Chapter11_认识与学习Bash

认识bash这个shell

硬件、内核与shell

- P20 图0-18
- 操作系统是一组软件，由于这组软件在控制整个硬件与管理系统的活动监测，如果这组软件能被用户随意操作，若用户应用不当，将会使得整个系统崩溃
- shell提供用户操作 系统的接口
- 能够操作应用程序的接口都能称为shell，狭义的shell指的是命令行方面的软件（bash等），广义的shell包括图形界面的软件（图形界面也能够操作各种应用程序来调用内核工作）
- 可能学过linux系统编程之后会有更为深入的认识，先这样吧。。。

为何要学命令行界面的shell

- 《鸟哥》P295给出了很多理由，总之好好学shell
- 反正想学的会看，不想学的再好也不会看，多说无用。。。

系统的合法shell与/etc/shells功能

- Linux默认使用 bash(Bourne Again SHell)
- 系统合法的shell记录在 **`/etc/shells`** 文件中
    - 系统某些服务在运行过程中，会借助 `/etc/shells` 检查用户能够使用的shells
- 命令 **`cat /etc/passwd`** 可以查看登录时取得的是什么shell

bash shell的功能

- 命令记忆能力(history)，用过的命令保存在 **`~/.bash_history`** 中
    - 本次登录使用的命令存于内存中，注销系统后写入该文件；最好将记录的命令数目减少一点。
- 命令与文件补全功能(**`[Tab]`** 键)
- 命令别名设置功能(alias)
    - 直接的 alias 命令，可以查看目前的命令别名有哪些
    - **`alias lm = 'ls-al'`** 设置命令别名
- 作业控制(job control)、前台(foreground)、后台控制(background)
- 程序脚本(shell script)
    - 将管理系统常需执行的命令写成一个文件，并且该文件可以通过交互的方式来进行主机的检测工作，也可以通过shell提供的环境变量及相关命令来进行设计
- 通配符(Wildcard)

bash shell的内置命令：type

- **`type [-tpa] command_name`**
- 查询名为 command_name 的命令 commandh是 **外部命令还是内置在bash中的**

命令的执行

- 参见Chapter5

---

shell的变量功能

什么是变量

- 变量就是以一组文字或符号等，来替代一些设置或者是一串保留的数据

变量的设置规则

- 变量名未被设置时，内容默认为空
- 变量名与变量内容之间用 **等号=** 相连，且 **等号两边不能有空格**
- 变量名只能由 **字母、数字** 组成，且 **开头不能是数字**
- 系统默认变量名均为大写，自己设置的小写
- 变量内容还有空格，可以用 **双引号或单引号** 括起，但两者有区别
  - 双引号内的特殊字符如$等，可以保有原本的特性； **`echo $name -> TYQ`**
  - 单引号内的特殊字符仅为一个字符（纯文本）;**`echo &name -> &name`**
- 可以用转义字符"\"将特殊符号(`[Enter]`,`$`,`\`,空格,!等)变成一般字符
- 如果命令中还包含命令，内部命令用 **反单引号括起** 或用 **`$(command)`**
- 增加某个变量的内容
  - **`PATH=$PATH`:新内容**
  - **`PATH=${PATH}`:新内容**
  - **`PATH="$PATH"`:新内容**
- 变量需要在其他子进程中使用，需要 **以export来使变量变成环境变量**
  - **`export variable_name`**

变量的显式(`echo`)

- 三种方式显式变量
  - **`echo $variable_name`**
  - **`echo ${variable_name}`**
  - **`echo "$variable_name"`**

取消变量(`unset`)

- **`unset variable_name`**

环境变量的功能

- 用命令 **'env'** 查看环境变量
- 常见环境变量
  - HOME，当前用户的主文件夹的绝对路径
  - SHELL，当前用的哪个shell，**Linux默认/bin/bash**
  - HISTSIZE，history记录的历史命令的条数由这个值来设置
  - PATH，执行文件查找的路径（以及查找顺序），**目录与目录中间以冒号(:)分隔**
  - LANG，语系数据，后面有详解
  - RANDOM，随机数生成器，bash中 RANDOM的内容介于 **0~32767** 之间，其他范围需要使用表达式
- 用命令 **`set`** 查看所有变量（环境变量和自定义变量）
  - HISTFILE，历史命令记录的放置文件，隐藏文件

