
wbluo
[wenbluo@phxdpeet1019 test]$ echo $b
robert
[wenbluo@phxdpeet1019 test]$
```

```
####等价于在父shell 上输入一遍 parent.sh command
##less parent.sh
a='wbluo'
b='robert'
export a ;
```

-->bash 与 dot 命令的区别

## H2 Summary :

该命令可以当前shell 中执行file 的内容。也就是说，file中命令就像是你直接输入的一样，由当前shell 执行，不是在子shell 中。（父Shell 和 子Shell 概念）

```
[wenbluo@phxdpeet1019 test]$ bash chid_2.sh
wbluo
```

```
[wenbluo@phxdpeet1019 test]$. chid_2.sh
wbluo
robert
wenbluo

#less chid_2.sh
echo $a
echo $b
echo $c
```

```
[wenbluo@phxdpeet1019 test]$ bash chid_2.sh
wbluo
```

```
[wenbluo@phxdpeet1019 test]$. parent.sh
[wenbluo@phxdpeet1019 test]$ bash chid_2.sh
wbluo
```

```
wenbluo
#less parent.sh
a='wbluo'
b='robert'
c='wenbluo'
export a
export c
```

#### H4 Bash

background:

如果文件第一行的前两个字符是 #! ， 那么余下的部分就指定了该文件的解析器，因此

```
#! /bin/ksh
#! /bin/bash
```

**文件不一定是sh文件：**

- **whereis bash** : 查看你bash所在的位置:

```
[wenbluo@phxdpeet1019 test]$ whereis bash
bash: /bin/bash /etc/bash.bashrc /usr/share/man/man1/bash.1.gz
```

```
[wenbluo@phxdpeet1019 test]$ whereis sh
sh: /bin/sh /usr/share/man/man1/sh.1.gz
/usr/share/man/man1p/sh.1p.gz
```

- 如何运行程序:

有两种方法：一种是显式制定 BASH 去执行：

**bash hello 或 sh hello** (这里 sh 是指向 bash 的一个链接，“**lrwxrwxrwx 1 root root 4 Aug 20 05:41 /bin/sh -> bash**”)

或者可以先将 hello 文件改为**可以执行的文件**，然后直接运行它

```
[wenbluo@phxdpeet1019 bin]$ bash chid_2.sh
hello wenbluo
```

```
#!/bin/bash
echo $a
echo $b
```

```

echo $c

###在/bin/bash 目录下直接运行

[wenbluo@phxdpeet1019 bin]$ cd $HOME/etl_home/sql_test/test/
[wenbluo@phxdpeet1019 test]$ chid_2.sh
-bash: chid_2.sh: command not found
由于不是在/bin目录下 或者说在 $PATH 路径下没有找到该文件
[wenbluo@phxdpeet1019 test]$./chid_2.sh
hello wenbluo
通过将当前目录. 运行该文件

[wenbluo@phxdpeet1019 test]$ parent.sh
bash: parent.sh: command not found
[wenbluo@phxdpeet1019 test]$ bash parent.sh
##第二种方法显示 Bash调度文件
wbluo

```

- 
- 
- 
- 
- 

#### • **#!/bin/bash**

**#!** 是说明 **hello** 这个文件的类型的，有点类似于 **Windows** 系统下用不同文件后缀来表示不同文件类型的意思（但不相同）。“**/bin/bash**”就表明该文件是一个 **BASH** 程序，需要由 **/bin** 目录下的 **bash** 程序来解释执行

- 
- . ./parent.sh

```
[wenbluo@phxdpeet1019 test]$ echo $c
wenbluo
```

```
#less parent.sh
a='wbluo'
b='robert'
c='wenbluo'
export c
```

```
##因为parent.sh是子shell
##当运行 . ./parent.sh 就好比 在父shell 环境下再次输入同Parent.sh
code
```

## H2 Source

- 与**dot** 命令相同，都是执行文件
- **source file <--> . file**

## ``命令

### 反引号

```
[wenbluo@phxdpeet1019 ~]$ echo the data and time is : $date
the data and time is :
[wenbluo@phxdpeet1019 ~]$ echo the data and time is : date
the data and time is : date
[wenbluo@phxdpeet1019 ~]$ echo the data and time is : $(date)
the data and time is : Thu May 16 03:32:15 -07 2019
[wenbluo@phxdpeet1019 ~]$ echo the data and time is : `date`
the data and time is : Thu May 16 03:32:22 -07 2019
[wenbluo@phxdpeet1019 ~]$
```

当shell扫描命令行时，它识别出了反引号，于是期望接下来的是一个命令。date 是一个返回当地时间的一个命令。在老的版本中依旧采用``形式，新形式采用\$()

- [chewu@phxdpeet1019.phx.ebay.com:/home/chewu] gap=expr  
\$first\_stamp - \$end\_time  
-bash: 1557104401: command not found  
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu] gap=`expr  
\$first\_stamp - \$end\_time`  
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu] echo \$gap  
-12202
- start\_time=date -d '2019-05-06 22:30:03' +%s  
-bash: -d: command not found  
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu/etl\_home/cronlog]  
start\_time=`date -d '2019-05-06 22:30:03' +%s`  
[chewu@phxdpeet1019.phx.ebay.com:/home/chewu/etl\_home/cronlog]  
echo \$start\_time  
1557207003
- time1=date +'Y%m%d-%H%M%S'  
-bash: +Y%m%d-%H%M%S: command not found  
[wenbluo@phxdpeet1019 ~]\$ date +'Y%m%d-%H%M%S'
20190515-203314  
[wenbluo@phxdpeet1019 ~]\$ time1=`date +'Y%m%d-%H%M%S'`  
[wenbluo@phxdpeet1019 ~]\$ echo \$time1
20190515-203334
- 

## H2 Sudo

- sudo :superuser do :允许已经验证的用户用其他用户运行指令；
- A non-root user with sudo privileges -->how to grant the sudo rights?
-

## H2 空指令

- : 空占位符，不执行操作

```
echo hi robert ,wenbluo ,welcome to the shell world!
echo how old are u ?
read age
if [$age -gt 20]
then
: ##echo 'You are old enough'
else
echo 'You are novice'
fi
export age
bash

[wenbluo@phxdpeet1019 test]$ bash w_3.sh
hi robert ,wenbluo ,welcome to the shell world!
how old are u ?
22 ##当大于等于 20的时候，不操作
[wenbluo@phxdpeet1019 test]$
```

## H2 Jobid

background : 目前是交互式的命令行，我们可以将一些**job**放到后台运行，**前台照样可以操作其他事情**

特殊字符： &

1. 通过**jobs** 查看运行的job [只能在当前session 有效，而且是系统变量，无需要指定那个文件夹]

也可以通过 **ps -aux |grep xxx or ps -ef |grep xx** :显示所有进程id

```
jobs
[1]+ Stopped ./jobs_2
jobs -1
[1]- 48028 Stopped (tty input) ./jobs_2
[2]+ 21491 Stopped (tty input) ./parent_2.sh
48028 : 表示进程id， ./jobs_2 :表示命令行
[1],[2]:表示任务编号
+:表示当运行的job ， -:表示下一个job(当前作业之后的一个作业)
jobs的状态可以是running, stopped, Terminated
```

```
[yuebl@lvsdpeet1001.vs.ebay.com:/dw/etl/home/prod/land/dw_clsfid/job_runner/weekly_sc/cfg]jobs -l
[1]- 16328 Running nohup /home/$(whoami)/etl_home/bin/crontab.wrapper.ksh /export/home/$(whoami)/etl_home/bin/dw_clsfid.job_runner_v2.ksh
[2]+ 10135 Running nohup /home/$(whoami)/etl_home/bin/crontab.wrapper.ksh /export/home/$(whoami)/etl_home/bin/dw_clsfid.job_runner_v2.ksh
[yuebl@lvsdpeet1001.vs.ebay.com:/dw/etl/home/prod/land/dw_clsfid/job_runner/weekly_sc/cfg]kill %2
-->
```

## 如何让**10135** 相对**16328** 后运行

- **+-** :并不是先后意思，而是`+`: 当前运行的job , `-`: 表示是当前作业之后的一个作业

并不是指先运行 `+`,后再 运行`-`...

```
[wenbluo@lvsdpeet1001 test]$ jobs -1
[1] 22748 Running nohup ./while_2.sh > test_2 &
[2]- 22834 Running nohup ./while_1.sh > test_1 &
[3]+ 22969 Running nohup ./while_1.sh > test_3 &
[wenbluo@lvsdpeet1001 test]$ cat test_2 |wc -l
25
[wenbluo@lvsdpeet1001 test]$ cat test_1 |wc -l
10
[wenbluo@lvsdpeet1001 test]$ cat test_3 |wc -l
10
[wenbluo@lvsdpeet1001 test]$ cat test_3 |wc -l
11
[wenbluo@lvsdpeet1001 test]$ cat test_1 |wc -l
13
[wenbluo@lvsdpeet1001 test]$ cat test_2 |wc -l
33
[wenbluo@lvsdpeet1001 test]$ _
```

- `bg`

将任务放在后台运行

```
[wenbluo@phxdpeet1019 bin]$ ps -aux
dw_adm 20870 0.0 0.0 6256 584 ? S May07
0:00 /ebay/uc4/agent-bin/Linux_x86_64_12.0.3_B1634/bin./ucx
dw_adm 20887 0.0 0.0 107024 1640 ? S 20:10
0:00 /bin/ksh /ebay/uc4/agent-tmp/JDQPGQLB.TXT
^Z #####ctrl+z
[1]+ Stopped ps -aux
[wenbluo@phxdpeet1019 bin]$ fg
ps -aux
```

- `fg` : foreground

将任务放在前台运行

当有多个任务时候:

```
[ARES]/export/home/b_clefd > jobs -l
[1] 5775 Running nohup hive -f motor_tmp_2019_05_10_tmp.hql --hiveconf clefd_start_dt='2016-07-01'
-hiveconf clefd_end_dt='2016-10-01' &
[2]- 5897 Running nohup hive -f motor_tmp_2019_05_10_tmp.hql --hiveconf clefd_start_dt='2016-04-01'
-hiveconf clefd_end_dt='2016-07-01' &
[3]+ 6028 Running nohup hive -f motor_tmp_2019_05_10_tmp.hql --hiveconf clefd_start_dt='2016-01-01'
-hiveconf clefd_end_dt='2016-04-01' &
[ARES]/export/home/b_clefd >
```

特点： 有三个jobs , 执行顺序: 6028 >5897 >5775

-->困惑: 难道不是并行运算? NO

`fg %1` :表示将任务1 调前台运行; **##并不是pid**

```
[wenbluo@phxdpeet1019 bin]$ jobs
```

```
[1]- Running nohup ksh
refresh_weekly_sc_core_tmp.ksh 2019-04-28 2019-05-04 >
000001.log 2>&1 &
[2]+ Stopped ps -aux
```

###可以看到有两个任务；

```
[wenbluo@phxdpeet1019 bin]$ fg %2 ##也可以fg 2，但是一般推荐前者
ps -aux
###将任务 2 放到前台运行；
```

```
Signal 18 (CONT) caught by ps (procps version 3.2.8).
[wenbluo@phxdpeet1019 bin]$ jobs
[1]+ Running nohup ksh
refresh_weekly_sc_core_tmp.ksh 2019-04-28 2019-05-04 >
000001.log 2>&1 &
这个时候只有一个任务在后台运行
[wenbluo@phxdpeet1019 bin]$
```

- kill

```
]$ jobs
[1] Stopped ./jobs_2
[2]- Stopped ./parent_2.sh
[3]+ Stopped ./explicit.sh
[wenbluo@phxdpeet1019 test]$ kill %%
###终止当前运行的指令（./explicit.sh）
[wenbluo@phxdpeet1019 test]$ jobs
[1]- Stopped ./jobs_2
[2]+ Stopped ./parent_2.sh
[3] Terminated ./explicit.sh
[wenbluo@phxdpeet1019 test]$ kill %2
###终止任务id 为2的jobs
```

```
[ARES]/export/home/b_clefd > jobs -l
[1]- 6806 Running nohup hive -f motor_tmp_2019_05_10_tmp.hql --hiveconf clefd_start_dt='2016-01-01' -hiveconf clefd_end_dt='2016-04-01' &
[2]- 6807 Running nohup hive -f motor_tmp_2019_05_10_tmp.hql --hiveconf clefd_start_dt='2016-04-01' -hiveconf clefd_end_dt='2016-07-01' &
[3]+ 6808 Running nohup hive -f motor_tmp_2019_05_10_tmp.hql --hiveconf clefd_start_dt='2016-07-01' -hiveconf clefd_end_dt='2016-10-01' &
[ARES]/export/home/b_clefd > kill 3
bash: kill: [3] - Operation not permitted
[ARES]/export/home/b_clefd > kill %3
[ARES]/export/home/b_clefd > jobs -l
```

注意打%

```
-hiveconf clefd_end_dt='2016-10-01'
[ARES]/export/home/b_clefd > jobs -l
[1]- 6806 Running nohup hive -f motor_tmp_2019_05_10
-hiveconf clefd_end_dt='2016-04-01' &
[2]+ 6807 Running nohup hive -f motor_tmp_2019_05_10
-hiveconf clefd_end_dt='2016-07-01' &
[ARES]/export/home/b_clefd >
```

or 直接kill pid

```
-hiveconf clscfd_end_dt='2016-07-01' &
[ARES]/export/home/b_clscfd > kill 6807
[ARES]/export/home/b_clscfd > jobs -l
[1]- 6806 Running nohup hive -f motor
hiveconf clscfd_end_dt='2016-04-01' &
[2]+ 6807 Exit 143 nohup hive -f motor
-hiveconf clscfd_end_dt='2016-07-01'
[ARES]/export/home/b_clscfd >
```

-->拓展

- 对于正在前台执行的job,我们可以`ctrl+Z`, 暂停(挂起), 然后 bg number 防止后台运行 -->`前提是是没有 & 时候`  
`ctrl+c` 表示终止;

2.

- `nohup refresh_weekly_sc_core.ksh 2019-04-14 2019-04-20 & ### &`表示后台指令 运行 `nohup` 表示防止挂起,或者会话意外关闭 (断开链接)

-->instance :

```
ksh target_table_load_handler.ksh
dw_clscfd.clscfd_daily_mkt_sc_wbluo_2019_04_29 tdd2
dw_clscfd.clscfd_daily_mkt_sc_wbluo.sql start_date=2019-01-01
end_date=2019-01-02 >test_log_2019_0420.log 2>&1 &
[1] 31619
```

2:

```
while_test 文件
COUNTER=0
while [$COUNTER -ge 0]
do
 ((COUNTER=$COUNTER+1))
 sleep 5
 echo $COUNTER
done

bash while_test &
[1] 25435 ##先返回pid
[wenbluo@phxdpeet1019 test]$ 1 ##后面运行成功后, 回向前端吐数据 , 如何
写了输出重定向的话, 就不会影响前端;
2
3
^C ####其次我就算终止这个pid, 其实并没有终止掉;
[wenbluo@phxdpeet1019 test]$ 4
5
```

##只有通过kill 25435 方能跳出死循环 or 关闭session

### 3: nohup +&的区别

nohup/

## H2 Ps

background:**Linux**中的**ps**命令是**Process Status**的缩写

类似**window**下的**资源管理器**,但是**unxi**下的只是显示瞬间行程状态

### H3 常见指令

```
[wenbluo@phxdpeet1019 ~]$ ps -u
Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
wenbluo 5886 0.0 0.0 120900 2116 pts/20 Ss 22:21 0:00 bash
wenbluo 12333 0.0 0.0 120908 2228 pts/46 Ss+ 21:59 0:00 -bash
wenbluo 21899 13.0 0.0 122700 1416 pts/29 R+ 22:40 0:00 ps -u
[wenbluo@phxdpeet1019 ~]$ ps -ux
```

- PID:cmd 的id
- TTY:命令所运行的位置
- TIME:运行该命令所占用CPU处理时间
- CMD:该进程的运行命令
- %CPU:进程CPU占用率
- %MEM:进程内存占用率
- STAT :状态码
  - 1. R:RUNNING
  - 2. S:SLEEPING 休眠中,受阻,在等待某个条件的形成或接受到信号
  - 3. T:TERMINATE
  - 4. s:进程的领导者(在他下面有子进程)
  - 5. +:位于后台的进程组

a: 表示该hostname下的终端机所有程序

u:表示该whoami 下(用户为主)的格式显示程序状况 -->**只展示用户调度的任务**

```
[wenbluo@lvsdpeet1001 test]$ ps -u
Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
wenbluo 12941 0.0 0.0 120908 2140 pts/11 Ss 00:00 0:00 bash
wenbluo 33008 0.0 0.0 120908 2100 pts/11 S 00:59 0:00 bash
wenbluo 34679 0.0 0.0 120908 2120 pts/11 S 00:59 0:00 /bin/bash
wenbluo 46338 0.0 0.0 122700 1400 pts/11 R+ 01:14 0:00 ps -u
[wenbluo@lvsdpeet1001 test]$
```

x:表示显示所有程序

```
ps -u wenbluo ## ps -u 亦可以
####查看特定用户的进程
```

PID	TTY	TIME	CMD
1423	pts/29	00:00:00	ps
5636	?	00:00:00	sshd
5886	pts/29	00:00:00	bash
11991	?	00:00:00	sshd
12333	pts/46	00:00:00	bash

