

环境搭建

2022年4月7日 10:59

安装Linux虚拟机

CentOS7的下载、安装和配置

13.1万 202 2020-07-09 22:45:37 未经作者授权，禁止转载



CentOS7的安装和配置

4



3人正在看，已装填 10 条弹幕



A 发个友善的弹幕见证当下

弹幕礼仪 >

发送

1603

821

2020

740

稿件投诉

记笔记

⋮

屏幕剪辑的捕获时间: 2023/1/13 21:25

(UP主分享CentOS 7、SecureCRT)

配置网卡

知乎

首发于 雷哥带你学Linux云计算

```
Linux localhost.localdomain 3.18.0-327.61.1.x86_64 #1 SMP Thu Nov 19 22:18:57 UTC 2015 x86_64  
[root@localhost ~]# pwd  
/root  
[root@localhost ~]# ls  
anaconda-ks.cfg
```

知乎 @我心飞翔

E 目录

配置网卡

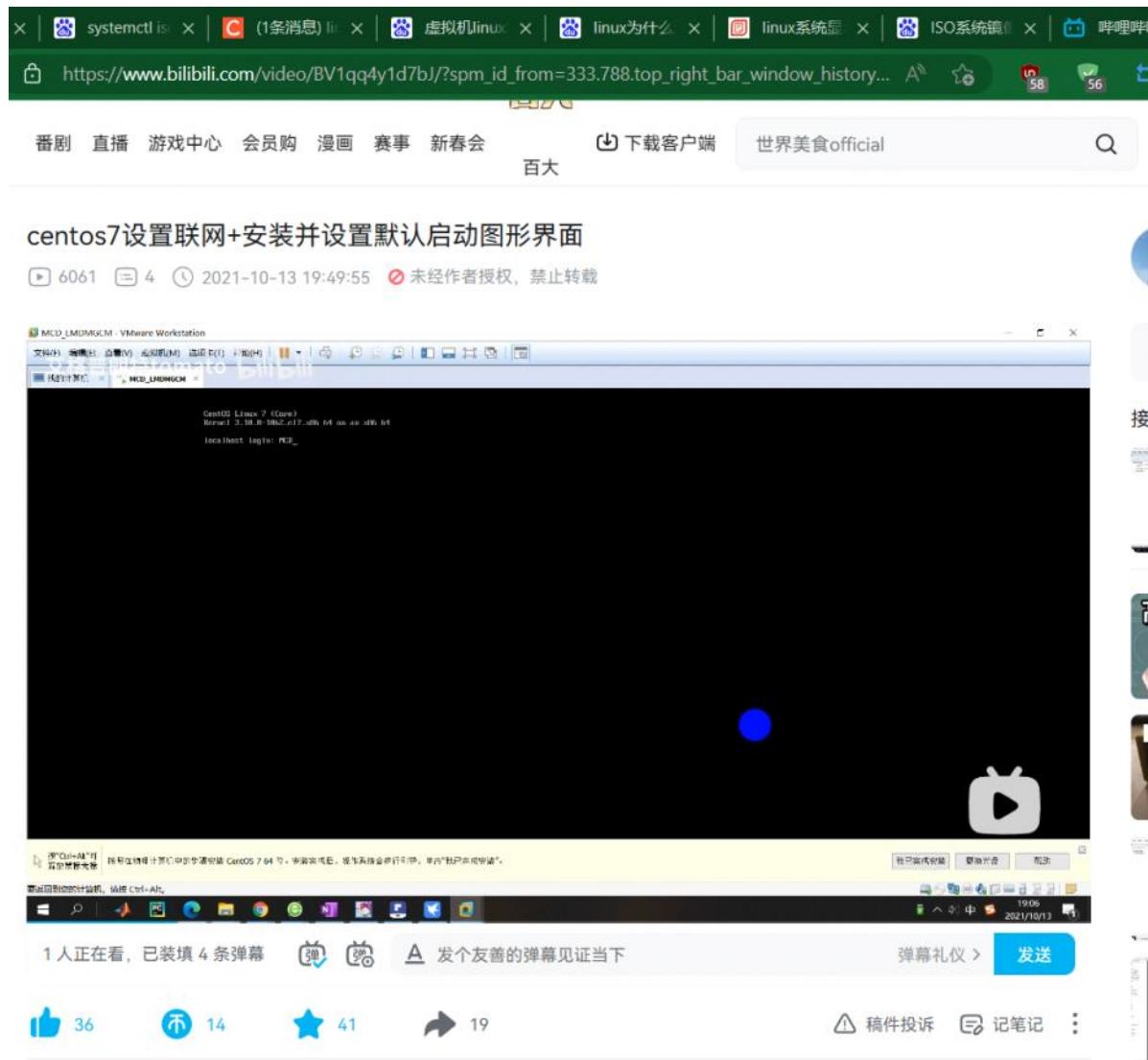
系统安装完毕，但是还不能和外界连通，这时候需要为操作系统设置网络：

在系统命令行下输入nmtui命令然后回车，然后通过上下左右键和tab键选择对应的选项进行配置。为了方便展示，老师把各步骤的截图合并并标上号：



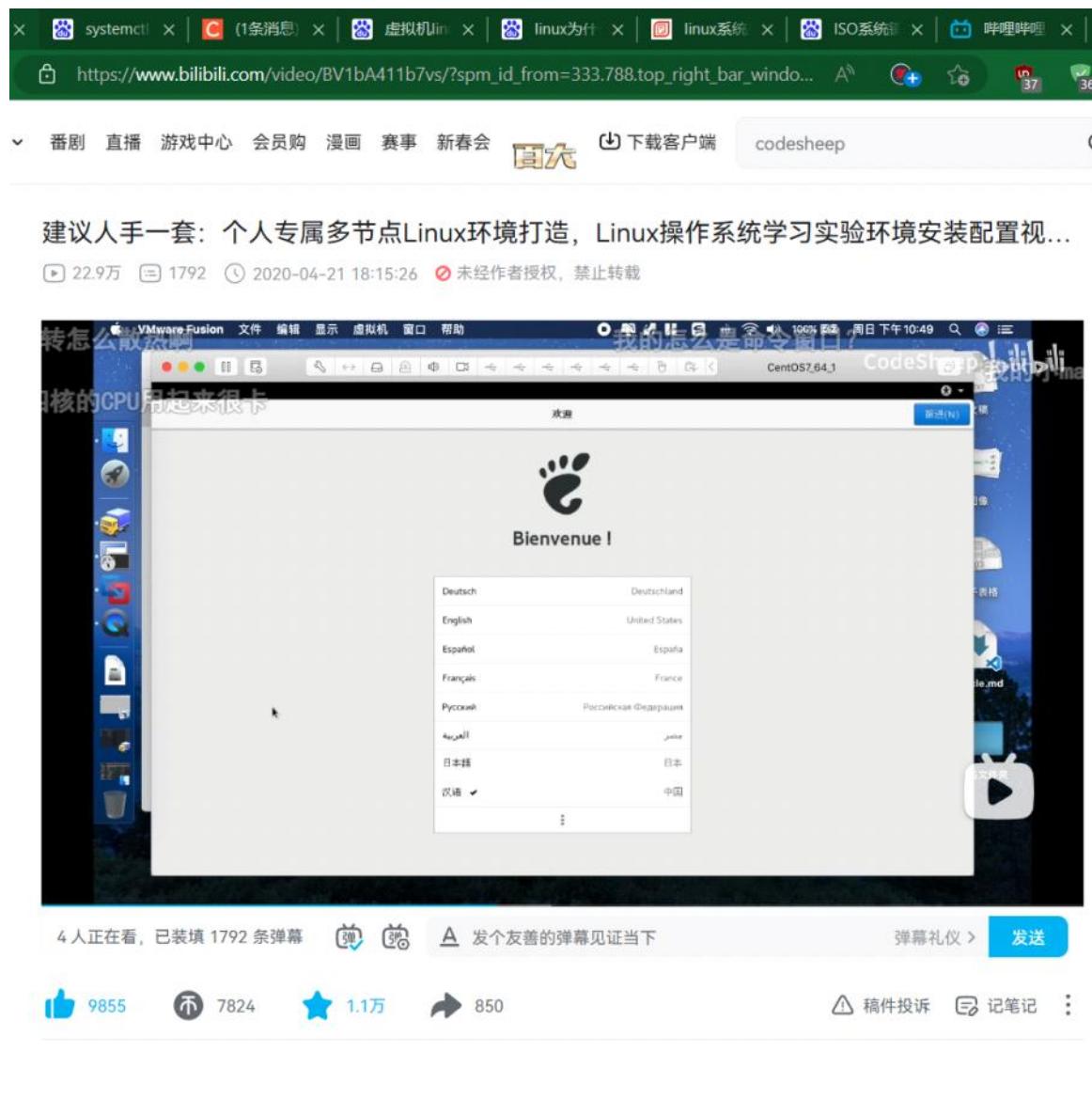
安装图形界面

(前面步骤完成时已经可以联网，因此视频跳到3:10看)



屏幕剪辑的捕获时间: 2023/1/13 21:50

进一步打造



屏幕剪辑的捕获时间: 2023/1/13 21:56

root 密码: 848610

luhao 密码: 77158yanYY



屏幕剪辑的捕获时间: 2023/1/13 21:58

切换命令行界面和图形界面

8. 修改CentOS默认启动模式为图形化模式

以命令 `systemctl get-default` 可查看当前默认的模式为multi-user.target, 即命令行模式

```
[root@localhost network-scripts]# systemctl get-default  
multi-user.target  
[root@localhost network-scripts]# CSDN @小果子^_^
```

需要以命令 `systemctl set-default graphical.target` 修改为图形界面模式

```
1 # 修改模式命令：  
2 systemctl set-default graphical.target # 将默认模式修改为图形界面模式  
3 systemctl set-default multi-user.target # 将默认模式修改为命令行模式  
4  
[root@localhost network-scripts]# systemctl set-default graphical.target  
Removed symlink /etc/systemd/system/default.target.  
Created symlink from /etc/systemd/system/default.target to /usr/lib/systemd/system/graphical.target.  
[root@localhost network-scripts]# A CSDN @小果子^_^
```

再次以命令 `systemctl get-default` 即可查看当前修改后的默认模式为graphical.target, 即图形界面模式

9. 重启CentOS, 检验GUI界面效果

屏幕剪辑的捕获时间: 2023/1/15 20:08

init 3	命令行界面
init 5	图形界面

centos7+版本

在图形界面使用 **ctrl+alt+F2** 切换到dos界面 dos界面 **ctrl+alt+F2** 切换回图形界面；
在命令上 输入 **init 3** 命令 切换到dos界面 输入 **init 5**命令 切换到图形界面；

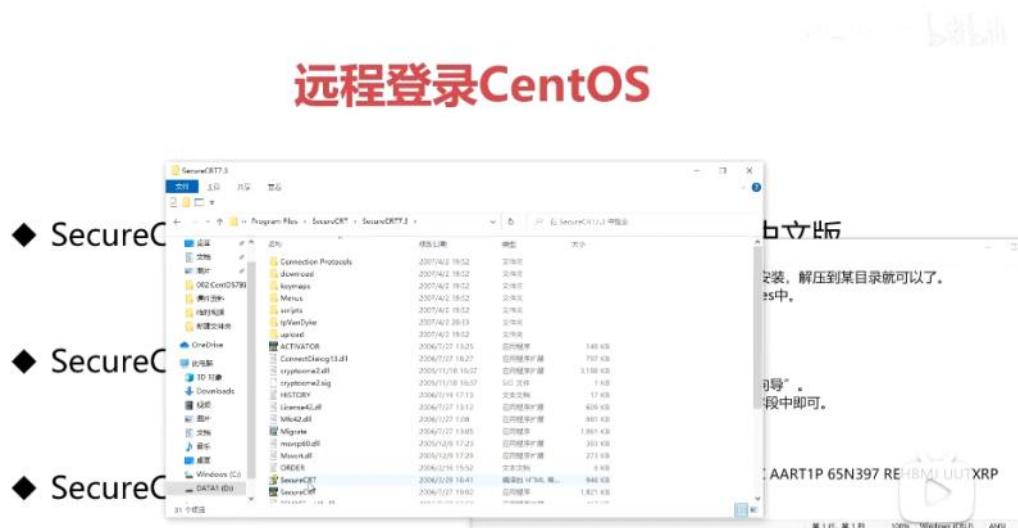
```
password.  
Last failed login: Fri Jul  2 17:39:58 CST 2021 on tty1  
There was 1 failed login attempt since the last successful login.  
Last login: Fri Jul  2 14:35:34 from 192.168.160.1  
[root@localhost ~]# init 5_
```

屏幕剪辑的捕获时间: 2023/1/15 20:09

SecureCRT

SecureCRT8.5的下载、安装和注册

2.7万 71 2020-07-10 21:34:38 未经作者授权，禁止转载



屏幕前的捕获时间: 2023/1/15 20:12

淘宝购买SecureCRT和SecureFX，然后可以结合

word的说明教程和上方的视频教程

(淘宝的SecureFX 9.1不能用，客服协助下载英文版SecureCRT 9.3和SecureFX 9.3，并成功激活)

使用SecureCRT进行两个系统间的文件传输

屏幕剪辑的捕获时间: 2023/1/16 10:14

注意开SecureCRT 9.3或SecureFX 9.3时不能开着VPN



建议人手一套：个人专属多节点Linux环境打造，Linux操作系统学习实验环境安装配置视...

22.9万 1795 2020-04-21 18:15:26 未经作者授权，禁止转载



2 人正在看，已装填 1795 条弹幕



发个友善的弹幕见证当下

弹幕礼仪 >

发送



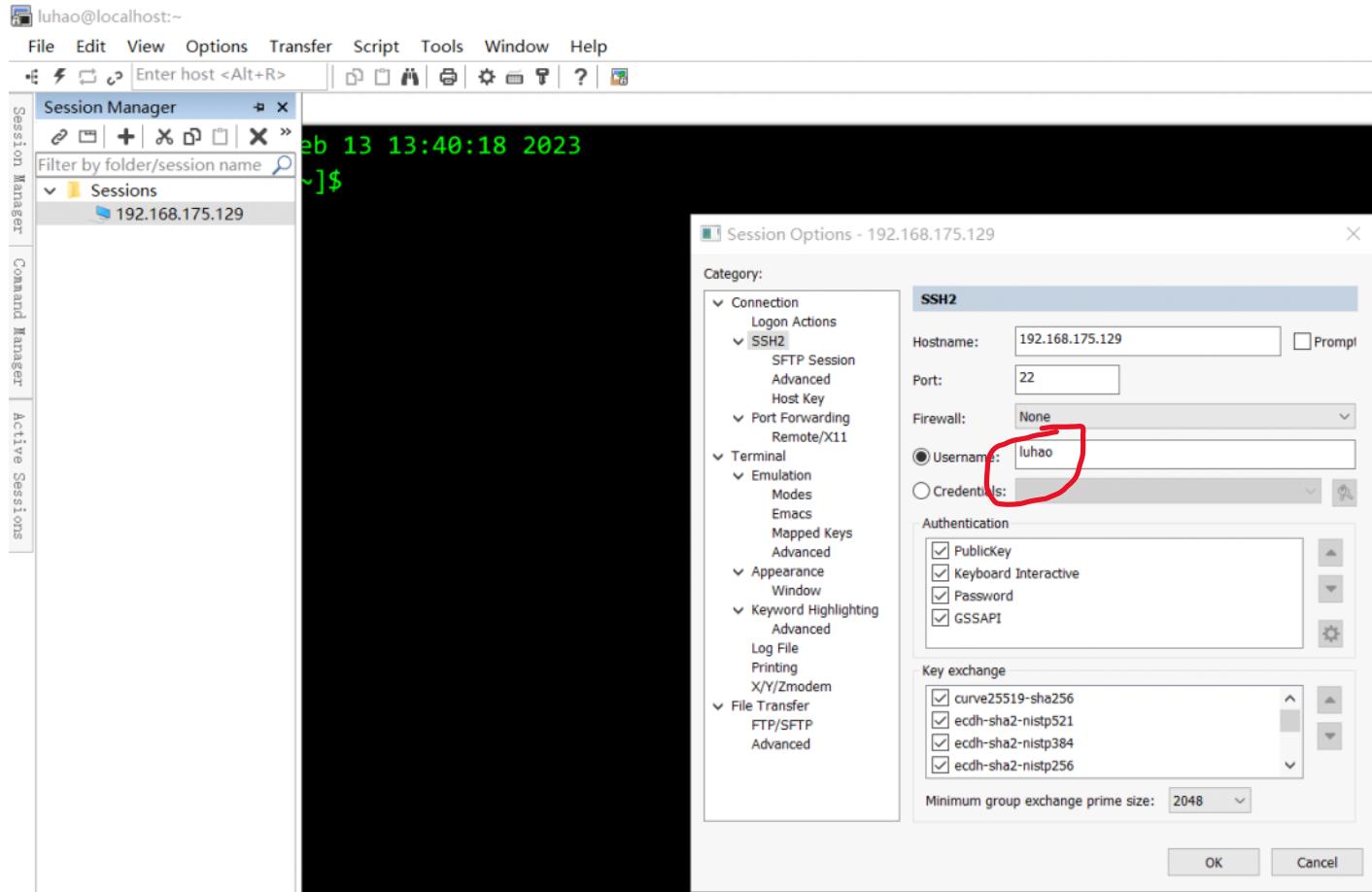
稿件投诉

记笔记



解决SecureFX中文乱码的方法

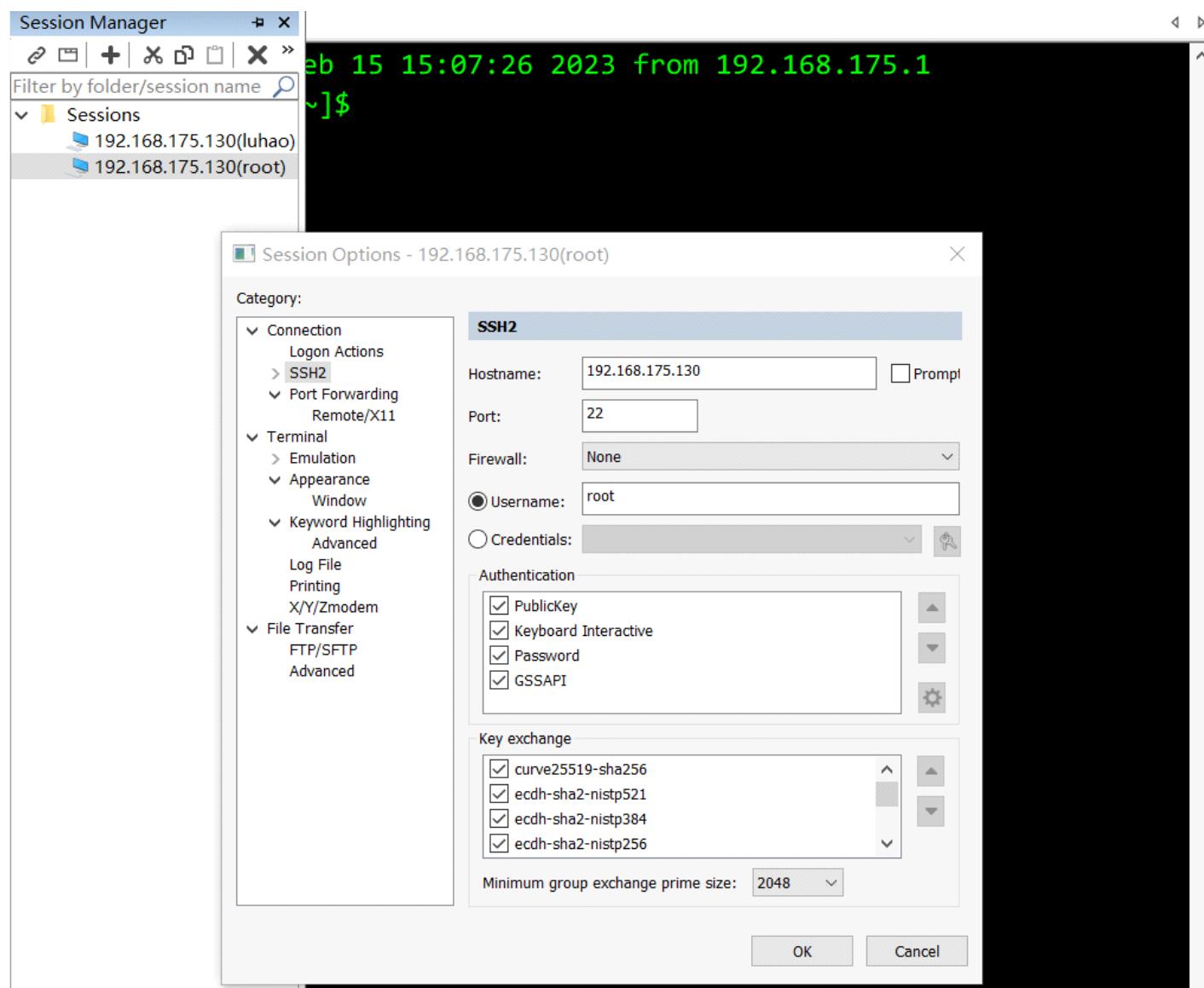
来自 <https://blog.csdn.net/qg_45780458/article/details/123187459>



屏幕剪辑的捕获时间: 2023/2/13 13:59

改成如上图这样，可以登录普通账号。

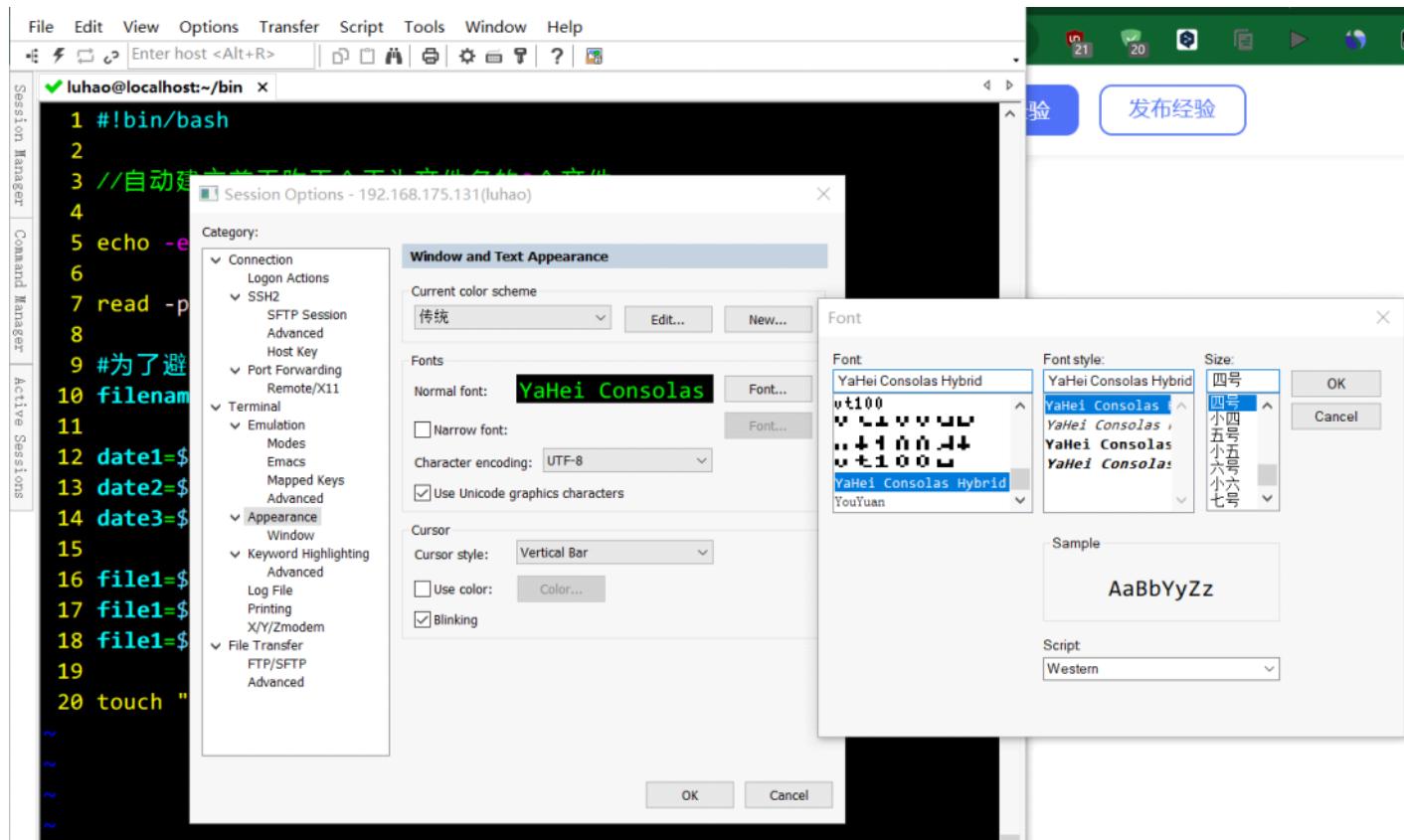
进一步修改：



屏幕剪辑的捕获时间: 2023/2/15 15:10

如上图, 已添加root用户的远程登录

修改字体与光标样式:



屏幕剪辑的捕获时间: 2023/3/24 11:33

出现如下情况:

```
[root@localhost ~]# gcc code1.c  
bash: gcc: 未找到命令...
```

屏幕剪辑的捕获时间: 2023/3/6 14:50

解决:

```
[root@localhost ~]# yum install gcc  
已加载插件：fastestmirror, langpacks  
Determining fastest mirrors  
* base: mirrors.bfsu.edu.cn
```

屏幕剪辑的捕获时间: 2023/3/6 14:51

vim的代替物:

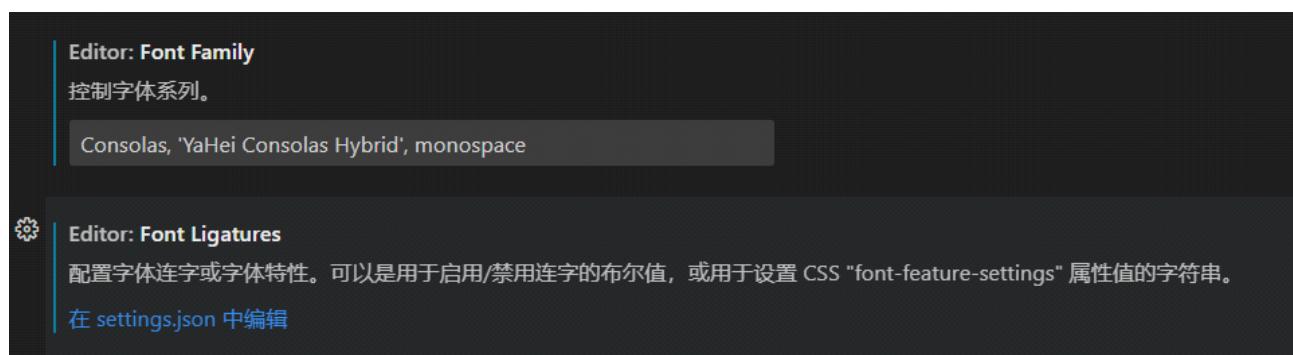
SecureFX配合VS code

The screenshot shows a dual-pane interface of VS Code. On the left is a terminal window titled 'luhao@localhost:~/bin' displaying a bash script execution. On the right is a code editor window titled '\$ ans_yn.sh' containing a shell script named 'ans_yn.sh'. The script prompts the user for input ('请输入您的选择(y/n):') and handles responses ('y', 'n', 'Y', 'N', 'interrupt'). It also includes error handling for invalid inputs.

```
#!/bin/bash
#显示用户的选择
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/~/.bin
read -p "请输入您的选择(y/n):" yn
#错误写法
if [ "${yn}"=="Y" ] || [ "${yn}"=="y" ];then
    echo "continue"
    exit 0
fi
if [ "${yn}" == "Y" ] || [ "${yn}" == "y" ];then
    echo "continue"
    exit 0
fi
if [ "${yn}" == "N" ] || [ "${yn}" == "n" ];then
    echo "interrupt"
    exit 0
fi
echo "我不能理解您的选择" && exit 0
```

屏幕剪辑的捕获时间: 2023/3/26 11:16

修改VS Code的字体:



屏幕剪辑的捕获时间: 2023/3/26 14:01

The screenshot shows the full content of the 'settings.json' file in the VS Code editor. It includes the previously highlighted properties and other standard settings like 'autoGuessEncoding' and 'fontLigatures'.

```
{
    "files.autoGuessEncoding": true,
    "editor.fontSize": 16,
    "security.workspace.trust.enabled": false,
    "editor.fontLigatures": false,
    "editor.fontWeight": "bold",
    "editor.fontFamily": "Consolas, 'YaHei Consolas Hybrid', monospace"
}
```

屏幕剪辑的捕获时间: 2023/3/26 14:02

第4章

2022年4月7日 11:00

```
[luhao@localhost ~]$ date  
2023年 02月 13日 星期一 14:23:22 CST
```

SecureCRT中可以正常显示中文，但CentOS中是全英文

屏幕剪辑的捕获时间: 2023/2/13 14:24

ctrl+alt+F2~F6 登录tty2~tty6

[Tab] 命令补全，文件名补全

ctrl+c 中断正在运行中的程序

ctrl+d 等价于输入exit

[Shift]+{[Page Up] | [Page Down]}

命令 --help 求助说明

man 命令或文件 操作说明，查看文件格式

/word 以在man page中查找word这个关键字

man -f 命令或文件 查和命令或文件有关的说明文件，相当于找命令或文件的完整名称

等价于whatis 命令或文件

```
[luhao@localhost ~]$ man -f cat  
cat (1)           - concatenate files and print on the standard ou...  
cat (1p)          - concatenate and print files
```

屏幕剪辑的捕获时间: 2023/3/9 16:11

man -k 命令或文件 在系统说明文件中找含关键字的所有说明文件

等价于apropos 命令或文件

```
[luhao@localhost ~]$ man -k cat  
alloca (3)        - allocate memory that is automatically freed  
backtrace (3)      - support for application self-debugging  
backtrace_symbols (3) - support for application self-debugging  
backtrace_symbols_fd (3) - support for application self-debugging  
calloc (3)         - allocate and free dynamic memory
```

屏幕剪辑的捕获时间: 2023/3/9 16:14

查询并打开指定的文件

```
[luhao@localhost ~]$ man -f cat
cat (1)           - concatenate files and print on the standard ou..
cat (1p)          - concatenate and print files
[luhao@localhost ~]$ man 1p cat
```

屏幕剪辑的捕获时间: 2023/3/30 9:46

info 命令 以多个可超链接的节点的界面来查看命令的说明文件,
相当于网页版说明文件

```
[luhao@localhost ~]$ man -f info
info (1)           - read Info documents
info (5)           - readable online documentation
[luhao@localhost ~]$ info 5 info
```

屏幕剪辑的捕获时间: 2023/3/9 16:16

/usr/share/doc/ 额外的说明文件

指令名称后括号内的数字的含义:

1 Executable programs or shell commands	用户在 shell 环境中可以操作的指令或可执行文件
2 System calls (functions provided by the kernel)	系统核心可呼叫的函数与工具等
3 Library calls (functions within program libraries)	一些常用的函数(function)与函数库(library), 大部分为 C 的函数库(libc)
4 Special files (usually found in /dev)	装置文件的说明, 通常在/dev 下的文件
5 File formats and conventions eg /etc/passwd	配置文件或者是某些文件的格式
6 Games	游戏(games)
7 Miscellaneous (including macro packages and convention)	惯例与协议等, 例如 Linux 文件系统、网络协议、ASCII code 等等的说明
8 System administration commands (usually only for root)	系统管理员可用的管理指令
9 Kernel routines [Non standard]	跟 kernel 有关的文件

ff(7)

屏幕剪辑的捕获时间: 2023/2/14 14:18

屏幕剪辑的捕获时间: 2023/2/14 14:55

日期时间: date

日历: cal

计算器: bc

文本编辑器: nano

数据同步写入硬盘: sync

重启: reboot

系统停止: halt

关机: shutdown、poweroff

第5章

2023年2月13日 19:29

ls 文件或目录
(详见第6章)

NAME

ls - list directory contents

SYNOPSIS

ls [OPTION]... [FILE]...

DESCRIPTION

List information about the FILEs (the current directory by default). Sort entries alphabetically if none of **-cftuvSUX** nor **--sort** is specified.

Mandatory arguments to long options are mandatory for short options too.

-a, --all

do not ignore entries starting with .

屏幕剪辑的捕获时间: 2023/2/15 14:16

-l use a long listing format

-d: 若为目录，则仅列出目录名，而不显示目录下的内容

-d, --directory

list directories themselves, not their contents

屏幕剪辑的捕获时间: 2023/3/17 15:04

文件属性与权限:

```
-rw-r--r--. 1 root      root     1864 May 4 18:01 initial-setup-ks.cfg  
[ 1 ][ 2 ][ 3 ][ 4 ][ 5 ][ 6 ] [ 7 ]  
[ 权限 ][ 连结 ][ 拥有者 ][ 群组 ][ 文件容量 ][ 修改日期 ][ 档名 ]
```

屏幕剪辑的捕获时间: 2023/2/15 14:43

- 第一个字符代表这个文件是『目录、文件或链接文件等等』:
 - 当为[**d**]则是目录, 例如上表档名为『.config』的那一行;
 - 当为[-]则是文件, 例如上表档名为『initial-setup-ks.cfg』那一行;
 - 若是[**1**]则表示为连结档(link file);
 - 若是[**b**]则表示为装置文件里面的可供储存的接口设备(可随机存取装置);
 - 若是[**c**]则表示为装置文件里面的串行端口设备, 例如键盘、鼠标(一次性读取装置)。
- 接下来的字符中, 以三个为一组, 且均为『**rwx**』 的三个参数的组合。其中, [r]代表可读(read)、[w]代表可写(write)、[x]代表可执行(execute)。要注意的是, 这三个权限的位置不会改变, 如果没有权限, 就会出现减号[-]而已。

屏幕剪辑的捕获时间: 2023/2/15 14:33

linux中cd命令为什么有时候加 / 反而进不去

时间: 2020-06-24

182*****5776

cd zxz在你当前目录下2113用的是相对路径cd/zxz进不去5261是4102因为你要进的是/root/zxz这是一个绝对路径在root下没有zxz你就进1653不去了顺那个cd/lost+found能进的也不是你home下的是进的/root下的lost+found

186*****5246

cd后第一个字符是/的表示绝对路径, 不带斜杠的表示相对路径。因为你当前路径在/下, 所以你cd/lost+found和cdlost+found效果一样

屏幕剪辑的捕获时间: 2023/2/15 16:43

(错误写法)

```
[root@localhost tmp]# cd /testing  
-bash: cd: /testing: 没有那个文件或目录  
[root@localhost tmp]# cd testing  
[root@localhost testing]#
```

(正确写法一)

```
[root@localhost tmp]# cd testing  
[root@localhost testing]# [REDACTED]
```

(正确写法二)

屏幕剪辑的捕获时间: 2023/2/15 16:45

```
[root@localhost tmp]# cd /tmp/testing  
[root@localhost testing]# [REDACTED]
```

屏幕剪辑的捕获时间: 2023/2/15 16:47

ls [选项] 文件(或多个文件或目录名)	查看列表
chgrp [-R] 用户组名 文件或目录名	修改所属的用户组, -R: 连同子目录下的所有内容都更新为该用户组
chown [-R] 账号名称 文件名或目录	修改拥有者
chown 账号名称:用户组 文件名	修改拥有者和用户组
chown .用户组 文件名	修改用户组
cp 源文件 目标文件	复制文件
chmod [-R] 权限数字 文件或目录	修改权限(法一, 数字法)
u	
g	r
chmod o +或-或= w 文件或目录	修改权限(法二, 符号法)
a	x

注: 文件的x是能否被执行, 目录的x是能否进入该目录

su - 用户名	切换身份
mkdir 目录名	建立新目录
touch 目录名/文件名	在目录下建立文件
cat 文件	读出文件内容

```
[root@localhost tmp]# ls -ld testing testing/testing  
drwxr--r--. 2 root root 21 2月 15 16:34 testing  
-rw-----. 1 root root 0 2月 15 16:34 testing/testing
```

屏幕剪辑的捕获时间: 2023/2/17 11:27

用luhao账号查看/tmp目录的权限

```
[luhao@localhost tmp]$ ls -ld /tmp  
drwxrwxrwt. 31 root root 4096 2月 17 13:22 /tmp
```

屏幕剪辑的捕获时间: 2023/2/17 13:23

屏幕剪辑的捕获时间: 2023/2/17 13:25

```
[luhao@localhost tmp]$ cd testing  
-bash: cd: testing: 权限不够
```

屏幕剪辑的捕获时间: 2023/2/17 13:27

用root修改目录的权限

```
[root@localhost ~]# chown luhao /tmp/testing
[root@localhost ~]#
[root@localhost ~]# ls -ld /tmp/testing
drwxr--r--. 2 luhao root 21 2月 15 16:34 /tmp/testing
```

屏幕剪辑的捕获时间: 2023/2/17 13:29

luhao可以进入该目录

```
[luhao@localhost tmp]$ cd testing  
[luhao@localhost testing]$
```

屏幕剪辑的捕获时间: 2023/2/17 13:31

testing文件不可以读，但可以删除

```
[luhao@localhost testing]$ ls -l  
总用量 0  
-rw-----. 1 root root 0 2月 15 16:34 testing  
[luhao@localhost testing]$ rm testing  
rm: 是否删除有写保护的普通空文件 "testing" ? n
```

屏幕剪辑的捕获时间: 2023/2/17 13:33

用户的家目录

```
[luhao@localhost /]$ cd ~luhao  
[luhao@localhost ~]$ ls  
1.txt 公共 模板 视频 图片 文档 下载 音乐 桌面
```

(写法二: cd ~)

(若当前在/home)

cd /var/log 绝对路径(一定由根目录写起)

cd ../var/log 相对路径(即相对于当前工作目录的路径)

第6章

2023年2月19日 16:14

cd [相对路径或绝对路径]	切换目录
pwd [-P]	显示当前目录 -P: 显示出真正路径, 而非链接路径
mkdir 目录名	建立一个新目录
mkdir -p 目录名/目录名/...	递归建立多个目录
mkdir -m 权限数字 目录名	建立目录并设置其权限
rmdir 目录名	删除一个空目录(即被删除的目录里不能存在其他目录或文件)
rmdir -p 目录名/目录名/...	递归删除多个空目录
rm 文件名	删除文件
rm -r 目录名	删除该目录下和目录下的所有东西 (目录下非空也可以直接删除)

.	此层目录
..	上层目录
-	前一个工作目录
~	目前使用者身份所在的家目录
~abc	abc这个使用者的家目录

```
[root@localhost testing]# rm -r test1
rm : 是否进入目录"test1"? y
rm : 是否进入目录"test1/test2"? y
rm : 是否删除目录 "test1/test2/test3" ? y
rm : 是否删除目录 "test1/test2" ? y
rm : 是否删除目录 "test1" ? y
[root@localhost testing]#
```

屏幕剪辑的捕获时间: 2023/2/20 16:48

echo	打印
mv 原目录/文件 目标目录	移动文件
ls	文件与目录的查看

```
[root@study ~]# ls [-aAdFlnrRSt] 文件名或目录名称..
[root@study ~]# ls [--color={never,auto,always}] 文件名或目录名称..
[root@study ~]# ls [--full-time] 文件名或目录名称..
选项与参数:
-a : 全部的文件, 连同隐藏档( 开头为 . 的文件) 一起列出来(常用)
-A : 全部的文件, 连同隐藏档, 但不包括 . 与 .. 这两个目录
-d : 仅列出目录本身, 而不是列出目录内的文件数据(常用)
-f : 直接列出结果, 而不进行排序 (ls 预设会以档名排序! )
-F : 根据文件, 目录等信息, 给予附加数据结构, 例如:
    *:代表可执行文件; /:代表目录; =:代表 socket 文件; -:代表 FIFO 文件;
-h : 将文件容量以人类较易读的方式(例如 GB, KB 等等)列出来;
-i : 列出 inode 号码, inode 的意义下一章将会介绍;
-l : 长数据串行出, 包含文件的属性与权限等等数据: (常用)
-n : 列出 UID 与 GID 而非使用者与群组的名称 (UID 与 GID 会在账号管理提到! )
-r : 将排序结果反向输出, 例如: 原本档名由小到大, 反向则为由大到小;
-R : 连同子目录内容一起列出来, 等于该目录下的所有文件都会显示出来;
-S : 以文件容量大小排序, 而不是用档名排序;
-t : 依时间排序, 而不是用档名。
--color=never : 不要依据文件特性给予颜色显示;
--color=always : 显示颜色
--color=auto : 让系统自行依据设定来判断是否给予颜色
--full-time : 以完整时间模式 (包含年、月、日、时、分) 输出
--time={atime,ctime} : 输出 access 时间或改变权限属性时间 (ctime)
    而非内容变更时间 (modification time)
```

原剪辑的捕获时间: 2023/2/21 13:55

蓝色: 目录

白色: 一般文件

cp [选项] 源目录/源文件 目标目录/目标文件(可与源文件名称不同)	源文件复制到目标目录下
cp [选项] 源目录/源文件 目标目录	源文件复制到目标目录下
cp [选项] 源目录/源文件 目标文件(可与源文件名称不同)	目标文件创建在当前目录下
cp [选项] 源文件 目标文件(可与源文件名称不同)	目标文件创建在当前目录下
cp [-ar] 源目录 目标目录	源目录下所有内容复制到目标目录下
cp [选项] 源目录/源文件 源目录/源文件 目标目录	多个文件一次复制到同个目录下

```
[root@study ~]# cp [-adfilprs] 来源文件(source) 目标文件(destination)
[root@study ~]# cp [options] source1 source2 source3 .... directory
选项与参数:
-a : 相当于 -dr --preserve=all 的意思, 至于 dr 请参考下列说明: (常用)
-d : 若来源文件为链接文件的属性(link file), 则复制链接文件属性而非文件本身;
-f : 为强制(force)的意思, 若目标文件已经存在且无法开启, 则移除后再尝试一次;
-i : 若目标文件(destination)已经存在时, 在覆盖时会先询问动作的进行(常用)
-l : 进行硬式连结(hard link)的连结档建立, 而非复制文件本身;
-p : 连同文件的属性(权限、用户、时间)一起复制过去, 而非使用默认属性(备份常用);
-r : 递归持续复制, 用于目录的复制行为: (常用)
-s : 复制成为符号链接文件 (symbolic link), 亦即『快捷方式』文件;
-u : destination 比 source 旧才更新 destination, 或 destination 不存在的情况下才复制。
--preserve=all : 除了 -p 的权限相关参数外, 还加入 SELinux 的属性, links, xattr 等也复制了。
最后需要注意的, 如果来源档有两个以上, 则最后一个目的文件一定要是『目录』才行!
```

屏幕剪辑的捕获时间: 2023/2/21 14:22

rm [选项] 文件或目录

移除文件或目录

```
[root@study ~]# rm [-fir] 文件或目录
```

选项与参数:

-f : 就是 force 的意思, 忽略不存在的文件, 不会出现警告诉讯息;

-i : 互动模式, 在删除前会询问使用者是否动作

-r : 递归删除啊! 最常用在目录的删除了! 这是非常危险的选项!!!

屏幕剪辑的捕获时间: 2023/2/21 15:05

通配符*

mv [选项] 源文件或源目录 源文件或源目录 目标目录

移动文件与目录, 或重命名

mv 目录名1 目录名2

将目录名1更名为目录名2

```
[root@study ~]# mv [-fiu] source destination
```

```
[root@study ~]# mv [options] source1 source2 source3 .... directory
```

选项与参数:

-f : force 强制的意思, 如果目标文件已经存在, 不会询问而直接覆盖;

-i : 若目标文件 (destination) 已经存在时, 就会询问是否覆盖!

-u : 若目标文件已经存在, 且 source 比较新, 才会更新 (update)

屏幕剪辑的捕获时间: 2023/2/21 15:13

basename

取得最后的文件名

dirname

取得目录名

```
[root@study ~]# basename /etc/sysconfig/network
```

network <== 很简单! 就取得最后的档名~

```
[root@study ~]# dirname /etc/sysconfig/network
```

/etc/sysconfig <== 取得的变成目录名了!

屏幕剪辑的捕获时间: 2023/3/21 11:29

cat [选项] 目录/文件名

查看文件内容

```
[root@study ~]# cat [-AbEnTv]
```

选项与参数:

-A : 相当于 -vET 的整合选项, 可列出一些特殊字符而不是空白而已;

选项与参数:

- A : 相当于 -vET 的整合选项, 可列出一些特殊字符而不是空白而已;
- b : 列出行号, 仅针对非空白行做行号显示; 空白行不标行号?
- E : 将结尾的断行字符 \$ 显示出来;
- n : 打印出行号, 连同空白行也会有行号, 与 -b 的选项不同;
- T : 将 [tab] 按键以 ^I 显示出来;
- v : 列出一些看不出来的特殊字符

屏幕剪辑的捕获时间: 2023/2/21 15:59

tac 目录/文件	从最后一行开始显示, 直到第一行(反向打印)
nl [选项] 文件	添加行号打印

[root@study ~]# nl [-bnw] 文件

选项与参数:

- b : 指定行号指定的方式, 主要有两种:
 - b a : 表示不论是否为空行, 也同样列出行号(类似 cat -n);
 - b t : 如果有空行, 空的那一行不要列出行号(默认值);
- n : 列出行号表示的方法, 主要有三种:
 - n ln : 行号在屏幕的最左方显示;
 - n rn : 行号在自己字段的最右方显示, 且不加 0 ;
 - n rz : 行号在自己字段的最右方显示, 且加 0 ;
- w : 行号字段的占用的字符数。

屏幕剪辑的捕获时间: 2023/2/22 14:35

```
[root@study ~]# nl -b a -n rz -w 3 /etc/issue
001      \S
002      Kernel \r on an \m
003
# 变成仅有 3 位数啰~
```

屏幕剪辑的捕获时间: 2023/2/22 14:36

more 文件	一页一页翻动(不能向前翻页)
---------	----------------

- 空格键 (space): 代表向下翻一页;
- Enter : 代表向下翻『一行』;
- /字符串 : 代表在这个显示的内容当中, 向下搜寻『字符串』这个关键词;
- :f : 立刻显示出文件名以及目前显示的行数;
- q : 代表立刻离开 more , 不再显示该文件内容。
- b 或 [ctrl]-b : 代表往回翻页, 不过这动作只对文件有用, 对管线无用。

屏幕剪辑的捕获时间: 2023/2/22 14:39

less 文件

一页一页翻动(能向前翻页)

- 空格键 : 向下翻动一页;
- [pagedown]: 向下翻动一页;
- [pageup] : 向上翻动一页;
- /字符串 : 向下搜寻『字符串』的功能;
- ?字符串 : 向上搜寻『字符串』的功能;
- n : 重复前一个搜寻 (与 / 或 ? 有关!)
- N : 反向的重复前一个搜寻 (与 / 或 ? 有关!)
- g : 前进到这个资料的第一行去;
- G : 前进到这个数据的最后一行去 (注意大小写);
- q : 离开 less 这个程序;

屏幕剪辑的捕获时间: 2023/2/22 14:42

head [-n number] 文件

前面number行打印

head [-n -number] 文件

后面number行不打印

tail [-n number] 文件

打印最后number行

tail [-n +number] 文件

打印number行后的内容

od [选项] 文件

打印非纯文本文件

[root@study ~]# od [-t TYPE] 文件

选项或参数:

-t : 后面可以接各种『类型 (TYPE)』的输出, 例如:

a : 利用默认的字符来输出;

c : 使用 ASCII 字符来输出

d[size] : 利用十进制(decimal)来输出数据, 每个整数占用 size bytes ;

f[size] : 利用浮点数(floating)来输出数据, 每个数占用 size bytes ;

o[size] : 利用八进制(octal)来输出数据, 每个整数占用 size bytes ;

x[size] : 利用十六进制(hexadecimal)来输出数据, 每个整数占用 size bytes ;

屏幕剪辑的捕获时间: 2023/2/22 15:03

使用管道, 实现打印输入字符对应的ASCII码

第0个字节

```
[luhao@localhost ~]$ echo abcABC | od -t dCc
00000000  97   98   99   65   66   67   10
          a     b     c     A     B     C    \n
```

```
[luhao@localhost ~]$ echo abcABC | od -t dCc  
0000000 97 98 99 65 66 67 10  
      a   b   c   A   B   C   \n  
0000007
```

屏幕剪辑的捕获时间: 2023/2/22 15:23

命令换行: 反斜杠后接回车键

命令分隔: 使用分号

```
[root@study ~]# date; ls -l /etc/man_db.conf ; ls -l --time=atime /etc/man_db.conf ; \  
> ls -l --time=ctime /etc/man_db.conf # 这两行其实是同一行喔! 用分号隔开
```