    - MAILCHECK，与邮件有关，按其值，每过一段时间就扫描一次信箱，看有没有新信
    - PS1，通过它设置命令提示符的样式，很多参数，P306
    - PS2，设置 输入多行 时的提示符 **>**
    - $，目前这个shell所使用的PID(Process ID), **echo $$** 打印输出其值
    - ?，这个变量是 **上一个执行的命令的回传值** ，上一个命令执行成功则回传0，上一个命令执行失败则回传对应的错误代码
- export（自定义变量与环境变量）
    - 自定义变量与环境变量的差异在于 **该变量是否会被子进程所继续引用**
    - 子进程 **仅会继承父进程的环境变量** ，不会继承父进程的自定义变量
    - 子进程的PID **不同于** 父进程的PID
    - 使用命令 **`export variable_name`** ，将名为variable_name的变量变为环境变量，子进程便可以继承该变量了

影响显示结果的语系变量(locale)

- 使用命令 **`locale`** 可以查看所用Linux支持多少语系
- 如果 **其他的语系变量都未设置** ， **且已经设置LANG或者是LC_ALL** ，则其他的语系变量就会被这两个变量所替代
- 为什么Linux主机在终端机接口(tty1~tty6)环境下，即使合理设置LANG的值，还是容易出现一堆乱码？
    - 因为Linux主机的 **终端机接口环境下** 是无法显式像中文这么复杂的编码文字的，产生乱码；可以通过安装中文化接口的软件解决
    - 如果是在Windows主机中 以 **远程连接服务器的软件** 连接到主机的话，其命令行界面是可以看到中文的

变量的有效范围

- 被export后的变量，可以被称为“环境变量”，它可以被子进程所引用，但是其他的自定义变量内容就不会存在于子进程中
    - 因为启动一个shell，操作系统会分配一个记忆块（内存空间）给shell使用，其中是可让子进程取用的环境变量
    - 子shell可以将父shell的环境变量所在的记忆块导入自己的环境变量记忆块中

变量键盘读取、数组与声明(read,declare,array)

- read
    - 读取来自键盘输入的变量，常用于shell script中
    - **`read [-pt] variable`**
        - `-p`，后可以接双引号括起来的提示信息
        - `-t`，设置等待用户输入的时间（用户可能一直不输入，则程序一直等待）
- declare/typeset
    - declare和typeset功能相同，声明变量的类型，**变量类型默认为“字符串”**
    - 使用declare时如果没有接任何参数，bash会将所有的变量名称与内容输出
    - declare语法
        - `-a`，将变量声明为数组(array)类型
        - `-i`，将变量声明为整型(integer)类型， **bash默认最多能到达整数类型**
        - `-x`，**`declare -x variable_name`** 和 **`export variable_name`** 功能一样，将后面名为variable_name的变量设置成环境变量
        - `+x`，**`declare +x variable_name`** 将环境变量编程自定义变量
        - `-r`，将后面的变量设置成只读(readonly)类型，不可被修改重设

- array
  - **`array_name[index]=content`**
  - 输出用 **`echo ${array_name[index]}`**
  - bash提供的是一维数组

与文件系统及程序的限制关系(ulimit)

- bash可以限制用户的某些系统资源
- 命令 **`ulimit -a`** ，显式当前用户的所有限制数据数值
- 具体设置P312
- 复原ulimit的设置最简单的方法是 **注销再登录** ，否则需要 **重新以 ulimit设置**

变量内容的删除、替代与替换

- 删除
  - **`${variable_name#删除内容格式}`**
  - **`${variable_name%删除内容格式}`**
  - **`${variable_name##删除内容格式}`**
  - **`${variable_name%%删除内容格式}`**
  - `#` ，从头开始删
  - `%` ，从尾开始删
  - `#,%`，后面即要删除的内容格式
  - `#` **或** `%`，单个，删除最短符合要求的
  - `##` **或** `%%`，双个，删除最长符合要求的
- 替代
  - **`${variable_name/old_String/new_String}`**
    - 将变量中 **第一个** old_string替换成new_String
  - **`${variable_name//old_String/new_String}`**
    - 将变量中 **所有** old_string替换成new_String
- 替换
  - 主要是P315、P316理解不透彻 总结不出规律，然后导致记不住

命令别名与历史命令

命令别名设置：alias,unalias

- 命令**`alias alias_name='command/...'`** ，为命令或变量起别名
- 命令**`unalias alias_name`** ，去掉别名（别名失效）

历史命令：history

- 命令 **`history [n]`**
  - 没有n，显式所有的历史命令
  - n为某个整数，则显式 **最近的n条命令**
- 其余参数参见P318-319
- 使用过的命令会被写入（记录在）**`~/.bash_history`** 中
  - 在关机注销时才会被写入，或者用命令， **`history -w`** 立即更新（之前是记录在内存中的）
- **`~/.bash_history`** 中记录的命令数由 HISTSIZE 决定