```
ps -aux | grep weekly_sc
```

```
ps -ef 查看进程 :execute
```

```
[wenbluo@phxdpeetl019 ~]$ ps -ef |grep weekly
chewu 9020 40408 0 03:33 ? 00:00:00 /bin/ksh /home/chewu/etl_home/bin/target_table_load_handler.ksh
weekly_sc_upd daily td2 dw_clsfed.clsfd.daily_motor_sc.sql start_date=2019-04-14 end_date=2019-04-20
chewu 9536 9020 0 03:33 ? 00:00:00 ksh -x /home/chewu/etl_home/bin/target_table_load.ksh dw_clsfed.
upd_daily td2 dw_clsfed.clsfd.daily_motor_sc.sql
chewu 15251 15235 0 00:00 ? 00:00:00 /bin/sh -c /home/chewu/etl_home/bin/crontab.wrapper.ksh /home/c
n/dw_clsfed.to_hive_weekly_schedule_pet.ksh > /dev/null 1>&2
chewu 15255 15251 0 00:00 ? 00:00:00 /usr/bin/ksh -eu /home/chewu/etl_home/bin/crontab.wrapper.ksh /
ome/bin/dw_clsfed.to_hive_weekly_schedule_pet.ksh
chewu 15484 15255 0 00:00 ? 00:00:00 /home/chewu/etl_home/bin/dw_clsfed.to_hive_weekly_schedule_pet.k
ome/bin/dw_clsfed.to_hive_weekly_schedule_pet.ksh
chewu 19414 34204 0 03:18 ? 00:00:00 /bin/ksh /home/chewu/etl_home/bin/target_table_load_handler.ksh
weekly_sc_upd daily td2 dw_clsfed.step2_clsfed.daily_sc_mkt_mtvc.sql start_date=2019-04-14 end_date=2019-04-20
chewu 20347 19414 0 03:18 ? 00:00:00 ksh -x /home/chewu/etl_home/bin/target_table_load.ksh dw_clsfed.
upd_daily td2 dw_clsfed.step2_clsfed.daily_sc_mkt_mtvc.sql
wenbluo 24679 22694 0 03:37 pts/0 00:00:00 grep weekly
chewu 34204 1 0 03:13 ? 00:00:00 ksh refresh_weekly_sc_marketing.ksh 2019-04-14 2019-04-20
chewu 40408 1 0 03:13 ? 00:00:00 ksh refresh_weekly_sc_motor.ksh 2019-04-14 2019-04-20
chewu 48107 1 0 03:13 ? 00:00:00 ksh refresh_weekly_sc_core.ksh 2019-04-14 2019-04-20
```

```
ps -auf | grep 16187 ## 16187 是pid : 进程id
```

```
[ARES]/export/home/b_clsfed > jobs -l
[1]+ 3124 Stopped (tty input) hive
[2]+ 16187 Running hive -e 'select count(1)from DW_CLSFED.VST_MOTOR_DAILY_NEW_2019_05_09' &
[ARES]/export/home/b_clsfed > 2019-05-10 03:27:55,259 Stage-1 map = 7%, reduce = 0%, Cumulative CPU 230.23 sec

[ARES]/export/home/b_clsfed ps -aux |grep 16187
Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ
b_clsfed 16187 35.5 0.6 4013996 590600 pts/19 Sl 03:27 0:19 /usr/java/latest/bin/java -Xmx4000m -server -Dlog4j.
nfiguration=log4j.properties -Dcom.sun.management.jmxremote.authenticate=true -Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.password.file=/apache/hadoop/conf/jmxremote.password -Dhadoop.log.dir=/apache/hadoop/logs
-Dhadoop.log.file=hadoop.log -Dhadoop.home.dir=/ebay/apache/hadoop-2.7.1.2.4.2.0-258 -Dhadoop.id.str=hadoop -Dhadoop.root.logger=INFO,console -Djava.library.path=/apache/hadoop/lib/native:/apache/hadoop/lib/native/Linux-amd64-64:/apac
/hadoop/lib/native/Linux-amd64-64/lib:/ebay/apache/hadoop-2.7.1.2.4.2.0-258/lib/native -Dhadoop.policy.file=hadoop-p
olicy.xml -Djava.net.preferIPv4Stack=true -Xms384m -Xmx2048m -Dhadoop.security.logger=INFO,NullAppender org.apache.hadoop.util.RunJar /apache/hive/lib/hive-cll-1.2.1000.2.4.2.57-1.jar org.apache.hadoop.hive.cli.CliDriver -hiveconf mapreduc
job.queue.name=hddq-other-fin -e select count(1)from DW_CLSFED.VST_MOTOR_DAILY_NEW_2019_05_09
b_clsfed 16982 0.0 0.0 103320 912 pts/11 R+ 03:28 0:00 grep 16187
[ARES]/export/home/b_clsfed > ps -au |grep 16187
Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ
b_clsfed 17430 0.0 0.0 103320 884 pts/11 S+ 03:28 0:00 grep 16187
```

## H2 Set 命令

### H3 -XV

background :代码调试

- set -xv 查看所有传参后命令行内容，有助于分析实际执行的是什么命令，一种分析日志

--refer to : bash.sh

1. ++: 表示输出结果

2. set -x :表示调试开始

set +x :表示调试结束

3.

```
[02:43:14]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice vi bash.sh
[02:43:32]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice bash bash.sh
+(6)echo
bash.sh: line 6: echo: command not found
+(8)set +x

a value is :1 , the cumulate value is :0
a value is :2 , the cumulate value is :1
a value is :3 , the cumulate value is :3
a value is :4 , the cumulate value is :6
a value is :5 , the cumulate value is :10
a value is :6 , the cumulate value is :15
```

#### 4. 也可以直接写

bash -x bash.sh ## 并不用在bash.sh 内部显式写 set -x

```
*c
[02:52:56]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice bash -x bash.sh robert
+(2)var=robert
+(6)echo robert
bash.sh: line 6: echo: command not found
+(10)echo robert
robert
+(12)bash /home/wenbluo/etl_home/sql_test/test/while_2.sh
a value is :1 , the cumulate value is :0
a value is :2 , the cumulate value is :1
a value is :3 , the cumulate value is :3
a value is :4 , the cumulate value is :6
a value is :5 , the cumulate value is :10
a value is :6 , the cumulate value is :15
a value is :7 , the cumulate value is :21
a value is :8 , the cumulate value is :28
a value is :9 , the cumulate value is :36
a value is :10 , the cumulate value is :45
a value is :11 , the cumulate value is :55
a value is :12 , the cumulate value is :66
```

## H2 Shift

位置参数左移; shift : 表示\$2 --> \$1 ; \$5 -->\$4 :默认是移动一位

shift 2 : 表示 \$3 --> \$1

- **\$0** :不移动
- **Bsh** 定义了9个位置变量, 从 **\$1** 到 **\$9**,

### 1. 遍历所有参数

```
#!/bin/bash
if [$# -le 0];then
 echo "usage bash $0 arg1 arg2 .."
 exit 0
fi

while [[$# != 0]];do
 echo "第一个参数 $1, 参数个数${#}"
 shift
done

##if 里面再循环可以吗?
```

```
第一个参数 4 5, 参数个数5
[03:16:53]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice bash shfit_practice.sh 1 23 4 5
第一个参数 1, 参数个数4
第一个参数 23, 参数个数3
第一个参数 4, 参数个数2
第一个参数 5, 参数个数1
[03:17:01]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice _
```

### 2.

## H2 Eval

- 告知shell取出eval的参数, 重新运算求出参数的内容

- `a='ls |more'`

```
[wenbluo@phxdpeet1019 cfg]$ echo $a
ls |more
[wenbluo@phxdpeet1019 cfg]$ $a
ls: cannot access |more: No such file or directory
[wenbluo@phxdpeet1019 cfg]$ eval $a
1
break_point.seq
date_range.cfg
```

## H2 paste

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (master)
$ ls -l
Business English.md Computer 通信.md ECG data flow.md ggplot.md
Hive vs Presto.md Hive ware house .md hql 优化.md marking_scorecard.md - Shortcut.lnk*
R.md self-prespect.md server sql.md shell.md
Spark.md Tableau.md Teradata.md 数据库理论.md

wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (master)
$ ls
'Business English.md' 'Hive vs Presto.md' R.md Spark.md
'Computer 通信.md' 'Hive ware house .md' self-prespect.md Tableau.md
'ECG data flow.md' 'hql 优化.md' 'server sql.md' Teradata.md
ggplot.md 'marking_scorecard.md - Shortcut.lnk'* shell.md
'database theory.md'

wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (master)
$ ls -al
```

虽然可以指定以四列作为展示，但是是否能更加美丽的展示呢？

### 1. 列合并文件

```
[wenbluo@lvsdpeet1001 shell_book_practice]$ cat file1
Linux
Unxia
solar
hupux
window

[wenbluo@lvsdpeet1001 shell_book_practice]$ cat file2
suse
fedora
centos
oel
ubuntu
[wenbluo@lvsdpeet1001 shell_book_practice]$ _
```

```
ubuntu
[wenbluo@lvsdpeet1001 shell_book_practice]$ cat file2 | paste -d ',' file1 -
;Linux ,suse
;Unxia ,fedora
;solar,centos
;hupux,oel
;window,ubuntu
;
```

2. Storage -Net app ,Disk

```
[wenbluo@lvsdpeet1001 shell_book_practice]$ paste -d ',' file2 file1
suse,Linux
fedora ,Unxia
centos,solar
oel,hupux
ubuntu,window
,[wenbluo@lvsdpeet1001 shell_book_practice]$ _
```

### 3. 列转行

`paste -s file2`

```
[wenbluo@lvsdpeetl001 shell_book_practice]$ paste -d ',' file2 file1 > file3
[wenbluo@lvsdpeetl001 shell_book_practice]$ paste -s file3
suse,Linux fedora ,Unxia centos,solar oel,hupux ubuntu,window ,
[wenbluo@lvsdpeetl001 shell_book_practice]$ paste -s -d '||' file3
suse,Linux |fedora ,Unxia |centos,solar|oel,hupux|ubuntu,window|,
[wenbluo@lvsdpeetl001 shell_book_practice]$ _
```

行转列

```
suse,Linux |fedora ,Unxia |centos,solar|oel,hupux|ubuntu,window|,
[wenbluo@lvsdpeetl001 shell_book_practice]$ paste - - < file3
suse,Linux fedora ,Unxia
centos,solar oel,hupux
ubuntu,window ,
[wenbluo@lvsdpeetl001 shell_book_practice]$ _
```

4.

## H2 Email

Foxmail

<https://www.linuxdashen.com/ubuntu%E6%90%AD%E5%BB%BA%E7%AE%80%E6%98%93postfix%E9%82%AE%E7%AE%B1%E6%9C%8D%E5%8A%A1%E5%99%A8>

<https://www.linuxbabe.com/mail-server/setup-basic-postfix-mail-sever-ubuntu-14-04>

- sudo apt-get install mailutils 下载 [postfix](#) STMP  
sudo apt autoremove mailutils :删除
- FQDN
  1. [FQDN + 在Linux系统中查看IP地址](#)
    - 查看ip 通过 ip addr show
    - or download tools : sudo apt install net-tools & ifconfig
  - 2.

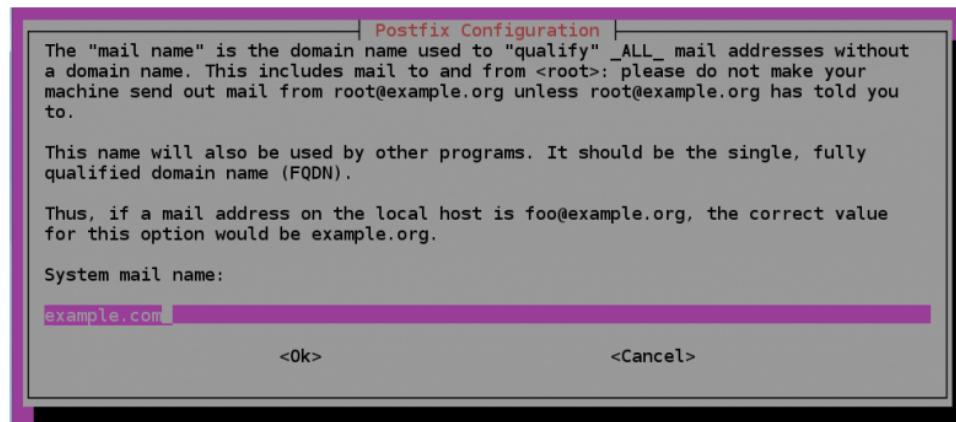
目前是通过mailx 发送邮件

- 第三方： 163 发送邮箱 : smtp : mailrc
- 自己搭建邮箱系统发送邮件 :postfix

[postfix配置基本信息](#)

- vi /etc/postfix/main.cf
  - using :[sudo dpkg-reconfigure postfix](#) to reconfigured postfix
  - using :[sudo service postfix restart](#) to restart postfix
- hostname:

On the ‘**System mail name**’ field, enter your domain name (e.g. example.com NOT ‘mail.example.com’)



- vi /etc/aliases

- ```

smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
myhostname = L-SHC-16505239.corp.ebay.com
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
mydomain = corp.ebay.com
myorigin = /etc/mailname
#mydestination = $myhostname, L-SHC-16505239.corp.ebay.com, localhost.corp.ebay.com, localhost
mydestination = L-SHC-16505239.corp.ebay.com, localhost.corp.ebay.com, localhost
relayhost =
mynetworks = 127.0.0.0/8 [:ffff:127.0.0.0]/104 [:1]/128 ,10.224.70.42
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = localhost
inet_protocols = ipv4
"/etc/postfix/main.cf" 52L, 1650C

```

H3 postfix 指令

1. commands

- mailq :邮箱队列 :正在排队发送的邮件id
- sudo postsuper -d ALL :清空队列

```
mailq | grep wenbluo@corp.ebay.com | awk '{print $1}' |sudo postsuper -d -
:删除某个queue id : 具体某个邮件 : 注意最末尾 - 不可以少
```

•

1:用什么邮箱发送? 2: 邮箱user & pwd

- mailx -s "JOB_ID: \${JOB_ID} RUN SUCCESSFULLY!" DL-eBay-MPT-IMD-CLS-FD-SAE@ebay.com;
- echo 'hello world' | mail -s 'hello robert' wenbluo@ebay.com
- mail -s 'hello weekly_sc_cored' wenbluo@ebay.com <
\$HOME/etl_home/cronlog/dw_clsfd.cls_cronjob.\$(date -d yesterday +%Y%m%d).log

#将文件 xxx.log , 发送出来

| Arguments | Comments |
|-----------|----------|
| -s | subject |

| Arguments | Comments |
|-----------|----------|
| -c | 抄送 |
| -b | 秘抄 |
| -a | 附件 |
| | |
| | |
| | |

H3 Sample

Fri 9/6/2019 2:28 PM
Wenbin\> <wenbluo@lvsdpeetl001.lvs.ebay.com>
attachment a file
To ● Luo, Robert(AWF)
Cc ● Luo, Robert(AWF)
while_1.sh 348 bytes

```
#!/bin/bash
a=3
echo $a
while [ $a -le 4 ];
do
echo 'hello world'
sleep 5
done
```

mail -s 'attachment a file ' -a while_1.sh -b
wenbluo@ebay.com -c wenbluo@ebay.com
wenbluo@ebay.com <while_1.sh>

Reply Reply All Forward IM
Fri 9/6/2019 2:25 PM
Wenbin\> <wenbluo@lvsdpeetl001.lvs.ebay.com>
attachment a file
To ○ while_1.sh@lvsdpeetl001.lvs.ebay.com; ○ -a@lvsdpeetl001.lvs.ebay.com; ● Luo, Robert(AWF)

```
#!/bin/bash
a=3
echo $a
while [ $a -le 4 ];
do
echo 'hello world'
sleep 5
done
```

mail -s 'attachment a file ' wenbluo@ebay.com -a
while_1.sh < while_1.sh

summary :

1. mail 最后一个参数是 收件人,多个收件人用 空格 表示
2. 关键字 放在最后参数之前, 不然最终结果跟图二一样, 当作收件人..

- 发送html 格式邮件

•

H2 Diff

H2 Read & Write

- cat date_range.cfg | tr '#' '' | read START_DT END_DT

-->读文件：并将参数传给 START_DT END_DT

```
cat date_range.cfg | tr '#' '' | read date_1 date_2
[wenbluo@phxdpeetl019 cfg]$ echo $date_1

[wenbluo@phxdpeetl019 cfg]$ echo $date_2
```

###困惑怎么没有数据? date_1 /date_2?

```
###因为bash 不支持，强制转化为ksh 就能执行
$ /bin/ksh
###先进入ksh 环境
$ cat date_range.cfg|tr '#' '' |read date_1 date_2
$ echo $date_1
20190407
$ echo $date_2
20190504
$ exit
###返回到 /bin/bash 环境
```

- \$ read name

```
wenbluo ###等待从键盘输入
[wenbluo@phxdpeetl019 test]$ echo $name
wenbluo
```

```
$ read -p 'enter your name: ' name_1 name_2 name_3
### -p : parameter :定义参数
enter your name: 1 2 3
[wenbluo@phxdpeetl019 test]$ echo $name_1
1
[wenbluo@phxdpeetl019 test]$ echo $name_2
2
[wenbluo@phxdpeetl019 test]$ echo $name_3
3
[wenbluo@phxdpeetl019 test]$ echo $name_1 $name_2 $name_3
1 2 3
```

```
$ read -t5 -p 'enter you name ,you have 5 seconds:' name_5
enter you name ,you have 5 seconds:[wenbluo@phxdpeetl019
test]$ echo $name_5
### read -t :提供了时间限制，计时输入
[wenbluo@phxdpeetl019 test]$ read -t5 -p 'enter you name ,you
have 5 seconds:' name_5
enter you name ,you have 5 seconds:hi world
[wenbluo@phxdpeetl019 test]$ echo $name_5
hi world
```

H2 Export

Background:

1. Set export attribute for shell variables. 父shell 与 子shell, 继承父shell 所有参数以及属性;
2. 一个shell 中的系统变量会被复制到子shell 中 (用export 定义的变量)
3. 一个shell 中的系统环境变量只对该shell 或者它的子shell 有效, 该shell 结束时变量消失 (并不能返回到父shell 中) 脚本执行完后该子shell 自动退出;

```
vi explicit.sh
a='i am apple'
b='i am banana'
c='i am cherry'
export a
##export b
export c
bash
```

```
a='robert'
[wenbluo@phxdpeet1019 test]$ bash explicit.sh
[wenbluo@phxdpeet1019 test]$ echo $a
i am apple
[wenbluo@phxdpeet1019 test]$ exit
exit ##强制退出explicit.sh
[wenbluo@phxdpeet1019 test]$ echo $a
robert
```

```
[wenbluo@phxdpeet1019 test]$ bash explicit.sh
[wenbluo@phxdpeet1019 test]$ echo $a
i am apple
[wenbluo@phxdpeet1019 test]$ echo $c
i am cherry
[wenbluo@phxdpeet1019 test]$ echo $b
```

####说明shell 结束时候, 变量消失! , 除非显示export !