屏幕剪辑的捕获时间: 2023/2/22 15:28

touch [选项] 文件

可以创建并修改文件的mtime、 atime

```
[root@study ~]# touch [-acdmt] 文件  
选项与参数:  
-a : 仅修订 access time;  
-c : 仅修改文件的时间, 若该文件不存在则不建立新文件;  
-d : 后面可以接欲修订的日期而不用目前的日期, 也可以使用 --date="日期或时间"  
-m : 仅修改 mtime ;  
-t : 后面可以接欲修订的时间而不用目前的时间, 格式为[YYYYMMDDhhmm]
```

屏幕剪辑的捕获时间: 2023/2/22 16:25

文件的默认权限: -rw-rw-rw-

目录的默认权限: drwxrwxrwx

当前用户建立文件或目录时的权限: 默认值-umask的值

注: umask指的是默认值需要减掉的权限

umask 002

在默认权限下对others身份减掉写权限

chattr [+-=] [选项] 文件或目录

配置文件隐藏属性

```
[root@study ~]# chattr [+-=][ASacdstu] 文件或目录名称
选项与参数:
+ : 增加某一个特殊参数，其他原本存在参数则不动。
- : 移除某一个特殊参数，其他原本存在参数则不动。
= : 设定一定，且仅有后面接的参数

A : 当设定了 A 这个属性时，若你有存取此文件(或目录)时，他的访问时间 atime 将不会被修改，  
可避免 I/O 较慢的机器过度的存取磁盘。(目前建议使用文件系统挂载参数处理这个项目)
S : 一般文件是异步写入磁盘的(原理请参考前一章 sync 的说明)，如果加上 S 这个属性时，  
当你进行任何文件的修改，该更动会『同步』写入磁盘中。
a : 当设定 a 之后，这个文件将只能增加数据，而不能删除也不能修改数据，只有 root 才能设定这属性
c : 这个属性设定之后，将会自动的将此文件『压缩』，在读取的时候将会自动解压缩，  
但是在储存的时候，将会先进行压缩后再储存(看来对于大文件似乎蛮有用的！)
d : 当 dump 程序被执行的时候，设定 d 属性将可使该文件(或目录)不会被 dump 备份
i : 这个 i 可就很厉害了！他可以让一个文件『不能被删除、改名、设定连结也无法写入或新增数据！』  
对于系统安全性有相当大的帮助！只有 root 能设定此属性
s : 当文件设定了 s 属性时，如果这个文件被删除，他将会被完全的移除出这个硬盘空间，  
所以如果误删了，完全无法救回来了喔！
u : 与 s 相反的，当使用 u 来配置文件案时，如果该文件被删除了，则数据内容其实还存在磁盘中，  
可以使用来救援该文件喔！

注意 1：属性设定常见的是 a 与 i 的设定值，而且很多设定值必须要身为 root 才能设定
注意 2：xfs 文件系统仅支援 AadiS 而已
```

屏幕剪辑的捕获时间: 2023/2/23 15:24

isattr [选项] 文件或目录 显示文件隐藏属性

```
[root@study ~]# lsattr [-adR] 文件或目录
选项与参数:
-a : 将隐藏文件的属性也秀出来；
-d : 如果接的是目录，仅列出目录本身的属性而非目录内的文件名；
-R : 连同子目录的数据也一并列出来！
```

```
[root@study tmp]# chattr +aiS attrtest
[root@study tmp]# lsattr attrtest
--S-ia----- attrtest
```

屏幕剪辑的捕获时间: 2023/2/23 15:27

文件的特殊权限:

SUID: 4

SGID: 2

SBIT: 1

(书P198)

- SUID 权限仅对二进制程序(binary program)有效;
- 执行者对于该程序需要具有 x 的可执行权限;
- 本权限仅在执行该程序的过程中有效 (run-time);
- 执行者将具有该程序拥有者 (owner) 的权限。

屏幕剪辑的捕获时间: 2023/3/31 10:39

- SGID 对二进制程序有用;
 - 程序执行者对于该程序来说, 需具备 x 的权限;
 - 执行者在执行的过程中将会获得该程序群组的支持!
-
- 用户若对于此目录具有 r 与 x 的权限时, 该用户能够进入此目录;
 - 用户在此目录下的有效群组(effective group)将会变成该目录的群组;
 - 用途: 若用户在此目录下具有 w 的权限(可以新建文件), 则使用者所建立的新文件, 该新文件的群组与此目录的群组相同。

屏幕剪辑的捕获时间: 2023/3/31 10:40

SBIT:

- 当用户对于此目录具有 w, x 权限, 亦即具有写入的权限时;
- 当用户在该目录下建立文件或目录时, 仅有自己与 root 才有权力删除该文件

屏幕剪辑的捕获时间: 2023/3/31 10:41

```
[root@study tmp]# chmod 4755 test; ls -l test <==加入具有 SUID 的权限  
-rwsr-xr-x 1 root root 0 Jun 16 02:53 test
```

```
[root@study tmp]# chmod 1755 test; ls -l test <==加入 SBIT 的功能!  
-rwxr-xr-t 1 root root 0 Jun 16 02:53 test  
[root@study tmp]# chmod 7666 test; ls -l test <==具有空的 SUID/SGID 权限  
-rw$rw$rwT 1 root root 0 Jun 16 02:53 test
```

屏幕剪辑的捕获时间: 2023/2/23 16:05

file 文件

观察文件类型

```
[luhao@localhost ~]$ file ~/.bashrc
/home/luhao/.bashrc: ASCII text
```

屏幕剪辑的捕获时间: 2023/2/23 16:13

which [-a] 命令名

根据PATH环境变量规定的路径, 查找命令的完整文件名

```
[root@study ~]# which [-a] command
```

选项或参数:

-a : 将所有由 PATH 目录中可以找到的指令均列出, 而不止第一个被找到的指令名称

屏幕剪辑的捕获时间: 2023/2/23 16:41

whereis是找系统中某些特定目录下的文件

locate是利用数据库来查找文件名

```
[root@study ~]# whereis [-bmsu] 文件或目录名
```

选项与参数:

- l : 可以列出 whereis 会去查询的几个主要目录而已
- b : 只找 binary 格式的文件
- m : 只找在说明文件 manual 路径下的文件
- s : 只找 source 来源文件
- u : 搜寻不在上述三个项目当中的其他特殊文件

屏幕剪辑的捕获时间: 2023/2/23 16:46

```
[root@study ~]# locate [-ir] keyword
```

选项与参数:

- i : 忽略大小写的差异;
- c : 不输出档名, 仅计算找到的文件数量
- l : 仅输出几行的意思, 例如输出五行则是 -l 5
- S : 输出 locate 所使用的数据库文件的相关信息, 包括该数据库纪录的文件/目录数量等
- r : 后面可接正规表示法的显示方式

屏幕剪辑的捕获时间: 2023/2/23 16:52

更新locate数据库: updatedb

find

```
[root@study ~]# find [PATH] [option] [action]
```

选项与参数：

1. 与时间有关的选项：共有 -atime, -ctime 与 -mtime，以 -mtime 说明

-mtime n : n 为数字，意义为在 n 天之前的『一天之内』被更动过内容的文件；

-mtime +n : 列出在 n 天之前(不含 n 天本身)被更动过内容的文件档名；

-mtime -n : 列出在 n 天之内(含 n 天本身)被更动过内容的文件档名。

-newer file : file 为一个存在的文件，列出比 file 还要新的文件档名

其中，[PATH]可以指定多个目录

范例一：将过去系统上面 24 小时内有更动过内容 (mtime) 的文件列出

```
[root@study ~]# find / -mtime 0
```

那个 0 是重点！0 代表目前的时间，所以，从现在开始到 24 小时前，

有变动过内容的文件都会被列出来！那如果是三天前的 24 小时内？

find / -mtime 3 有变动过的文件都被列出的意思！

范例二：寻找 /etc 底下的文件，如果文件日期比 /etc/passwd 新就列出

```
[root@study ~]# find /etc -newer /etc/passwd
```

-newer 用在分辨两个文件之间的新旧关系是很有用的！

选项与参数：

2. 与使用者或组名有关的参数：

-uid n : n 为数字，这个数字是用户的账号 ID，亦即 UID，这个 UID 是记录在 /etc/passwd 里面与账号名称对应的数字。这方面我们会在第四篇介绍。

-gid n : n 为数字，这个数字是组名的 ID，亦即 GID，这个 GID 记录在 /etc/group，相关的介绍我们会第四篇说明～

-user name : name 为使用者账号名称喔！例如 dmtsa

-group name: name 为组名喔，例如 users :

-nouser : 寻找文件的拥有者不存在 /etc/passwd 的人！

-nogroup : 寻找文件的拥有群组不存在于 /etc/group 的文件！

当你自行安装软件时，很可能该软件的属性当中并没有文件拥有者，

这是可能的！在这个时候，就可以使用 -nouser 与 -nogroup 搜寻。

范例三：搜寻 /home 底下属于 dmtsaI 的文件

```
[root@study ~]# find /home -user dmtsaI  
# 这个东西也很有用的～当我们要找出任何一个用户在系统当中的所有文件时，  
# 就可以利用这个指令将属于某个使用者的所有文件都找出来喔！
```

范例四：搜寻系统中不属于任何人的文件

```
[root@study ~]# find / -nouser  
# 透过这个指令，可以轻易的就找出那些不太正常的文件。如果有找到不属于系统任何人的文件时，  
# 不要太紧张，那有时候是正常的～尤其是你曾经以原始码自行编译软件时。
```

选项与参数：

3. 与文件权限及名称有关的参数：

-name filename：搜寻文件名为 filename 的文件；

-size [+|-]SIZE：搜寻比 SIZE 还要大(+)或小(-)的文件。这个 SIZE 的规格有：

c：代表 byte， k：代表 1024bytes。所以，要找比 50KB

还要大的文件，就是『 -size +50k 』

-type TYPE：搜寻文件的类型为 TYPE 的，类型主要有：一般正规文件 (f)，装置文件 (b, c)，
目录 (d)，连结档 (l)，socket (s)，及 FIFO (p) 等属性。

-perm mode：搜寻文件权限『刚好等于』 mode 的文件，这个 mode 为类似 chmod
的属性值，举例来说， -rwsr-xr-x 的属性为 4755 ！

-perm -mode：搜寻文件权限『必须要全部囊括 mode 的权限』的文件，举例来说，

我们要搜寻 `-rwxr--r--`，亦即 0744 的文件，使用 `-perm -0744`。

当一个文件的权限为 `-rwsr-xr-x`，亦即 4755 时，也会被列出来，

因为 `-rwsr-xr-x` 的属性已经囊括了 `-rwxr--r--` 的属性了。

`-perm /mode`：搜寻文件权限『包含任一 mode 的权限』的文件，举例来说，我们搜寻 `-rwxr-Xr-X`，亦即 `-perm /755` 时，但一个文件属性为 `-rw-----` 也会被列出来，因为他有 `-rw....` 的属性存在！

范例五：找出档名为 `passwd` 这个文件

```
[root@study ~]# find / -name passwd
```

范例五-1：找出文件名包含了 `passwd` 这个关键词的文件

```
[root@study ~]# find / -name "*passwd*"
```

利用这个 `-name` 可以搜寻档名啊！默认是完整文件名，如果想要找关键词，

可以使用类似 * 的任意字符来处理

范例六：找出 `/run` 目录下，文件类型为 `Socket` 的档名有哪些？

```
[root@study ~]# find /run -type s
```

这个 `-type` 的属性也很有帮助喔！尤其是要找出那些怪异的文件，

例如 `socket` 与 `FIFO` 文件，可以用 `find /run -type p` 或 `-type s` 来找！

范例七：搜寻文件当中含有 `SGID` 或 `SUID` 或 `SBIT` 的属性

```
[root@study ~]# find / -perm /7000
```

所谓的 7000 就是 `---s---s---t`，那么只要含有 `s` 或 `t` 的就列出，所以当然要使用 `/7000`，

使用 `-7000` 表示要同时含有 `---s---s---t` 的所有三个权限。而只需要任意一个，就是 `/7000` ~ 瞭乎？

屏幕剪辑的捕获时间: 2023/2/24 11:38

查找含`passwd`的关键字：“`*passwd*`” 其中*是通配符

4. 额外可进行的动作：

`-exec command`：`command` 为其他指令，`-exec` 后面可再接额外的指令来处理搜寻到的结果。

`-print`：将结果打印到屏幕上，这个动作是预设动作！

范例八：将上个范例找到的文件使用 `ls -l` 列出来～

```
[root@study ~]# find /usr/bin /usr/sbin -perm /7000 -exec ls -l {} \;
```

注意到，那个 `-exec` 后面的 `ls -l` 就是额外的指令，指令不支持命令别名，

屏幕剪辑的捕获时间: 2023/2/26 14:42

不列出子目录占用的内存

第7章

2023年3月4日 14:56

ls -i

观察文件占用的inode号码

查看安装的Linux支持的文件系统

```
[root@localhost ~]# ls -l /lib/modules/$(uname -r)/kernel/fs
总用量 20
-rw-r--r--. 1 root root 6136 1月 26 00:54 binfmt_misc.ko.xz
drwxr-xr-x. 2 root root 25 2月 11 16:45 btrfs
drwxr-xr-x. 2 root root 30 2月 11 16:45 cachefiles
drwxr-xr-x. 2 root root 24 2月 11 16:45 ceph
drwxr-xr-x. 2 root root 24 2月 11 16:45 cifs
drwxr-xr-x. 2 root root 26 2月 11 16:45 cramfs
drwxr-xr-x. 2 root root 23 2月 11 16:45 dlm
drwxr-xr-x. 2 root root 26 2月 11 16:45 exofs
drwxr-xr-x. 2 root root 24 2月 11 16:45 ext4
drwxr-xr-x. 2 root root 60 2月 11 16:45 fat
drwxr-xr-x. 2 root root 27 2月 11 16:45 fscache
drwxr-xr-x. 2 root root 42 2月 11 16:45 fuse
drwxr-xr-x. 2 root root 24 2月 11 16:45 gfs2
drwxr-xr-x. 2 root root 25 2月 11 16:45 iso9660
```

屏幕剪辑的捕获时间: 2023/2/28 15:47

cat /proc/filesystems

查看系统已经加载到内存中支持的文件系统

```
[root@localhost ~]# cat /proc/filesystems
nodev    sysfs
nodev    rootfs
nodev    ramfs
nodev    bdev
nodev    proc
nodev    cgroup
nodev    cpuset
nodev    tmpfs
nodev    devtmpfs
nodev    debugfs
nodev    securityfs
nodev    sockfs
nodev    dax
nodev    hnfs
```

屏幕剪辑的捕获时间: 2023/2/28 20:06

df [选项] [目录或文件名]

列出文件系统的整体磁盘使用量

```
[root@study ~]# df [-ahikHTm] [目录或文件名]
```

选项与参数:

-a : 列出所有的文件系统, 包括系统特有的 /proc 等文件系统;

-k : 以 KBytes 的容量显示各文件系统;

-m : 以 MBytes 的容量显示各文件系统;

-h : 以人们较易阅读的 GBytes, MBytes, KBytes 等格式自行显示;

-H : 以 M=1000K 取代 M=1024K 的进位方式;

-T : 连同该 partition 的 filesystem 名称 (例如 xfs) 也列出;

-i : 不用磁盘容量, 而以 inode 的数量来显示

```
[root@localhost ~]# df -T /boot
文件系统      类型    1K-块   已用   可用   已用% 挂载点
/dev/sda1      xfs     1038336 236672 801664    23% /boot
```

屏幕剪辑的捕获时间: 2023/2/28 20:16

du [选项] [文件或目录名称]

查看目录所占的磁盘空间

```
[root@study ~]# du [-ahskm] 文件或目录名称
```

选项与参数:

- a : 列出所有的文件与目录容量, 因为默认仅统计目录底下的文件量而已。
- h : 以人们较易读的容量格式 (G/M) 显示;
- s : 列出总量而已, 而不列出每个各别的目录占用容量;
- S : 不包括子目录下的总计, 与 -s 有点差别。
- k : 以 KBytes 列出容量显示;
- m : 以 MBytes 列出容量显示;

```
[root@localhost ~]# du
```

```
4      ./cache/abrt
4      ./cache/gdm
8      ./cache/imsettings
0      ./cache/libgweather
0      ./cache/evolution/addressbook/trash
0      ./cache/evolution/addressbook
0      ./cache/evolution/calendar/trash
0      ./cache/evolution/calendar
0      ./cache/evolution/mail/trash
0      / cache/evolution/mail
```

屏幕剪辑的捕获时间: 2023/2/28 20:28

```
[root@localhost ~]# du -s  
27108 .  
[root@localhost ~]# du -S  
4      ./cache/abrt  
4      ./cache/gdm  
8      ./cache/imsettings  
0      ./cache/libgweather  
0      ./cache/evolution/addressbook/trash  
0      ./cache/evolution/addressbook  
0      ./cache/evolution/calendar/trash  
0      ./cache/evolution/calendar  
0      ./cache/evolution/mail/trash  
0      ./cache/evolution/mail
```

屏幕剪辑的捕获时间: 2023/2/28 20:42

-S 显示某目录下的全部数据, 计算文件夹大小的时候不计入子文件夹的大小。

来自 <https://www.zhihu.com/question/39735415/answer/104190858>

-s 显示某目录占用多少容量

ln 源目录名/源文件名 目标目录名/目标文件名 建立硬链接

```
[root@localhost ~]# ln /etc/crontab .  
[root@localhost ~]#  
[root@localhost ~]#  
[root@localhost ~]# ll -i /etc/crontab ./crontab  
16954925 -rw-r--r--. 2 root root 451 6月 10 2014 ./crontab  
16954925 -rw-r--r--. 2 root root 451 6月 10 2014 /etc/crontab
```

屏幕剪辑的捕获时间: 2023/3/1 14:21

ln -s 源目录名/源文件名 目标目录名/目标文件名 建立符号链接

```
[root@localhost ~]# ln -s /etc/crontab ./crontab2
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# ll -i /etc/crontab ./crontab2
33940814 lrwxrwxrwx. 1 root root 12 3月 1 14:36 ./crontab2 -> /etc/crontab
16954925 -rw-r--r--. 2 root root 451 6月 10 2014 /etc/crontab
```

屏幕剪辑的捕获时间: 2023/3/1 14:39

建立一个新目录时，新的目录链接数为2，而上层目录的链接数会增加1

lsblk [选项] [设备名] 查看磁盘列表

```
[root@study ~]# lsblk [-dfimpt] [device]
```

选项与参数：

- d : 仅列出磁盘本身，并不会列出该磁盘的分区数据
- f : 同时列出该磁盘内的文件系统名称
- i : 使用 ASCII 的线段输出，不要使用复杂的编码（再某些环境下很有用）
- m : 同时输出该装置在 /dev 底下的权限数据（rwx 的数据）
- p : 列出该装置的完整文件名！而不是仅列出最后的名字而已。
- t : 列出该磁盘装置的详细数据，包括磁盘队列机制、预读写的数据量大小等

屏幕剪辑的捕获时间: 2023/3/1 15:13

查看UUID（通用唯一标识符）：lsblk -f 或 blkid

parted 磁盘名称 print 列出磁盘相关信息

磁盘分区：GPT分区表使用gdisk，MBR分区表使用fdisk

```
[root@study ~]# fdisk /dev/sda
```

Command (**m** for help): **m** <== 输入 **m** 后，就会看到底下这些指令介绍
Command action

- a toggle a bootable flag
- b edit bsd disklabel
- c toggle the dos compatibility flag
- d delete a partition** <==删除一个 partition
- l list known partition types
- m print this menu**
- n add a new partition** <==新增一个 partition
- o create a new empty DOS partition table
- p print the partition table** <==在屏幕上显示分区表
- q quit without saving changes** <==不储存离开 fdisk 程序
- s create a new empty Sun disklabel
- t change a partition's system id
- u change display/entry units
- v verify the partition table
- w write table to disk and exit** <==将刚刚的动作写入分区表
- x extra functionality (experts only)

屏幕剪辑的捕获时间: 2023/3/2 16:29

partprobe -s

更新以显示分区表

mkfs.xfs

磁盘格式化（创建文件系统）为xfs

```
[root@study ~]# mkfs.xfs [-b bsize] [-d parms] [-i parms] [-l parms] [-L label] [-f] \
[-r parms] 装置名称
```

选项与参数：

关于单位：底下只要谈到『数值』时，没有加单位则为 bytes 值，可以用 k,m,g,t,p (小写)等来解释
比较特殊的是 s 这个单位，它指的是 sector 的『个数』喔！

-b : 后面接的是 block 容量，可由 512 到 64k，不过最大容量限制为 Linux 的 4k 哟！

-d : 后面接的是重要的 data section 的相关参数值，主要的值有：

agcount=数值 : 设定需要几个储存群组的意思(AG)，通常与 CPU 有关

agsize=数值 : 每个 AG 设定为多少容量的意思，通常 agcount/agsize 只选一个设定即可

屏幕剪辑的捕获时间: 2023/3/2 16:27

mkfs[tab][tab]	查看系统支持的文件系统的格式化的类型
mkfs -t 选择的文件系统格式 设备名称	磁盘格式化

xfs_repair [选项] 设备名称	文件系统错乱时，使用，以进行文件系统检验
----------------------	----------------------

mount	将文件系统挂载到Linux系统
-------	-----------------

```
[root@study ~]# mount -a
[root@study ~]# mount [-l]
[root@study ~]# mount [-t 文件系统] LABEL='' 挂载点
[root@study ~]# mount [-t 文件系统] UUID='' 挂载点 # 鸟哥近期建议用这种方式喔!
[root@study ~]# mount [-t 文件系统] 装置文件名 挂载点
```

选项与参数：

-a : 依照配置文件 /etc/fstab 的数据将所有未挂载的磁盘都挂载上来

-l : 单纯的输入 mount 会显示目前挂载的信息。加上 -l 可增列 Label 名称！

-t : 可以加上文件系统种类来指定欲挂载的类型。常见的 Linux 支持类型有：xfs, ext3, ext4, reiserfs, vfat, iso9660(光盘格式), nfs, cifs, smbfs (后三种为网络文件系统类型)

-n : 在默认的情况下，系统会将实际挂载的情况实时写入 /etc/mtab 中，以利其他程序的运作。
但在某些情况下(例如单人维护模式)为了避免问题会刻意不写入。此时就得要使用 -n 选项。

-o : 后面可以接一些挂载时额外加上的参数！比方说账号、密码、读写权限等：

- async, sync: 此文件系统是否使用同步写入 (sync) 或异步 (async) 的内存机制，请参考[文件系统运作方式](#)。预设为 async。
- atime,noatime: 是否修订文件的读取时间(atime)。为了效能，某些时刻可使用 noatime
- ro, rw: 挂载文件系统成为只读(ro) 或可擦写(rw)
- auto, noauto: 允许此 filesystem 被以 mount -a 自动挂载(auto)
- dev, nodev: 是否允许此 filesystem 上，可建立装置文件？ dev 为可允许
- suid, nosuid: 是否允许此 filesystem 含有 uid/gid 的文件格式？
- exec, noexec: 是否允许此 filesystem 上拥有可执行 binary 文件？
- user, nouser: 是否允许此 filesystem 让任何使用者执行 mount ？一般来说，mount 仅有 root 可以进行，但下达 user 参数，则可让一般 user 也能够对此 partition 进行 mount 。
- defaults: 默认值为：rw, suid, dev, exec, auto, nouser, and async

remount: 重新挂载，这在系统出错，或重新更新参数时，很有用！

屏幕剪辑的捕获时间: 2023/3/2 13:59

范例：使用相同的方式，将 /dev/vda5 挂载于 /data/ext4

```
[root@study ~]# blkid /dev/vda5  
/dev/vda5: UUID="899b755b-1da4-4d1d-9b1c-f762adb798e1" TYPE="ext4"  
  
[root@study ~]# mkdir /data/ext4  
[root@study ~]# mount UUID="899b755b-1da4-4d1d-9b1c-f762adb798e1" /data/ext4  
[root@study ~]# df /data/ext4
```

屏幕剪辑的捕获时间: 2023/3/2 14:22

mount --bind 源目录名 目标目录名

将某个目录挂载到其他目录

umount

将设备文件卸载

mknod

设置硬件文件名

```
[root@study ~]# mknod 装置文件名 [bcp] [Major] [Minor]
```

选项与参数:

装置种类:

b : 设定装置名称成为一个周边储存设备文件, 例如磁盘等;

c : 设定装置名称成为一个周边输入设备文件, 例如鼠标/键盘等;

p : 设定装置名称成为一个 FIFO 文件;

Major : 主要装置代码;

Minor : 次要装置代码;

屏幕剪辑的捕获时间: 2023/3/2 14:35

xfs_admin

修改XFS文件系统的UUID和Label name

```
[root@study ~]# xfs_admin [-lu] [-L label] [-U uuid] 装置文件名
```

选项与参数:

-l : 列出这个装置的 label name

-u : 列出这个装置的 UUID

-L : 设定这个装置的 Label name

屏幕剪辑的捕获时间: 2023/3/2 14:38

uuidgen

产生新的UUID

范例: 利用 uuidgen 产生新 UUID 来设定 /dev/vda4, 并测试挂载

```
[root@study ~]# umount /dev/vda4          # 使用前, 请先卸除!
```

```
[root@study ~]# uuidgen
```

e0fa7252-b374-4a06-987a-3cb14f415488 # 很有趣的指令! 可以产生新的 UUID 呀!

```
[root@study ~]# xfs_admin -u /dev/vda4
```

UUID = e0a6af55-26e7-4cb7-a515-826a8bd29e90

```
[root@study ~]# xfs_admin -U e0fa7252-b374-4a06-987a-3cb14f415488 /dev/vda4
```

Clearing log and setting UUID

屏幕剪辑的捕获时间: 2023/3/2 14:43

tune2fs

修改ext4的label name与UUID

设置启动挂载: 修改/etc/fstab文件中的内容

例题：

假设我们要将 /dev/vda4 每次开机都自动挂载到 /data/xfs , 该如何进行?

