## Bash

Bourne Again Shell（/bin/bash）

#! ：一个约定的标记，它告诉系统这个脚本需要什么解释器来执行，即使用哪一种Shell，告诉系统其后路径所指定的程序即解释此脚本文件的Shell程序。

扩展名： .sh

echo：向窗口输出文本。

注释：

以“#”开头的行是注释，会被解释器忽略。没有多行注释，只有单行注释。

遇到大段的代码需要临时注释起来，过一会儿又取消注释，每一行加个#符号太费力了，可以把这一段要注释的代码用一对花括号括起来，定义成一个函数，没有地方调用这个函数，这块代码就不会执行，达到了和注释一样的效果。

运行Shell脚本有两种方法：

1．作为可执行程序

将代码保存为 .sh 文件，

|  |
| --- |
| #!/bin/bash  echo "Hello World!" |

cd到相应目录：

|  |
| --- |
| bogon:Desktop apple$ chmod +x ./t.sh #使脚本具有执行权限  bogon:Desktop apple$ ./t.sh #执行脚本  Hello World! |

注意：

执行：**./t.sh** 告诉系统在当前目录下寻找运行程序t.sh 。

执行：**t.sh** 系统会去PATH里寻找有没有t.sh ，而只有/bin,/sbin,/usr/bin等在PATH里，你的当前目录通常不在PATH里。

|  |
| --- |
| bogon:Desktop apple$ ./t.sh  -bash: ./t.sh: Permission denied |

2．作为解释器参数

直接运行解释器，其参数就是shell脚本的文件名：

|  |
| --- |
| bogon:Desktop apple$ /bin/sh t.sh  Hello World! |

这种方式运行脚本，不需要在第一行指定解释器信息，写了也么有用。

Shell变量：

定义变量：

|  |
| --- |
| your\_name=”mengyueping.com” |

注意：变量名和等号之间不能有空格。变量名命名规则：

* 首字符必须为字母（a-z， A-Z）。
* 中间不能有空格，可以使用下划线（\_） 。
* 不能使用标点符号。
* 不能使用bash里的关键字（可用help命令查看保留关键字）

可以用语句变量赋值：

|  |
| --- |
|  |

变量类型：

* 局部变量：在脚本或命令中定义，仅在当前shell实例中有效，其他shell启动的程序不能访问局部变量。
* 环境变量：所有的程序，包括shell启动的程序，都能访问环境变量，有些程序需要环境变量来保证其正常运行。必要的时候shell脚本也可以定义环境变量。
* shell变量：有shell程序设置的特殊变量。Shell变量中有一部分是环境变量，有一部分是局部变量，这些变量保证了shell的正常运行。

Shell字符串：

Shell传递参数：

处理参数：

|  |  |
| --- | --- |
| 参数处理 | 说明 |
| $# | 传递到脚本的参数个数 |
| $\* | 以一个单字符串显示所有向脚本传递的参数。  如："$\*"用 " 括起来的情况、以"$1 $2 … $n"的形式输出所有参数。 |
| $$ | 脚本运行的当前进程ID号 |
| $! | 后台运行的最后一个进程的ID号 |
| $@ | 与 $\* 相同，但是使用时加引号，并在引号中返回每个参数。  如 "$@"用 " 括起来的情况、以"$1" "$2" … "$n" 的形式输出所有参数。 |
| $- | 显示Shell使用的当前选项，与set命令功能相同。 |
| $? | 显示最后命令的退出状态。0表示没有错误，其他任何值表明有错误。 |

算术运算符：

|  |  |  |
| --- | --- | --- |
| 运算符 | 说明 | 举例 |
| + | 加法 | `expr $10 + $20` 结果为30 |
| - | 减法 | `expr $10 - $20` 结果为-10 |
| \* | 乘法 | `expr $10 \\* $20` 结果为200 |
| / | 除法 | `expr $20 / $10` 结果为2 |
| % | 取余 | `expr $20 % $10` 结果为0 |
| = | 赋值 | a=$b 把变量b的值赋给a |
| == | 相等。  用于比较两个数字，相同则返回true . | [$a == $b] 返回 false |
| != | 不相等。用于比较两个数字，不相同则返回true。 | [$a != &b] 返回true |
|  |  |  |