4. exit命令 显示退出当前shell ;

-->类似css 继承

- shell 仅仅是一个程序，类似windows一样是一个应用程序，(excel),运行多个命令就好比打开多个excel文档
- 每一个shell 建立的变量都是局部变量

```
[wenbluo@phxdpeet1019 test]$ bash parent.sh
[wenbluo@phxdpeet1019 test]$ bash child.sh
hello wenbluo

#less parent.sh
a='wbluo' ### 也可以写成 export a ='wbluo'
b='robert'
export a

#less child.sh
echo $a
echo $b
```

- 子shell 不能更改父shell 参数

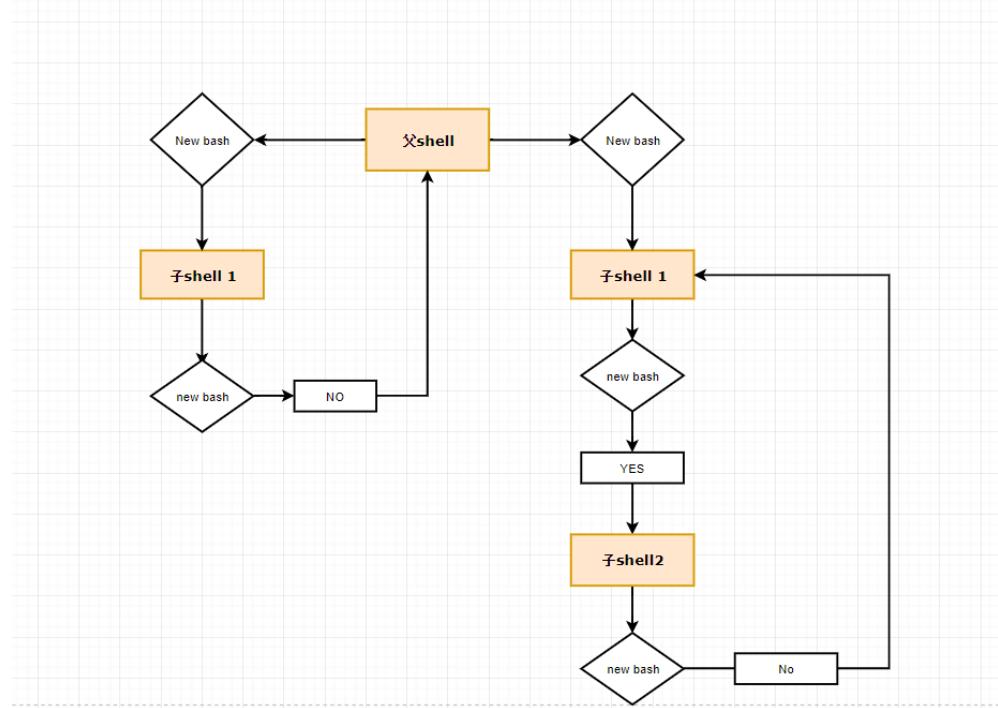
```
[wenbluo@phxdpeet1019 test]$ export myglobal=10
[wenbluo@phxdpeet1019 test]$ bash var4.sh
myglobal=33
[wenbluo@phxdpeet1019 test]$ echo $myglobal
10

#less var4.sh
#!/bin/bash
myglobal=33
echo myglobal=$mgloba
```

- 如何传递局部变量呢？因为最终会返回到开机shell程序...

shell 与 export命令

用户登录到Linux系统后，系统将启动一个用户shell。在这个shell中，可以使用shell命令或声明变量，也可以创建并运行shell脚本程序。运行shell脚本程序时，系统将创建一个子shell。此时，系统中将有两个shell，一个是登录时系统启动的shell，另一个是系统为运行脚本程序创建的shell。当一个脚本程序运行完毕，脚本shell将终止，返回到执行该脚本之前的shell。



为了能让局部变量能够传递下去，需要在子shell 中再开启一个子shell，那么会子shell2 退出，则是返回到shell 1 而不是父shell

```

less parent.sh
a='wbluo'
b='robert'
c='wenbluo'
echo $a
export c
export a
####没有开启新的shell

```

```

parent_2.sh
#!/bin/bash
a='wbluo'
b='robert'
c='wenbluo'
echo $a
export a
export c
echo $c
bash ##---开启新的shell (子shell)

```

```

less chid_2.sh
#!/bin/bash
echo $a
echo $b
echo $c

```

```

[wenbluo@phxdpeetl019 test]$ bash parent.sh
wbluo
[wenbluo@phxdpeetl019 test]$ bash chid_2.sh

```

```
[wenbluo@phxdpeet1019 test]$ bash parent_2.sh
wbluo
wenbluo
[wenbluo@phxdpeet1019 test]$ bash chid_2.sh
wbluo

wenbluo
```

- ksh 与 sh 的区别

1. 00 20 * * * /home/chewu/etl_home/bin/crontab.wrapper.ksh
/export/home/chewu/etl_home/bin/dw_clsf.job_runner_v2.ksh wenbluo_test >
/dev/null 2>&1

H2 WGET

it's download tools , just like :迅雷

- for ubuntu using

```
sudo apt update
sudo apt install wget
```

--> what is apt ? 跟python pip 一样，是包管理系统。

The *apt* command is a powerful command-line tool, which works with Ubuntu's *Advanced Packaging Tool* (APT) performing such functions as installation of new software packages, upgrade of existing software packages, updating of the package list index, and even upgrading the entire Ubuntu system.

-->apt & apt-get 的区别:

简单来说就是: apt = apt-get、apt-cache 和 apt-config 中最常用命令选项的集合。

[apt-vs-apt-get](#)

- for centos

```
sudo yum install wget
```

--> what is yum ?

-

一般来说著名的linux系统基本上分两大类:

- RedHat系列: Redhat、Centos、Fedora等
- Debian系列: Debiar、Ubuntu等

RedHat 系列

1. 常见的安装包格式 rpm包, 安装rpm包的命令是 "rpm -参数"
2. 包管理工具 yum
3. 支持tar包

Debian系列

1. 常见的安装包格式 deb包, 安装deb包的命令是 "dpkg -参数"
2. 包管理工具 apt-get
3. 支持tar包

yum可以用于运作rpm包, 例如在Fedora系统上对某个软件的管理:

1. 安装: yum install
2. 卸载: yum remove
3. 更新: yum update

apt-get可以用于运作deb包, 例如在Ubuntu系统上对某个软件的管理:

1. 安装: apt-get install
2. 卸载: apt-get remove

- examples

[Downloading files recursively.](#)

- sudo apt upgrade :更新软件

•

H3 Important args

wget --help

- wget -b xxx :后台下载
 - wget -c xxx :断点继续下载 / continue
 - wget -t xxxx : wget --tries=10 xxx : 表示retry event if connection is refused
 - wget --user-agent xxx : using agent to download
1. how to check whether i use the proxy or not ?
 - 2.

H2 SFTP/SCP/WSCP

- 跟ssh用法一样，先connet远程机子

```
sftp> exit  
[wenbluo@lvsdpeetl005 test]$ sftp yuebli@lvsdpeetl001.lvs.ebay.com;  
Connecting to lvsdpeetl001.lvs.ebay.com...  
sftp>
```

通过 help 指令，可以查看有哪些cmd

```
Available commands:  
bye                                Quit sftp  
cd path                             Change remote directory to 'path'  
chgrp grp path                     Change group of file 'path' to 'grp'  
chmod mode path                     Change permissions of file 'path' to 'mode'  
chown own path                      Change owner of file 'path' to 'own'  
df [-hi] [path]                      Display statistics for current directory or  
                                      filesystem containing 'path'  
exit                                Quit sftp  
get [-P] remote-path [local-path]   Download file
```

- 通过get指令，将远程的file cp 到local： ll命令

```
!ll /home/yuebli/etl_home/tmp/td2/dw_clsfld_monthly_user_sc_be_extract.2019-07-06.t  
sftp> get dw_clsfld.clsfd_monthly_user_sc_be_extract.2019-07-06.t  
Fetching /home/yuebli/etl_home/tmp/td2/dw_clsfld.clsfd_monthly_user_sc_be_extract.2019-07-06.t to dw_clsfld.clsfd_monthly_user_sc_be_extract.2019-07-06.t  
/home/yuebli/etl_home/tmp/td2/dw_clsfld.clsfd_monthly_user_sc_be_extract.2019-07-06.t 100% 207 0.2KB/s 00:00  
sftp> ll  
arg_2.sh      date_range.cfg  
arg.sh        dd.txt  
case_test_2   dir_1  
case_test.txt dispfile  
child_2.sh    dw_clsfld.clsfd_monthly_user_sc_be_extract.2019-07-06.t  
child_2.sh    explicit  
child.sh      explicit.sh  
child.txt     jobs_1  
jobs_2         partn_2.sh    wbluo_1.txt  
log.txt       reorient.txt  wbluo_2.txt  
nohup.out     sed_1.txt    wbluo_3.txt  
number        sed_2.txt    while_test  
parent_2.sh   to          parent.sh  vartest4  
parent.sh     vartest5  
parent.txt    vartest5  
parnt_2.sh   vi_indent.txt
```

- 通过put指令，将local file upload remote server

```
sftp> put number  
Uploading number to /home/yuebli/etl_home/tmp/td2/number  
number  
sftp>
```

- scp

```
scp ./while_test  
wenbluo@phxdpeetl019.phx.ebay.com:/home/wenbluo/etl_home/sql_test/test/while_test  
wenbluo@phxdpeetl019.phx.ebay.com's password:
```

1. scp 如何做到免密操作？

同样将src 的id_rsa.pub cp 到 destin 的authorized_keys 中

2.

- df -h 查看空间容量

```
exit  
sftp> df -h  
      Size      Used      Avail      (root)      %Capacity  
      2.4TB     638GB     1.8TB     1.8TB      25%  
sftp> _
```

H2 Awk

[tutorial](#)

H3 重要参数

| Argument | comment |
|----------|----------------------|
| -F | 决定分隔符 (Fields : IFS) |
| -f | 执行awk 文件 |
| -v | 定义变量 |
| ~ /!~ | 匹配or 不匹配 |

原理：

awk '{print}' info.txt :

将info.txt 每一行记录 按顺序执行代码块 --{xxx}, 此时代码块的内容只有一个print 函数

--》 等价于：

awk '{print \$0}' info.txt

awk 与 print 函数 结合使用，并不是使用echo

H3 NF&NR&FNR

1. NF&NR&FNR

NF : number of fields

目前记录被分割的字段数

```
apps
[wenbluo@lvsdpeetl001 shell_book_practice]$ cat info.txt
Linux - SystemAdminin
Database - Oracle , MySql etc
Security -FireWall ,Network ,Online Security etc.
Cool - Website
Storage -Net app ,Disk
[wenbluo@lvsdpeetl001 shell_book_practice]$ awk '{print "NF=""NF"\t"$NF }' info.txt
NF=3      SystemAdminin
NF=6      etc
NF=6      etc.          默认是以 空格 or\t 制表符 作为分隔符
NF=3      Website
NF=4      ,Disk
[wenbluo@lvsdpeetl001 shell_book_practice]$
```

-->指定分隔符

```
[wenbluo@lvsdpeetl001 shell_book_practice]$ awk [-F'-'] '{print $2 }' info.txt
SystemAdminin
Oracle , MySQL etc
FireWall ,Network ,Online Security etc.
Website
Net app ,Disk
[wenbluo@lvsdpeetl001 shell_book_practice]$
```

NR: number of records

在awk处理多个输入文件的时候，在处理完第一个文件后，NR并不会从1开始，而是继续累加，因此就出现了FNR，每当处理一个新文件的时候，FNR就从1开始计数，FNR可以理解为File Number of Record

FNR: number of rows of fields

2. 合并文件，并打印出行号

```
awk '{print NR, $0}' class class2
1 zhaoyun 87 82
2 gunyu 22 19
3 liubei 80 98
4 caocao 12 12
5 guojia 99 88 61
```

##\$0 ,打印出当前记录的内容。

打印最后一列数据

```
5 guojia 99 88 61
[wenbluo@lvsdpeetl001 shell_book_practice]$ awk '{print NR, $0}' class class2 | awk '{print "NF="NF,$NF}'
NF=4 82
NF=4 19
NF=4 98
NF=4 12
NF=5 61
[wenbluo@lvsdpeetl001 shell_book_practice]$
```

H3 正则表达式

正则表达式必须放在斜杠内

```
1. [wenbluo@lvsdpeetl001 shell_book_practice]$ awk '/Oracle/{print $0}' info.txt
Database - Oracle , MySQL etc
[wenbluo@lvsdpeetl001 shell_book_practice]$
```

grep -i work xxx.file

可以用：

awk /work/ xxx.file

```
2. awk:           ^ syntax error
[wenbluo@lvsdpeetl001 shell_book_practice]$ awk 'BEGIN {FS="-"} $2 ~ "etc" {print $1}' info.txt
Linux
Database
Security
Cool
Storage
[wenbluo@lvsdpeetl001 shell_book_practice]$
```

###与上面等价

```
awk 'BEGIN {FS="-"} {if {$2 ~ "etc"} {print $1}}' info.txt
```

先指定： 分割符是 - : 然后匹配第二列 =='etc' 的行记录，最终打印第一列

BEGIN 块 awk:允许你定义一个BEGIN 块，表示在开始处理输入文件中的文本前
执行一些初始话代码

H3 布尔逻辑变量

&&: 逻辑与 || :逻辑或

```

awk 'BEGIN {FS="-"} $2 ~ "Network|network" || $2 ~ "Disk" {print
$2}' info.txt
FireWall ,Network ,Online Security etc.
network, microsoft
Net app ,Disk
###查看第二列 为disk Or network

```

- ```

i find the file
[03:30:46]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice grep -il table * && echo "i find the file"
table_test.txt
i find the file
[03:30:51]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice

```

只有当command 1 【grep -il table \* 执行成功】，才再运行echo

```

-rw-rw-r-- 1 wenbluo wenbluo 145 Sep 6 00:07 table_test.txt
[03:33:31]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice grep -il world * && echo 'i find the file'
[03:34:04]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice echo $?
[03:34:08]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice

```

### H3 IF & WHILE & FOR

- ```

[wenbluo@lvsdpeetl001 shell_book_practice]$ awk '{for (i=1;i<=4;i++) print $i}' class2
caocao
12
12
guojia
99
88
61

```

H3 QA

- 如何输出hello world ,已经info.txt 行数，输出多个hello world

```

~/etl_home/sql_test/test/shell_book_practice awk -F '-' '{echo hello world}' info.txt
~/etl_home/sql_test/test/shell_book_practice awk -F '-' 'hello world' info.txt
~/etl_home/sql_test/test/shell_book_practice awk -F '-' 'hello world' <info.txt
~/etl_home/sql_test/test/shell_book_practice awk -F '-' '{echo "hello world"}' <info.txt
~/etl_home/sql_test/test/shell_book_practice _

[20:05:47]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice awk -F '-' '{print 12}' info.txt
12
12
12
12
12

```

- echo & print 区别

```

[20:13:40]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice awk -F '-' "{echo $var}" info.txt
[20:13:51]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice awk -F '-' "{echo $0}" info.txt
[20:15:26]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice awk -F '-' "{print $0}" info.txt
0
0
0
0
0
0
[20:15:31]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice awk -F '-' '{print $0}' info.txt
Linux - SystemAdmin
Database - Oracle , MySql etc
Security - Firewall , Network , Online Security etc.
Cool - Website
Windows - network, microsoft
Storage - Net app , Disk
[20:15:40]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice _

```

- [wenbluo@lvsdpeetl001 shell_book_practice]\$ awk -F '-' '{ print \$2 }' < info.txt


```
SystemAdminin
      Oracle , MySql etc
      FireWall ,Network ,Online Security etc.
      Website
      Net app ,Disk
```
- [wenbluo@lvsdpeetl001 shell_book_practice]\$ awk -F '-' "{ print \$2 }" < info.txt


```
Linux - SystemAdminin
      Database - Oracle , MySql etc
      Security -FireWall ,Network ,Online Security etc.
      Cool - Website
      Storage -Net app ,Disk
```
- [wenbluo@lvsdpeetl001 shell_book_practice]\$

如何输出更加整洁？ 去掉多余的空格、缩进？

H1 Sudo

- sudob_clsf &sudodw_adm
- sudo 默认到root user
- sudo yuebli xxx : sudo 到yuebli权限
-

H3 QA

1. sudob_clsf 与 ssh yuebli@lvsdpeetl001.lvs.ebay.com 有什么区别？
- 2.