答：

首先, 请用 nano 将底下这一行写入 /etc/fstab 最后面中;

```
[root@study ~]# nano /etc/fstab
UUID="e0fa7252-b374-4a06-987a-3cb14f415488" /data/xfs xfs defaults 0 0
```

屏幕剪辑的捕获时间: 2023/3/2 16:37

镜像文件不刻录就挂载使用:

```
[root@study ~]# ll -h /tmp/CentOS-7.0-1406-x86_64-DVD.iso
-rw-r--r--. 1 root root 3.9G Jul 7 2014 /tmp/CentOS-7.0-1406-x86_64-DVD.iso
# 看到上面的结果吧! 这个文件就是映象档, 文件非常的大吧!
```

```
[root@study ~]# mkdir /data/centos_dvd
[root@study ~]# mount -o loop /tmp/CentOS-7.0-1406-x86_64-DVD.iso /data/centos_dvd
[root@study ~]# df /data/centos_dvd
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/loop0        4050860  4050860         0 100% /data/centos_dvd
# 就是这个项目! .iso 映象文件内的所有数据可以在 /data/centos_dvd 看到!
```

屏幕剪辑的捕获时间: 2023/3/2 15:04

在原分区中制作出想要的分区: 利用dd建立一个空文件, 对文件格式化后进行挂载(书P252)

创建内存交换分区(P253)

使用文件创建内存交换分区(P254)

parted

同时支持MBR和GPT的分区

```
[root@study ~]# parted [装置] [指令 [参数]]
```

选项与参数:

指令功能:

新增分区: mkpart [primary|logical|extended] [ext4|vfat|xfs] 开始 结束

显示分区: print

删除分区: rm [partition]

范例一：以 parted 列出目前本机的分区表资料

```
[root@study ~]# parted /dev/vda print
```

Model: Virtio Block Device (virtblk)	<==磁盘接口与型号
Disk /dev/vda: 42.9GB	<==磁盘文件名与容量
Sector size (logical/physical): 512B/512B	<==每个扇区的大小
Partition Table: gpt	<==是 GPT 还是 MBR 分区
Disk Flags: pmbr_boot	

屏幕剪辑的捕获时间: 2023/3/2 16:13

umount

卸载

parted 设备名称 rm 号码

删除分区

第8章

2023年3月4日 14:58

常见的压缩文件的扩展名：

*.Z	compress 程序压缩的文件；
*.zip	zip 程序压缩的文件；
*.gz	gzip 程序压缩的文件；
*.bz2	bzip2 程序压缩的文件；
*.xz	xz 程序压缩的文件；
*.tar	tar 程序打包的数据，并没有压缩过；
*.tar.gz	tar 程序打包的文件，其中并且经过 gzip 的压缩
*.tar.bz2	tar 程序打包的文件，其中并且经过 bzip2 的压缩
*.tar.xz	tar 程序打包的文件，其中并且经过 xz 的压缩

gzip [选项] 文件名

gzip程序压缩文件

```
[dmtsai@study ~]$ gzip [-cdtv#] 檔名
```

```
[dmtsai@study ~]$ zcat 檔名.gz
```

选项与参数：

-c : 将压缩的数据输出到屏幕上，可透过数据流重导向来处理；

-d : 解压缩的参数；

-t : 可以用来检验一个压缩文件的一致性～看看文件有无错误；

-v : 可以显示出原文件/压缩文件案的压缩比等信息；

-# : # 为数字的意思，代表压缩等级，-1 最快，但是压缩比最差、-9 最慢，但是压缩比最好！预设是 -6

屏幕剪辑的捕获时间: 2023/3/4 15:11

```
[luhao@localhost tmp]$ gzip -v services
services:          79.7% -- replaced with services.gz
[luhao@localhost tmp]$ ls -l /etc/services services.gz
-rw-r--r--. 1 root  root  670293 6月    7 2013 /etc/services
-rw-r--r--. 1 luhao luhao 136088 3月   15:15 services.gz
```

屏幕剪辑的捕获时间: 2023/3/4 15:19

zcat、zmore、zless读取纯文本文件被压缩后的文件的内容

gzip -d 压缩文件名

解压缩

gzip [-c] 文件名 > 压缩后的文件名

压缩文件并保留原本的文件

```
[luhao@localhost tmp]$ gzip -c services > services.gz
[luhao@localhost tmp]$
[luhao@localhost tmp]$
[luhao@localhost tmp]$ ls
services
services.gz
SSH-1MNOXkyuRs
```

屏幕剪辑的捕获时间: 2023/3/4 16:04

grep --color "leo" /etc/passwd

文本搜索工具: grep

来自 <<http://c.biancheng.net/linux/grep.html>>

在passwd文件中搜索leo并高亮显示

zgrep [选项] 要搜索的字符串 文件名

在压缩文件中搜索字符串

bzip2 [选项] 文件名

bzip2程序压缩文件

```
[dmtsai@study ~]$ bzip2 [-cdkzv#] 檔名
```

```
[dmtsai@study ~]$ bzcat 檔名.bz2
```

选项与参数:

-c : 将压缩的过程产生的数据输出到屏幕上!

-d : 解压缩的参数

-k : 保留源文件, 而不会删除原始的文件喔!

-z : 压缩的参数 (默认值, 可以不加)

-v : 可以显示出原文件/压缩文件案的压缩比等信息;

-# : 与 gzip 同样的, 都是在计算压缩比的参数, -9 最佳, -1 最快!

屏幕剪辑的捕获时间: 2023/3/4 16:21

bzip2 [-c] 文件名 > 压缩后的文件名

压缩文件并保留原本的文件

xz [选项] 文件名

xz程序压缩文件

```
[dmtsai@study ~]$ xz [-dtlkc#] 檔名
```

```
[dmtsai@study ~]$ xcat 檔名.xz
```

选项与参数：

-d : 就是解压缩啊！

-t : 测试压缩文件的完整性，看有没有错误

-l : 列出压缩文件的相关信息

-k : 保留原本的文件不删除～

-c : 同样的，就是将数据由屏幕上输出的意思！

-# : 同样的，也有较佳的压缩比的意思！

屏幕剪辑的捕获时间: 2023/3/4 16:25

xz [-k] 文件名

压缩文件并保留原本的文件

```
[luhao@localhost tmp]$ xz -k services
```

```
[luhao@localhost tmp]$ ll
```

总用量 1016

-rw-r--r--.	1	luhao	luhao	670293	3月	4 15:30	services
-rw-rw-r--.	1	luhao	luhao	123932	3月	4 16:22	services.bz2
-rw-rw-r--.	1	luhao	luhao	136088	3月	4 15:34	services.gz
-rw-r--r--.	1	luhao	luhao	99608	3月	4 15:30	services.xz

屏幕剪辑的捕获时间: 2023/3/4 16:29

tar

将多个文件或目录包成一个大文件，打包

```
[dmtsa1@study ~]$ tar [-z|-j|-J] [cv] [-f 待建立的新檔名] filename... <==打包与压缩  
[dmtsa1@study ~]$ tar [-z|-j|-J] [tv] [-f 既有的 tar 檔名] <==察看檔名  
[dmtsa1@study ~]$ tar [-z|-j|-J] [xv] [-f 既有的 tar 檔名] [-C 目录] <==解压缩
```

选项与参数:

- c : 建立打包文件，可搭配 -v 来察看过程中被打包的档名(filename)
- t : 察看打包文件的内容含有哪些档名，重点在察看『档名』就是了；
- x : 解打包或解压缩的功能，可以搭配 -C (大写) 在特定目录解开
特别留意的是， -c, -t, -x 不可同时出现在一串指令列中。
- z : 透过 gzip 的支持进行压缩/解压缩：此时档名最好为 *.tar.gz
- j : 透过 bzip2 的支持进行压缩/解压缩：此时档名最好为 *.tar.bz2
- J : 透过 xz 的支持进行压缩/解压缩：此时档名最好为 *.tar.xz
特别留意， -z, -j, -J 不可以同时出现在一串指令列中
- v : 在压缩/解压缩的过程中，将正在处理的文件名显示出来！相当于ls -l 列出详细信息
- f filename: -f 后面要立刻接要被处理的档名！建议 -f 单独写一个选项啰！(比较不会忘记)
- C 目录 : 这个选项用在解压缩，若要在特定目录解压缩，可以使用这个选项。

其他后续练习会使用到的选项介绍：

- p(小写) : 保留备份数据的原本权限与属性，常用于备份(-c)重要的配置文件
- P(大写) : 保留绝对路径，亦即允许备份数据中含有根目录存在之意；
- exclude=FILE: 在压缩的过程中，不要将 FILE 打包！

屏幕剪辑的捕获时间: 2023/3/4 16:36

用tar备份目录

```
[dmtsaï@study ~]$ su - # 因为备份 /etc 需要 root 的权限，否则会出现一堆错误
[ root@study ~]# time tar -zpcv -f /root/etc.tar.gz /etc
tar: Removing leading `/' from member names <==注意这个警告讯息
/etc/
....(中间省略)....
/etc/hostname
/etc/aliases.db

real    0m0.799s  # 多了 time 会显示程序运作的时间！看 real 就好了！花去了 0.799s
user    0m0.767s
sys     0m0.046s

# 由于加上 -v 这个选项，因此正在作用中的文件名就会显示在屏幕上。
# 如果你可以翻到第一页，会发现出现上面的错误讯息！底下会讲解。
# 至于 -p 的选项，重点在于『保留原本文件的权限与属性』之意。
```

屏幕剪辑的捕获时间: 2023/3/4 16:49

在特定目录下解压

```
[ root@study ~]# tar -jxv -f /root/etc.tar.bz2 -C /tmp
```

屏幕剪辑的捕获时间: 2023/3/5 14:13

- 压 缩: tar -jcv -f filename.tar.bz2 要被压缩的文件或目录名称
- 查 询: tar -jtv -f filename.tar.bz2
- 解压缩: tar -jxv -f filename.tar.bz2 -C 欲解压缩的目录

屏幕剪辑的捕获时间: 2023/3/5 14:14

解开单一文件:

```
# 1. 先找到我们要的档名，假设解开 shadow 文件好了:
[ root@study ~]# tar -jtv -f /root/etc.tar.bz2 | grep 'shadow'
----- root/root      721 2015-06-17 00:20 etc/gshadow
----- root/root     1183 2015-06-17 00:20 etc/shadow-
----- root/root     1210 2015-06-17 00:20 etc/shadow <==这是我们要的!
```

```
# 1. 先找到我们要的档名，假设解开 shadow 文件好了：  
[root@study ~]# tar -jtv -f /root/etc.tar.bz2 | grep 'shadow'  
----- root/root      721 2015-06-17 00:20 etc/gshadow  
----- root/root      1183 2015-06-17 00:20 etc/shadow-  
----- root/root     1210 2015-06-17 00:20 etc/shadow <==这是我们要的！  
----- root/root      707 2015-06-17 00:20 etc/gshadow-  
# 先搜寻重要的档名！其中那个 grep 是『撷取』关键词的功能！我们会在第三篇说明！  
# 这里您先有个概念即可！那个管线 | 配合 grep 可以撷取关键词的意思！
```

2. 将该文件解开！语法与实际作法如下：

```
[root@study ~]# tar -jxv -f 打包檔.tar.bz2 待解开档名  
[root@study ~]# tar -jxv -f /root/etc.tar.bz2 etc/shadow  
etc/shadow  
[root@study ~]# ll etc  
total 4  
----- 1 root root 1210 Jun 17 00:20 shadow  
# 很有趣！此时只会解开一个文件而已！不过，重点是那个档名！你要找到正确的档名。  
# 在本例中，你不能写成 /etc/shadow！因为记录在 etc.tar.bz2 内的并没有 / 之故！
```

屏幕剪辑的捕获时间: 2023/3/5 14:18

打包某目录，但不含该目录下的某些文件

```
[root@study ~]# tar -jcv -f /root/system.tar.bz2 --exclude=/root/etc* \  
> --exclude=/root/system.tar.bz2 /etc /root
```

屏幕剪辑的捕获时间: 2023/3/5 14:26

仅备份比某个时刻还要新的文件

```
# 1. 先由 find 找出比 /etc/passwd 还要新的文件
[ root@study ~]# find /etc -newer /etc/passwd
....(过程省略)....
# 此时会显示出比 /etc/passwd 这个文件的 mtime 还要新的档名,
# 这个结果在每部主机都不相同! 您先自行查阅自己的主机即可, 不会跟鸟哥一样!
```

```
[ root@study ~]# ll /etc/passwd
-rw-r--r--. 1 root root 2092 Jun 17 00:20 /etc/passwd
```

```
# 2. 好了, 那么使用 tar 来进行打包吧! 日期为上面看到的 2015/06/17
[ root@study ~]# tar -jcv -f /root/etc.newer.then.passwd.tar.bz2 \
> --newer-mtime="2015/06/17" /etc/*
tar: Option --newer-mtime: Treating date `2015/06/17' as 2015-06-17 00:00:00
tar: Removing leading `/' from member names
/etc/abrt/
....(中间省略)....
/etc/alsa/
/etc/yum.repos.d/
....(中间省略)....
tar: /etc/yum.repos.d/CentOS-fasttrack.repo: file is unchanged; not dumped
# 最后行显示的是『没有被备份的』, 亦即 not dumped 的意思!
```

屏幕剪辑的捕获时间: 2023/3/5 14:32

一边打包, 一边解开

```
# 1. 将 /etc 整个目录一边打包一边在 /tmp 解开
[ root@study ~]# cd /tmp
[ root@study tmp]# tar -cvf - /etc | tar -xvf -
# 这个动作有点像是 cp -r /etc /tmp 啦～依旧是具有用途的！
# 要注意的地方在于输出档变成 - 而输入档也变成 - ，又有一个 | 存在～
# 这分别代表 standard output, standard input 与管线命令啦！
# 简单的想法中，你可以将 - 想成是在内存中的一个装置(缓冲区)。
# 更详细的数据流与管线命令，请翻到 bash 章节啰！
```

屏幕剪辑的捕获时间: 2023/3/5 14:42

xfs文件系统的备份: xfsdump

xfs文件系统的还原: xfsrestore

建立镜像文件: mkisofs

光盘刻录工具: cdrecord

备份完整的硬盘或硬盘分区: dd

dd if=输入文件名 of=输出文件名 bs=一个扇区的大小 count=多少个扇区

```
[ root@study ~]# dd if="input_file" of="output_file" bs="block_size" count="number"
```

选项与参数:

if : 就是 input file 哪～也可以是装置喔！

of : 就是 output file 哪～也可以是装置；

bs : 规划的一个 block 的大小，若未指定则预设是 512 bytes(一个 sector 的大小)

count: 多少个 bs 的意思。

范例一：将 /etc/passwd 备份到 /tmp/passwd.back 当中

```
[ root@study ~]# dd if=/etc/passwd of=/tmp/passwd.back
4+1 records in
4+1 records out
2092 bytes (2.1 kB) copied, 0.000111657 s, 18.7 MB/s
```

屏幕剪辑的捕获时间: 2023/3/5 16:05

备份任何东西: cpio (需要搭配find和管道来操作)

```
[root@study ~]# cpio -ovcB > [file|device] <==备份  
[root@study ~]# cpio -ivcd < [file|device] <==还原  
[root@study ~]# cpio -ivct < [file|device] <==察看
```

备份会使用到的选项与参数：

- o : 将数据 copy 输出到文件或装置上
- B : 让预设的 Blocks 可以增加至 5120 bytes , 预设是 512 bytes !
这样的好处是可以让大文件的储存速度加快(请参考 i-nodes 的观念)

还原会使用到的选项与参数：

- i : 将数据自文件或装置 copy 出来系统当中
- d : 自动建立目录！使用 cpio 所备份的数据内容不见得会在同一层目录中，因此我们必须要让 cpio 在还原时可以建立新目录，此时就得要 -d 选项的帮助！
- u : 自动的将较新的文件覆盖较旧的文件！
- t : 需配合 -i 选项，可用在"察看"以 cpio 建立的文件或装置的内容

一些可共享的选项与参数：

- v : 让储存的过程中文件名可以在屏幕上显示
- c : 一种较新的 portable format 方式储存

屏幕剪辑的捕获时间: 2023/3/5 16:16

file 文件

查看文件是什么文件格式

第9章

2023年3月6日 15:43

/bin/vi 文件名

使用vi, 进入一般命令模式来操作文件
或写为vi 文件名

```
[luhao@localhost ~]$ vi welcome.txt
```

屏幕剪辑的捕获时间: 2023/3/6 15:57

vim具有通过缓存来恢复未保存的数据的功能

ctrl+z

程序到后台执行

出现缓存文件:

```
[luhao@localhost vitest]$ ll -a
总用量 36
drwxrwxr-x.  2 luhao luhao    72 3月   7 11:25 .
drwxrwxrwt. 52 root  root   4096 3月   7 11:23 ..
-rw-r--r--.  1 luhao luhao   4852 3月   6 19:58 man_db.conf
-rw-r--r--.  1 luhao luhao 16384 3月   7 11:25 .man_db.conf.swp
-rw-rw-r--.  1 luhao luhao   4847 3月   6 19:54 man.test.config
```

屏幕剪辑的捕获时间: 2023/3/7 11:27

输入alias, 存在

```
alias vi='vim'
```

屏幕剪辑的捕获时间: 2023/3/7 15:38

表明vi已被vim替换

可视区块 (书P300)

vim 文件1 文件2

多文件编辑

:sp

多窗口功能, 同一个文件显示在两个窗口

```
# their PATH environment variable. For details see the manpath(5) man page.  
#  
# Lines beginning with '#' are comments and are ignored. Any combination  
# of  
# tabs or spaces may be used as 'whitespace' separators.  
man_db.conf 9,1 4%  
# I am a student...  
#  
#  
# This file is used by the man-db package to configure the man and cat paths.  
@  
man_db.conf 1,1 顶端
```

屏幕剪辑的捕获时间: 2023/3/7 16:11

关键词补全 (书P303)

vim ~/.vimrc 通过配置文件直接规定vim默认的操作环境

修改终端(bash)的语系以匹配文件的中文编码: 书P306

DOS(Windows)系统的文件格式和Linux系统的文件格式相互转换:

(先安装dos2unix)

```
[root@localhost ~]# yum install dos2unix
```

屏幕剪辑的捕获时间: 2023/3/7 20:14

字符转换:

```
[dmtsai@study ~]$ dos2unix [-kn] file [newfile]  
[dmtsai@study ~]$ unix2dos [-kn] file [newfile]
```

选项与参数:

-k : 保留该文件原本的 mtime 时间格式 (不更新文件上次内容经过修订的时间)
-n : 保留原本的旧档, 将转换后的內容输出到新文件, 如: dos2unix -n old new

屏幕剪辑的捕获时间: 2023/3/7 20:16

```
[root@localhost vitest]# unix2dos -k man_db.conf  
[root@localhost vitest]# dos2unix -k -n man_db.conf man_db.conf.linux
```

屏幕剪辑的捕获时间: 2023/3/7 20:26

注 -k -n 必须分开写, 不能写成-kn

```
[root@localhost vitest]# file man_db.conf*
man_db.conf:          ASCII text, with CRLF line terminators
man_db.conf.linux: ASCII text
```

屏幕剪辑的捕获时间: 2023/3/7 20:28

文件的语系编码转换: iconv

```
[dmtsai@study ~]$ iconv --list
[dmtsai@study ~]$ iconv -f 原本编码 -t 新编码 filename [-o newfile]
```

选项与参数:

- list : 列出 iconv 支持的语系数据
- f : from , 亦即来源之意, 后接原本的编码格式;
- t : to , 亦即后来的新编码要是什么格式;
- o file: 如果要保留原本的文件, 那么使用 -o 新档名, 可以建立新编码文件。

屏幕剪辑的捕获时间: 2023/3/8 19:17

范例一: 将 /tmp/vitest/vi.big5 转成 utf8 编码吧!

```
[dmtsai@study ~]$ cd /tmp/vitest
[dmtsai@study vitest]$ iconv -f big5 -t utf8 vi.big5 -o vi.utf8
[dmtsai@study vitest]$ file vi*
vi.big5: ISO-8859 text, with CRLF line terminators
vi.utf8: UTF-8 Unicode text, with CRLF line terminators
# 是吧! 有明显的不同吧! ^_^
```

屏幕剪辑的捕获时间: 2023/3/8 19:19

注用file查看文件编码格式是ISO-8859, 其实iconv不支持的, 故尝试写成GB2312

```
[root@localhost ~]# file code1.c
code1.c: C source, ISO-8859 text
[root@localhost ~]# iconv -f GB2312 -t UTF8 code1.c -o code1_UTF8.c
[root@localhost ~]# ls -l code1*
-rw-r--r--. 1 root root 285 4月    7 2022 code1.c
-rw-r--r--. 1 root root 318 3月    9 14:24 code1_UTF8.c
[root@localhost ~]# vim code1_UTF8.c
```

屏幕剪辑的捕获时间: 2023/3/9 14:27

发现中文显示正常：

```
#define _CRT_SECURE_NO_WARNINGS 1
#include <stdio.h>
int main(void)
{
    int a /*这也是一条注释*/ = 1;
    printf("I am a smart computer\n");//这是一条注释
    printf("my favorite number is %d because it is the first.\n", a
);/*这也是一条注释*/
    return 0;
}
//这是一条注释
/*这也是一条注释*/
```

屏幕剪辑的捕获时间: 2023/3/9 14:26

ctrl+z

停止进程并放入后台

第10章

2023年3月9日 14:58

当前的Linux可以使用的shells:

/bin/bash
/bin/tcsh
/bin/csh

记录可用的shells的文件:

/etc/shells

Linux默认的shell:

bash即/bin/bash

系统登录后不同用户取得的shell记录在/etc/passwd中

```
[luhao@localhost bin]$ cat /etc/passwd
luhao:x:1000:1000:luhao:/home/luhao:/bin/bash
```

屏幕剪辑的捕获时间: 2023/3/9 15:11

```
[luhao@localhost bin]$ grep --color 'root' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

屏幕剪辑的捕获时间: 2023/3/9 15:09

家目录下的.bash_history记录了历史命令

查看命令别名: alias

```
[luhao@localhost ~]$ alias
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -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'
```

屏幕剪辑的捕获时间: 2023/3/9 15:43

注普通用户会有vi = 'vim'的命令别名, 但root没有这一条。

alias 别名='命令'

设置命令别名

```
[luhao@localhost ~]$ alias lm='ls -al'
```

type [选项] 命令名

查询命令是否为bash的内置命令
类似which命令的作用

```
[dmtsa@study ~]$ type [-tpa] name
```

选项与参数：

- : 不加任何选项与参数时, type 会显示出 name 是外部指令还是 bash 内建指令
- t : 当加入 -t 参数时, type 会将 name 以底下这些字眼显示出他的意义:
 - file : 表示为外部指令;
 - alias : 表示该指令为命令别名所设定的名称;
 - builtin : 表示该指令为 bash 内建的指令功能;
- p : 如果后面接的 name 为外部指令时, 才会显示完整文件名;
- a : 会由 PATH 变量定义的路径中, 将所有含 name 的指令都列出来, 包含 alias

屏幕剪辑的捕获时间: 2023/3/9 16:21

\[Enter]

命令以两行来书写

热键(组合键):

ctrl+u	从光标处向前删除命令串
ctrl+k	从光标处向后删除命令串
ctrl+a	光标移到命令串最前
ctrl+e	光标移到命令串最后

echo \$变量名
echo \${变量名}
读出变量的内容
读出变量的内容