注意：条件表达式要放在方括号之间，并且要有空格。

关系运算符：

a=10 b=20

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

布尔运算符：

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

逻辑运算符：

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

字符串运算符：

|  |  |  |
| --- | --- | --- |
| 运算符 | 说明 | 举例 |
| = | 检测两个字符串是否相等，相等返回true | [ $a = $b ] 返回false |
| != | 检测两个字符串是否相等，不相等返回true | [ $a != $b ] 返回true |
| -z | 检测字符串长度是否为0，为0返回true | [ -z $a ]返回false |
| -n | 检测字符串长度是否为0，不为0返回true | [ -n $a ] 返回true |
| str | 检测字符串是否为空，不为空返回true | [ $a ] 返回true |

文件测试运算符：

文件测试运算符用于：检测Unix文件的各种属性。

|  |  |  |
| --- | --- | --- |
| 操作符 | 说明 | 举例 |
| -b file | 检测文件是否是**块设备文件**，如果是，则返回true | [ -b $file ] 返回false |
| -c file | 检测文件是否是**字符设备文件**，如果是，则返回true | [ -c $file ] 返回false |
| -d file | 检测文件是否是**目录**，如果是，则返回true | [ -d $file ] 返回false |
| -f file | 检测文件是否是**普通文件**（既不是目录，也不是设备文件），如果是，则返回true | [ -f $file ] 返回true |
| -g file | 检测文件是否**设置了SGID 位**，如果是，则返回true | [ -g $file ] 返回false |
| -k file | 检测文件是否设置了**粘着位（Sticky Bit）**,如果是，则返回true | [ -k $file ] 返回false |
| -p file | 检测文件是否是**有名管道**，如果是，则返回true | [ -p $file ] 返回false |
| -u file | 检测文件是否**设置了SUID位**，如果是，则返回true | [ -u $file ] 返回false |
| -r file | 检测文件是否**可读**，如果是，则返回true | [ -r $file ] 返回true |
| -w file | 检测文件是否**可写**，如果是，则返回true | [ -w $file ] 返回true |
| -x file | 检测文件是否**可执行**，如果是，则返回true | [ -x $file ] 返回true |
| -s file | 检测文件是否**为空**（文件大小是否大于0），不为空返回true | [ -s $file ] 返回true |
| -e file | 检测文件（包括目录）是否**存在**，如果是，则返回true. | [ -e $file ] 返回true |

printf的转义序列：

|  |  |
| --- | --- |
| 序列 | 说明 |
| \a | 警告字符，通常为ASCII的BEL字符 |
| \b | 后退 |
| \c | 抑制（不显示）输出结果中任何结尾的换行字符（只在%b格式指示符控制下的参数字符串中有效），而且，任何留在参数里的字符、任何接下来的参数以及任何留在格式字符串中的字符，都被忽略 |
| \f | 换页（formfeed） |
| \n | 换行 |
| \r | 回车（Carriage return） |
| \t | 水平制表符 |
| \v | 垂直制表符 |
| \\ | 一个字面上的反斜杠字符 |
| \ddd | 表示1到3位数八进制值的字符。仅在格式字符串中有效。 |
| \0ddd | 表示1到3位的八进制值字符 |
|  |  |

Shell test命令：

数值测试：

|  |  |
| --- | --- |
| 参数 | 说明 |
| -eq | 等于 则为真 |
| -ne | 不等于 则为真 |
| -gt | 大于 则为真 |
| -lt | 小于 则为真 |
| -ge | 大于等于 则为真 |
| -le | 小于等于 则为真 |
|  |  |

字符串测试：

|  |  |
| --- | --- |
| 运算符 | 说明 |
| = | 等于 则为真 |
| != | 不等于 则为真 |
| -z 字符串 | 字符串的长度为零 则为真 |
| -n 字符串 | 字符串的长度不为零 则为真 |
|  |  |

文件测试：

|  |  |
| --- | --- |
| 操作符 | 说明 |
| -e 文件名 | 如果文件存在 则为真 |
| -r 文件名 | 如果文件存在且可读 则为真 |
| -w 文件名 | 如果文件存在且可写 则为真 |
| -x 文件名 | 如果文件存在且可执行 则为真 |
| -s 文件名 | 如果文件存在且至少有一个字符 则为真 |
| -d 文件名 | 如果文件存在且为目录 则为真 |
| -f 文件名 | 如果文件存在且为普通文件 则为真 |
| -c 文件名 | 如果文件存在且为字符型特殊文件 则为真 |
| -b 文件名 | 如果文件存在且为块特殊文件 则为真 |