H1 Mount

[tutorial](#)

background

shell /unix 系统只有一个根目录 /

并不像window 系统，有C:D:E:F:G etc 分区硬盘

因此，如果要读取根目录之外的文件，这个时候需要挂载，目录即为“挂载点”。

总而言之，mount挂载的作用，就是将一个设备（通常是存储设备）挂接到一个已存在的目录上。

- 如何查看挂载了那些设备？

cat /etc/mtab

- sudo mount /dev/vdc /mnt/shared
- sudo mount /dev/vdd /opt/data
- sudo mount /dev/vde /export/home
-

Define Function

[date_add](#)

H1 Crontab

[Crontab](#)

1. Linux下的任务调度分为两类：系统任务调度和用户任务调度

2. minute hour day month week command

其中：

minute：表示分钟，可以是从0到59之间的任何整数。

hour：表示小时，可以是从0到23之间的任何整数。

day：表示日期，可以是从1到31之间的任何整数。

month：表示月份，可以是从1到12之间的任何整数。

week：表示星期几，可以是从0到7之间的任何整数，这里的0或7代表星期日。

command：要执行的命令，可以是系统命令，也可以是自己编写的脚本文件。

| Crontab格式说明 | | | | | |
|--------------|--------------|--------------|--------------|-------------|----------------|
| * | * | * | * | * | command |
| 取值范围
0-59 | 取值范围
0-23 | 取值范围
1-31 | 取值范围
1-12 | 取值范围
0-7 | 需要执行的命令 |
| | | | | | 星期几 |
| | | | | | 月份 |
| | | | | | 几号 |
| | | | | | 小时 |
| | | | | | 分钟 |
| | | | | | OPSSers.org©羽飞 |

3.

除了数字还有几个特殊的符号就是 "*"、"/" 和 "-"、","：

*代表所有的取值范围内的数字;

"/"代表步进设置,"*/5"表示每5个单位;如0-59/2定义每两分钟执行一次;如0-12/2,的偶数点执行;

"-"代表从某个数字到某个数字

", "分开了几个离散的数字。

以下举几个例子说明问题：

```
1 | 0 6 * * * echo "Good morning." >> /tmp/test.txt #每天早上6点
2 | 0 */2 * * * echo "Have a break now." >> /tmp/test.txt 每两个小时
3 | 0 11 4 * 1-3 command line #每个月的4号和每个礼拜的礼拜一到礼拜三的早上11点
4 | 0 4 1 1 * command line #1月1日早上4点
```

拓展：“%”

- 特殊字符，相当于回车
-

4.

实例1：每1分钟执行一次command

命令：

* * * * * command

实例2：每小时的第3和第15分钟执行

命令：

3,15 * * * * command

实例3：在上午8点到11点的第3和第15分钟执行

命令：

3,15 8-11 * * * command

实例4：每隔两天的上午8点到11点的第3和第15分钟执行

命令：

3,15 8-11 */2 * * command

实例5：每个星期一的上午8点到11点的第3和第15分钟执行

命令：

3,15 8-11 * * 1 command

实例6：每晚的21:30重启cmh

```
00 01 10 * * /home/chewu/etl_home/bin/crontab.wrapper.ksh
/export/home/chewu/etl_home/bin/dw_clsfid.job_runner_v2.ksh
motor_sc_monthly > /dev/null 2>&1
```

###每月10号运行

5.

```

crontab [-u user] [-e | -l | -r]

2. 命令功能:
通过crontab命令, 我们可以在固定的间隔时间执行指定的系统指令或 shell script脚本。时间间隔的单位可以是分钟、小时、
3. 命令参数:
-u user: 用来设定某个用户的crontab服务, 例如, "-u ixdba"表示设定ixdba用户的crontab服务, 此参数一般有root用户来
file: file是命令文件的名字, 表示将file做为crontab的任务列表文件并载入crontab。如果在命令行中没有指定这个文件, crontab
-e: 编辑某个用户的crontab文件内容。如果不指定用户, 则表示编辑当前用户的crontab文件。
-l: 显示某个用户的crontab文件内容。如果不指定用户, 则表示显示当前用户的crontab文件内容。
-r: 从/var/spool/cron目录中删除某个用户的crontab文件, 如果不指定用户, 则默认删除当前用户的crontab文件。
-i: 在删除用户的crontab文件时给确认提示。

```

6. 执行文件:

```

crontab -l grep bsc_weekly_table_archive.sql

15 0 * * 0
/home/chewu/etl_home/bin/target_table_load_handler.ksh
dw_clsfid.x td2 bsc_weekly_table_archive.sql

###单独调度运行archieve.sql文件 , 该文件就是delete old_table
, insert table ;等dml操作;

00 20 * * * /home/chewu/etl_home/bin/crontab.wrapper.ksh
/export/home/chewu/etl_home/bin/dw_clsfid.job_runner_v2.ksh
wenbluo_test > /dev/null 2>&1

#####运行批处理文件, 调用crontab.wrapper.ksh

```

question :

- wenbluo_test 在哪个目录下?

/dw/etl/home/prod/land/dw_clsfid/job_runner ,是一个document

```

[wenbluo@phxdpeet1019 job_runner]$ cd wenbluo_test
[wenbluo@phxdpeet1019 wenbluo_test]$ ll
total 12
drwxrwxr-x 2 wenbluo wenbluo 4096 Mar 28 06:14 cfg
drwxrwxr-x 2 wenbluo wenbluo 4096 Mar 28 01:23 dat
-rw-rw-r-- 1 wenbluo wenbluo 142 Mar 28 01:49
job_steps.tmp

```

- crontab** 有两种调度方式: 一种是**tagert_table_load_handler.ksh** 一种是**crontab.wrapper.ksh and job_runner_v2.ksh**

7. 注释

```

      * date >>~/date.txt
[wenbluo@phxdpeet1019 ~]$ crontab -e
* * * * 1 $date >> ~/date.txt
* * * * 1 echo 'hello world' >> ~/world.txt
21 23 1 1 1 echo date >>~/test.txt
* * * * 1 date >>~/test.txt

```

如何全部注释掉

- 通过添加 #
- %s /^#/g ;

8.

H1 Special Argument

| 变量 | 含义 |
|------|---|
| \$0 | 当前脚本的文件名或开机启动运行的那个版本shell |
| \$n | 传递给脚本或函数的参数。n 是一个数字，表示第几个参数。例如，第一个参数是\$1，第二个参数是\$2。 |
| \$# | 传递给脚本或函数的参数个数。 |
| \$* | 传递给脚本或函数的所有参数。 |
| \$@ | 传递给脚本或函数的所有参数。被双引号(" ")包含时，与 \$* 稍有不同，下面将会讲到。 |
| \$? | 上个命令的退出状态，或函数的返回值。 |
| \$\$ | 当前Shell进程ID。对于 Shell 脚本，就是这些脚本所在的进程ID。 |
| \$_ | <p>前一个命令最后一个参数</p> <pre>they are different [wenbluo@lvsdpeetl001 test]\$ if test \$num == \$num2; then echo 'they are same'; else echo "they are different"; fi they are different [wenbluo@lvsdpeetl001 test]\$ if test -e xv_test.sh ; then echo "the file \$_ exists" ;else echo "the file \$_ vanish ";fi the file xv_test.sh exists [wenbluo@lvsdpeetl001 test]\$ _</pre> <p>f</p> <pre>./while_1.sh [wenbluo@lvsdpeetl001 test]\$ echo 1 2 3 1 2 3 [wenbluo@lvsdpeetl001 test]\$ echo \$_ 3</pre> |
| !! | <p>6.非常实用的 !! 操作符</p> <p>你可以使用(!!)来运行或者改变之前的命令。它会调用最近使用的命令来调整当前命令，给大家展示一下使用场景。</p> <p>昨天我运行了一个获取IP的Shell命令：</p> <pre>\$ ip addr show grep inet grep -v 'inet6' grep -v '127.0.0.1' awk '{print \$2}' cut -f 1 -d/</pre> <p>突然我意识到需要将结果重定向到 ip.txt 中，这时你应该想到用“UP”键恢复上一个命令再加上 '>ip.txt' 命令来重定向进去：</p> <pre>\$ ip addr show grep inet grep -v 'inet6' grep -v '127.0.0.1' awk '{print \$2}' cut -f 1 -d/ > ip.txt</pre> <p>感谢这次救命的“UP”键。那么再考虑下这个场景，如果我需要运行下面的这个脚本：</p> |

| 变量 | 含义 |
|-----|--|
| IFS | <p>internal field separator :分隔符号</p> <pre>[03:07:19]wenbluo@lvsdpeetl001 ~ echo \$text aa bb xx [03:07:24]wenbluo@lvsdpeetl001 ~ IFS=' ' [03:07:36]wenbluo@lvsdpeetl001 ~ echo \$text aa bb xx [03:07:48]wenbluo@lvsdpeetl001 ~ _</pre> |

- .\$\$的运用

表示进程id号，其实shell也是一个进程，用来控制底层 unix 环境or shell 环境的程序

如何将dir_26 目录下的文件直接搬到dir_25 目录下，并且删除dir_26 目录？

```
[wenbluo@lvsdpeetl001 dir_26]$ ls
while_test while_test_2
[wenbluo@lvsdpeetl001 dir_26]$ ls -al
total 12
drwxrwxr-x 2 wenbluo wenbluo 4096 Aug 21 01:50 .
drwxrwxr-x 3 wenbluo wenbluo 4096 Aug 21 01:50 ..
-rw-rw-r-- 1 wenbluo wenbluo 251 Jun 19 03:32 while_test
-rw-rw-r-- 1 wenbluo wenbluo 0 Aug 21 01:46 while_test_2
[wenbluo@lvsdpeetl001 dir_26]$ pwd
/home/wenbluo/etl_home/sql_test/test/dir_25/dir_26
[wenbluo@lvsdpeetl001 dir_26]$ mv * /tmp/tmp$$/
[wenbluo@lvsdpeetl001 dir_26]$ ls /tmp/tmp$$/
while_test while_test_2
[wenbluo@lvsdpeetl001 dir_26]$ cd ..
[wenbluo@lvsdpeetl001 dir_25]$ rm -r dir_26
[wenbluo@lvsdpeetl001 dir_25]$ mv /tmp/tmp$$/* .
[wenbluo@lvsdpeetl001 dir_25]$ ls -l
total 4
-rw-rw-r-- 1 wenbluo wenbluo 251 Jun 19 03:32 while_test
-rw-rw-r-- 1 wenbluo wenbluo 0 Aug 21 01:46 while_test_2
[wenbluo@lvsdpeetl001 dir_25]$ pwd
/home/wenbluo/etl_home/sql_test/test/dir_25
[wenbluo@lvsdpeetl001 dir_25]$
```

- 首先将dir_26 目录下的文件，全部搬运到 /tmp /tmp\$\$ 目录下
- 最后再将 /tmp/tmp\$\$/* 所有文件 搬运到 dir_25 下

/tmp 目录特点就是，当前会话关闭后，就会自动删除所有文件。

watch out !

```
不能通过 mv /home/wenbluo/etl_home/sql_test/test/dir_25/dir_26
/home/wenbluo/etl_home/sql_test/test/dir_25/ 命名错误方式
```

- \$?

返回最近一次操作的返回码：0 表示correct 非0值表示error

```
cat arg.sh
#!/bin/bash
echo "filename:$0"
echo "First parameter :$1"
echo "seconde parameter :$2"
echo "quoted values :$*"
```

```

echo "quoted values:$@"
echo "total parameter number:$#"
echo 'hello boy'

$ ./arg.sh wenbluo robert
###直接怎么传参!
filename:./arg.sh
First parameter :wenbluo
seconde parameter :robert
quoted values :wenbluo robert
quoted values:wenbluo robert
total parameter number:2
hello boy

```

how to use \$@

```

#! /bin/bash
echo "number od arguments passed is $#"
for arg in "$@"
do
echo $arg
done

```

```

[wenbluo@phxdpeetl019 test]$ ./arg_2.sh 'a b' c
number o arguments passed is 2
a b
c

```

```

[03:10:15]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test  IFS='%'
[03:11:29]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test . arg.sh wenbluo robert
filename:-bash
First parameter :wenbluo
seconde parameter :robert
quoted values :wenbluo%robert
quoted values:wenbluo robert
total parameter number:2
hello boy
[03:11:31]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test _
```

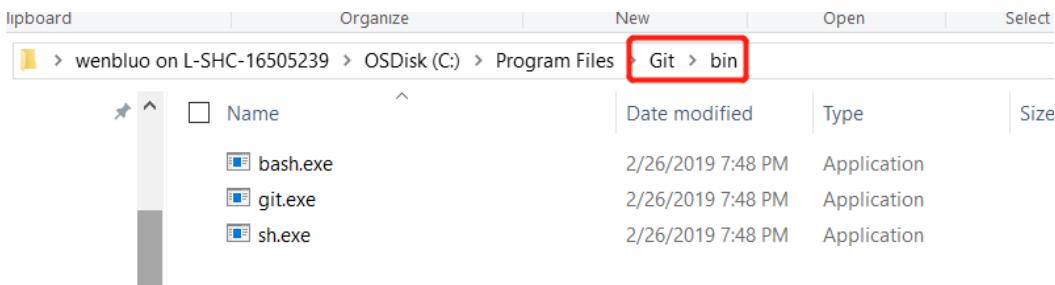
1. 可以看到 \$*, 变量之间是以 % : 等价于 "\$1 % \$2" : 其中 % 是依据 **IFS**
 2. 然后%@ : 表示变量之间是有间隔的, 等价于 "\$1" ,"\$2"
-

H1 Git/Github

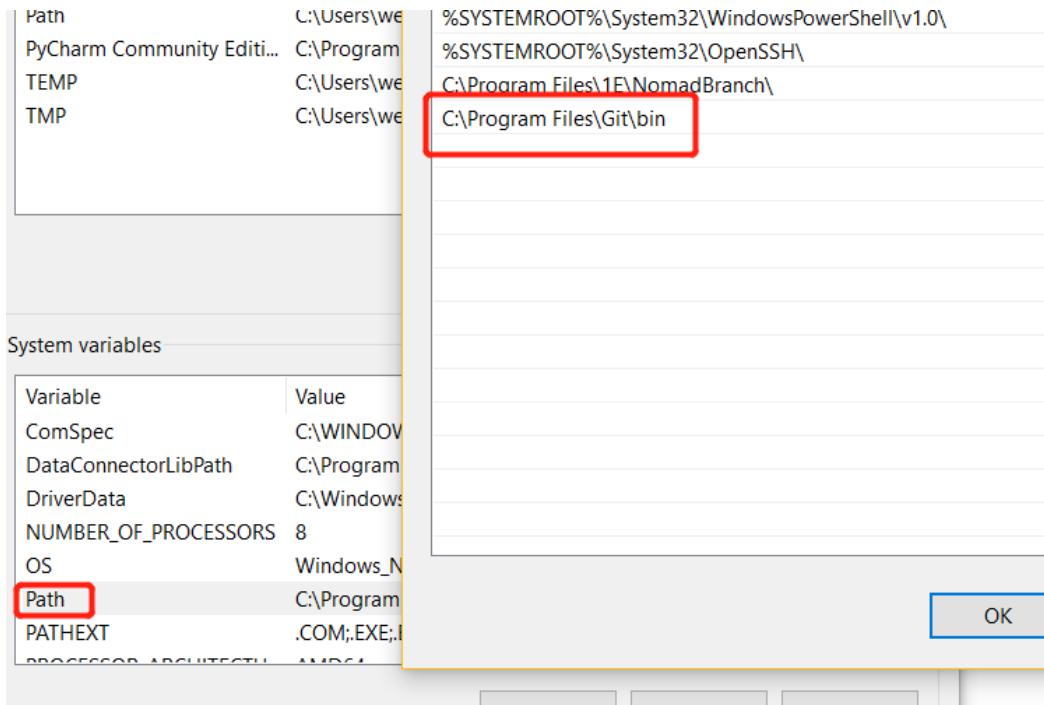
windows 运行shell脚本-->git

[git bash &github&git-bash 详细教程](#)

git: 目前世界上最先进的分布式**版本控制**系统.