```
[luhao@localhost ~]$ echo $HOME  
/home/luhao  
[luhao@localhost ~]$ cd $HOME  
[luhao@localhost ~]$
```

屏幕剪辑的捕获时间: 2023/3/9 19:52

设置或修改变量的内容

```
[luhao@localhost ~]$ echo $myname  
  
[luhao@localhost ~]$ myname=luhao  
[luhao@localhost ~]$ echo $myname  
luhao
```

屏幕剪辑的捕获时间: 2023/3/9 19:54

unset 变量名称

取消设置变量, 即
清空该变量的内容

```
[luhao@localhost ~]$ unset myname  
[luhao@localhost ~]$  
[luhao@localhost ~]$ echo $myname
```

屏幕剪辑的捕获时间: 2023/3/9 20:03

父进程的自定义变量无法在子进程中使用

```
[luhao@localhost ~]$ echo $name  
luhao's name  
[luhao@localhost ~]$ bash  
[luhao@localhost ~]$ echo $name
```

```
[luhao@localhost ~]$ exit  
exit  
[luhao@localhost ~]$ echo $name  
luhao's name
```

屏幕剪辑的捕获时间: 2023/3/10 15:02

export 变量名

将变量设置成环境变量

在一串命令的执行中，还需要其他额外命令所提供的信息时，使用\$(命令)

```
[luhao@localhost ~]$ uname -r  
3.10.0-1160.83.1.el7.x86_64  
[luhao@localhost ~]$ cd /lib/modules/$(uname -r)/kernel  
[luhao@localhost kernel]$ █
```

屏幕剪辑的捕获时间: 2023/3/10 15:09

查找与要求相关的文件名，并输出这些文件的权限

例：列出每个和crontab相关的文件名的权限

```
[luhao@localhost ~]$ ls -ld $(locate crontab)
-rw-----. 1 root root 541 1月 14 2022 /etc/anacrontab
-rw-r--r--. 1 root root 451 6月 10 2014 /etc/crontab
-rw-r--r--. 1 luhao luhao 451 6月 10 2014 /home/luhao/文档/crontab
-rwsr-xr-x. 1 root root 57576 1月 14 2022 /usr/bin/crontab
drwxr-xr-x. 2 root root 21 1月 13 21:42 /usr/share/doc/man-pages-overrides-7.9.0/crontabs
-rw-r--r--. 1 root root 17738 10月 1 2020 /usr/share/doc/man-pages-overrides-7.9.0/crontabs/COPYING
-rw-r--r--. 1 root root 2626 1月 14 2022 /usr/share/man/man1/crontab.1.gz
-rw-r--r--. 1 root root 4229 6月 10 2014 /usr/share/man/man1p/crontab.1p.gz
-rw-r--r--. 1 root root 1121 6月 10 2014 /usr/share/man/man4/crontab.s.4.gz
-rw-r--r--. 1 root root 1658 1月 14 2022 /usr/share/man/man5/anacron.tab.5.gz
-rw-r--r--. 1 root root 4980 1月 14 2022 /usr/share/man/man5/crontab.5.gz
-rw-r--r--. 1 root root 2566 12月 16 2020 /usr/share/vim/vim74/syntax/crontab.vim
```

屏幕剪辑的捕获时间: 2023/3/10 15:24

列出所有默认的环境变量: env或export

```
SSH_AUTH_SOCK=/tmp/ssh-5QoXy9ZiWJ/agent.1552
MAIL=/var/spool/mail/luhao
PATH=/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/luhao/.local/bin:/home/luhao/bin
PWD=/home/luhao
LANG=zh_CN.UTF-8
SELINUX_LEVEL_REQUESTED=
HISTCONTROL=ignoredups
SHLVL=1
HOME=/home/luhao
LOGNAME=luhao
XDG_DATA_DIRS=/home/luhao/.local/share/flatpak/exports/share:/var/lib/flatpak/exports/share:/usr/local/share:/usr/share
SSH_CONNECTION=192.168.175.1 7415 192.168.175.131 22
LESSOPEN=||/usr/bin/lesspipe.sh %
XDG_RUNTIME_DIR=/run/user/1000
_=/usr/bin/env
OLDPWD=/lib/modules/3.10.0-1160.83.1.el7.x86_64/kernel
```

屏幕剪辑的捕获时间: 2023/3/10 15:39

使用变量

```
[luhao@localhost etc]$ cd HOME  
-bash: cd: HOME: 没有那个文件或目录  
[luhao@localhost etc]$ cd $HOME  
[luhao@localhost ~]$
```

屏幕剪辑的捕获时间: 2023/3/10 15:44

获取随机数(0~32767)

```
[luhao@localhost ~]$ echo $RANDOM  
1284
```

屏幕剪辑的捕获时间: 2023/3/10 15:47

获取随机数(0~9)

利用declare声明数值类型

```
[luhao@localhost ~]$ declare -i number=$RANDOM*10/32767; echo $number  
8
```

屏幕剪辑的捕获时间: 2023/3/10 16:29

观察所有变量(环境变量和自定义的变量): set

由于屏幕有限, 只能看到部分的局部的输出结果, 如何操作才能看到全部的输出结果?

使用管道|less

```
[luhao@localhost ~]$ set |less
```

屏幕剪辑的捕获时间: 2023/3/10 16:45

提示字符的设置: 环境变量PS1

进程号: PID

echo \$?

关于上个执行命令的返回值

```
[dmtsaï@study ~]$ echo $SHELL  
/bin/bash <==可顺利显示！没有错误！  
[dmtsaï@study ~]$ echo $?  
0 <==因为没问题，所以回传值为 0  
[dmtsaï@study ~]$ 12name=VBird  
bash: 12name=VBird: command not found... <==发生错误了！bash 回报有问题  
[dmtsaï@study ~]$ echo $?  
127 <==因为有问题，回传错误代码(非为 0)  
# 错误代码回传值依据软件而有不同，我们可以利用这个代码来搜寻错误的原因喔！  
[dmtsaï@study ~]$ echo $?  
0  
# 嘿！怎么又变成正确了？这是因为 "?" 只与『上一个执行指令』有关，  
# 所以，我们上一个指令是执行『 echo $? 』，当然没有错误，所以是 0 没错！
```

屏幕剪辑的捕获时间: 2023/3/10 19:54

```
export 变量名称 自定义变量转化成环境变量  
[dmtsaï@study ~]$ export  
declare -x HISTSIZE="1000"  
declare -x HOME="/home/dmtsaï"  
declare -x HOSTNAME="study.centos.vbird"  
declare -x LANG="zh_TW.UTF-8"  
declare -x LC_ALL="en_US.utf8"  
# 后面的鸟哥就都直接省略了！不然....浪费版面~ ^_^
```

屏幕剪辑的捕获时间: 2023/3/10 19:59

```
locale -a 查看Linux支持的语系
```

```
[luhao@localhost ~]$ locale -a | less
```

屏幕剪辑的捕获时间: 2023/3/11 16:26

```
locale 显示目前的语系
```

```
[luhao@localhost ~]$ locale
LANG=zh_CN.UTF-8
LC_CTYPE="zh_CN.UTF-8"
LC_NUMERIC="zh_CN.UTF-8"
LC_TIME="zh_CN.UTF-8"
LC_COLLATE="zh_CN.UTF-8"
LC_MONETARY="zh_CN.UTF-8"
LC_MESSAGES="zh_CN.UTF-8"
LC_PAPER="zh_CN.UTF-8"
LC_NAME="zh_CN.UTF-8"
LC_ADDRESS="zh_CN.UTF-8"
LC_TELEPHONE="zh_CN.UTF-8"
LC_MEASUREMENT="zh_CN.UTF-8"
LC_IDENTIFICATION="zh_CN.UTF-8"
LC_ALL=
```

屏幕剪辑的捕获时间: 2023/3/11 16:29

注

Linux终端环境下若没有安装中文界面的软件就无法显示中文，但Windows远程连接到主机可以看到中文

系统默认语系定义位置: /etc/locale.conf 可以用nano自行修改

```
[luhao@localhost locale]$ cat /etc/locale.conf
LANG="zh_CN.UTF-8"
```

屏幕剪辑的捕获时间: 2023/3/11 16:36

修改为英文语系输出

```
[dmtsai@study ~]$ locale
LANG=zh_TW.UTF-8
LC_CTYPE="zh_TW.UTF-8"
LC_NUMERIC="zh_TW.UTF-8"
LC_TIME="zh_TW.UTF-8"
```

```
[dmtsai@study ~]$ LANG=en_US.utf8; locale  
[dmtsai@study ~]$ export LC_ALL=en_US.utf8; locale # 你就会看到与上头有不同的语系啰!
```

屏幕剪辑的捕获时间: 2023/3/11 16:43

read [选项] 变量名

自定义变量内容由键盘输入

```
[dmtsai@study ~]$ read [-pt] variable
```

选项与参数:

-p : 后面可以接提示字符!

-t : 后面可以接等待的『秒数!』这个比较有趣~不会一直等待使用者啦!

屏幕剪辑的捕获时间: 2023/3/11 20:16

```
[luhao@localhost ~]$ read test
```

```
123456
```

```
[luhao@localhost ~]$ echo $test
```

```
123456
```

屏幕剪辑的捕获时间: 2023/3/11 20:16

范例二: 提示使用者 30 秒内输入自己的大名, 将该输入字符串作为名为 named 的变量内容

```
[dmtsai@study ~]$ read -p "Please keyin your name: " -t 30 named
```

Please keyin your name: VBird Tsai <==注意看, 会有提示字符喔!

```
[dmtsai@study ~]$ echo ${named}
```

VBird Tsai <==输入的数据又变成一个变量的内容了!

屏幕剪辑的捕获时间: 2023/3/11 20:17

使用declare声明变量的类型

```
[dmtsai@study ~]$ declare [-aixr] variable
```

选项与参数:

-a : 将后面名为 variable 的变量定义成为数组 (array) 类型

-i : 将后面名为 variable 的变量定义成为整数数字 (integer) 类型

-x : 用法与 export 一样, 就是将后面的 variable 变成环境变量;

-r : 将变量设定成为 readonly 类型, 该变量不可被更改内容, 也不能 unset

屏幕剪辑的捕获时间: 2023/3/11 20:24

```
[luhao@localhost ~]$ echo $sum  
100+200  
[luhao@localhost ~]$ declare -i sum=100+200  
[luhao@localhost ~]$ echo $sum  
300
```

屏幕剪辑的捕获时间: 2023/3/11 20:26

注变量类型默认为字符串

范例二：将 sum 变成环境变量

```
[dmtsaiai@study ~]$ declare -x sum  
[dmtsaiai@study ~]$ export | grep sum  
declare -ix sum="450" <==果然出现了！包括有 i 与 x 的宣告！
```

范例三：让 sum 变成只读属性，不可更动！

```
[dmtsaiai@study ~]$ declare -r sum  
[dmtsaiai@study ~]$ sum=tesgting  
-bash: sum: readonly variable <==老天爷～不能改这个变数了！
```

范例四：让 sum 变成非环境变量的自定义变量吧！

```
[dmtsaiai@study ~]$ declare +x sum <== 将 - 变成 + 可以进行『取消』动作  
[dmtsaiai@study ~]$ declare -p sum <== -p 可以单独列出变量的类型  
declare -ir sum="450" <== 看吧！只剩下 i, r 的类型，不具有 x 嘍！
```

屏幕剪辑的捕获时间: 2023/3/11 20:30

将-变成+可进行取消的操作

创建数组变量并显示内容：

```
[luhao@localhost ~]$ var[0]=\"small min\"\n[luhao@localhost ~]$ var[1]=\"big min\"\n[luhao@localhost ~]$ var[2]=\"nice min\"\n[luhao@localhost ~]$ echo ${var[0]}\nsmall min[0]\n[luhao@localhost ~]$ echo ${var[0]}\nsmall min\n[luhao@localhost ~]$ echo "${var[0]},${var[1]},${var[2]}\"\nsmall min,big min,nice min"
```

屏幕剪辑的捕获时间: 2023/3/11 20:37

注必须以echo \${变量}来读取

限制用户某些系统资源: ulimit

屏幕剪辑的捕获时间: 2023/3/11 20:41

范例二：限制用户仅能建立 10MBytes 以下的容量的文件

```
[dmtsai@study ~]$ ulimit -f 10240\n[dmtsai@study ~]$ ulimit -a | grep 'file size'\ncore file size          (blocks, -c) 0\nfile size                (blocks, -f) 10240 <==最大量为 10240Kbytes, 相当 10Mbytes\n\n[dmtsai@study ~]$ dd if=/dev/zero of=123 bs=1M count=20\nFile size limit exceeded (core dumped) <==尝试建立 20MB 的文件, 结果失败了!\n\n[dmtsai@study ~]$ rm 123 <==赶快将这个文件删除啰! 同时你得要注销再次的登入才能解开 10M 的限制
```

屏幕剪辑的捕获时间: 2023/3/11 20:47

变量内容的删除与替换(P332)

变量的测试与内容替换(P333)

unalias 命令名

将命令别名删除

查看历史命令: history

```
[dmtsai@study ~]$ history [-raw] histfiles
```

选项与参数:

```
n    : 数字，意思是『要列出最近的 n 笔命令行表』的意思！  
-c    : 将目前的 shell 中的所有 history 内容全部消除  
-a    : 将目前新增的 history 指令新增入 histfiles 中，若没有加 histfiles，  
      则预设写入 ~/.bash_history  
-r    : 将 histfiles 的内容读到目前这个 shell 的 history 记忆中；  
-w    : 将目前的 history 记忆内容写入 histfiles 中！
```

屏幕剪辑的捕获时间: 2023/3/13 14:10

历史命令记录的条数和HISTFILESIZE变量的内容有关

利用history来执行命令：

```
[dmtsai@study ~]$ !number  
[dmtsai@study ~]$ !command  
[dmtsai@study ~]$ !!
```

选项与参数：

number : 执行第几笔指令的意思；

command : 由最近的指令向前搜寻『指令串开头为 command』的那个指令，并执行；

!! : 就是执行上一个指令(相当于按↑按键后，按 Enter)

```
[dmtsai@study ~]$ history  
66 man rm  
67 alias  
68 man history  
69 history  
[dmtsai@study ~]$ !66 <==执行第 66 笔指令  
[dmtsai@study ~]$ !! <==执行上一个指令，本例中亦即 !66  
[dmtsai@study ~]$ !al <==执行最近以 al 为开头的指令(上头列出的第 67 个)
```

屏幕剪辑的捕获时间: 2023/3/13 14:18

type -a 命令名

查看命令查找的顺序

```
[luhao@localhost ~]$ alias echo='echo -n'  
[luhao@localhost ~]$ type -a echo  
echo 是 'echo -n' 的别名  
echo 是 shell 内嵌  
echo 是 /bin/echo  
echo 是 /usr/bin/echo
```

屏幕剪辑的捕获时间: 2023/3/13 14:30

修改bash的登录与欢迎信息: /etc/issue

```
[root@localhost ~]# nano /etc/issue
```

屏幕剪辑的捕获时间: 2023/3/13 14:39

不注销后再登录，直接读取配置文件

non-login shell 的配置文件的读取(R341)

查看按键的含义：stty

```
[dmt sai@study ~]$ stty [-a]
```

选项与参数.

-a : 将目前所有的 stty 参数列出来:

范例一：列出所有的按键与按键内容

```
[dmt saini@study ~]$ stty sane
```

speed 38400 baud; rows 20; columns 90; line = 0;

`intr = ^C; quit = ^\}; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2 = <undef>;`

同萼煎银的插花时间 2022/3/13 15:23

组合按键	执行结果
Ctrl + C	终止目前的命令
Ctrl + D	输入结束 (EOF)，例如邮件结束的时候；
Ctrl + M	就是 Enter 啦！
Ctrl + S	暂停屏幕的输出
Ctrl + Q	恢复屏幕的输出
Ctrl + U	在提示字符下，将整列命令删除
Ctrl + Z	『暂停』目前的命令

屏幕剪辑的捕获时间: 2023/3/13 15:27

通配符:

符号	意义
*	代表『0个到无穷多个』任意字符
?	代表『一定有一个』任意字符
[]	同样代表『一定有一个在括号内』的字符(非任意字符)。例如 [abcd] 代表『一定有一个字符，可能是 a, b, c, d 这四个任何一个』
[-]	若有减号在中括号内时，代表『在编码顺序内的所有字符』。例如 [0-9] 代表 0 到 9 之间的所有数字，因为数字的语系编码是连续的！
[^]	若中括号内的第一个字符为指数符号 (^)，那表示『反向选择』，例如 [^abc] 代表 一定有一个字符，只要是非 a, b, c 的其他字符就接受的意思。

屏幕剪辑的捕获时间: 2023/3/13 16:13

```
[dmtsaï@study ~]$ LANG=C                                     <==由于与编码有关，先设定语系一下

范例一：找出 /etc/ 底下以 cron 为开头的档名
[dmtsaï@study ~]$ ll -d /etc/cron*      <==加上 -d 是为了仅显示目录而已

范例二：找出 /etc/ 底下文件名『刚好是五个字母』的文件名
[dmtsaï@study ~]$ ll -d /etc/?????      <==由于 ? 一定有一个，所以五个 ? 就对了

范例三：找出 /etc/ 底下文件名含有数字的文件名
[dmtsaï@study ~]$ ll -d /etc/*[0-9]*  <==记得中括号左右两边均需 *

范例四：找出 /etc/ 底下，档名开头非为小写字母的文件名：
[dmtsaï@study ~]$ ll -d /etc/[^a-z]*  <==注意中括号左边没有 *

范例五：将范例四找到的文件复制到 /tmp/upper 中
[dmtsaï@study ~]$ mkdir /tmp/upper; cp -a /etc/[^a-z]* /tmp/upper
```

屏幕剪辑的捕获时间: 2023/3/13 16:18

特殊符号：

符号	内容
#	批注符号：这个最常被使用在 script 当中，视为说明！在后的数据均不执行
\	跳脱符号：将『特殊字符或通配符』还原成一般字符
	管线 (pipe)：分隔两个管线命令的界定(后两节介绍)；
;	连续指令下达分隔符：连续性命令的界定 (注意！与管线命令并不相同)
~	用户的家目录
\$	取用变数前导符：亦即是变量之前需要加的变量取代值
&	工作控制 (job control)：将指令变成背景下工作
!	逻辑运算意义上的『非』 not 的意思！
/	目录符号：路径分隔的符号
>, >>	数据流重导向：输出导向，分别是『取代』与『累加』
<, <<	数据流重导向：输入导向 (这两个留待下节介绍)
'	单引号，不具有变量置换的功能 (\$ 变为纯文本)
''	具有变量置换的功能！ (\$ 可保留相关功能)
``	两个「`」中间为可以先执行的指令，亦可使用 \$()
()	在中间为子 shell 的起始与结束
{ }	在中间为命令区块的组合！

屏幕剪辑的捕获时间: 2023/3/13 16:19

重定向

1. 标准输入 (stdin) : 代码为 0 , 使用 < 或 << ;
2. 标准输出 (stdout): 代码为 1 , 使用 > 或 >> ;
3. 标准错误输出(stderr): 代码为 2 , 使用 2> 或 2>> ;

屏幕剪辑的捕获时间: 2023/3/14 14:31

```
[luhao@localhost ~]$ ll / > ~/rootfile  
[luhao@localhost ~]$ ll ~/rootfile  
-rw-rw-r--. 1 luhao luhao 1018 3月 14 14:33 /home/luhao/rootfile
```

屏幕剪辑的捕获时间: 2023/3/14 14:34

- 1> : 以覆盖的方法将『正确的数据』输出到指定的文件或装置上;
- 1>>: 以累加的方法将『正确的数据』输出到指定的文件或装置上;
- 2> : 以覆盖的方法将『错误的数据』输出到指定的文件或装置上;
- 2>>: 以累加的方法将『错误的数据』输出到指定的文件或装置上;

屏幕剪辑的捕获时间: 2023/3/14 14:50

范例三: 承范例二, 将 stdout 与 stderr 分存到不同的文件去

```
[dmtsai@study ~]$ find /home -name .bashrc > list_right 2> list_error
```

屏幕剪辑的捕获时间: 2023/3/14 14:51

/dev/null可以吃掉任何导向这个装置的信息:

范例四: 承范例三, 将错误的数据丢弃, 屏幕上显示正确的数据

```
[dmtsai@study ~]$ find /home -name .bashrc 2> /dev/null
```

/home/dmtsai/.bashrc <==只有 stdout 会显示到屏幕上, stderr 被丢弃了

屏幕剪辑的捕获时间: 2023/3/22 11:03

将正确与错误数据写入同一文件

范例五：将指令的数据全部写入名为 list 的文件中

```
[dmtsai@study ~]$ find /home -name .bashrc > list 2> list <==错误
[dmtsai@study ~]$ find /home -name .bashrc > list 2>&1 <==正确
[dmtsai@study ~]$ find /home -name .bashrc &> list <==正确
```

屏幕剪辑的捕获时间: 2023/3/14 14:55

用cat来建立一个文件

```
[luhao@localhost ~]$ cat > catfile
ceshi
123
```

屏幕剪辑的捕获时间: 2023/3/14 15:04

最后按ctrl+d结束输入

同时使用<和>

范例七：用 stdin 取代键盘的输入以建立新文件的简单流程

```
[dmtsai@study ~]$ cat > catfile < ~/.bashrc
[dmtsai@study ~]$ ll catfile ~/.bashrc
-rw-r--r--. 1 dmtsai dmtsai 231 Mar  6 06:06 /home/dmtsai/.bashrc
-rw-rw-r--. 1 dmtsai dmtsai 231 Jul  9 18:58 catfile
# 注意看，这两个文件的大小会一模一样！几乎像是使用 cp 来复制一般！
```

屏幕剪辑的捕获时间: 2023/3/14 15:30

<<代表结束的输入字符

```
[dmtsai@study ~]$ cat > catfile << "eof"
> This is a test.
> OK now stop
> eof  <==输入这关键词，立刻就结束而不需要输入 [ctrl]+d
```

```
[dmtsai@study ~]$ cat catfile
This is a test.
OK now stop      <==只有这两行，不会存在关键词那一行！
```

屏幕剪辑的捕获时间: 2023/3/14 15:33

将信息以标准错误的格式输出

```
[dmtsai@study ~]$ echo "error message" 2> /dev/null 1>&2
```

屏幕剪辑的捕获时间: 2023/3/14 15:44

命令;命令

不考虑命令相关性的连续命令执行

&&和||

指令下达情况	说明
cmd1 && cmd2	1. 若 cmd1 执行完毕且正确执行(\$?=0)，则开始执行 cmd2。 2. 若 cmd1 执行完毕且为错误 (\$?≠0)，则 cmd2 不执行。
cmd1 cmd2	1. 若 cmd1 执行完毕且正确执行(\$?=0)，则 cmd2 不执行。 2. 若 cmd1 执行完毕且为错误 (\$?≠0)，则开始执行 cmd2。

屏幕剪辑的捕获时间: 2023/3/14 15:55

范例一：使用 ls 查阅目录 /tmp/abc 是否存在，若存在则用 touch 建立 /tmp/abc/hehe

```
[dmtsaï@study ~]$ ls /tmp/abc && touch /tmp/abc/hehe
ls: cannot access /tmp/abc: No such file or directory
# ls 很干脆的说明找不到该目录，但并没有 touch 的错误，表示 touch 并没有执行
```

```
[dmtsaï@study ~]$ mkdir /tmp/abc
[dmtsaï@study ~]$ ls /tmp/abc && touch /tmp/abc/hehe
[dmtsaï@study ~]$ ll /tmp/abc
-rw-rw-r--. 1 dmtsaï dmtsaï 0 Jul  9 19:16 hehe
```

屏幕剪辑的捕获时间: 2023/3/14 15:55

范例二：测试 /tmp/abc 是否存在，若不存在则予以建立，若存在就不作任何事情

```
[dmtsaï@study ~]$ rm -r /tmp/abc                         <==先删除此目录以方便测试
[dmtsaï@study ~]$ ls /tmp/abc || mkdir /tmp/abc
ls: cannot access /tmp/abc: No such file or directory <==真的不存在喔!
[dmtsaï@study ~]$ ll -d /tmp/abc
drwxrwxr-x. 2 dmtsaï dmtsaï 6 Jul  9 19:17 /tmp/abca <==结果出现了！有进行 mkdir
```

范例三：我不清楚 /tmp/abc 是否存在，但就是要建立 /tmp/abc/hehe 文件