Bash Shell的操作环境

路径与命令查找程序

- 命令运行的顺序
  - 以相对路径或绝对路径执行命令，例如 **"/bin/ls"** 或 **"./ls"**
  - 由alias找到该命令来执行
  - 由bash内置的(builtin)命令来执行
  - 通过 **$PATH** 这个变量的顺序找到的第一个命令来执行

bash的登录与欢迎信息：/etc/issuc , /etc/motd

- 在终端机接口 **(tty1~tty6)** 登录的时候，会有几行提示的字符串，它们在 **`/etc/issue`**
- issue这个文件的内容也可以使用 **反斜杠作为变量调用** ，比如 **\d 表示本地端时间的日期**
- 具体参见 **`man issue 或者 man mingetty`**

bash的环境配置文件

- 前面谈到的命令、自定义的变量在 **注销bash后就会失效** ，要想保留设置，就要 **将这些设置写入配置文件**
- login shell
  - **login shell** ：取得bash时需要完整的登录流程
  - 比如，由 **tty1~tty6** 登录， **需要输入用户的账号密码**
- non-login shell
  - 取得bash接口的方法不需要重复登录
  - 以X的图形界面启动终端机， **不需要用户名密码** ，执行bash命令，子进也不用输入
- **`/etc/profile`**（login shell 才会读取）（全局设置）
  - 每个用户登录取得bash时一定会读取的配置文件。通过它可以设置所有用户的环境
  - 主要设置的参数
    - PATH，依据 **UID** 决定PATH变量要不要含有sbin的系统命令目录
    - USER，根据用户的账号设置此变量内容
    - HOSTNAME，依据hostname命令决定此变量内容
    - HISTSIZE，历史命令记录条数
  - 还会调用以下文件
    - **`/etc/inputrc`**，如果用户没有自定义输入的按键功能，/etc/profile设置“INPUTRC=/etc/inputrc”
    - **`/etc/profile.d/*.sh`**，规定bash操作接口的颜色、语系、ll与ls命令的别名等等
    - **`/etc/sysconfig/i18n`**，决定bash默认使用何种语系，最重要的变量是LANG
- **`~/.bash_profile`**（login shell 才会读取）（个人设置）
  - bash读完整体环境设置的 `/etc/profile` 并借此调用其他配置文件之后，会读取用户的个人配置文件
  - login shell 的bash环境中，读取的个人偏好配置文件主要有三个
    - **`~/.bash_profile`**
    - **`~/.bash_login`**
    - **`~/.profile`**
  - bash的login shell 设置只会读取上面三个文件中的一个，读取顺序依照上面的顺序

（ **读取第一个存在的** ）
    - 这么多是为了照顾其他shell转换过来的用户的习惯
    - 鸟哥的CentOS的 ~/.bash_profile 中（我的ubuntu是 ~/.profile）
        - 判断 **~/.bashrc** 文件是否存在，存在则调用（最终调用的还是 **~/.bashrc** ）
        - 还改变 PATH 变量的值
- **source**（读入环境配置文件的命令）
    - 配置文件写完后，用 **source config_file_name** 使配置文件生效，而不用注销重启
    - 遇到的事情：装了Scala，在 **/etc/profile** 中设置了环境变量，source了之后可以用，但是重新登录（X Window 登录），就不能用了，现在知道是因为 non-login shell 登录，不读取 /etc/profile ，source之后解决
- **~/.bashrc**（non-login shell 会读）
    - login shell 登录，会读取 /etc/profile 和 ~/.profile（ubuntu下，可能还有别的，见上面），~/.profile会调用读取 ~/.bashrc
    - non-login shell 则只会读取 ~/.bashrc（上面的例子也证明了），
- 其他相关配置文件
    - **/etc/man.config**，规定执行man的时候该去哪里查看数据的路径设置
    - **~/.bash_history**，记录历史命令，每次登录bash后，bash会先读取这个文件，将所有的历史命令读入内存
    - **~/.bash_logout**，设置注销bash后系统要完成的操作