将二进制文件放置到**path**环境参数中



- 打开本地repository (仓库//或者叫目录)的文件夹 -->这样所有文件都可以被git管理起来

每个文件的，每个文件的修改，删除，**Git**都能跟踪，以便任何时刻都可以追踪历史，或者在将来某个时刻还可以将文件”还原”。以便任何时刻都可以追踪历史，或者在将来某个时刻还可以将文件”还原”。

H2 Concept

1. pull/push

同步更新repository里的文件信息

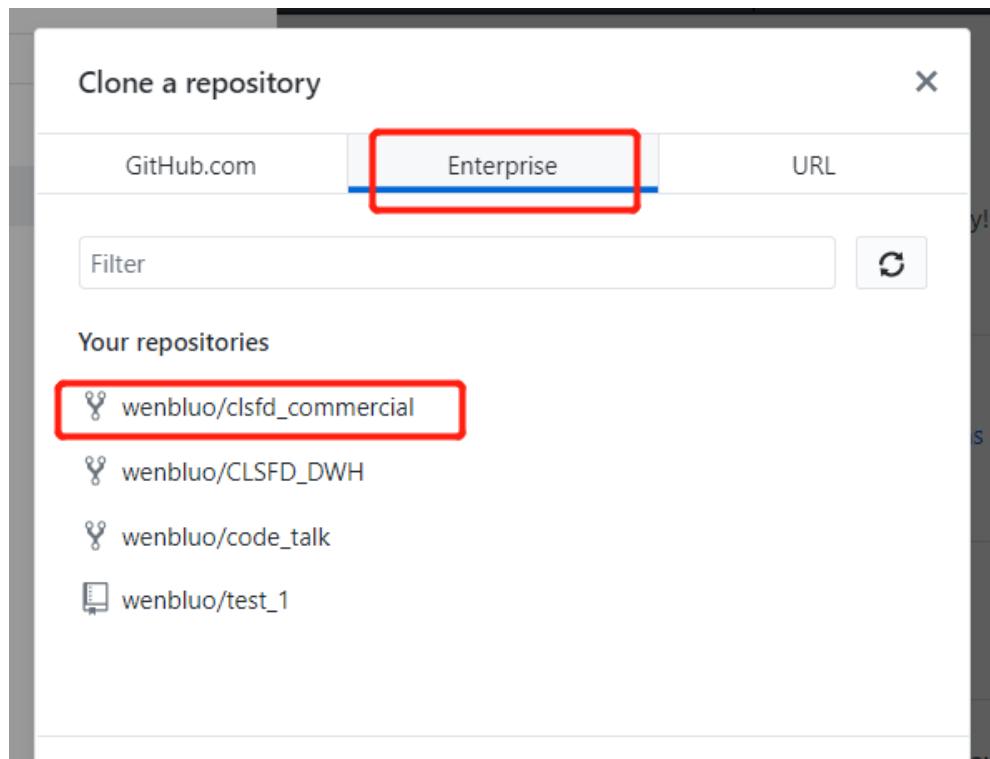
2. fork

完全赋值别人的repository 到自己repository中

3. star

收藏

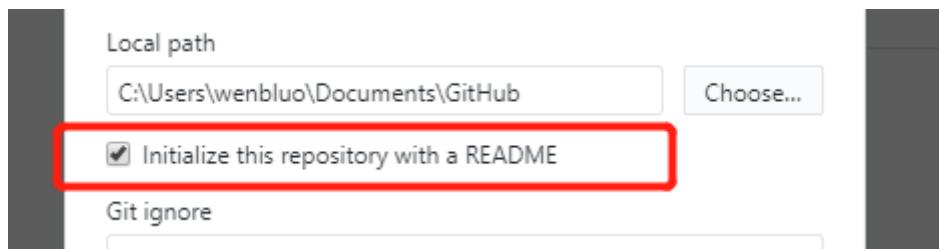
4. clone



将云端的repository 同步到本地

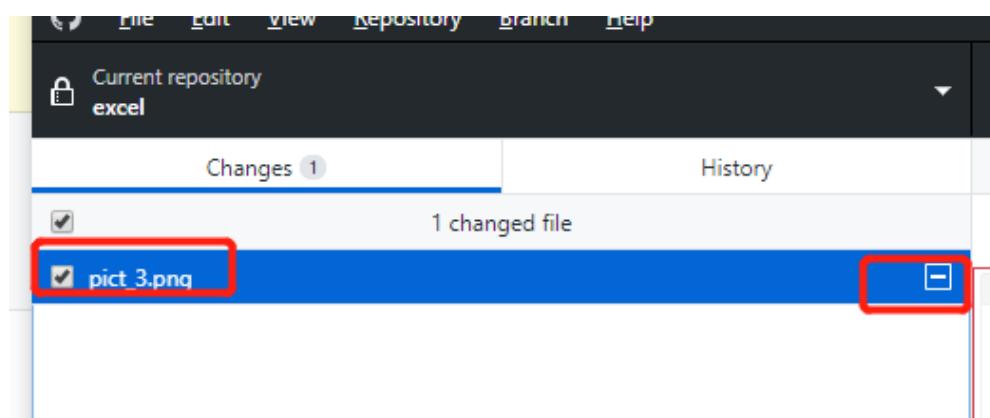
H2 QA

- 在每次创建新的 repository 时候，为什么总是要勾上 initialize this repository with a README ?



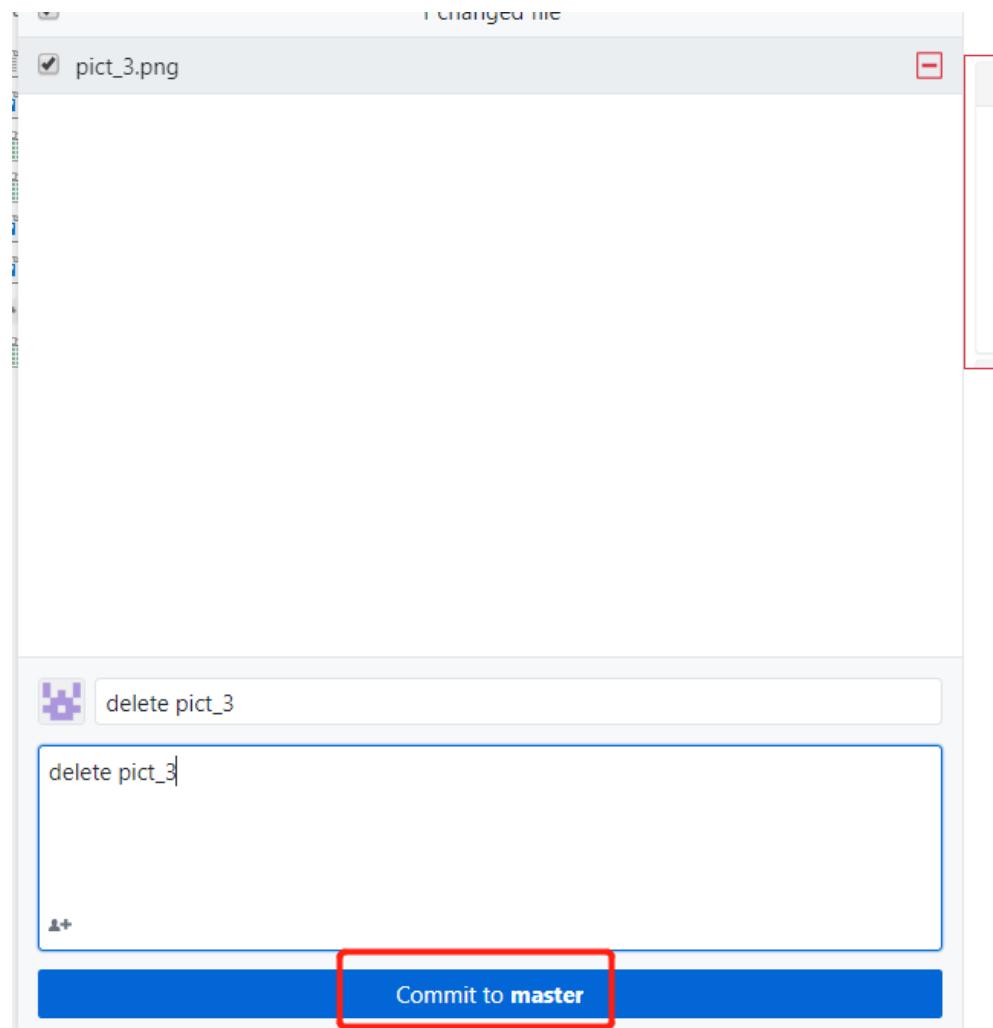
2. Github desktop

监控作用：当本地repository（目录）下的文件有变动的时候，会及时反馈在desktop界面，然后查看变更记录，仓库里的文件变动都会被github记录下来



对于变动不合理的，你可以点击右侧 - 符号，回滚到未变更之前的状态；

其次也可以先同步到本地，本机上编辑文件，然后sync到github网站上；



当同意该变更时候，我们commit 提交这次变更，那么仓库就会少了pict_3文件

No local changes

You have no uncommitted changes in your repository! Here are some friendly suggestions for what to do next.



Push 1 commit to the origin remote

You have one local commit waiting to be pushed to GitHub

Always available in the toolbar when there are local commits waiting to be pushed or

Ctrl P

Push origin

最后点击push ,那么云端就会同步本地repository,删掉pict_3文件;

3.

H1 Blog

[shell 学习的第一天](#)

H1 Practice

[10 mins to learn shell](#)

1.

H1 Printf & Echo

H2 ECHO

- **1、echo**

在shell中，内建（builtin）命令echo，格式如下：

```
1 echo [-neE] [arg ...]
```

echo命令用于输出各参数arg，参数间以空格分隔，结尾是个换行符。选项“-n”禁止输出结尾的换行符。对于一些反斜线“\”转义的特殊字符，在echo命令中默认不进行转义，选项“-e”启用转义，“-E”禁止转义。

下面是反斜线“\”转义的特殊字符。

| 1 | \a | 警告（响铃） | 登录后复制 |
|----|-------------|-----------------------|-------|
| 2 | \b | 退格删除 | |
| 3 | \c | 禁止继续输出 | |
| 4 | \e | 转义字符 | |
| 5 | \E | 转义字符 | |
| 6 | \f | 换页 | |
| 7 | \n | 新行 | |
| 8 | \r | 换行 | |
| 9 | \t | 水平制表符 | |
| 10 | \v | 垂直制表符 | |
| 11 | \\\ | 反斜线 | |
| 12 | \0nnn | 八进制数nnn表示的八位字符 | |
| 13 | \xHH | 十六进制数HH表示的八位字符 | |
| 14 | \uHHHH | 一到四个十六进制数表示的Unicode字符 | |
| 15 | \UHHHHHHHHH | 一到八个十六进制数表示的Unicode字符 | |

echo -e : 表示转义 echo -E :停止转义

application :

```
[03:12:22]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test echo -E '\a hello world'
\a hello world
[03:14:18]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test echo -e '\n hello world'

hello world
[03:14:21]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test echo -e '\t hello world'
    hello world
[03:14:25]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test echo -e '\\ hello world'
\ hello world
[03:14:30]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test echo -e '\r hello world'
hello world
[03:14:58]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test echo -e 'hello robert \r hello world'
hello world
[03:15:10]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test echo -e 'hello robert \n hello world'
hello robert
hello world
[03:15:20]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test
```

H1 Tips

1. Ctrl+u : 清空整行

2. Tab:自动补充

3. echo设置颜色

- 通过 echo -e 调用颜色
- 字体颜色后面有个m
- \033[:表示颜色提示符开始
- \003[m]:表示颜色提示符结束

```

字颜色: 30——37
echo -e "\033[30m 黑色字 \033[0m"
echo -e "\033[31m 红色字 \033[0m"
echo -e "\033[32m 绿色字 \033[0m"
echo -e "\033[33m 黄色字 \033[0m"
echo -e "\033[34m 蓝色字 \033[0m"
echo -e "\033[35m 紫色字 \033[0m"
echo -e "\033[36m 天蓝字 \033[0m"
echo -e "\033[37m 白色字 \033[0m"

字背景颜色范围: 40——47
echo -e "\033[40;37m 黑底白字 \033[0m"
echo -e "\033[41;37m 红底白字 \033[0m"
echo -e "\033[42;37m 绿底白字 \033[0m"
echo -e "\033[43;37m 黄底白字 \033[0m"
echo -e "\033[44;37m 蓝底白字 \033[0m"
echo -e "\033[45;37m 紫底白字 \033[0m"
echo -e "\033[46;37m 天蓝底白字 \033[0m"
echo -e "\033[47;30m 白底黑字 \033[0m"

```

最后面控制选项说明

\33[0m 关闭所有属性

\33[1m 设置高亮度

4. 修改bash提示符ps1 /ps2 ..

| arg | comment |
|-----|-----------|
| \u | username |
| \h | hostname |
| \w | 完全路径 |
| \t | date:当前时间 |
| | |
| | |

```
export PS1="\e[0;36m [\t]\u@\h \w \e[m "
```

- \e[:表示颜色提升符开始
- \e[m :表示颜色提示符结束
- x;ym:颜色格式 0; 36 --青色 0; 34--蓝色 0; 32--绿色 0; 31--红色

H1 Function

•

H1 Programming

#! :(shebang) 告知用哪个解析器去解析文本,

H2 Data type

H4 Array

array : 从1开始计数

```
[08:05:40]wenbluo@lvsdpeetl001 ~ cd etl_home/sql_test/test/  
[08:05:47]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test cd shell_book_practice/  
[08:05:49]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice arr[@]='hello world'  
[08:09:07]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice arr[1]='hello robert'  
[08:09:18]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice arr[2]='hello wenbluo'  
[08:09:24]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice echo ${arr[*]}  
1. hello world hello robert hello wenbluo  
[08:09:37]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice echo ${#arr[*]}  
3  
[08:09:47]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice echo $arr  
hello world  
[08:10:05]wenbluo@lvsdpeetl001 ~/etl_home/sql_test/test/shell_book_practice ls
```

H2 [] & [[]] & (0)

- `[]` : 是一个关键字

表 6.4 “[]” 和 “[]” 之间的不同的比较

| Feature | [] | [] | Example |
|-----------------------------|--------|-----------------|--|
| String comparison | > | > | [[a > b]] "a does not come before b"
[a > b] "a does not come before b" |
| | < | < | [[az < za]] && echo "az comes before za"
[az < za] && echo "az comes before za" |
| Expression grouping | (...) | \(... \) | [[\$var = img* && (\$var = *.png \$var = *.jpg)]] &&
echo "\$var starts with img and ends with .jpg or .png" |
| Pattern matching | = | (not available) | [[\$name = a*]] echo "name does not start with an 'a': \$name" |
| Regular Expression matching | =~ | (not available) | [[\$(date) =~ ^Fri\ ...\\ 13]] && echo "It's Friday the 13th!" |

H2 分号

reference

隔断每个语法关键字或命令

- 如果写成单行，则用 ; 区分代码块
- 如果写成多行，则用\n（换行符）区分代码块

```
-bash: [: missing ]  
[wenbluo@lvsdpeetl001 test]$ if [ $name != 0 ] ; then echo "hello world" ; fi  
hello world  
[wenbluo@lvsdpeetl001 test]$ if [ $name != 0 ]  
> then echo 'hello world'  
> fi  
hello world  
[wenbluo@lvsdpeetl001 test]$ _
```

H2 Character

- concatenate

```

value1='wenbluo'
[wenbluo@phxdpeet1019 bin]$ echo $value1
wenbluo
[wenbluo@phxdpeet1019 bin]$ echo ${value1}"-
wenbluo-
[wenbluo@phxdpeet1019 bin]$ echo ${value1}'1'
wenbluo1
[wenbluo@phxdpeet1019 bin]$ echo ${value1}'2344'
wenbluo2344
[wenbluo@phxdpeet1019 bin]$ valu2='robert'
[wenbluo@phxdpeet1019 bin]$ echo ${value1}${valu2}
wenbluorobert

```

- processing skills

1. \$ echo \$name
dw_clsfed.clsfd_daily_onsiterev.sql
\$ echo \${name%.*}
dw_clsfed.clsfd_daily_onsiterev
\$ echo \${name%%.*}
dw_clsfed

##%表示从右往左匹配
##.表示分割符
%%表示第几个分割符
* 表示被删除的字符串

2. \$ echo \${name#*.}
clsfd_daily_onsiterev.sql
\$ echo \${name##*.}
sql
#表示从左往右匹配
3. echo \${#name}
34
###求字符串长度

H2 Date

通过 date --help，可以查看帮助手册

1. 设置时间格式

```

date '+%Y%m%d-%H%M%S'
20190507-001840

```

###注意大小写

2. #获取昨天时间

```

date -d yesterday
Mon May 6 00:24:45 -07 2019
#获取前两天时间 : 当前时间减2d

```

```
date -d -2day
Mon May  6 00:28:08 -07 2019
#获取明天时间
date -d tomorrow
Wed May  8 00:28:23 -07 2019
#获取后天时间  :当前时间加2d
date -d 2day +'%Y%m%d'
20190509
```

3. 时间运算--本质将时间转化为时间戳形式

```
###此时的date 不再是指系统变量date , 此时是一种函数
###通过man date 可以查看字典
###date -d  : 表示display time described by string ,not 'now'
###

date -d "$date" +%s
1557212400
date -d "2019-01-01" +%s
1546326000

###将时间戳转化为Y-M-D
date -d @1546326000 +'%Y%m%d'
20190101

###也可以转化为天数 ---通过man date 查看具体参数 是j or d or D
etc
date -d "$date" +%j
127
```

4.