```
[dmtsaï@study ~]$ ls /tmp/abc || mkdir /tmp/abc && touch /tmp/abc/hehe
```

屏幕剪辑的捕获时间: 2023/3/14 16:00

管道: |

注管道命令仅会处理标准输出，必须要能够接受来自前一个命令的数据成为标准输入继续处理才行

将一段信息的某一段给切出来: cut (将一行信息当中，取出某部分)

```
[dmtsai@study ~]$ cut -d'分隔字符' -f fields <=用于有特定分隔字符  
[dmtsai@study ~]$ cut -c 字符区间 <=用于排列整齐的讯息
```

选项与参数：

-d : 后面接分隔字符。与 -f 一起使用；

-f : 依据 -d 的分隔字符将一段讯息分区成为数段，用 -f 取出第几段的意思；

-c : 以字符 (characters) 的单位取出固定字符区间；

范例一：将 PATH 变量取出，我要找出第五个路径。

```
[dmtsai@study ~]$ echo ${PATH}  
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/dmtsai/.local/bin:/home/dmtsai/bin  
#      1      |      2      |      3      |      4      |      5      |      6      |  
  
[dmtsai@study ~]$ echo ${PATH} | cut -d ':' -f 5  
# 如同上面的数字显示，我们是以『:』作为分隔，因此会出现 /home/dmtsai/.local/bin  
# 那么如果想要列出第 3 与第 5 呢？，就是这样：  
[dmtsai@study ~]$ echo ${PATH} | cut -d ':' -f 3,5
```

屏幕剪辑的捕获时间: 2023/3/14 16:22

范例二：将 export 输出的讯息，取得第 12 字符以后的所有字符串

```
[dmtsai@study ~]$ export  
declare -x HISTCONTROL="ignoredups"  
declare -x HISTSIZE="1000"  
declare -x HOME="/home/dmtsai"  
declare -x HOSTNAME="study.centos.vbird"  
.....(其他省略).....
```

注意看，每个数据都是排列整齐的输出！如果我们不想要『 declare -x 』时，就得这么做：

```
[dmtsai@study ~]$ export | cut -c 12-  
HISTCONTROL="ignoredups"  
HISTSIZE="1000"  
HOME="/home/dmtsai"  
HOSTNAME="study.centos.vbird"  
.....(其他省略).....
```

知道怎么回事了吧？用 -c 可以处理比较具有格式的输出数据！

我们还可以指定某个范围的值，例如第 12-20 的字符，就是 cut -c 12-20 等等！

范例三：用 last 将显示的登入者的信息中，仅留下用户大名

```
[dmtsai@study ~]$ last  
root pts/1 192.168.201.101 Sat Feb 7 12:35 still logged in  
root pts/1 192.168.201.101 Fri Feb 6 12:13 - 18:46 (06:33)  
root pts/1 192.168.201.254 Thu Feb 5 22:37 - 23:53 (01:16)
```

last 可以输出『账号/终端机/来源/日期时间』的数据，并且是排列整齐的

```
[dmtsai@study ~]$ last | cut -d ' ' -f 1
```

由输出的结果我们可以发现第一个空白分隔的字段代表账号，所以使用如上指令：

但是因为 root pts/1 之间空格有好几个，并非仅有一个，所以，如果要找出
pts/1 其实不能以 cut -d ' ' -f 1,2 哪！输出的结果会不是我们想要的。

屏幕剪辑的捕获时间: 2023/3/14 16:31

grep [选项] 文件

分析一行信息，若有所需要的信息，就拿出该行
当作为管道命令时文件名可以不写

```
[dmtsaï@study ~]$ grep [-acinv] [--color=auto] '搜寻字符串' filename
```

选项与参数：

- a：将 binary 文件以 text 文件的方式搜寻数据
- c：计算找到‘搜寻字符串’的次数
- i：忽略大小写的不同，所以大小写视为相同
- n：顺便输出行号
- v：反向选择，亦即显示出没有‘搜寻字符串’内容的那一行！
- color=auto：可以将找到的关键词部分加上颜色的显示喔！

屏幕剪辑的捕获时间: 2023/3/14 16:36

排序: sort、wc、uniq

```
[dmtsai@study ~]$ sort [-fbMnrtuk] [file or stdin]
```

选项与参数：

- f : 忽略大小写的差异，例如 A 与 a 视为编码相同；
- b : 忽略最前面的空格符部分；
- M : 以月份的名字来排序，例如 JAN, DEC 等等的排序方法；
- n : 使用『纯数字』进行排序(默认是以文字型态来排序的)；
- r : 反向排序；
- u : 就是 uniq ，相同的数据中，仅出现一行代表；
- t : 分隔符，预设是用 [tab] 键来分隔；
- k : 以那个区间 (field) 来进行排序的意思

范例一：个人账号都记录在 /etc/passwd 下，请将账号进行排序。

```
[dmtsai@study ~]$ cat /etc/passwd | sort
```

```
abrt:x:173:173::/etc/abrt:/sbin/nologin
```

```
adm:x:3:4:adm:/var/adm:/sbin/nologin
```

```
alex:x:1001:1002::/home/alex:/bin/bash
```

鸟哥省略很多的输出～由上面的数据看起来，sort 是预设『以第一个』数据来排序，

而且默认是以『文字』型态来排序的喔！所以由 a 开始排到最后啰！

范例二：/etc/passwd 内容是以 : 来分隔的，我想以第三栏来排序，该如何？

```
[dmtsai@study ~]$ cat /etc/passwd | sort -t ':' -k 3
```

```
root:x:0:0:root:/root:/bin/bash
```

```
dmtsai:x:1000:1000:dmtsai:/home/dmtsai:/bin/bash
```

```
alex:x:1001:1002::/home/alex:/bin/bash
```

```
arod:x:1002:1003::/home/arod:/bin/bash
```

看到特殊字体的输出部分了吧？怎么会这样排列啊？呵呵！没错啦～

如果是以文字型态来排序的话，原本就会是这样，想要使用数字排序：

```
# cat /etc/passwd | sort -t ':' -k 3 -n
```

这样才行啊！用那个 -n 来告知 sort 以数字来排序啊！

范例三：利用 last ，将输出的数据仅取账号，并加以排序

```
[dmtsai@study ~]$ last | cut -d ' ' -f1 | sort
```

屏幕剪辑的捕获时间: 2023/3/14 16:52

排序完后, 将重复的数据仅列出一个显示: uniq (需要先经过排序)

```
[dmtsai@study ~]$ uniq [-ic]
```

选项与参数:

-i : 忽略大小写字符的不同;

-c : 进行计数

范例一: 使用 last 将账号列出, 仅取出账号栏, 进行排序后仅取出一位;

```
[dmtsai@study ~]$ last | cut -d ' ' -f1 | sort | uniq
```

屏幕剪辑的捕获时间: 2023/3/14 16:53

统计文件内的字数(英文), 行数, 字符数: wc

```
[dmtsai@study ~]$ wc [-lwm]
```

选项与参数:

-l : 仅列出行;

-w : 仅列出多少字(英文单字);

-m : 多少字符;

范例一: 那个 /etc/man_db.conf 里面到底有多少相关字、行、字符数?

```
[dmtsai@study ~]$ cat /etc/man_db.conf | wc
```

131 723 5171

输出的三个数字中, 分别代表: 『行、字数、字符数』

范例二: 我知道使用 last 可以输出登入者, 但是 last 最后两行并非账号内容, 那么请问, 我该如何以一行指令串取得登入系统的总人次?

```
[dmtsai@study ~]$ last | grep [a-zA-Z] | grep -v 'wtmp' | grep -v 'reboot' | \> grep -v 'unknown' | wc -l
```

由于 last 会输出空白行, wtmp, unknown, reboot 等无关账号登入的信息, 因此, 我利用 # grep 取出非空白行, 以及去除上述关键词那几行, 再计算行数, 就能够了解啰!

屏幕剪辑的捕获时间: 2023/3/14 19:57

cat /etc/passwd | wc -l

查看在账号文件中记录的账号数

双向重定向: tee

(将数据流分送到文件与屏幕)

```
[dmtsai@study ~]$ tee [-a] file
```

选项与参数:

-a : 以累加 (append) 的方式, 将数据加入 file 当中!

屏幕剪辑的捕获时间: 2023/3/15 14:45

删除一段信息中的指定的文字或进行文字替换: tr

```
[dmtsai@study ~]$ tr [-ds] SET1 ...
```

选项与参数:

-d : 删除讯息当中的 SET1 这个字符串;

-s : 取代掉重复的字符!

范例一: 将 last 输出的讯息中, 所有的小写变成大写字符:

```
[dmtsai@study ~]$ last | tr '[a-z]' '[A-Z]'
```

事实上, 没有加上单引号也是可以执行的, 如: [last | tr [a-z] [A-Z]]

范例二: 将 /etc/passwd 输出的讯息中, 将冒号 (:) 删除

```
[dmtsai@study ~]$ cat /etc/passwd | tr -d ':'
```

范例三: 将 /etc/passwd 转存成 dos 断行到 /root/passwd 中, 再将 ^M 符号删除

```
[dmtsai@study ~]$ cp /etc/passwd ~/passwd && unix2dos ~/passwd
```

```
[dmtsai@study ~]$ file /etc/passwd ~/passwd
```

```
/etc/passwd: ASCII text
```

/home/dmtsai/passwd: ASCII text, with CRLF line terminators <==就是 DOS 断行

```
[dmtsai@study ~]$ cat ~/passwd | tr -d '\r' > ~/passwd.linux
```

那个 \r 指的是 DOS 的断行字符, 关于更多的字符, 请参考 man tr

```
[dmtsai@study ~]$ ll /etc/passwd ~/passwd*
```

```
-rw-r--r--. 1 root    root   2092 Jun 17 00:20 /etc/passwd  
-rw-r--r--. 1 dmtsai dmtsai 2133 Jul  9 22:13 /home/dmtsai/passwd  
-rw-rw-r--. 1 dmtsai dmtsai 2092 Jul  9 22:13 /home/dmtsai/passwd.linux  
# 处理过后，发现文件大小与原本的 /etc/passwd 就一致了！
```

屏幕剪辑的捕获时间: 2023/3/15 14:54

将tab转换成空格: col

```
[dmtsai@study ~]$ col [-xb]
```

选项与参数:

-x : 将 tab 键转换成对等的空格键

范例一：利用 cat -A 显示出所有特殊按键，最后以 col 将 [tab] 转成空白

```
[dmtsai@study ~]$ cat -A /etc/man_db.conf <==此时会看到很多 ^I 的符号，那就是 tab
```

```
[dmtsai@study ~]$ cat /etc/man_db.conf | col -x | cat -A | more
```

```
# 嘿嘿！如此一来，[tab] 按键会被取代成为空格键，输出就美观多了！
```

屏幕剪辑的捕获时间: 2023/3/15 14:59

head -n 数字 文件 文件 ...

从头显示对应数字个数量的行数

对比两个文件的数据相关性，有相同数据的行，则将两行贴在一起: join

```
[dmtsai@study ~]$ join [-t i12] file1 file2
```

选项与参数:

-t : join 默认以空格符分隔数据，并且比对『第一个字段』的数据，

如果两个文件相同，则将两笔数据联成一行，且第一个字段放在第一个！

-i : 忽略大小写的差异；

-1 : 这个是数字的 1，代表『第一个文件要用那个字段来分析』的意思；

-2 : 代表『第二个文件要用那个字段来分析』的意思。

范例一：用 root 的身份，将 /etc/passwd 与 /etc/shadow 相关数据整合成一栏

```
[root@study ~]# head -n 3 /etc/passwd /etc/shadow
```

```
==> /etc/passwd <=
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin

==> /etc/shadow <=
root:$6$wtbCCce/PxMeE5wm$KE2IfSJr...:16559:0:99999:7:::
bin:*:16372:0:99999:7:::
daemon:*:16372:0:99999:7:::

# 由输出的资料可以发现这两个文件的最左边字段都是相同账号！且以 : 分隔

[ root@study ~]# join -t ':' /etc/passwd /etc/shadow | head -n 3
root:x:0:0:root:/root:/bin/bash:$6$wtbCCce/PxMeE5wm$KE2IfSJr...:16559:0:99999:7:::
bin:x:1:1:bin:/bin:/sbin/nologin:*:16372:0:99999:7:::
daemon:x:2:2:daemon:/sbin:/sbin/nologin:*:16372:0:99999:7:::

# 透过上面这个动作，我们可以将两个文件第一字段相同者整合成一列！

# 第二个文件的相同字段并不会显示(因为已经在最左边的字段出现了啊！)
```

范例二：我们知道 /etc/passwd 第四个字段是 GID，那个 GID 记录在 /etc/group 当中的第三个字段，请问如何将两个文件整合？

```
[ root@study ~]# head -n 3 /etc/passwd /etc/group
==> /etc/passwd <=
root:x:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin

==> /etc/group <=
root:x:0:
bin:x:1:
daemon:x:2:
# 从上面可以看到，确实有相同的部分喔！赶紧来整合一下！
```

```
[root@study ~]# join -t ':' -1 4 /etc/passwd -2 3 /etc/group | head -n 3
0:root:x:0:root:/bin/bash:root:x:
1:bin:x:1:bin:/bin:/sbin/nologin:bin:x:
2:daemon:x:2:daemon:/sbin:/sbin/nologin:daemon:x:
# 同样的，相同的字段部分被移动到最前面了！所以第二个文件的内容就没再显示。
# 请读者们配合上述显示两个文件的实际内容来比对！
```

屏幕剪辑的捕获时间: 2023/3/15 15:09

直接将两个文件的内容的两行贴在一起: paste

```
[dmtsa1@study ~]$ paste [-d] file1 file2
```

选项与参数:

-d : 后面可以接分隔字符。预设是以 [tab] 来分隔的！

- : 如果 file 部分写成 -，表示来自 standard input 的资料的意思。

范例一：用 root 身份，将 /etc/passwd 与 /etc/shadow 同一行贴在一起

```
[root@study ~]# paste /etc/passwd /etc/shadow
```

```
root:x:0:0:root:/bin/bash root:$6$wtbCCce/PxMeE5wm$KE2IfSJr...:16559:0:99999:7:::
```

```
bin:x:1:1:bin:/bin:/sbin/nologin bin:*:16372:0:99999:7:::
```

```
daemon:x:2:2:daemon:/sbin:/sbin/nologin daemon:*:16372:0:99999:7:::
```

注意喔！同一行中间是以 [tab] 按键隔开的！

范例二：先将 /etc/group 读出(用 cat)，然后与范例一贴上一起！且仅取出前三行

```
[root@study ~]# cat /etc/group|paste /etc/passwd /etc/shadow -l|head -n 3
```

这个例子的重点在那个 - 的使用！那玩意儿常常代表 stdin 呀！

屏幕剪辑的捕获时间: 2023/3/15 15:19

将tab转换成自定义个数量的空格: expand

```
[dmtsa1@study ~]$ expand [-t] file
```

选项与参数:

-t : 后面可以接数字。一般来说，一个 tab 按键可以用 8 个空格键取代。

我们也可以自行定义一个 [tab] 按键代表多少个字符呢！

屏幕剪辑的捕获时间: 2023/3/15 15:24

将大文件划分成几个小文件: split

```
[dmtsai@study ~]$ split [-b1] file PREFIX
```

选项与参数:

-b : 后面可接欲分区成的文件大小, 可加单位, 例如 b, k, m 等;

-l : 以行数来进行分区。

PREFIX : 代表前导符的意思, 可作为分区文件的前导文字。

范例一: 我的 /etc/services 有六百多 K, 若想要分成 300K 一个文件时?

```
[dmtsai@study ~]$ cd /tmp; split -b 300k /etc/services services
```

```
[dmtsai@study tmp]$ ll -k services*
```

```
-rw-rw-r--. 1 dmtsai dmtsai 307200 Jul 9 22:52 servicesaa
```

```
-rw-rw-r--. 1 dmtsai dmtsai 307200 Jul 9 22:52 servicesab
```

```
-rw-rw-r--. 1 dmtsai dmtsai 55893 Jul 9 22:52 servicesac
```

那个档名可以随意取的啦! 我们只要写上前导文字, 小文件就会以

xxxaa, xxxab, xxxac 等方式来建立小文件的!

范例三: 使用 ls -al / 输出的信息中, 每十行记录成一个文件

```
[dmtsai@study tmp]$ ls -al / | split -l 10 - lsroot
```

```
[dmtsai@study tmp]$ wc -l lsroot*
```

```
10 lsrootaa
```

```
10 lsrootab
```

```
4 lsrootac
```

```
24 total
```

重点在那个 - 啦! 一般来说, 如果需要 stdout/stdin 时, 但偏偏又没有文件,

有的只是 - 时, 那么那个 - 就会被当成 stdin 或 stdout ~

屏幕剪辑的捕获时间: 2023/3/15 15:52

注

-会被当成是stdin或stdout

xargs是给命令传递参数的一个过滤器,也是组合多个命令的一个工具.它把一个数据流分割为一些足够小的块,以方便过滤器和命令进行处理.由

此命令也是后置引用的一个强有力的替换。在一般使用过多参数的命令替换失败的时候，用xargs来替换它一般都能成功。通常情况下，xargs从管道的输出作为输入或者stdin中读取数据，但是它也能够从文件的输出中读取数据。

来自 <<https://www.bbsmax.com/A/lk5aEAWa51/>>

需要将一些命令串在一起，但是其中一个命令不接受管道输入的情况。在这种情况下，我们就可以使用xargs命令。xargs可以将一个命令的输出作为参数发送给另一个命令。

来自 <<https://baijiahao.baidu.com/s?id=1675957551367318263&wfr=spider&for=pc>>

```
[dmtsai@study ~]$ xargs [-0epn] command
```

选项与参数：

- 0：如果输入的stdin含有特殊字符，例如`，\，空格键等等字符时，这个-0参数可以将他还原成一般字符。这个参数可以用于特殊状态喔！
- e：这个是EOF(end of file)的意思。后面可以接一个字符串，当xargs分析到这个字符串时，就会停止继续工作！
- p：在执行每个指令的argument时，都会询问使用者的意思：
- n：后面接次数，每次command指令执行时，要使用几个参数的意思。
当xargs后面没有接任何的指令时，默认是以echo来进行输出喔！

屏幕剪辑的捕获时间: 2023/3/15 16:10

```
[dmtsai@study ~]$ cut -d ':' -f 1 /etc/passwd | head -n 3 | xargs -n 1 id  
uid=0(root) gid=0(root) groups=0(root)  
uid=1(bin) gid=1(bin) groups=1(bin)  
uid=2(daemon) gid=2(daemon) groups=2(daemon)  
# 透过-n来处理，一次给予一个参数，因此上述的结果就OK正常的显示啰！
```

屏幕剪辑的捕获时间: 2023/3/15 16:11

范例四：找出/usr/sbin底下具有特殊权限的档名，并使用ls -l列出详细属性

```
[dmtsai@study ~]$ find /usr/sbin -perm /7000 | xargs ls -l  
-rwx--s--x. 1 root lock      11208 Jun 10 2014 /usr/sbin/lockdev  
-rwsr-xr-x. 1 root root     113400 Mar  6 12:17 /usr/sbin/mount.nfs  
-rwxr-sr-x. 1 root root     11208 Mar  6 11:05 /usr/sbin/netreport  
.....(底下省略).....  
# 聪明的读者应该会想到使用『ls -l $(find /usr/sbin -perm /7000)』来处理这个范例！  
# 都OK！能解决问题的方法，就是好方法！
```

屏幕剪辑的捕获时间: 2023/3/15 16:12

第11章

2023年3月18日 13:40

注正则表达式与通配符是完全不同的东西

一些特殊符号的意义：

特殊符号	代表意义
[alnum:]	代表英文大小写字符及数字，亦即 0-9, A-Z, a-z
[alpha:]	代表任何英文大小写字符，亦即 A-Z, a-z
[blank:]	代表空格键与 [Tab] 按键两者
[cntrl:]	代表键盘上面的控制按键，亦即包括 CR, LF, Tab, Del.. 等等
[digit:]	代表数字而已，亦即 0-9
[graph:]	除了空格符（空格键与 [Tab] 按键）外的其他所有按键
[lower:]	代表小写字符，亦即 a-z
[print:]	代表任何可以被打印出来的字符
[punct:]	代表标点符号（punctuation symbol），亦即：“ ’ ? ! ; : # \$...”
[upper:]	代表大写字符，亦即 A-Z
[space:]	任何会产生空白的字符，包括空格键，[Tab], CR 等等
[xdigit:]	代表 16 进位的数字类型，因此包括：0-9, A-F, a-f 的数字与字符

屏幕剪辑的捕获时间: 2023/3/18 14:09

grep的一些高级选项：

```
[dmtsa@study ~]$ grep [-A] [-B] [--color=auto] '搜寻字符串' filename
```

选项与参数：

-A : 后面可加数字，为 after 的意思，除了列出该行外，后续的 n 行也列出来；
-B : 后面可加数字，为 before 的意思，除了列出该行外，前面的 n 行也列出来；
--color=auto 可将正确的那个撷取数据列出颜色

屏幕剪辑的捕获时间: 2023/3/18 14:17

```
[luhao@localhost ~]$ dmesg | grep -n -A2 -B2 'XFS'
1760-[    2.003396] e1000 0000:02:01.0 eth0: (PCI:66MHz:32-bit) 00:0c:29
:f4:77:e6
1761-[    2.003400] e1000 0000:02:01.0 eth0: Intel(R) PRO/1000 Network C
onnection
1762:[    2.230217] SGI XFS with ACLs, security attributes, no debug ena
bled
1763:[    2.232154] XFS (dm-0): Mounting V5 Filesystem
1764:[    2.395255] XFS (dm-0): Starting recovery (logdev: internal)
1765:[    2.445683] XFS (dm-0): Ending recovery (logdev: internal)
1766-[    2.805984] systemd-journald[89]: Received SIGTERM from PID 1 (s
ystemd).
1767-[    2.923345] random: crng init done
--
1791-[   12.452804] input: PC Speaker as /devices/platform/pcspkr/input/
input5
1792-[   13.052900] cryptd: max_cpu_qlen set to 1000
1793:[   13.130878] XFS (sda1): Mounting V5 Filesystem
1794-[   13.166037] Adding 2097148k swap on /dev/mapper/centos-swap.  Pr
iority:-2 extents:1 across:2097148k FS
1795-[   13.331633] ppdev: user-space parallel port driver
--
1798-[   13.384317] alg: No test for __gcm-aes-aesni (__driver-gcm-aes-a
esni)
1799-[   13.384417] alg: No test for __generic-gcm-aes-aesni (__driver-g
eneric-gcm-aes-aesni)
1800:[   13.693605] XFS (sda1): Starting recovery (logdev: internal)
1801:[   13.734738] XFS (sda1): Ending recovery (logdev: internal)
1802-[   15.233979] floppy0: no floppy controllers found
1803-[   15.234031] work still pending
```

屏幕剪辑的捕获时间: 2023/3/18 14:20

区分

```
[luhao@localhost ~]$ grep -n '[^a-z]o' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
3:Football game is not use feet only.
5:However, this dress is about $ 3183 dollars.^M
15:You are the best is mean you are the no. 1.
[luhao@localhost ~]$ grep -n '^*[a-z]' regular_express.txt
2:apple is my favorite food.
4:this dress doesn't fit me.
10:motorcycle is cheap than car.
12:the symbol '*' is represented as start.
18:google is the best tools for search keyword.
19:gooooooogle yes!
20:go! go! Let's go.
```

屏幕剪辑的捕获时间: 2023/3/20 10:22

```
[luhao@localhost ~]$ grep -n '^[^[:alpha:]]' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
21:# I am VBird
```

屏幕剪辑的捕获时间: 2023/3/20 10:26

任意一个字符.与重复字符*

```
[luhao@localhost ~]$ grep -n 'g..d' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
9:Oh! The soup taste good.^M
16:The world <Happy> is the same with "glad".
```

```
[luhao@localhost ~]$ grep -n 'ooo*' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
2:apple is my favorite food.
3:Football game is not use feet only.
9:Oh! The soup taste good.^M
18:google is the best tools for search keyword.
19:oooooooole yes!
```

屏幕剪辑的捕获时间: 2023/3/20 10:53

找含有2个o以上的字符串行:

```
[luhao@localhost ~]$ grep -n 'gooo*g' regular_express.txt
18:google is the best tools for search keyword.
19:oooooooole yes!
```

```
[luhao@localhost ~]$ grep -n 'go\{2,\}g' regular_express.txt
18:google is the best tools for search keyword.
19:oooooooole yes!
```

屏幕剪辑的捕获时间: 2023/3/20 11:06

sed [选项] [操作]

将数据进行替换、删除、新增、选取等操作

作用于一整个行的处理