Shell 输入/输出重定向：

|  |  |
| --- | --- |
| 命令 | 说明 |
| command>file | 将输出重定向到file |
| command<file | 将输入重定向file |
| command>>file | 将输出以追加的方式重定向到file |
| n>file | 将文件描述符为n的文件重定向到file |
| n>>file | 将文件描述符为n的文件以追加的方式重定向到file |
| n>&m | 将输出文件m和n合并 |
| n<&m | 将输入文件m和n合并 |
| <<tag | 将开始标记tag和结束标记tag之间的内容作为输入 |
|  |  |

注意： 文件描述符0通常是标准输入（STDIN）, 1是标准输出（STDOUT）, 2是标准错误输出（STDERR）.

### Shell中的关联数组

关联数组：

普通数组：只能使用整数作为数组索引。

关联数组：可以使用字符串作为数组索引。

申明关联数组变量：

**#declare –A ass\_array1**

数组名[索引]=变量值

**#ass\_array1[index1]=pear**

## vim

编辑器设置

v方式一：临时性质的设置

方式二：永久性质的设置（修改**vimrc**文件）

语法高亮： **syntax on**

显示行号： **set number**

自动缩进：

**set autoindent**

**set cindent**

自动加入文件头：

shell高亮显示：(centos中能够使用，mac终端中不起作用)

基本格式：

**echo –e 终端颜色 + 显示内容 + 结束后的颜色**

echo -e "\e[1;30m www.mengyueping.com Hi~ \e[1;0m"

echo -e "\e[1;35m" "www.mengyueping.com Hi~" $(tput sgr0)

\e[1;0m" ： 使用颜色闭合，将终端颜色重新设置成黑色0m.

$(tput sgr0) ：屏幕终端进行重置成默认色。

## Shell典型应用

场景脚本结构：

Input

控制脚本

monitor\_man.sh

应用运行分析

日志分析

提取系统信息

Shell脚本应用场景：

提取Linux 操作系统信息

获取操作系统运行状态

分析应用状态

应用日志分析

shell能做什么？

Shell的开发环境及使用特性？

明白shell对于资源监控功能作用：总结出linux系统的资源使用，以及每个性能参数含义。

Shell将操作系统、应用、日志连接起来。

主控脚本实现

系统信息及系统状态分析

Nginx和mysql应用状态分析

应用日志分析

shell对于系统资源提取及运行状态分析

shell分析程序或者应用状态

shell日志提取分析

## Linux

Linux严格区分大小写；

Linux中所有内容以文件形式保存，包括硬件。一切内容皆文件。

Linux**不依靠扩展名区分文件类型**。

约定俗成的扩展名：(这些是给管理员使用的)

压缩包： **.gz** 、**.bz2** 、 **.tar.bz2** 、 **.tgz** 等

二进制软件包： **.rpm**

网页文件： **.html .php**

脚本文件： **.sh**

配置文件： **.conf**

字符界面优势：

字符界面占用的系统资源更少。

字符界面减少了出错、被攻击的可能性。

磁盘分区：

使用分区编辑器（partition editor）在磁盘上划分几个逻辑部分。 碟片一旦划分成数个分区（Partition），不同类的目录与文件可以存储进不同的分区。

分区类型：（分区号1234只能给主分区和扩展分区，不能给逻辑分区，逻辑分区是从5开始的。）

主分区：最多只能有4个。

扩展分区：

最多只能有1个。

主分区加扩展分区最多有4个。

不能写入数据，只能包含逻辑分区。

逻辑分区：

格式化：

（高级格式化）又称逻辑格式化，它是指根据用户选定的文件系统（如FAT16、FAT32、NTFS、EXT2、EXT3、EXT4等），在磁盘的特定区域写入特定数据，在分区中划出一片用于存放文件分配表、目录表等用于文件管理的磁盘空间。

硬件设备文件名：(/dev ：专门用来存放硬件文件)

|  |  |
| --- | --- |
| 硬件 | 设备文件名 |
| IDE硬盘 | /dev/hd[a-d] |
| SCSI/SATA/USB硬盘 | /dev/sd[a-p] |
| 光驱 | /dev/cdrom或/dev/hdc |
| 软盘 | /dev/fd[0-1] |
| 打印机（25针） | /dev/lp[0-2] |
| 打印机（USB） | /dev/usb/lp[0-15] |
| 鼠标 | /dev/mouse |