H2 Wrapper

-

```

cat refresh_weekly_sc_core.ksh
#!/usr/bin/ksh

start_date=$1
end_date=$2

ksh /home/${whoami}/etl_home/bin/target_table_load_handler.ksh
dw_clsfid.step2_clsfid_daily_sc_core_mtrc td2
dw_clsfid.step2_clsfid_daily_sc_core_mtrc.sql
start_date=$start_date end_date=$end_date
ksh /home/${whoami}/etl_home/bin/target_table_load_handler.ksh
dw_clsfid.step3_clsfid_daily_sc_cluster_target td2
dw_clsfid.step3_clsfid_daily_sc_cluster_target.sql

ksh /home/${whoami}/etl_home/bin/target_table_load_handler.ksh
dw_clsfid.step2_clsfid_weekly_sc_core_mtrc td2
dw_clsfid.step2_clsfid_weekly_sc_core_mtrc.sql
start_date=$start_date end_date=$end_date
ksh /home/${whoami}/etl_home/bin/target_table_load_handler.ksh
dw_clsfid.step3_clsfid_weekly_sc_cluster_target td2
dw_clsfid.step3_clsfid_weekly_sc_cluster_target.sql

• refresh_weekly_sc_core.ksh 2019-01-01 2019-02-28

```

H2 Process controls

- if..then..fi
 - if..then..else..fi
- 如果为真 0 , 执行then 语句 , 并忽略else代码块 ,
如果为假(非0), 执行else代码块;

- if..then..elif..then..else..fi

```

a=10
b=20
if [ $a -gt $b ]
then
echo "$a -ge $b : a大于b"
elif [ $a -lt $b ]
then
echo "$a -lt $b : a小于b"
else
echo "$a -eq $b: a等于b"
fi

```

困惑一定要打分号?

```

55
56
57 if [ ! -e "$DATE_DIR/$RUN_KYLIN_ID" ]; then
58   echo "$DATE_DIR/$RUN_KYLIN_ID not exists"
59   mkdir -p $DATE_DIR/$RUN_KYLIN_ID
60 fi
61
62 if [ ! -f ${DATE_DIR}/${RUN_KYLIN_ID}.last_distro_tm.dat ]

```

；必不可少， 表示语句块的作用

```
[wenbluo@lvsdpeetl001 shell_book_practice]$ for file in `ls`  
> do  
> echo $file  
> done;  
checkDaysOfWeek_case_mode.sh  
checkDaysOfWeek.sh  
compareNumber.sh  
forlooplinux.sh  
simplenestedfor.sh  
[wenbluo@lvsdpeetl001 shell_book_practice]$ for file in `ls` ; do echo $file; done;
```

- for 循环：

格式： **for xx ;do xxxx done ;**

In 方法：

```
$ for loop in 1 12 3  
> do echo "the value is :$loop"  
> done  
the value is :1  
the value is :12  
the value is :3  
  
## for do done  
## 系统会先创建 创建一个变量 loop 然后赋值给loop;
```

C 方法： **for ((xxx;xxx;xxx))**

```
#!/bin/bash  
a=0  
b=0  
for (( $a; $a<=20; ))  
do  
let b=$b+$a  
let a=$a+1  
echo "a value is :$a , the cumulate value is :$b"  
sleep 2  
done
```

- while 循环：

格式： **while condition do done**

```
#!/bin/bash  
read -p "enter your num and name:" num name    ###不支持ksh指令  
while [ $num -lt 10 ]  
do  
#print($num)  
echo $num  
#$num+=1  
#$num=$num+1  
let num=$num+1
```

```

name=`expr $name + 1`
#((name+1))
sleep 2
# sleep(2) Error
echo $name
done

###while do done

```

1. 写无限循环: true or : 占位符

```

^C
[wenbluo@lvsdpeetl001 shell_book_practice]$ while :
> do
> echo "do nothing ,if you want to stop ,please enter CTRL+C"
> sleep 5
> done
do nothing ,if you want to stop ,please enter CTRL+C
do nothing ,if you want to stop ,please enter CTRL+C

```

2.

- Case

```

case &cntry in
'za') clsfd_site_list=201;;
'uk') clsfd_site_list=1011;;
'sg') clsfd_site_list=198;;
*) echo "cannnot decide a proper clsfd_site_id list for input
value cntry="
esac

```

- 除了*)模式，各个分支中`;;`是必须的，`;;`相当于其他语言中的`break`
- | 分割多个模式，相当于`or`

--> 拓展case ...esac

-

H1 Git



background :分布式版本控制系统

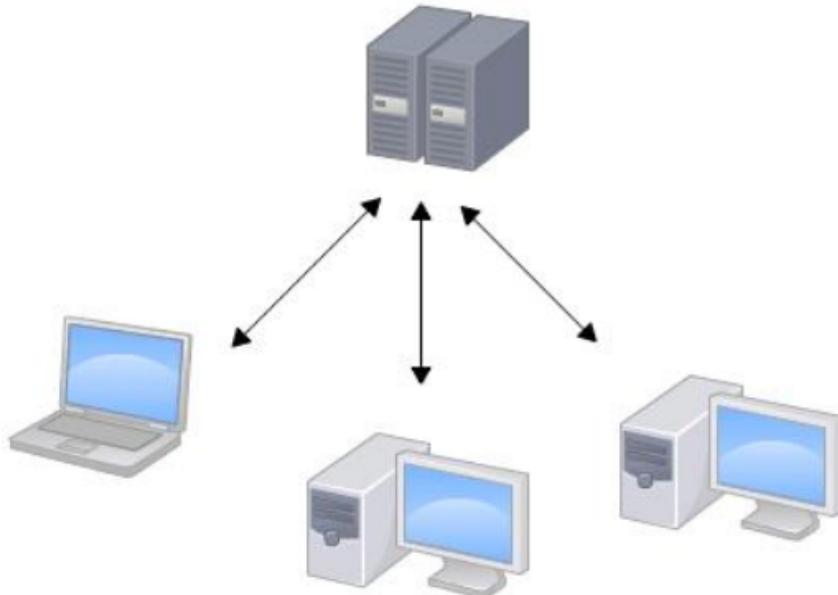
[Git命令笔记](#)

廖雪峰git

H2 集中式vs分布式

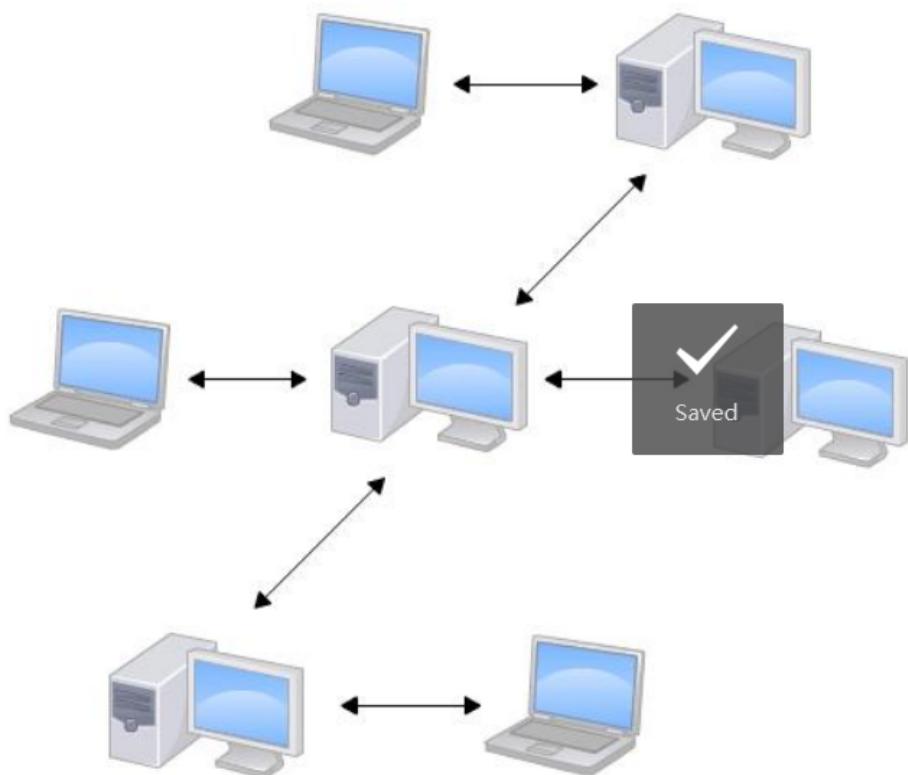
分布式的优势

- 集中式：所有人都可以连接服务器，读取并更新文件。劣势：当出现单点故障时候，谁都无法操作。如果数据丢失，没有良好的完善机制。



- 分布式：每台电脑都有一份完整的版本库--git pull

分布式控制系统，通常也有一台中央服务器的电脑，但是这个服务器的作用仅仅是方便“交换”大家的修改，没有它大家也可以干活，只是交换修改不方便而已 --git add /commit /branch /merge



H2 Config

H3 alias

```
git config --global alias.zbbix 'xxxxx' <--> git zbbix
```

```
$ git config --global alias.co checkout -- git co  
$ git config --global alias.br branch  
$ git config --global alias.ci commit  
$ git config --global alias.st status
```

```
git config --global user.name "wenbluo"
```

```
git config --global user.email "wenbluo@ebay.com"
```

也可以定义每个 repository 的 local user.name & local user.email

The screenshot shows two terminal windows side-by-side. The left window displays a series of git config commands setting global aliases and user information. The right window shows the usage of the git config command, listing the search locations for configuration files: global, system, local, worktree, file, blob, and a help message for reading from a blob object.

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (master)  
$ git config --local user.name 'privatewenbluo'  
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (master)  
$ git config --local user.email 'wbluo_rober@163.com'  
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (master)  
$ git config -l  
core.symlinks=false  
core.autocrlf=true  
core.fscache=true  
color.diff=auto  
color.status=auto  
color.branch=auto  
color.interactive=true  
help.format=html  
rebase.autosquash=true  
http.sslbackend=schannel  
credential.helper=cmanager  
user.email=wenbluo@ebay.com  
user.name=wenbluo  
filter.lfs.process=git-lfs filter-process  
filter.lfs.required=true  
filter.lfs.clean=git-lfs clean -- %F  
filter.lfs.smudge=git-lfs smudge -- %F  
http.sslVerify=true  
core.repositoryformatversion=0  
core.fildemode=false  
core.bare=false  
core.logallrefupdates=true  
core.symlinks=false  
core.ignorecase=true  
gui.wmstate=normal  
gui.geometry=1034x563+96+96 216 255  
user.name=privatewenbluo  
user.email=wbluo_rober@163.com
```

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (master)  
$ git config  
usage: git config [<options>]  
  
Config file location  
  --global      use global config file  
  --system      use system config file  
  --local       use repository config file  
  --worktree    use per-worktree config file  
  -f, --file <file> use given config file  
  --blob <blob-id> read config from given blob object
```

ps: 因为Git是分布式版本控制系统，所以，每个机器都必须自报家门：你的名字和 Email地址

git config -l : 查看所有的配置信息

git config --help

H2 Repository

版本库又叫仓库，简单来说就是一个目录，然后目录下的文件（增删改），都可以被git监控

1. 初始化本地仓库

```
git init
```

```
$ git init  
Initialized empty Git repository in  
C:/Users/wenbluo/Desktop/wbluo/git/learngit/.git/
```

然后就在目录learngit ,生成一个.git 文件 -->请勿改动

-->

如何结束inti 初始化? undo init .

直接删除掉 .git 目录就行

```
rm -rf .git #####r remove -r .git
```

2. 添加文件

```
git add readme.txt
```

3. 提交文件

```
git commit -m "i wrote a txt "
```

```
$ git commit -m "wrote a readme file"  
[master (root-commit) ffbec42] wrote a readme file  
1 file changed, 3 insertions(+)  
create mode 100644 readme.txt
```

1 file changed ,3insertions :表示一个文件上传，并且总行数是3行
-m: comment 备注一定要，这样可以知道每次操作是什么，便于追溯历史文件

ps : add 与 commit 区别

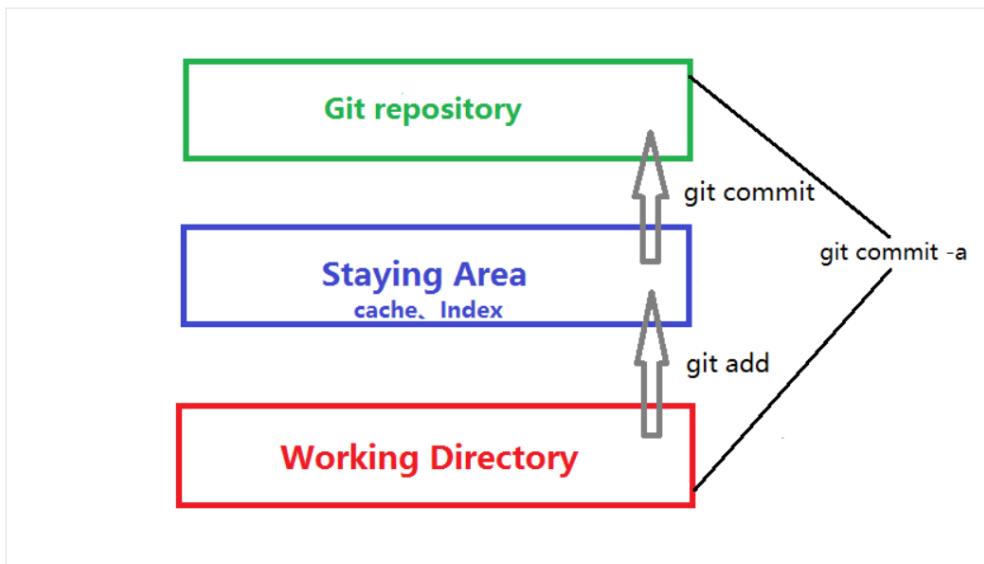
- 文件的三种状态

对于任何一个文件，在git 内都只有三种状态： committed, staged ,modified

- 三种工作区域： working directory ,staging directory ,repository

分别存放三种状态的文件： modified , staged, committed

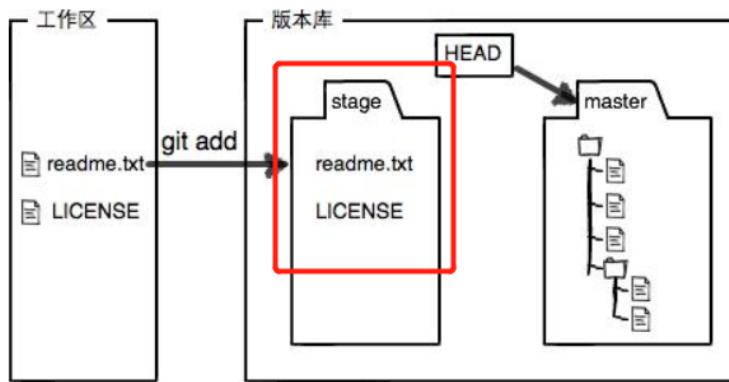
4. 添加及提交文件



- commit 可以一次提交多个文件

第一步是用`git add`把文件添加进去，实际上就是把文件修改添加到暂存区；

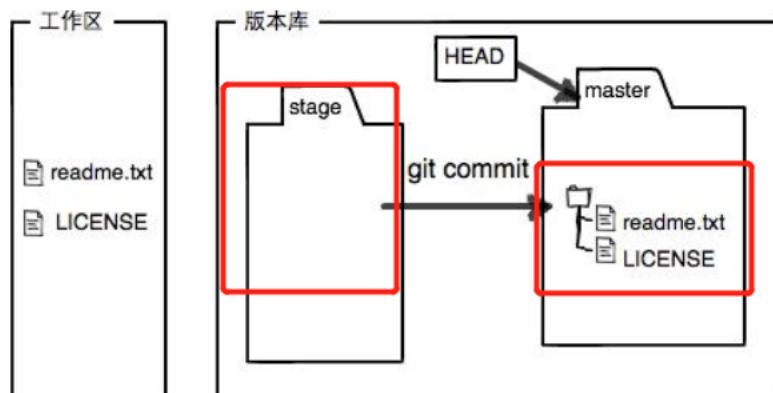
现在暂存区的状态就变成这样了：



- 像这样，你不断对文件进行修改，然后不断提交修改到版本库里，就好比玩RPG游戏时，每通过一关就会自动把游戏状态存盘，如果某一关没过去，你还可以选择读取前一关的状态。有些时候，在打Boss之前，你会手动存盘，以便万一打Boss失败了，可以从最近的地方重新开始。Git也是一样，每当你觉得文件修改到一定程度的时候，就可以“保存一个快照”，这个快照在Git中被称为commit

第二步是用`git commit`提交更改，实际上就是把暂存区的所有内容提交到当前分支。

现在版本库变成了这样，暂存区就没有任何内容了：



第一次修改 -> `git add` -> 第二次修改 -> `git commit`

你看，我们前面讲了，Git管理的是修改，当你用`git add`命令后，在工作的第一次修改被放入暂存区，准备提交，但是，在工作的第二次修改并没有放入暂存区，所以，`git commit`只负责把暂存区的修改提交了，也就是第一次的修改被提交了，第二次的修改不会被提交。

- git rm : 删除文件

```
[06:24:27]wenbluo@rnoetl-preprod004-tess0025 ~/stm2rno/bin git status
# On branch cfg_0919
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       deleted:   cfgGen.py
#       deleted:   stm2rno_cfgGen.py
#
no changes added to commit (use "git add" and/or "git commit -a")
[06:24:31]wenbluo@rnoetl-preprod004-tess0025 ~/stm2rno/bin git rm cfgGen.py stm2rno_cfgGen.py
rm 'bin/cfgGen.py'
rm 'bin/stm2rno_cfgGen.py'
[06:25:02]wenbluo@rnoetl-preprod004-tess0025 ~/stm2rno/bin git commit -m 'to delete the historical files'
[cfg_0919 03a85ec] to delete the historical files
2 files changed, 134 deletions(-)
delete mode 100644 bin/cfgGen.py
delete mode 100644 bin/stm2rno_cfgGen.py
[06:25:24]wenbluo@rnoetl-preprod004-tess0025 ~/stm2rno/bin git checkout master
```

4. git status

掌握仓库当前的状态

```
$ git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   readme.txt
###告知有一个新文件添加到仓库中，待提交

$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in
working directory)

    modified:   readme.txt

no changes added to commit (use "git add" and/or "git commit -a")
###告知有一个文件被修改，但还没被提交

###当想知道到底变动了什么？可以通过git diff readme.txt

$ git diff readme.txt
diff --git a/readme.txt b/readme.txt
index 8797631..4d53f0e 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1,3 +1,2 @@
-git is a version control system. ##删除
+git is a distributed version control system.
 git is free software. ##新增

###舍弃变动
git checkout -- readme.txt ###返回上一个版本
```