```
[dmtsa@study ~]$ sed [-nefr] [动作]
```

选项与参数:

-n : 使用安静(silent)模式。在一般 sed 的用法中，所有来自 STDIN 的数据一般都会被列出到屏幕上。

但如果加上 -n 参数后，则只有经过 sed 特殊处理的那一行(或者动作)才会被列出来。

-e : 直接在指令列模式上进行 sed 的动作编辑;

-f : 直接将 sed 的动作写在一个文件内， -f filename 则可以执行 filename 内的 sed 动作；
-r : sed 的动作支持的是延伸型正规表示法的语法。(预设是基础正规表示法语法)
-i : 直接修改读取的文件内容，而不是由屏幕输出。

动作说明： [n1[,n2]]function

n1, n2 : 不见得会存在，一般代表『选择进行动作的行数』，举例来说，如果我的动作是需要在 10 到 20 行之间进行的，则『 10,20[动作行为] 』

function 有底下这些咚咚：

a : 新增， a 的后面可以接字符串，而这些字符串会在新的一行出现(目前的下一行)～
c : 取代， c 的后面可以接字符串，这些字符串可以取代 n1,n2 之间的行！
d : 删除，因为是删除啊，所以 d 后面通常不接任何咚咚；
i : 插入， i 的后面可以接字符串，而这些字符串会在新的一行出现(目前的上一行)；
p : 打印，亦即将某个选择的数据印出。通常 p 会与参数 sed -n 一起运作～
s : 取代，可以直接进行取代的工作哩！通常这个 s 的动作可以搭配正规表示法！
例如 1,20s/old/new/g 就是啦！

屏幕剪辑的捕获时间: 2023/3/20 11:20

```
[luhao@localhost ~]$ nl /etc/passwd | sed '2,5d'
      1  root:x:0:0:root:/root:/bin/bash
      6  sync:x:5:0:sync:/sbin:/bin/sync
      7  shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
      8  halt:x:7:0:halt:/sbin:/sbin/halt
      9  mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
     10  operator:x:11:0:operator:/root:/sbin/nologin
     11  games:x:12:100:games:/usr/games:/sbin/nologin
```

屏幕剪辑的捕获时间: 2023/3/20 11:21

```
[luhao@localhost ~]$ nl /etc/passwd | sed '2,5a append data'
    1  root:x:0:0:root:/root:/bin/bash
    2  bin:x:1:1:bin:/bin:/sbin/nologin
append data
    3  daemon:x:2:2:daemon:/sbin:/sbin/nologin
append data
    4  adm:x:3:4:adm:/var/adm:/sbin/nologin
append data
    5  lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
append data
    6  sync:x:5:0:sync:/sbin:/bin/sync
```

屏幕剪辑的捕获时间: 2023/3/20 11:24

取出想要显示的行号对应的内容:

```
[luhao@localhost ~]$ nl /etc/passwd | sed -n '2,5p'
    2  bin:x:1:1:bin:/bin:/sbin/nologin
    3  daemon:x:2:2:daemon:/sbin:/sbin/nologin
    4  adm:x:3:4:adm:/var/adm:/sbin/nologin
    5  lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

屏幕剪辑的捕获时间: 2023/3/20 11:29

sed 's/要被替换的字符串/新的字符串/g' '部分数据的查找与替换:'
[luhao@localhost ~]\$ /sbin/ifconfig | grep 'inet '
 inet 192.168.175.131 netmask 255.255.255.0 broadcast 192.168.175.255
 inet 127.0.0.1 netmask 255.0.0.0
 inet 192.168.122.1 netmask 255.255.255.0 broadcast 192.168.122.255

```
[luhao@localhost ~]$ /sbin/ifconfig | grep 'inet '|sed 's/^.*inet//g'
192.168.175.131  netmask 255.255.255.0  broadcast 192.168.175.255
127.0.0.1  netmask 255.0.0.0
192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
```

```
[luhao@localhost ~]$ /sbin/ifconfig | grep 'inet '|sed 's/^.*inet//g'|sed 's/ *netmask.*$/g'
192.168.175.131
127.0.0.1
192.168.122.1
```

屏幕剪辑的捕获时间: 2023/3/20 13:47

swd后如果要接超过两个以上的操作时，每个操作前要加-e

```
[dmtsa@study testpw]$ cat /etc/passwd | sed -e '4d' -e '6c no six line' > passwd.new
# 注意一下， sed 后面如果要接超过两个以上的动作时，每个动作前面得加 -e 才行！
# 透过这个动作，在 /tmp/testpw 里面便有新旧的 passwd 文件存在了！
```

屏幕剪辑的捕获时间: 2023/3/22 9:53

找到含MAN字符串的行:

```
[luhao@localhost ~]$ cat /etc/man_db.conf |grep 'MAN'
# MANDATORY_MANPATH                         manpath_element
# MANPATH_MAP                   path_element   manpath_element
# MANDB_MAP                      global_manpath [relative_catpath]
# every automatically generated MANPATH includes these fields
#MANDATORY_MANPATH                  /usr/src/pvm3/man
MANDATORY_MANPATH                 /usr/man
MANDATORY_MANPATH                 /usr/share/man
MANDATORY_MANPATH                 /usr/local/share/man
# set up PATH to MANPATH mapping
# *PATH*                                *MANPATH*
```

删掉注释后的内容:

```
[luhao@localhost ~]$ cat /etc/man_db.conf |grep 'MAN'|sed 's/#.*$///g'

MANDATORY_MANPATH                 /usr/man
MANDATORY_MANPATH                 /usr/share/man
MANDATORY_MANPATH                 /usr/local/share/man

MANPATH_MAP           /bin          /usr/share/man
```

删掉空行:

```
[luhao@localhost ~]$ cat /etc/man_db.conf |grep 'MAN'|sed 's/#.*$///g' \
> | sed '/^$/d'
MANDATORY_MANPATH                 /usr/man
MANDATORY_MANPATH                 /usr/share/man
MANDATORY_MANPATH                 /usr/local/share/man
MANPATH_MAP           /bin          /usr/share/man
MANPATH_MAP           /usr/bin      /usr/share/man
```

屏幕剪辑的捕获时间: 2023/3/20 14:00

sed -i

支持修改源文件的内容

范例六：利用 sed 将 regular_express.txt 内每一行结尾若为 . 则换成 !

```
[dmtsaï@study ~]$ sed -i 's/\.$/!/g' regular_express.txt
```

上头的 -i 选项可以让你的 sed 直接去修改后面接的文件内容而不是由屏幕输出喔！

这个范例是用在取代！请您自行 cat 该文件去查阅结果啰！

范例七：利用 sed 直接在 regular_express.txt 最后一行加入『# This is a test』

```
[dmtsaï@study ~]$ sed -i '$a # This is a test' regular_express.txt
```

由于 \$ 代表的是最后一行，而 a 的动作是新增，因此该文件最后新增啰！

屏幕剪辑的捕获时间: 2023/3/20 14:04

grep仅支持基础正则表达式，而egrep，或grep -E 支持扩展正则表达式

格式化打印：printf

```
[dmtsaï@study ~]$ printf '打印格式' 实际内容
```

选项与参数：

关于格式方面的几个特殊样式：

\a 警告声音输出

\b 退格键(backspace)

\f 清除屏幕 (form feed)

\n 输出新的一行

\r 亦即 Enter 按键

\t 水平的 [tab] 按键

\v 垂直的 [tab] 按键

\xNN NN 为两位数的数字，可以转换数字成为字符。

关于 C 程序语言内，常见的变数格式

%ns 那个 n 是数字， s 代表 string ，亦即多少个字符；

%ni 那个 n 是数字， i 代表 integer ，亦即多少整数字数；

%N.nf 那个 n 与 N 都是数字， f 代表 floating (浮点)，如果有小数字数，

假设我共要十个位数，但小数点有两位，即为 %10.2f 哪！

屏幕剪辑的捕获时间: 2023/3/20 14:26

范例二：将上述资料关于第二行以后，分别以字符串、整数、小数点来显示：

```
[dmtsaï@study ~]$ printf '%10s %5i %5i %5i %8.2f \n' $(cat printf.txt | grep -v Name)
```

DmTsai	80	60	92	77.33
--------	----	----	----	-------

VBird	75	55	80	70.00
-------	----	----	----	-------

Ken	60	90	70	73.33
-----	----	----	----	-------

屏幕剪辑的捕获时间: 2023/3/24 10:48

```
awk '条件类型{操作} 条件类型{操作}...'
```

对一行当中分成数个字段处理
以行为一次的处理单位，以字段为最小的处理单位

```
[dmtsai@study ~]$ awk '条件类型 1{动作 1} 条件类型 2{动作 2} ...' filename
```

```
[luhao@localhost ~]$ last -n 5
luhao    pts/0        192.168.175.1      Tue Mar 21 10:44 still logged in
reboot   system boot  3.10.0-1160.83.1 Tue Mar 21 10:37 - 11:05 (00:27)
luhao    pts/0        192.168.175.1      Mon Mar 20 10:10 - crash (1+00:27)
)
reboot   system boot  3.10.0-1160.83.1 Mon Mar 20 10:09 - 11:05 (1+00:55)
)
luhao    pts/0        192.168.175.1      Sat Mar 18 14:11 - crash (1+19:57)
)

[luhao@localhost ~]$ last -n 5 | awk '{print $1"\t"$3}'
luhao    192.168.175.1
reboot   boot
luhao    192.168.175.1
reboot   boot
luhao    192.168.175.1
```

屏幕剪辑的捕获时间: 2023/3/21 11:08

变量名称	代表意义
NF	每一行 (\$0) 拥有的字段总数
NR	目前 awk 所处理的是『第几行』数据
FS	目前的分隔字符， 默认是空格键

屏幕剪辑的捕获时间: 2023/3/21 11:14

```
[dmtsai@study ~]$ last -n 5 | awk '{print $1 "\t" NR "\t" NF}'
dmtsai  lines: 1      columns: 10
dmtsai  lines: 2      columns: 10
dmtsai  lines: 3      columns: 10
dmtsai  lines: 4      columns: 10
dmtsai  lines: 5      columns: 9
# 注意喔，在 awk 内的 NR, NF 等变量要用大写，且不需要有钱字号 $ 啦！
```

屏幕剪辑的捕获时间: 2023/3/21 11:14

awk的逻辑运算符的使用:

```
[dmtsaï@study ~]$ cat /etc/passwd | awk '{FS=":"} $3 < 10 {print $1 "\t " $3}'  
root:x:0:0:root:/root:/bin/bash  
bin      1  
daemon   2  
....(以下省略)....  
  
[dmtsaï@study ~]$ cat /etc/passwd | awk 'BEGIN {FS=":"} $3 < 10 {print $1 "\t " $3}'  
root      0  
bin      1  
daemon   2  
.....(以下省略).....  
  
[dmtsaï@study ~]$ cat pay.txt | \  
> awk 'NR==1{printf "%10s %10s %10s %10s %10s\n", $1,$2,$3,$4,"Total" }  
> NR>=2{total = $2 + $3 + $4  
> printf "%10s %10d %10d %10d %10.2f\n", $1, $2, $3, $4, total}'  


| Name   | 1st   | 2nd   | 3th   | Total     |
|--------|-------|-------|-------|-----------|
| VBi rd | 23000 | 24000 | 25000 | 72000.00  |
| DMTsai | 21000 | 20000 | 23000 | 64000.00  |
| Bird2  | 43000 | 42000 | 41000 | 126000.00 |


```

屏幕剪辑的捕获时间: 2023/3/21 11:22

非变量的文字部分，要用双引号括起

在awk中，变量可以直接使用，不用加上\$号

对比文件的命令: diff (主要用于纯文本文件)

```
[dmtsai@study ~]$ diff [-bBi] from-file to-file
```

选项与参数:

from-file : 一个档名, 作为原始比对文件的档名;

to-file : 一个档名, 作为目的比对文件的档名;

注意, from-file 或 to-file 可以 - 取代, 那个 - 代表『Standard input』之意。

-b : 忽略一行当中, 仅有多个空白的差异(例如 "about me" 与 "about me" 视为相同)

-B : 忽略空白行的差异。

-i : 忽略大小写的不同。

范例一: 比对 passwd.old 与 passwd.new 的差异:

```
[dmtsai@study testpw]$ diff passwd.old passwd.new
```

4d3 <==左边第四行被删除 (d) 掉了, 基准是右边的第三行

< adm:x:3:4:adm:/var/adm:/sbin/nologin <==这边列出左边(<)文件被删除的那一行内容

6c5 <==左边文件的第六行被取代 (c) 成右边文件的第五行

< sync:x:5:0:sync:/sbin:/bin/sync <==左边(<)文件第六行内容

> no six line <==右边(>)文件第五行内容

很聪明吧! 用 diff 就把我们刚刚的处理给比对完毕了!

屏幕剪辑的捕获时间: 2023/3/22 9:59

范例一: 比对 passwd.old 与 passwd.new 的差异:

```
[dmtsai@study testpw]$ diff passwd.old passwd.new
```

4d3 <==左边第四行被删除 (d) 掉了, 基准是右边的第三行

< adm:x:3:4:adm:/var/adm:/sbin/nologin <==这边列出左边(<)文件被删除的那一行内容

6c5 <==左边文件的第六行被取代 (c) 成右边文件的第五行

< sync:x:5:0:sync:/sbin:/bin/sync <==左边(<)文件第六行内容

> no six line <==右边(>)文件第五行内容

很聪明吧! 用 diff 就把我们刚刚的处理给比对完毕了!

屏幕剪辑的捕获时间: 2023/3/22 10:06

diff以行为单位对比, cmp以字节为单位对比:

(cmp支持纯文本与二进制文件)

```
[dmtsai@study ~]$ cmp [-l] file1 file2
```

选项与参数：

-l : 将所有的不同点的字节处都列出来。因为 cmp 预设仅会输出第一个发现的不同点。

范例一：用 cmp 比较一下 passwd.old 及 passwd.new

```
[dmtsai@study testpw]$ cmp passwd.old passwd.new
```

```
passwd.old passwd.new differ: char 106, line 4
```

屏幕剪辑的捕获时间: 2023/3/22 10:09

将旧的文件升级成新的文件，或将新文件还原为旧文件，可以使用diff生成补丁文件，再用patch来操作

文件打印: pr

```
[dmtsai@study ~]$ pr /etc/man_db.conf
```

2014-06-10 05:35

/etc/man_db.conf

Page 1

```
#  
#  
# This file is used by the man-db package to configure the man and cat paths.  
# It is also used to provide a manpath for those without one by examining  
# configure script.
```

屏幕剪辑的捕获时间: 2023/3/22 10:26

找出在/etc下含有*号的文件与内容：

```
[dmtsai@study ~]$ grep '\*' $(find /etc -type f ) 2> /dev/null
```

如果只想列出档名而不要列出内容的话，使用底下的方式来处理即可喔！

```
[dmtsai@study ~]$ grep -l '\*' $(find /etc -type f ) 2> /dev/null
```

屏幕剪辑的捕获时间: 2023/3/22 11:14

搜索对象为整个系统：

```
[dmtsai@study ~]$ find / -type f 2> /dev/null | xargs -n 10 grep -l '\*'
```

屏幕剪辑的捕获时间: 2023/3/22 11:15

(指令列的长度有限制，当报错指令列表太长时可以考虑用管道和xargs)

第12章

2023年3月22日 11:17

shell是命令行模式下我们与系统沟通的一个接口，而脚本就是利用shell的命令(搭配正则表达式、管道、数据流重定向等功能)所写的程序。shell脚本不需要编译即可执行。

编写脚本的良好习惯：

```
[luhao@localhost bin]$ cat hello.sh
#!/bin/bash

#说明
#1.内容与功能
#2.版本信息
#3.作者与联络方式
#4.建立文件的日期
#5.历史记录

#2023.3.24 luhao

PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/~/bin

export PATH

echo -e "Hello World! \a\n"

exit 0
```

屏幕剪辑的捕获时间: 2023/3/24 10:36

编写交互式脚本：

```
1 #!/bin/bash
2
3 #交互式脚本
4
5 read -p "请输入您的姓 :" firstname
6 read -p "请输入您的名 :" lastname
7
8 echo "Your full name is:"
9
10 #这样会输出两行
11 #echo ${firstname}
12 #echo ${lastname}
13
14 echo ${firstname} ${lastname}
15
16 exit 0
```

屏幕剪辑的捕获时间: 2023/3/24 11:05

编写自动建立前天、昨天、今天为文件名的三个文件的脚本

```
1 #!/bin/bash
2
3 #自动建立前天昨天今天为文件名的3个文件
4
5 echo -e "我将使用touch命令"
6
7 read -p "请输入文件名：" fileuser
8
9 #为了避免使用者随意按Enter，故进行变量测试
10 filename=${fileuser:-"未命名"}
11
12 date1=$(date --date='2 days ago' +%Y%m%d)
13 date2=$(date --date='1 days ago' +%Y%m%d)
14 date3=$(date +%Y%m%d)
15
16 file1=${filename}${date1}
17 file2=${filename}${date2}
18 file3=${filename}${date3}
19
20 touch "${file1}"
21 touch "${file2}"
22 touch "${file3}"
```

屏幕剪辑的捕获时间: 2023/3/24 13:54

注区分\${}和\$()

编写整数数值运算脚本

```

1 #!/bin/bash
2
3 #整数乘法
4
5 echo -e "请输入两个整数 : \n"
6
7 read -p "第一个整数 :" firstn
8 read -p "第二个整数 :" secondn
9
10 declare -i total=${firstn}*${secondn}
11
12 #法二 :
13 #total=$(( ${firstn}*${secondn})) 
14 |
15 echo -e "${firstn}与${secondn}相乘得${total}\n"
16
17 exit 0
18

```

屏幕剪辑的捕获时间: 2023/3/24 14:06

编写计算指定精度的圆周率的脚本

```

[dmtsa@study bin]$ vim cal_pi.sh
#!/bin/bash
# Program:
#           User input a scale number to calculate pi number.
# History:
# 2015/07/16      VBird      First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH
echo -e "This program will calculate pi value. \n"
echo -e "You should input a float number to calculate pi value.\n"
read -p "The scale number (10~10000) ? " checking
num=${checking:-"10"}          # 开始判断有否有输入数值
echo -e "Starting calcuate pi value. Be patient."
time echo "scale=${num}; 4*a(1)" | bc -lq

```

屏幕剪辑的捕获时间: 2023/3/24 14:13

不使用source：脚本会在新的bash(即子进程)中执行，执行完毕后子进程内的所有数据会被删除。

利用source来执行脚本：在父进程中执行

自定义的变量可以在父进程中存在

注 脚本中不要写exit 0 否则脚本运行结束后用户会重新登录！

```
[luhao@localhost bin]$ source showname.sh
请输入您的姓 : a
请输入您的名 : b
Your full name is:
a b
[luhao@localhost bin]$ echo ${firstname}
a
```

屏幕剪辑的捕获时间: 2023/3/25 10:30

测试某样东西是否存在: test

```
[luhao@localhost bin]$ ll
总用量 16
-rw-rw-r--. 1 luhao luhao 0 3月 24 13:41 abc20230322
-rw-rw-r--. 1 luhao luhao 0 3月 24 13:41 abc20230323
-rw-rw-r--. 1 luhao luhao 0 3月 24 13:41 abc20230324
-rw-rw-r--. 1 luhao luhao 482 3月 24 13:54 create_3_filename.sh
-rwxrwxr-x. 1 luhao luhao 263 3月 24 10:25 hello.sh
-rw-rw-r--. 1 luhao luhao 285 3月 24 14:07 multiply.sh
-rwxrw-r--. 1 luhao luhao 236 3月 25 10:28 showname.sh
[luhao@localhost bin]$ test -e abc.txt && echo "exit" || echo "noexit"
noexit
```

屏幕剪辑的捕获时间: 2023/3/25 10:40

使用中括号进行数据的判断：书P397

- 在中括号 [] 内的每个组件都需要有空格键来分隔；
- 在中括号内的变数，最好都以双引号括号起来；
- 在中括号内的常数，最好都以单或双引号括号起来。

屏幕剪辑的捕获时间: 2023/3/25 11:27

示例:

```
[luhao@localhost ~]$ name="a bc"
[luhao@localhost ~]$ [ ${name} == "a" ]
-bash: [: 参数太多
[luhao@localhost ~]$ [ "${name}" == "a" ] && echo "yes" || echo "no"
no
```

屏幕剪辑的捕获时间: 2023/3/25 11:30

```
[dmtsai@study bin]$ vim ans_yn.sh
#!/bin/bash
# Program:
#           This program shows the user's choice
# History:
# 2015/07/16      VBird      First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

read -p "Please input (Y/N): " yn
[ "${yn}" == "Y" -o "${yn}" == "y" ] && echo "OK, continue" && exit 0
[ "${yn}" == "N" -o "${yn}" == "n" ] && echo "Oh, interrupt!" && exit 0
echo "I don't know what your choice is" && exit 0
```

屏幕剪辑的捕获时间: 2023/3/25 11:32

shell脚本的默认变量: (书P399)

/path/to/scriptname	opt1	opt2	opt3	opt4
\$0	\$1	\$2	\$3	\$4

屏幕剪辑的捕获时间: 2023/3/25 13:36

- \$# : 代表后接的参数『个数』, 以上表为例这里显示为『 4 』;
- \$@ : 代表『 "\$1" "\$2" "\$3" "\$4" 』之意, 每个变量是独立的(用双引号括起来);
- \$* : 代表『 "\$1\$c\$2\$c\$3\$c\$4" 』, 其中 c 为分隔字符, 默认为空格键, 所以本例中代表『 "\$1 \$2 \$3 \$4" 』之意。

屏幕剪辑的捕获时间: 2023/3/25 13:37

```
[dmtsai@study bin]$ vim how_paras.sh
#!/bin/bash
# Program:
#           Program shows the script name, parameters...
# History:
# 2015/07/16      VBird   First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin
export PATH

echo "The script name is      => ${0}"
echo "Total parameter number is => $#"
[ "$#" -lt 2 ] && echo "The number of parameter is less than 2. Stop here." && exit 0
echo "Your whole parameter is => '$@'"
echo "The 1st parameter       => ${1}"
echo "The 2nd parameter       => ${2}"
```

屏幕剪辑的捕获时间: 2023/3/25 13:37

```
[dmtsai@study bin]$ sh how_paras.sh theone haha quot
The script name is      => how_paras.sh      <==檔名
Total parameter number is => 3                  <==果然有三个参数
Your whole parameter is => 'theone haha quot' <==参数的内容全部
The 1st parameter       => theone            <==第一个参数
The 2nd parameter       => haha              <==第二个参数
```

屏幕剪辑的捕获时间: 2023/3/25 13:37

造成参数变量号码偏移: shift (书P400)

写使用条件判断式if...then代替中括号的判断式的脚本:

```
[ "${yn}" == "Y" -o "${yn}" == "y" ]
上式可替换为
[ "${yn}" == "Y" ] || [ "${yn}" == "y" ]
```

屏幕剪辑的捕获时间: 2023/3/26 10:23

写法一:

```
1 #!/bin/bash
2
3 #显示用户的选择
4
5 PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/~/.bin
6
7 read -p "请输入您的选择(y/n):" yn
8
9 #错误写法
10 # if [ "${yn}"=="Y" ] || [ "${yn}"=="y" ];then
11 #   echo "continue"
12 #   exit 0
13 # fi
14
15 if [ "${yn}" == "Y" ] || [ "${yn}" == "y" ];then
16   echo "continue"
17   exit 0
18 fi
19
20 if [ "${yn}" == "N" ] || [ "${yn}" == "n" ];then
21   echo "interrupt"
22   exit 0
23 fi
24
25 echo "我不能理解您的选择" && exit 0
```

屏幕剪辑的捕获时间: 2023/3/26 11:36

写法二:

```
1  #!/bin/bash
2
3  #显示用户的选择
4
5  PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/usr/local/share/
6
7  read -p "请输入您的选择(y/n):" yn
8
9  #错误写法
10 # if [ "${yn}"=="Y" ] || [ "${yn}"=="y" ];then
11 #   echo "continue"
12 #   exit 0
13 # fi
14
15 # if [ "${yn}" == "Y" ] || [ "${yn}" == "y" ];then
16 #   echo "continue"
17 #   exit 0
18 # elif [ "${yn}" == "N" ] || [ "${yn}" == "n" ];then
19 #   echo "interrupt"
20 #   exit 0
21 # else
22 #   echo"我不能理解您的选择"
23 #   exit 0
24 # fi
25
```