终端机的环境设置：stty,set

- 主要是设置按键，目前不用深究，有更紧急的（其实是记不住。。。）P325

通配符与特殊符号

- 常用的通配符
    - ***** ，0到无穷多个任意字符
    - **?** ，一定有一个任意字符
    - **[abc]** ， 一定有方括号内的一个字符（abc可以换）
    - **[a-z]** ， 表示含有 a~z（包括a和z）内的所有字符 中 任意多个字符（a~z可以换）
    - **[^abc]** ，一定有一个字符，并且不是方括号内的（abc可以替换）
- 特殊符号
    - P328，有很多，类似于系统保留字，命名时（尽量）不要使用

数据流重定向

- 数据流重定向就是将 某个命令执行后 应该要出现在屏幕上的数据 传输到其他的地方，例如文件或者是设备（打印机之类）

标准输入，标准（错误）输出

- standard output（标准输出）与 standard error output（标准错误输出）
    - 标准输出指的是命令执行成功后，回传的正确的信息，**代码为1，使用 1> 或 1>>**（数字1一般省略）
    - 标准错误输出指的是命令执行失败后，回传的错误信息，**代码为2，使用 2> 或 2>>**
    - 输出文件的创建规则：不存在则创建，存在 **> 覆盖 ， >> 累加**

- /dev/null垃圾桶黑洞设备
  - 有意想将错误信息忽略掉而不是显示或存储，只要将输出信息 重定向到 **/dev/null** 就行
  - 命令行和脚本中很有用
- 特殊写法
  - 将正确与错误信息按序写入同一个文件
  - **find /home -name .bashrc > list 2>&1**
  - **find /home -name .bashrc &> list**
- standard input: < 与 <<
  - 将原本需要由键盘输入的数据改由文件内容来替代
  - 举个栗子： **cat > filename < ~/.bashrc**，创建一个文本文件，并将 **~/.bashrc** 的内容输入
  - 举个栗子： **cat > filename << "abc"**，创建一个文本文件，当输入 abc 时，结束输入

命令执行的判断依据：;,&&,||

- 想要将很多命令一次输入执行，而不是分次执行
  - shell script
  - 一次输入多重命令
- cmd1 ; cmd2
  - 不考虑命令相关性的连续命令执行
  - cmd1执行完，不管成功与否，执行cmd2
- cmd1 && cmd2 **和** cmd1 || cmd2
  - 命令回传码，**echo $?** 输出打印上一个命令执行后的命令回传码
  - **若 cmd1 执行完毕且正确执行($?=0)，则开始执行 cmd2**
  - **若 cmd1 执行完毕且为错误($?≠0)，则 cmd2 不执行**
  - **若 cmd1 执行完毕且正确执行($?=0)，则 cmd2 不执行**
  - **若 cmd1 执行完毕且为错误($?≠0)，则开始执行 cmd2**
- 由于命令是一个接着一个执行的，因此，如果使用判断，那么 && 和 || 的顺序就不能搞错
  - 如果假设判断式有三个，顺序通常是这样的（《鸟哥》P333有个很有意思的例子，有助理解）：
  - **cmd1 && cmd2 || cmd3**

管道命令(pipe)

- 很多命令，P334~P343，因为暂时用不到，先快速了解一下，用到的时候学

选取命令:cut,grep

- 选取信息通常是针对“行”来分析的，并不是整篇信息分析的
- cut 命令，可以将一段信息的某一段“切”出来，但在处理多空格相连的数据时比较吃力
- grep 命令，分析一行信息，若其中有我们所需要的信息，就将该行拿出来

排序命令：sort,uniq,wc

- sort命令，可以根据不同的数据类型来排序
- uniq命令，可以将重复的行删除只显示一个
- wc命令，计算文件有多少字符，多少字，多少行

双向重定向：tee

- tee命令，能同时将数据流 送与 文件与屏幕

字符转换命令：tr,col,join,paste,expand

- tr命令，可以删除一段信息中的文字，或者进行文字信息的替换
- col命令，有特殊用途
- join命令，主要是将两个文件中有相同数据的那一行加在一起
    - 使用join之前，需要处理的文件应该要先经过排序(sort)处理，否则有些对比的项目会被略过
- paste命令，直接将两行贴在一起，中间用[tab]键隔开
- expand命令，将[tab]键转换成空格
    - 还有unexpand命令，将空格转换成[tab]

切割命令：split

- split命令，可以将一个大文件依据文件大小或行数切割成小文件

参数代换：xargs

- xargs命令，产生某个命令的参数（不懂。。。）

关于减号的用途

- 在管道命令中，经常会使用到 前一个命令 的stdout作为这次的stdin，某些命令需要用到文件名（例如tar）来进行处理，该stdin与stdout可以利用减号"-"来替代

Chapter11_认识与学习Bash End~