5. git log

```
![git_6] (C:\Users\wenbluo\Desktop\wbluo\git\git_6.png) #查看操作
轨迹

git log --pretty=oneline;
##简洁的形式 自上而下 以时间顺序 desc
f238418b2efa5acc47489087fd1d2e38d707fef2 (HEAD -> master)
append GPL
e9eef556941ce1a31385dbcefe2d9efc24d8d34b add distribute
ffbec428f968e71bf5f4ec1dfe9bd203134a8190 wrote a readme file
```

```

##如果想回到add distribute 版本
##采用 git reset --hard head^ or git reset --hard e9eef55

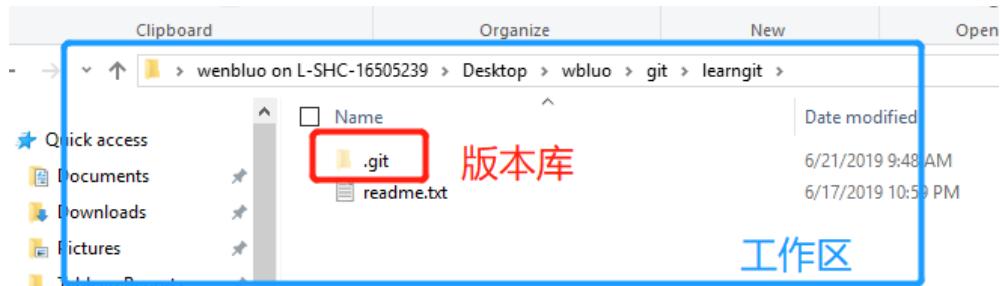
##当回滚到第二版本时候，发现append GPL 没有了！
$ git log --pretty=oneline
e9eef556941ce1a31385dbcef2d9efc24d8d34b (HEAD -> master) add
distribute
ffbec428f968e71bf5f4ec1dfe9bd203134a8190 wrote a readme file

###如果还想回到最后修改的版本 append GPL 怎么办
1: 当会话没有关闭时候，可以查看commit id
    git reset --hard f23841
2: 如果会话关闭了，不知道commit id 怎么办？
    $ git reflog
e9eef55 (HEAD -> master) HEAD@{0}: reset: moving to e9eef55
f238418 HEAD@{1}: reset: moving to f238418b
e9eef55 (HEAD -> master) HEAD@{2}: reset: moving to e9eef55
f238418 HEAD@{3}: reset: moving to f2384
e9eef55 (HEAD -> master) HEAD@{4}: reset: moving to head^
f238418 HEAD@{5}: commit: append GPL ##找到了append GPL
commit id
e9eef55 (HEAD -> master) HEAD@{6}: commit: add distribute
ffbec42 HEAD@{7}: commit (initial): wrote a readme file
#简介一点的
    git reflog | grep -i "append Gpl"
f238418 HEAD@{5}: commit: append GPL

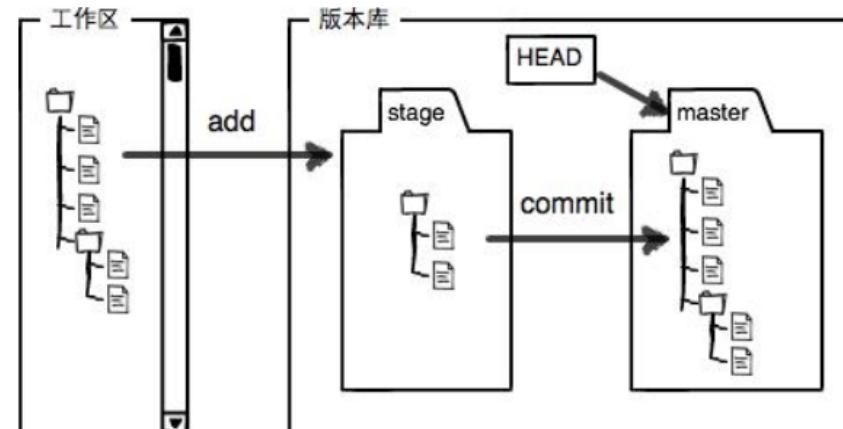
```

H2 Working directory and Repository

1. working directory :工作区



2. stage(index):暂缓区



```

$ git add .
$ git commit -m "upload ad_cube branch"
[master 1049009] upload ad_cube branch
 1 file changed, 0 insertions(+)

```

3. 删除文件

`git checkout -- test.out`

其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

4.

H2 Pull

`git checkout master`

`git pull` :更新当前分支所有信息到本地

```

wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/work/kylin cube/New folder/ad_cube (mas
ter)
$ git pull
warning: no common commits
remote: Enumerating objects: 112, done.
remote: Counting objects: 100% (112/112), done.
remote: Compressing objects: 100% (83/83), done.
Receiving objects: 20% (146209/723699), 85.83 MiB | 9.69 MiB/s

```

`git pull origin master` : 更新 origin 中 master 下最新文件信息

```

wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/work/kylin cube/New folder/ad_cube (mas
ter)
$ git pull
warning: no common commits
remote: Enumerating objects: 112, done.
remote: Counting objects: 100% (112/112), done.
remote: Compressing objects: 100% (83/83), done.
Receiving objects: 20% (146209/723699), 85.83 MiB | 9.69 MiB/s

```

`git pull typora typora` : 表示更新typora connect 下 typora 分支

```
remote: Enumerating objects: 112, done.  
remote: Counting objects: 100% (112/112), done.  
remote: Compressing objects: 100% (83/83), done.  
receiving objects: 49% (354/613) [23/699], 1.92 GiB | 9.55 MiB/s  
 1 folder  
    * gpplotmd  
      1/10/2019 2:51 PM  
      1 file  
    * Hive with Presto.md  
      4/11/2019 3:27 PM  
      1 file  
    * Hive 数据仓库.md  
      7/1/2019 10:08 AM  
      1 file  
    * hql 优化.md  
      1/3/2019 4:40 PM  
      1 file  
    * R.md  
      6/4/2019 9:03 PM  
      1 file  
    * self-prefect.md  
      1/15/2019 6:11 PM  
      1 file  
    * server sql.md  
      4/29/2019 5:40 PM  
      1 file  
    * shell.md  
      7/5/2019 6:45 PM  
      1 file  
    * Sparkmd  
      6/25/2019 10:59 AM  
      1 file  
    * Tableau.md  
      7/2/2019 2:53 PM  
      1 file  
    * Teradata.md  
      6/17/2019 1:37 PM  
      1 file  
    * 数据库理论.md  
      4/15/2019 1:34 PM  
      1 file  
  1 file  
  pull_test  
      7/5/2019 6:46 PM  
      File  
      1 KB  
  1 file  
  typora  
      git@github.corp.ebay.com:wenbluo/typora.git (fetch)  
      git@github.corp.ebay.com:wenbluo/typora.git (push)  
  wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (typora)  
  $ git pull typora typora  
  remote: Enumerating objects: 2, done.  
  remote: Counting objects: 100% (4/4), done.  
  remote: Compressing objects: 100% (2/2), done.  
  Unpacking objects: 100% (3/3), done.  
  Total 3 (delta 1), reused 0 (delta 0), pack-reused 0  
  From github.corp.ebay.com:wenbluo/typora  
    * branch            typora      -> FETCH_HEAD  
    * branch            typora      -> typora/typora  
  Updating d181bc..7b348f0  
  Fast-forward  
  pull test | 1 +  
  1 file changed, 1 insertion(+)  
  create mode 100644 pull_test  
  wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (typora)  
  $  
  Markdown File  
  29 KB  
  Hive数据仓库.md  
  back up my typora  
  a day ago
```

H2 Push

```
git remote set-url --push origin git@github.scm.corp.ebay.com:apdrm/DINT-CLSF.D.git
```

`git push origin master ##`将 master branch 上的信息推送到 origin 上

H3 QA

1. 如何将本地repository push 到 github上?

- step one :在github 上创建repository
 - step two : git remote add typora <https://github.com/privatewbluo/typora.git>
 - step three : git pull typora master :本地同步 github上typora master 分支信息
 - step four : git add & git commit
 - step five : git push typora 0904 将本地文件push到github上

-->question

- 既然可以直接**git push xxx branch** 那么为什么还要tag ?
 -

2. pull request



how to push the local file/directory to github #1
 Open privateewbluo wants to merge 1 commit into from

有什么作用？

privatewbluo wants to merge 1 commit into **master** from **0904**

Conversation 0 Commits 1 Checks 0 Files changed 1

privatewbluo commented 32 seconds ago
No description provided.

Owner +
Reviewers
No reviews

Assignees
No one—assign y

Labels
8fc79e6

可以看出是将**0904** 分支 合并到**master** 主分支上。

git checkout master

git merge 0904

2、pull request 的含义

解释一：

有一个仓库，叫**Repo A**，你如果要往里贡献代码，首先要Fork这个Repo，于是在你的Github账号下有了一个**Repo A2**。然后你在这个A2下工作，Commit，push等。然后你希望原始仓库**Repo A**合并你的工作，你可以在**Github**上发起一个Pull Request，意思是请求**Repo A**的所有者从你的A2合并分支。
如果被审核通过并正式合并，这样你就为项目**A**做贡献了。

3. 为什么在window desktop 操作了get merge 可以同步文件，但是到github 上没有实现同步呢？

| File | Action | Branch | Time |
|------------------|---------------------|---------------|----------------|
| self-prespect.md | upload my notebooks | branch:0904 | 3 hours ago |
| server sql.md | upload my notebooks | branch:0904 | 3 hours ago |
| shell.md | test git merge | branch:0904 | 3 minutes ago |
| 数据库理论.md | upload my notebooks | branch:0904 | 3 hours ago |
| server sql.md | upload my notebooks | branch:master | 3 hours ago |
| shell.md | change | branch:master | 35 minutes ago |
| 数据库理论.md | upload my notebooks | branch:master | 3 hours ago |

这个时候需要在**website** 点击 **pull requests**

privatewbluo / typora

Code Issues Pull requests Projects Wiki Security Insights Settings

store my own typora notebooks
Manage topics

13 commits 2 branches 0 releases 1 contributor

Branch: master ▾ New pull request

privatewbluo Merge pull request #3 from privatewbluo/0904 ... Latest commit 79c144c 2 minutes ago

H2 Merge

如何file 还在**working direcor**，那么不管在哪个**branch** 都能看到or 同步

如果做了**git commit** ,就只能在当下**branch** 查看

-->原理可以查看 **git add & git commit** 的区别

不过可以通过 **git merge** 操作，也可以在其他**branch** 同步

1. git checkout master

git merge 0904 #meger branch 0904 to master

2. git branch -d 0904 :删除分支

3.

The screenshot shows the GitHub repository overview for 'privatewbluo / typora'. It displays three branches: 'master' (Default), '0904', and another unnamed branch. A red box highlights the 'Default branch' section. Another red box highlights the 'Your branches' section where '0904' is listed. Below is the 'Active branches' section with the same information. At the top right, there is a 'Change default branch' button.

4. website 删除分支

The screenshot shows the GitHub repository overview for 'bit / test'. It displays three branches: 'master', 'test', and another unnamed branch. A red box highlights the 'Default branch' section. Another red box highlights the 'Your branches' section where 'test' is listed. A red arrow points from the 'New pull request' button in the 'Your branches' section to a URL at the bottom of the page: <https://blog.csdn.net/wangbinbin1>. A third red arrow points from the 'New pull request' button in the 'Stale branches' section to the same URL.

H2 Status

1. git status 可以查看变化

-->那么如果查看具体变化呢?

通过git diff shell.md

2. 那git diff 是跟 什么时候的file 作比较?

跟 **committed** 到**repository** 的文件做比较

--> git 文件有三种状态: modified,staged ,committed .

git add xxx --> staged

git commit xxx -m 'xxx' -->committed

H2 Fork

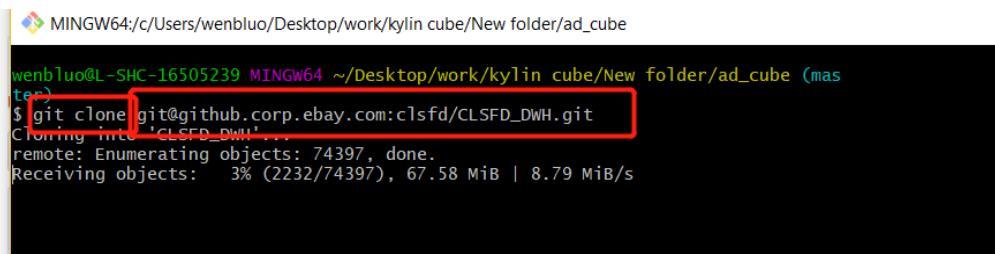
clone repository 到个人账户

The screenshot shows the GitHub fork page for a repository. At the top, there are navigation links for 'Marketplace' and 'Pricing'. A search bar is followed by 'Sign in' and 'Sign up' buttons. Below the search bar are buttons for 'Sponsor', 'Watch' (7,287), 'Star' (134,401), and 'Fork' (65,960). A red box highlights the 'Fork' button. A message at the bottom states: 'You must be signed in to fork a repository'.

Clone

[如何ping github.com 成功?](#)

从远程库 (origin) clone 到本地环境

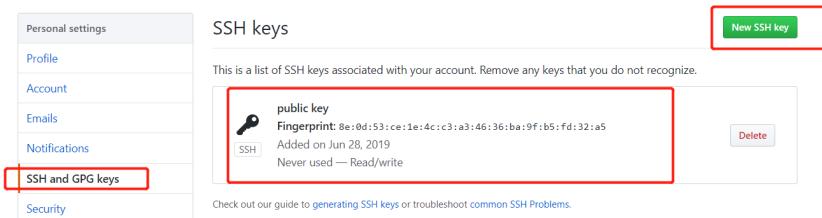


```
MINGW64:/c/Users/wenbluo/Desktop/work/kylin cube/New folder/ad_cube
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/work/kylin cube/New folder/ad_cube (master)
$ git clone git@github.corp.ebay.com:clsfd/CLSD_DWH.git
Cloning into 'CLSD_DWH'...
remote: Enumerating objects: 74397, done.
Receiving objects: 3% (2232/74397), 67.58 MiB | 8.79 MiB/s
```

为了避免每次远程clone repository 到本地 都要输入 账号与密码:

[可以采用远程ssh](#)

1.



Personal settings

Profile

Account

Emails

Notifications

SSH and GPG keys

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

public key
Fingerprint: 8e:0d:53:ce:1e:4c:c3:a3:46:36:ba:9f:b5:fd:32:a5
SSH
Added on Jun 28, 2019
Never used — Read/write

Delete

Check out our guide to generating SSH keys or troubleshoot common SSH Problems.

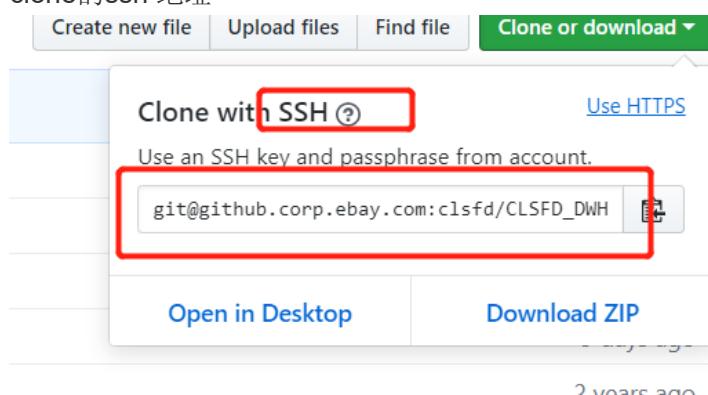
点击个人账户，添加pub key

- **ssh-keygen** :生成public key
- Paste the **content** of your public key here, usually from your `~/.ssh/id_rsa.pub`.

`clip < /c/Users/wenbluo/.ssh/id_rsa.pub` :将密码放到剪贴板

-->clip 是window 命令

2. clone的ssh 地址



Create new file Upload files Find file **Clone or download ▾**

Clone with SSH ⓘ Use HTTPS

Use an SSH key and passphrase from account.

git@github.corp.ebay.com:clsfd/CLSD_DWH

Open in Desktop Download ZIP

2 years ago

3.

4. git clone git@github.corp.ebay.com:clsfd/CLSD_DWH.git

clone 一个项目

---ecg 固定relese repository :git@github.corp.ebay.com:APD/DINT-CLSD_DWH.git

https://github.corp.ebay.com/APD/DINT-CLSD_DWH/commit/DW_PDP_DWRM3294

3

H3 QA

- 可以直接clone 不需要配置or identify?
 -

H2 Fetch

H2 Remote

2.1 setup 'origin'

- if you haven't cloned yet:

```
1 | cd <directory-of-choice>
2 | git clone git@github.scm.corp.ebay.com:apdrm/<project>.git
```

- in case you already cloned:

```
1 cd <existing_local_clone>
2 git remote set-url origin git@github.scm.corp.ebay.com:apdrm/<project>.git
```

2.2 update the push URL

```
1 # make sure we're inside the project root of the cloned repository
2 cd <project> # e.g. cd DINT-CORE
3
4 # e.g. git remote set-url --push origin git@github.scm.corp.ebay.com:APD/DINT-CORE.git
5 git remote set-url --push origin git@github.scm.corp.ebay.com:APD/<project>.git
```

只有clone之后才能添加remote？目的是能更改github上的repository

如何远程通信github 上的repository?

你已经在本地创建了一个git仓库后，又想在Github创建一个Git仓库，并且让这两个仓库进行远程同步，这样，GitHub上的仓库既可以作为备份，又可以让其他人通过该仓库协作，一举多得。

1. git remote set -url fetch

```
git remote ``set``-url origin  
git@github.scm.corp.ebay.com:apdrm/<project>.git
```

```
git remote set-url git@github.scm.corp.ebay.com:APD/DINT-  
CLSF.D.git
```

注意是 git 开头

HTTPS.