屏幕剪辑的捕获时间: 2023/3/26 13:40

写让用户输入选项的脚本 if...then...elif...then...else

```
1  #!/bin/bash
2
3  #让用户输入参数
4
5  if [ "${1}" == "hello" ];then
6    echo "Hello, how are you ?"
7  elif [ "${1}" == "" ];then
8    echo "你必须输入一个参数, ex>${0} someword"
9  fi
10
```

屏幕剪辑的捕获时间: 2023/3/26 14:07

编写自动检测端口的脚本

```
1 #!/bin/bash
2
3 #使用netstat和grep命令，检测是否开启www,SSH,FTP和Mail端口
4
5 echo "现在我将检测您的Linux服务器的四个主要网络端口"
6 echo -e "检测是否开启www,SSH,FTP和Mail端口\n"
7
8 testfile=/dev/shm/netstat_checking.txt
9
10 netstat -tuln > ${testfile}
11
12 testing=$(grep ":80" ${testfile})
13 if [ "${testing}" != "" ];then
14     echo "www 存在"
15 fi
16
17 testing=$(grep ":22" ${testfile})
18 if [ "${testing}" != "" ];then
19     echo "SSH 存在"
20 fi
21
22 testing=$(grep ":21" ${testfile})
23 if [ "${testing}" != "" ];then
24     echo "FTP 存在"
25 fi
26
27 testing=$(grep ":25" ${testfile})
28 if [ "${testing}" != "" ];then
29     echo "Mail 存在"
30 fi
```

屏幕剪辑的捕获时间: 2023/3/27 9:40

编写计算两个日期间隔的天数的脚本

```
[dmtsai@study bin]$ vim cal_retired.sh
#!/bin/bash
# Program:
#       You input your demobilization date, I calculate how many days before you demobilize.
# History:
# 2015/07/16      VBird      First release
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:/bin
export PATH

# 1. 告知用户这支程序的用途，并且告知应该如何输入日期格式？
echo "This program will try to calculate :"
echo "How many days before your demobilization date..."
read -p "Please input your demobilization date (YYYYMMDD ex>20150716): " date2

# 2. 测试一下，这个输入的内容是否正确？利用正规表示法啰～
date_d=$(echo ${date2} |grep '[0-9]\{8\}')    # 看看是否有八个数字
if [ "${date_d}" == "" ]; then
    echo "You input the wrong date format...."
    exit 1
fi

# 3. 开始计算日期啰～
declare -i date_dem=$((date --date="${date2}" +%s))      # 退伍日期秒数
declare -i date_now=$((date +%s))                          # 现在日期秒数
declare -i date_total_s=$(((date_dem)-${date_now}))      # 剩余秒数统计
declare -i date_d=$((($date_total_s)/60/60/24))          # 转为日数
if [ "${date_total_s}" -lt "0" ]; then                  # 判断是否已退伍
    echo "You had been demobilization before: " $((-1*${date_d})) " ago"
else
    declare -i date_h=$(((($date_total_s)-${date_d}*60*60*24))/60/60)
    echo "You will demobilize after ${date_d} days and ${date_h} hours."
fi
```

使用case...esac编写脚本

```
$ hello_3.sh > hello_2.sh  
C: > Users > lenovo > AppData > Local > Temp > $ hello_3.sh  
1 #!/bin/bash  
2  
3 #使用case...esac判断  
4  
5 case ${1} in  
6 "hello")  
7 echo "Hello, how are you ?"  
8 ;;  
9 "")  
10 echo "你必须输入一个参数, ex>${0} someword"  
11 ;;  
12 *)  
13 echo "使用方法 : ${0} hello"  
14 ;;  
15 esac  
16
```

使用case...esac编写脚本

```
$ show123.sh ×

C: > Users > lenovo > AppData > Local > Temp > $ show123.sh

1  #!/bin/bash
2
3  #使用case...esac判断
4
5  read -p "请输入您的选择：" choice
6
7  case ${choice} in
8  "one")
9  echo "ONE"
10 ;;
11 "two")
12 echo "TWO}"
13 ;;
14 *)
15 echo "使用方法： ${0} {one|two}"
16 ;;
17 esac
```

屏幕剪辑的捕获时间: 2023/3/27 10:19

使用函数功能来编写脚本

```
$ show123.sh × $ show123_2.sh ×  
C: > Users > lenovo > AppData > Local > Temp > $ show123_2.sh  
1 #!/bin/bash  
2  
3 #使用case...esac判断和函数  
4  
5 function printit()  
6 {  
7     #echo -n 可以不换行，继续在同一行显示  
8     echo -n "您的选择是："  
9 }  
10  
11 case ${1} in  
12 "one")  
13     printit;  
14     echo ${1} | tr '[a-z]' '[A-Z]'  
15     ;;  
16 "two")  
17     printit;  
18     echo ${1} | tr '[a-z]' '[A-Z]'  
19     ;;  
20 *)  
21     printit;  
22     echo "使用方法： ${0} {one|two}"  
23     ;;  
24 esac
```

屏幕剪辑的捕获时间: 2023/3/27 10:34

注

被调函数中的\$0 \$1等与脚本中的\$0 \$1是不同的，可以理解成向函数传入实参

```
$ show123.sh × $ show123_2.sh × $ show123_3.sh ×
C: > Users > lenovo > AppData > Local > Temp > $ show123_3.sh
1  #!/bin/bash
2
3  #使用case...esac判断和函数
4
5  function printit()
6  {
7      #echo -n 可以不换行，继续在同一行显示
8      | echo -n "您的选择是：${1}"
9  }
10
11 case ${1} in
12 "one")
13     printit 1;
14     echo "拜拜"
15 ;;
16 "two")
17     printit 2;
18     echo "拜拜"
19 ;;
20 *)
21     printit;
22     echo "拜拜"
23 ;;
24 esac
```

屏幕剪辑的捕获时间: 2023/3/27 10:51

while do done循环
(条件成立就进行循环)

```
$ yes_to_stop.sh × $ show123_3.sh
C: > Users > lenovo > AppData > Local > Temp > $ yes_to_stop.sh
1 #!/bin/bash
2
3 #使用while do done循环
4
5 while [ "${yn}" != "yes" -a "${yn}" != "YES" ]
6 do
7     read -p "输入yes或YES以终止此循环：" yn
8 done
9
10 echo "OK,退出了"
```

屏幕剪辑的捕获时间: 2023/3/27 11:03

until do done循环
(条件成立时终止循环)

```
$ yes_to_stop.sh      $ yes_to_stop_2.sh • $ cal_1_100.sh      $ show12
C: > Users > lenovo > AppData > Local > Temp > $ yes_to_stop_2.sh
1 #!/bin/bash
2
3 #使用until do done循环
4
5 until [ "${yn}" == "yes" -o "${yn}" == "YES" ]
6 do
7     read -p "输入yes或YES以终止此循环：" yn
8 done
9
10 echo "OK,退出了"
```

屏幕剪辑的捕获时间: 2023/3/27 11:09

编写计算1加到100的脚本

```
$ yes_to_stop.sh      $ yes_to_stop_2.sh      $ cal_1_100.sh ×      $ show123_3.sh
C: > Users > lenovo > AppData > Local > Temp > $ cal_1_100.sh
1  #!/bin/bash
2
3  #使用while do done循环，计算1加到100
4
5  sum=0;
6  i=0;
7
8  while [ "${i}" != "100" ]
9  do
10   i=$((i+1))
11   sum=$((sum+i))
12  done
13
14 echo "结果为${sum}"
```

屏幕剪辑的捕获时间: 2023/3/27 11:17

使用for...do...done循环, 编写检查用户ID的脚本

```
$ yes_to_stop.sh      $ yes_to_stop_2.sh      $ cal_1_100.sh      $ userid.sh ×
C: > Users > lenovo > AppData > Local > Temp > $ userid.sh
1  #!/bin/bash
2
3  #使用for...do...done循环，检查用户ID
4
5  users=$(cut -d ':' -f1 /etc/passwd)
6
7  for username in ${users}
8  do
9    id ${username} #id 用于打印用户的uid gid
10 done
11
```

屏幕剪辑的捕获时间: 2023/3/27 11:33

for...do...done要从1循环到100:

for num in \$(seq 1 100) 或写为 for num in {1..100}

编写给出目录下各文件的权限的脚本

```
op.sh X $ yes_to_stop_2.sh $ cal_1_100.sh $ userid.sh $ dir_perm.sh X □
C: > Users > lenovo > AppData > Local > Temp > $ dir_perm.sh
4
5   read -p "请输入一个目录：" dir
6
7   if [ "${dir}" == "" -o ! -d "${dir}" ];then
8       echo "${dir}不是一个目录！"
9       exit 1;
10  fi
11
12  filelist=$(ls ${dir})
13  for filename in ${filelist}
14  do
15      perm=""
16      test -r "${dir}/${filename}" && perm="${perm} readable"
17      test -w "${dir}/${filename}" && perm="${perm} writable"
18      test -x "${dir}/${filename}" && perm="${perm} executable"
19      echo "文件${dir}/${filename}的权限是：${perm}"
20  done
21
```

屏幕剪辑的捕获时间: 2023/3/27 13:52

for...do...done的数值处理

```
$ cal_1_100_2.sh × $ cal_1_100.sh
C: > Users > lenovo > AppData > Local > Temp > $ cal_1_100_2.sh
1 #!/bin/bash
2
3 #使用for...do...done循环，计算1加到100
4
5 read -p "请输入一个正整数n，我将计算1+2+...+n的值" n
6
7 # 写法一：
8 # declare -i s=0      #注：若写成s=0则最终结果会变成0+1+2+...+100
9
10 # for (( i=1;i<=${n};i=i+1 ))
11 # do
12 #     s=$((s+i))
13 # done
14
15 #写法二：
16 s=0
17
18 for (( i=1;i<=${n};i=i+1 ))
19 do
20     s=$((s+i))
21 done
22
23 echo "结果是：$s"
```

屏幕剪辑的捕获时间: 2023/3/28 10:22

使用随机数编写脚本

RANDOM会随机生成0~32767的数字

```
[luhao@localhost bin]$ echo ${RANDOM}
26304
[luhao@localhost bin]$ echo ${RANDOM}
25391
[luhao@localhost bin]$ echo ${RANDOM}
24866
```

屏幕剪辑的捕获时间: 2023/3/28 10:32

```
[dmtsa1@study ~]$ sh [-nvx] scripts.sh
```

选项与参数：

- n : 不要执行 script，仅查询语法的问题；
- v : 再执行 script 前，先将 scripts 的内容输出到屏幕上；
- x : 将使用到的 script 内容显示到屏幕上，这是很有用的参数！

屏幕剪辑的捕获时间: 2023/3/28 10:38

```
[luhao@localhost bin]$ bash -x cal_1_100_2.sh
+ read -p 请输入一个正整数n，我将计算1+2+...+n的值 n
请输入一个正整数n，我将计算1+2+...+n的值100
+ s=0
+ (( i=1 ))
+ (( i<=100 ))
+ s=1
+ (( i=i+1 ))
+ (( i<=100 ))
+ s=3
+ (( i=i+1 ))
+ (( i<=100 ))
+ s=6
+ (( i=i+1 ))
+ (( i<=100 ))
+ s=10
+ (( i=i+1 ))
+ (( i<=100 ))
+ s=15
```

屏幕剪辑的捕获时间: 2023/3/28 10:39

计算还有几天可以过生日的脚本

```

55 echo -e "I will calculate you brithday .Please input you brithday(MMDD)"
56 read brithday
57
58 brithday_temp=$(echo $brithday | grep '[0-9]\{4\}')
59
60 if [ "$brithday_temp" == "" ] ; then
61     echo "You input is wrong !!"
62     exit 0
63 fi
64
65 now=$(date +%m%d)
66
67 if [ "$brithday" == "$now" ] ; then
68 echo "Today is your brithday "
69 elif [ "$brithday" -gt "$now" ] ; then
70     year=$(date +%Y)
71     brithday_second=$(date --date="$year$brithday" +%s)
72
73     now_second=$(date +%s)
74
75     brithday_days_temp=$((brithday_second - now_second))
76     brithday_days=$((brithday_days_temp/60/60/24))
77     echo "your brithday have $brithday_days days "
78 else
79     year=$(( $(date +%Y) + 1))
80     brithday_second=$(date --date="$year$brithday" +%s)
81
82     now_second=$(date +%s)
83
84     brithday_days_temp=$((brithday_second-now_second))
85     brithday_days=$((brithday_days_temp/60/60/24))
86     echo "your brithday have $brithday_days days "
87 fi
88 exit 0

```

屏幕剪辑的捕获时间: 2023/3/29 13:34

显示信息的第一栏内容的脚本

```
$ first_column.sh ● $ cal_birthday.sh ● $ cal_1_100.sh ● $ check_fileordir.sh
C: > Users > lenovo > AppData > Local > Temp > $ first_column.sh
1 #!/bin/bash
2
3 #将/etc/passwd的第一栏取出，而且每一栏都
4 #以一行字符串The 1 account is “root”来显示
5
6 users=$(cut -d ':' -f1 /etc/passwd)
7
8 count=1;
9 for username in ${users}
10 do
11     echo "The ${count} account is \"${username}\""
12     count=$(( ${count} +1))
13 done
```

屏幕剪辑的捕获时间: 2023/3/30 9:16

第13章

2023年3月30日 9:13

Linux主机并不会直接认识账号名称，而是仅认识ID号码

/etc/passwd 存不同账号的账号名称、UID、GID、家目录、登录时获取的shell等信息

/etc/shadow 存不同账号的账号名称、密码

计算某个日期的累计日数(从1970年1月1日作为1开始累加)

```
[luhao@localhost ~]$ echo $(( $(date --date="2023/3/30" +%s)/86400+1 ))
19446
```

屏幕剪辑的捕获时间: 2023/3/30 10:15

有效用户组，有效用户组的切换(书P425)

有效与支持用户组的观察: groups

有效用户组的切换: newgrp

新增用户: useradd

passwd [选项] 账号

设置用户密码

注系统账号主要用于执行系统所需的服务的权限设置，所以系统账号默认不主动建立家目录

私有用户组机制(P428)

更详细的密码参数显示: chage -l 账号名

在useradd命令完成，创建了新账号后，想修改账号的一些相关设置: usermod

删除用户的相关数据: userdel

查询某人或自己的相关UID/GID信息: id 用户名称

修改个人相关信息，如姓名，电话等: chfn

chsh -l

列出系统上合法的shell

chsh -s shell的完整路径

修改自己的shell

建立用户组: groupadd

修改用户组的相关参数: groupmod

删除用户组: groupdel

让某个用户组有一个管理员: gpasswd(书P438)

ACL即访问控制列表，可以针对单一用户，单一文件或目录进行r、w、x权限的设置

设置某个文件/目录的ACL规范: setfacl

```
[root@study ~]# setfacl [-bkRd] [{-m|-x} acl 参数] 目标文件名
```

选项与参数：

-m：设定后续的 acl 参数给文件使用，不可与 -x 合用；

-x：删除后续的 acl 参数，不可与 -m 合用；

-b：移除『所有的』 ACL 设定参数；

-k：移除『预设的』 ACL 参数，关于所谓的『预设』参数于后续范例中介绍；

-R：递归设定 acl，亦即包括次目录都会被设定起来；

-d：设定『预设 acl 参数』的意思！只对目录有效，在该目录新建的数据会引用此默认值

屏幕剪辑的捕获时间: 2023/3/31 11:17

针对单一用户：

```
# 1. 针对特定使用者的方式：
```

```
# 设定规范：『 u:[使用者账号列表]:[rwx] 』，例如针对 vbird1 的权限规范 rx：
```

```
[root@study ~]# touch acl_test1
```

```
[root@study ~]# ll acl_test1
```

```
-rw-r--r-- 1 root root 0 Jul 21 17:33 acl_test1
```

```
[root@study ~]# setfacl -m u:vbird1:rx acl_test1
```

```
[root@study ~]# ll acl_test1
```

```
-rw-rxr--+ 1 root root 0 Jul 21 17:33 acl_test1
```

```
# 权限部分多了个 +，且与原本的权限 (644) 看起来差异很大！但要如何查阅呢？
```

```
[root@study ~]# setfacl -m u::rwx acl_test1
```

```
[root@study ~]# ll acl_test1
```

```
-rwxr-xr--+ 1 root root 0 Jul 21 17:33 acl_test1
```

```
# 设定值中的 u 后面无使用者列表，代表设定该文件拥有者，所以上面显示 root 的权限成为 rwx 了！
```

屏幕剪辑的捕获时间: 2023/3/31 11:20

获取某个文件/目录的ACL设置选项: getfacl

```
[root@study ~]# getfacl filename
```

选项与参数：

getfacl 的选项几乎与 setfacl 相同！所以鸟哥这里就免去了选项的说明啊！

屏幕剪辑的捕获时间: 2023/3/31 11:23

```
# 请列出刚刚我们设定的 acl_test1 的权限内容:  
[root@study ~]# getfacl acl_test1  
# file: acl_test1  <==说明档名而已!  
# owner: root      <==说明此文件的拥有者，亦即 ls -l 看到的第三使用者字段  
# group: root      <==此文件的所属群组，亦即 ls -l 看到的第四群组字段  
user::rwx          <==使用者列表栏是空的，代表文件拥有者的权限  
user:vbird1:r-x    <==针对 vbird1 的权限设定为 rx ，与拥有者并不同!  
group::r--         <==针对文件群组的权限设定仅有 r  
mask::r-x          <==此文件预设的有效权限 (mask)  
other::r--         <==其他人拥有的权限啰!
```

屏幕剪辑的捕获时间: 2023/3/31 11:23

针对用户组:

```
# 2. 针对特定群组的方式:  
# 设定规范: 『 g:[群组列表]:[rwx] 』，例如针对 mygroup1 的权限规范 rx :  
[root@study ~]# setfacl -m g:mygroup1:rx acl_test1  
[root@study ~]# getfacl acl_test1  
# file: acl_test1  
# owner: root  
# group: root  
user::rwx  
user:vbird1:r-x  
group::r--  
group:mygroup1:r-x  <==这里就是新增的部分！多了这个群组的权限设定！  
mask::r-x  
other::r--
```

屏幕剪辑的捕获时间: 2023/3/31 11:28

使用默认权限，实现设置目录内的未来的文件的ACL权限继承

```
# 设定规范：『 d:[ug]:使用者列表:[ rwx ] 』
```

```
# 让 myuser1 在 /srv/projecta 底下一直具有 rx 的预设权限！
```

```
[root@study ~]# setfacl -m d:u:myuser1:rx /srv/projecta
```

屏幕剪辑的捕获时间: 2023/3/31 14:00

取消某个账号的设置值:

```
# 1.2 针对每个设定值来处理，注意，取消某个账号的 ACL 时，不需要加上权限项目！
```

```
[root@study ~]# setfacl -x u:myuser1 /srv/projecta
```

```
[root@study ~]# setfacl -x d:u:myuser1 /srv/projecta
```

```
# 2.1 开始让 pro3 这个用户无法使用该目录啰！
```

```
[root@study ~]# setfacl -m u:pro3:- /srv/projecta
```

屏幕剪辑的捕获时间: 2023/4/1 9:41

切换身份: su

```
[root@study ~]# su [-l[m]] [-c 指令] [username]
```

选项与参数:

- : 单纯使用 - 如『 su - 』代表使用 login-shell 的变量文件读取方式来登入系统；若使用者名称没有加上去，则代表切换为 root 的身份。
- l : 与 - 类似，但后面需要加欲切换的使用者账号！也是 login-shell 的方式。
- m : -m 与 -p 是一样的，表示『使用目前的环境设定，而不读取新使用者的配置文件』
- c : 仅进行一次指令，所以 -c 后面可以加上指令喔！

屏幕剪辑的捕获时间: 2023/4/1 9:45

```
[dmtsa1@study ~]$ su - -c "head -n 3 /etc/shadow"
```

Password: <==这里输入 root 的密码喔！

```
root:$6$wtbCCce/PxMeE5wm$KE2IfSJr.YLP7Rca16oa/T7KFhOYO62vDnqfLw85...:16559:0:99999:7:::
```

```
bin:*:16372:0:99999:7:::
```

```
daemon:*:16372:0:99999:7:::
```

```
[dmtsa1@study ~]$ <==注意看，身份还是 dmtsa1 哟！继续使用旧的身份进行系统操作！
```

屏幕剪辑的捕获时间: 2023/4/1 9:55

用户以其他身份去执行命令: sudo

```
[root@study ~]# sudo [-b] [-u 新使用者账号]
```

选项与参数：

-b : 将后续的指令放到背景中让系统自行执行，而不与目前的 shell 产生影响

-u : 后面可以接欲切换的使用者，若无此项则代表切换身为 root 。

范例一：你想要以 sshd 的身份在 /tmp 底下建立一个名为 mysshd 的文件

```
[root@study ~]# sudo -u sshd touch /tmp/mysshd
```

```
[root@study ~]# ll /tmp/mysshd
```

```
-rw-r--r--. 1 sshd sshd 0 Jul 21 23:37 /tmp/mysshd
```

特别留意，这个文件的权限是由 sshd 所建立的情况喔！

范例二：你想要以 vbird1 的身份建立 ~vbird1/www 并于其中建立 index.html 文件

```
[root@study ~]# sudo -u vbird1 sh -c "mkdir ~vbird1/www; cd ~vbird1/www; \  
> echo 'This is index.html file' > index.html"
```

```
[root@study ~]# ll -a ~vbird1/www
```

```
drwxr-xr-x. 2 vbird1 vbird1 23 Jul 21 23:38 .
```

```
drwx----- 6 vbird1 vbird1 4096 Jul 21 23:38 ..
```

```
-rw-r--r--. 1 vbird1 vbird1 24 Jul 21 23:38 index.html
```

要注意，建立者的身份是 vbird1，且我们使用 sh -c "一串指令" 来执行的！

屏幕剪辑的捕获时间: 2023/4/1 10:08

root用visudo命令去修改/etc/sudoers, 让某些账号能使用全部或部分的root命令 (书P448)

PAM(插入式验证模块) (书P452)

相关的PAM文件: /etc/security/limits.conf

查询目前已经登录在系统上的用户: w或who

查询每个账号最近登录的时间: lastlog

用户对话: write、mesg、wall

用户邮箱: mail

检查账号的一些信息(如家目录是否存在): pwck

将/etc/passwd内的账号与密码移动到/etc/shadow中

读取未加密前的密码，并加密后，将加密后的密码写入/etc/passwd中，可用于更新某个账号的密码：

```
[root@study ~]# echo "vbird3:abcdefg" | chpasswd
```

现已被如下方法替代:

范例三：使用 standard input 建立用户的密码

```
[root@study ~]# echo "abc543CC" | passwd --stdin vbird2
Changing password for user vbird2.
passwd: all authentication tokens updated successfully.
```

编写用passwd --stdin大量创建账号的脚本:

```
[root@study ~]# vim accountadd.sh
#!/bin/bash
# This shell script will create amount of linux login accounts for you.
# 1. check the "accountadd.txt" file exist? you must create that file manually.
#     one account name one line in the "accountadd.txt" file.
# 2. use openssl to create users password.
# 3. User must change his password in his first login.
# 4. more options check the following url:
# http://linux.vbird.org/linux_basic/0410accountmanager.php#manual_amount
# 2015/07/22    VBird
export PATH=/bin:/sbin:/usr/bin:/usr/sbin

# 0. userinput
```

```

usergroup=""                      # if your account need secondary group, add here.
pwmech="openssl"                 # "openssl" or "account" is needed.
homeperm="no"                     # if "yes" then I will modify home dir permission to 711

# 1. check the accountadd.txt file
action="${1}"                     # "create" is useradd and "delete" is userdel.
if [ ! -f accountadd.txt ]; then
    echo "There is no accountadd.txt file, stop here."
    exit 1
fi

[ "${usergroup}" != "" ] && groupadd -r ${usergroup}
rm -f outputpw.txt
usernames=$(cat accountadd.txt)

for username in ${usernames}
do
    case ${action} in
        "create")
            [ "${usergroup}" != "" ] && usegrp="-G ${usergroup}" || usegrp=""
            useradd ${usegrp} ${username}           # 新增账号
            [ "${pwmech}" == "openssl" ] && usepw=$(openssl rand -base64 6) || usepw=${username}
            echo ${usepw} | passwd --stdin ${username} # 建立密码
            chage -d 0 ${username}                  # 强制登入修改密码
            [ "${homeperm}" == "yes" ] && chmod 711 /home/${username}
            echo "username=${username}, password=${usepw}" >> outputpw.txt
            ;;
        "delete")
            echo "deleting ${username}"
            userdel -r ${username}
            ;;
        *)
            echo "Usage: $0 [create|delete]"
            ;;
    esac
done

```

屏幕剪辑的捕获时间: 2023/4/2 14:36