分区设备文件名

设备文件名：

/dev/hda1 (IDE硬件接口)

/dev/sda1 (SCSI硬盘接口、SATA硬盘接口)

挂载

必须分区：

/ （根分区）

swap分区（交换分区，内存2倍，不超过2GB。内存2G以上，swap分区和内存一样大）

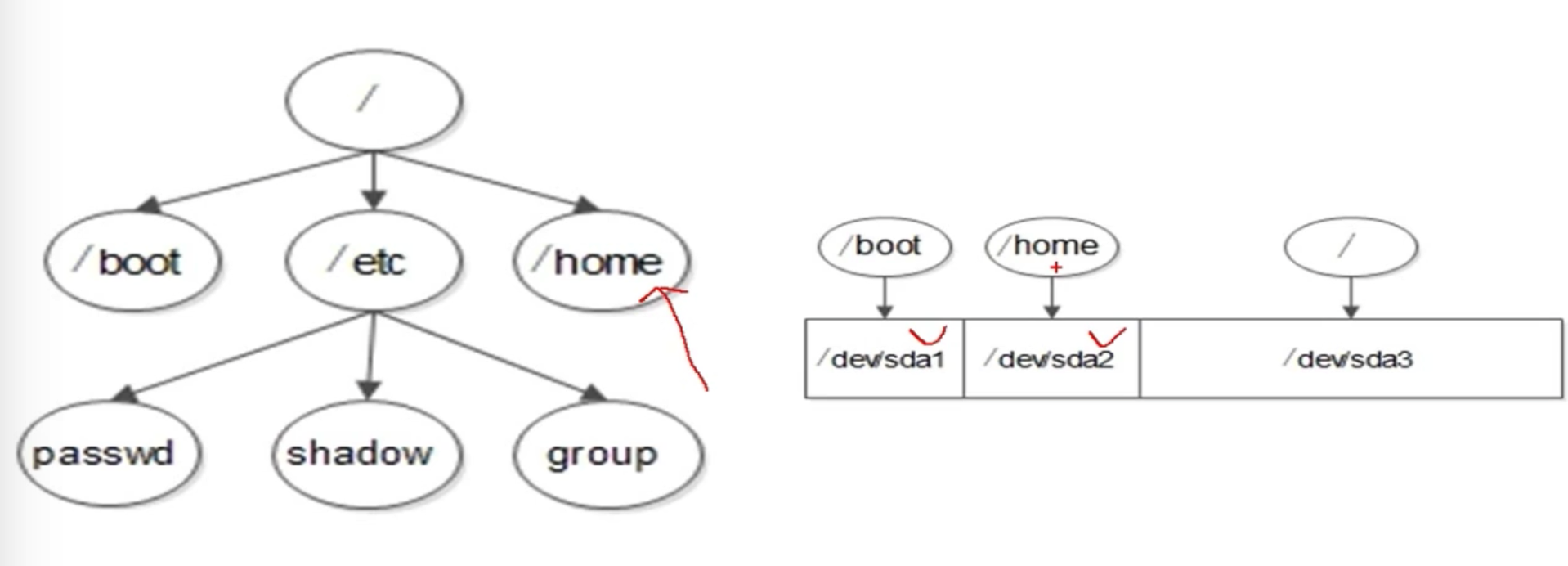
推荐分区：

/boot （启动分区，200MB）

文件系统结构：

从磁盘来看，属于不同分区。

从系统来看，/home、/etc、/boot 都属于/根目录下。



分区：把硬盘分为小的逻辑分区

格式化：写入文件系统

分区设备文件名：给每个分区定义设备文件名

挂载：给每个分区分配挂载点

centos安装日志：

/root/install.log : 存储了安装在系统中的软件包及其版本信息。

/root/install.log.syslog : 存储了安装过程中留下的事件记录

/root/anaconda-ks.cfg : 以Kickstart配置文件的格式记录安装过程中设置的选项信息。

### Linux常用命令

命令基本格式

**[root@localhost ~]#**

root代表当前登录用户

Linux中，管理员账号是root

文件处理命令

文件搜索命令

帮助命令

压缩与解压缩命令

关机和重启命令

其他常用命令