Always use the "git" user

All connections, including those for remote URLs, must be made as the "git" user. If you try to connect with your GitHub username, it will fail:

```
$ ssh -T GITHUB-USERNAME@github.com  
> Permission denied (publickey).
```

If your connection failed and you're using a remote URL with your GitHub username, you can change the remote URL to use the "git" user.

You should verify your connection by typing:

```
$ ssh -T git@github.com  
> Hi username! You've successfully authenticated...
```

```
PreferredAuthentications publickey
IdentityFile /c/Users/wenbluo/.ssh/private

Host github.corp.ebay.com
Hostname github.corp.ebay.com
User wenbluo
PreferredAuthentications publickey
IdentityFile /c/Users/wenbluo/.ssh/id_rsa

[...]
ssh github.corp.ebay.com
ssh wenbluo@github.corp.ebay.com: Permission denied (publickey).
[...]
$ git git@github.corp.ebay.com
git: 'git@github.corp.ebay.com' is not a git command. See 'git --help'.
[...]
$ ssh git@github.corp.ebay.com
[...]
$ ssh git@github.corp.ebay.com
PTY allocation request failed on channel 0
No connection could be made because the target machine actively refused it.
Connection to github.corp.ebay.com closed.

[...]
$ telnet www.SMC-Easy2SSH.MINGW64 / -sshd
[...]
```

H5 OA

- 如何将本地repository push 到github 上?

1. step_1: git init
 2. step_2: git add .
 3. step_3:git commit -m 'xxxx'

4. step_4:git remote add origin xxx

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora  
(master)  
$ git remote add orgin  
https://github.corp.ebay.com/wenbluo/typora.git  
  
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora  
(master)  
$ git remote -v  
orgin  
https://github.corp.ebay.com/wenbluo/typora.git  
(fetch)  
orgin  
https://github.corp.ebay.com/wenbluo/typora.git (push)  
##与github上的repository创建连接
```

5. step_5:git push -u origin master

2. git remote set-url push

```
git remote ``set``-url --push origin  
git@github.scm.corp.ebay.com:APD/<project>.git
```

```
git remote set-url --push origin  
git@github.scm.corp.ebay.com:APD/DINT-CLSF.D.git
```

- git remote add origin git@github.scm.corp.ebay.com:APD/DINT-CLSF.D.git
-->本地关联远端库: DINT-CLSF.D
-->Origin:远端库, git默认的叫法
- git push -u origin master
-->把本地库的内容推送到远程, 用git push命令, 实际上是把当前分支
master推送到远程。
- git push origin master

3. git remote -v

```
$ git remote -v origin git@github.scm.corp.ebay.com:APD/DINT-CLSF.D.git  
(fetch) origin git@github.scm.corp.ebay.com:APD/DINT-CLSF.D.git (push)
```

#这样就可以对project or repository DINT-CLSF.D 进行**fork or push**

```
4. $ git remote add typora  
git@github.corp.ebay.com:wenbluo/typora.git  
###添加远程链接 typora  
###说明远程链接库，并不一定非是 origin  
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora/.git  
(GIT_DIR!)  
$ git remote -v  
origin https://github.corp.ebay.com/APD/DINT-CLSF.D.git  
(fetch)  
origin https://github.corp.ebay.com/APD/DINT-CLSF.D.git (push)  
typora git@github.corp.ebay.com:wenbluo/typora.git (fetch)  
typora git@github.corp.ebay.com:wenbluo/typora.git (push)
```

git config 文件:

5. 如何去掉 remote ?

6.

git config --list :查看所有配置信息

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora/.git (GIT_DIR!)  
$ cat config  
[core]  
    repositoryformatversion = 0  
    filemode = false  
    bare = false  
    logallrefupdates = true  
    symlinks = false  
    ignorecase = true  
[remote "origin"]  
    url = https://github.corp.ebay.com/APD/DINT-CLSF.D.git  
    fetch = +refs/heads/*:refs/remotes/origin/*  
    pushurl = https://github.corp.ebay.com/APD/DINT-CLSF.D.git  
[branch "typora"]  
    remote = origin  
    merge = refs/heads/typora  
[remote "typora"]  
    url = git@github.corp.ebay.com:wenbluo/typora.git  
    fetch = +refs/heads/*:refs/remotes/typora/*
```

cat .git 目录下 config 文件可以看到 typora remote 还没有push 功能。

```
```shell  
$ git remote set-url --push typora
git@github.corp.ebay.com:wenbluo/typora.git

--push typora : 对typora link 添加push 功能
```

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (typora)
$ cat .git/config
[core]
 repositoryformatversion = 0
 filemode = false
 bare = false
 logallrefupdates = true
 symlinks = false
 ignorecase = true
[remote "origin"]
 url = https://github.corp.ebay.com/APD/DINT-CLSF.D.git
 fetch = +refs/heads/*:refs/remotes/origin/*
 pushurl = https://github.corp.ebay.com/APD/DINT-CLSF.D.git
[branch "typora"]
 remote = origin
 merge = refs/heads/typora
[remote "typora"]
 url = git@github.corp.ebay.com:wenbluo/typora.git
 fetch = +refs/heads/*:refs/remotes/typora/*
 pushurl = git@github.corp.ebay.com:wenbluo/typora.git
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (typora)
$
```

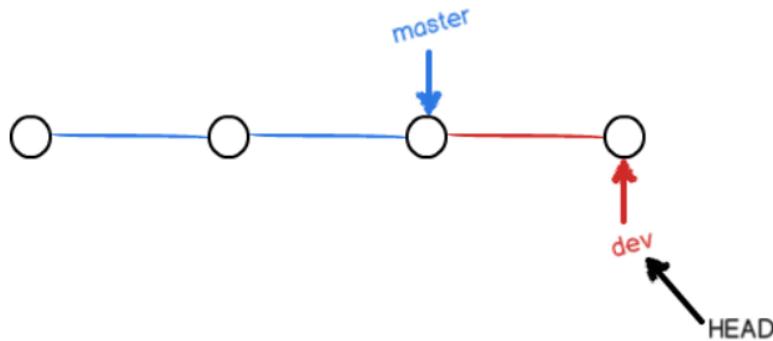
## H2 Master & Branch

background: 两个指针 : head & branch

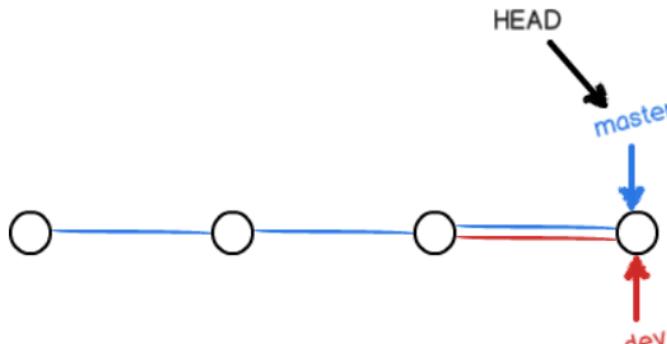


当你创建branch dev : git checkout -b dev

此时 head 指针指向 dev 分支，然后每次commit 是在dev 分支 上；



如果我们在dev 分支上的工作完成了，就可以将dev 分支合并master 上。git 最简单的办法，就是直接将master 指向dev 的当前提交。



```
go back to project root directory and get latest master
cd <project-root> # check with `pwd`
git checkout master ##切换到主分支
git pull # if this fails, run `git branch --set-upstream master
origin/master` : 更新 master 下所有文件信息;
git checkout -b ad_cube master ###查看branch
##fatal: A branch named 'ad_cube' already exists
```

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/work/kylin_cube/New folder/ad_cube/DINT-
$ git checkout -b ad_cube master
Switched to a new branch 'ad_cube'
M etl/sql/app_trust/app_trust.Bid_lmts_seg.del_ins.sql
M etl/sql/app_trust/app_trust.Buy2bid_chng_impact.del_ins.sql
M etl/sql/app_trust/app_trust.Early_arrival_data_slr_cntry.del_ins.sql
M etl/sql/app_trust/app_trust.Early_arrival_data_trans_site.del_ins.sql
M etl/sql/app_trust/app_trust.Exp_byrs_list.del_ins.sql
M etl/sql/app_trust/app_trustUPI_GMV.del_ins.sql
M etl/sql/app_trust/app_trustUPI_actions.del_ins.sql
M etl/sql/app_trust/app_trustUPI_main.del_ins.sql
ix.
```

```
go back to our branch and merge things from latest master
git checkout <branch>
git merge master
```

git checkout -b dev :

-->表示创建分支 dev ,并且切换到dev 该分支下;

###相当于下面两步操作  
`git branch dev`  
`git checkout dev ###切换到dev branch`

###查看分支  
`$ git branch`  
\* dev     ###前面\*表示当前分支  
  master  
###查看各个分支 最后提交的最后comment  
`git branch -v`

###查看已经合并的分支  
`git branch --merge --no-merge` :没有合并的分支

```
###删除分支
git branch -d xxxx | -D xxxx : 强制删除分支, 忽略是否有file ,没有提交到该分支上
```

```
###查看文档:
git branch --help
```

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/wbluo/git/learngit (master)
$ git checkout -b ad_cube
Switched to a new branch 'ad_cube'
g
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/wbluo/git/learngit (ad_cube)
$ git branch
* ad_cube
 master
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/wbluo/git/learngit (ad_cube)
$ |
```

**git checkout master** : 切换 **master** 主分支的时候, 发现是我们在**ad\_cube** 修改的 **readme.txt** 文件 并没有影响 **master** 上的文件 !

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/wbluo/git/learngit (ad_cube)
$ git add readme.txt
gi
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/wbluo/git/learngit (ad_cube)
$ git commit -m 'branch test'
[ad_cube 9503378] branch test
 1 file changed, 1 insertion(+)
不
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/wbluo/git/learngit (ad_cube)
$ git checkout master
Switched to branch 'master'
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/wbluo/git/learngit (master)
$ cat readme.txt
git is a distributed version control system.
git is free software.
git has a mutable index call stage.
git is free source !
git track changes.
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/wbluo/git/learngit (master)
$ |
```

```
git merge ad_cube
将ad_cube 合并到当前分支 :
将ad_cube 分支 与master 合并。
```

##此时 master readme.txt文件就会变更

```
git branch -d ad_cube ##删除分支 ad_cube
```

## 分支策略

在实际开发中，我们应该按照几个基本原则进行分支管理：

首先，`master` 分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面干活；

那在哪干活呢？干活都在`dev` 分支上，也就是说，`dev` 分支是不稳定的，到某个时候，比如1.0版本发布时，再把`dev` 分支合并到`master` 上，在`master` 分支发布1.0版本；

你和你的小伙伴们每个人都在`dev` 分支上干活，每个人都有自己的分支，时不时地往`dev` 分支上合并就可以了。

所以，团队合作的分支看起来就像这样：



## git checkout -b dev origin/dev : copy 远程repository 的一个branch dev

```
it
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/work/kylin_cube/New Folder/ad_cube (typora)
$ git checkout -b dev_test origin/dev
Checking out files: 100% (98822/98822), done.
Switched to a new branch 'dev_test'
Branch 'dev_test' set up to track remote branch 'dev' from 'origin'.
```

## Tips&QA

### 1. 如何安全branch 切换到其他他们branch /master ?

留心 staging areas / 缓冲区，那些没有提交的修改，他会和你即将检出的分支产生冲突，从而阻止你change branch .

### 2.

## H2 Tag

### Reference tag

git tag : 可以查看已经有那些标签

```
DWRM10389_1
DWRM10431
DWRM10431_1
DWRM10488
DWRM13678
DWRM13678_1
DWRM22382
DWRM22454
DWRM22461
DWRM22588
DWRM22588_1
DWRM22771
DWRM23061_1
DWRM23061_4
DWRM23290
DWRM23436
DWRM23437
DWRM23437_1
DWRM23618
DWRM23933
DWRM23933_1
DWRM23933_2
DWRM23933_3
DWRM24623
DWRM24623_1
..|
```

## Manage topics

Branch: master ▾ New pull request Create new file

Switch branches/tags

Find or create a branch...  
Branches Tags

0904 ...  
upload my notebooks  
upload my notebooks

master  
Hive vs Presto.md

既有branch 又有tags ,所以在 gti push typora branch/tag 都可以

-->

分享标签

默认情况下, `git push` 并不会把标签传送到远端服务器上, 只有通过显式命令才能分享标签到远端仓库。其命令格式如同推送分支, 运行 `git push origin [tagname]` 即可:

```
$ git push origin v1.5
Counting objects: 50, done.
Compressing objects: 100% (38/38), done.
Writing objects: 100% (44/44), 4.56 KiB, done.
Total 44 (delta 18), reused 8 (delta 1)
To git@github.com:schacon/simplegit.git
 * [new tag] v1.5 -> v1.5
```

如果要一次推送所有本地新增的标签上去 可以使用 `--tags` 选项:

通过github ,可以通过tag **DW\_PDP\_DWRM32943\_1**

1. 总共有35个变化，其中17个新增(绿色部分)，18个delete(红色部分)

Merge branch 'master' into ad\_cube

rebuild table ad\_cube ,add partitions clifd\_site\_id & clifd\_sum\_dt

master → DW\_PDP\_DWRM32943\_1

wenbluo committed an hour ago

Showing 1 changed file with 17 additions and 18 deletions.

1. 通过 `git show tag_name` 可以查看具体信息
- 2.

### H3 QA :difference Branch & Tag

- 什么时候创建tag ?

当下**repository** 的一个 **snapshot** ,一个镜像。是不可变的，但是**branch**下的**files**都是可以变的

- [tag name](#)
- 删除tag `git tag -d`
- 如何提交tag?
  1. `git push typora lear_tag_1`
  2. `git push typora --tag` : 提交所有当前branch下的所有tag

### • tag 与 release 关系

每有一个tag 就有一个release

## H2 Release

### [git to release](#)

1.

## H2 checkout

1. `git checkout -b ad_cube`

```
$ git checkout -b ad_cube
fatal: A branch named 'ad_cube' already exists.
```

2. `git checkout -- readme.txt`

恢复最近一次readme.txt 原始文件

- 注意: -- 两个横杠一个都不能少!

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/wbbluo/git/learngit (master)
$ git status
On branch master
Changes not staged for commit:
 (use "git add <file>..." to update what will be committed)
 (use "git checkout -- <file>..." to discard changes in working directory)

 modified: readme.txt

no changes added to commit (use "git add" and/or "git commit -a")

wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/wbbluo/git/learngit (master)
$ git checkout -- readme.txt

wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/wbbluo/git/learngit (master)
$ cat readme.txt
git is a distributed version control system.
git is free software.
git has a mutable index call stage.
git is free source !
git track changes.
```

1.

## H2 Stash

background:

当你进行一项目中的某一部分工作，里面的东西处于一个比较杂乱的状态，而你想转到其他分支上进行一些工作。问题是，你不想提交进行了一般的工作，否则以后你就无法回到这个工作点。

--> 为什么不可以打标签? tag , 这样可以有一个镜像

临时性的将当场工作“隐藏”起来，等以后恢复工作现场后继续工作. 然后就可以切换  
**git checkout master** ,等其他**branch** 工作;

**git stash ##**隐藏当前工作区

原理:

为了往堆栈推送一个新的储藏, 只要运行 **git stash:** , 生成一个id

**git stash list ##**查看有多少stash

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (0904)
$ git stash
Saved working directory and index [state WIP on 0904: 3832dd1 cal=sgit status]
```

```
wenbluo@L-SHC-16505239 MINGW64 ~/Desktop/typora (0904)
$ git stash list
stash@{0}: WIP on 0904: 3832dd1 cal=sgit status
stash@{1}: WIP on 0904: aac3c36 how to push the local file/directory to github
```

**git stash apply stash@{o}** : 通过apply id 来恢复

ps: git apply 只是调用栈列中的stash id ,隐藏的内容仍然在栈中。

**git stash drop** : 删除栈中apply id

**git stash pop =git stash apply + git stash drop**

## H2 Roll Back

##

## H2 QA

- Untracking file

```
NO COMMITS YET
Untracked files:
(use "git add <file>..." to include in what will be committed)
Business_English.md
"Computer \351\200\232\344\277\241.md"
ECG data flow.md
Hive vs Presto.md
Hive ware house .md
R.md
README.md
Spark.md
Tableau.md
Teradata.md
bootstrap/
gpplot.md
git_stash
"hq1 \344\274\230\345\214\226.md"
marking_scorecard.md - Shortcut.lnk
self-respect.md
server sql.md
shell.md
"\346\225\260\346\215\256\345\272\223\347\220\206\350\256\272.md"
```

- git checkout master error: pathspec 'master' did not match any file(s) known to git

正常情况下是可以回到master分支的

不过这时是报错误的，错误信息如下：

error: pathspec 'master' did not match any file(s) known to git.

这是因为，还没有文件被提交过。即没有commit过任何文件。

当commit过以后就可以切换分支了

备注：此时执行：git branch，只显示有dev这个branch。

不过我们可以直接再创建一个master出来。

下面是整个过程

- 如何在shell ps1 [命令提示符]显示branch？

```
1 | function git-branch-name {
2 | git symbolic-ref HEAD 2>/dev/null | cut -d"/" -f 3
3 |
4 | function git-branch-prompt {
5 | local branch=git-branch-name`
6 | if [$branch]; then printf "[%s]" $branch; fi
7 | }
8 | PS1="\u@\h \[\033[0;36m\]\W\[\033[0m\]\[\033[0;32m\]\$(git-branch-prompt)\[\033[0m\] \$ "
```

注意：上面的 `git symbolic-ref HEAD 2>/dev/null | cut -d"/" -f 3` 可以改成 `git symbolic-ref --short -q HEAD`，更简洁，而且测试有效。谢谢风之去向\_c305的建议；

