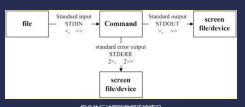
Bash

- type
  - 判断指令是否为bash shell的内置命令
  - 推荐可执行文件
- 变量
  - 设置规则
    - echo ${变量}　用法　echo　取用
    - 变量与值以=链接
      - 等号
      - =两边不能直接接空白字符
    - 变量值若有空白字符可使用单、双引号括起来
    - 内部除字符为纯文本　单、双引号
    - 内部除字符保持原本值　双
    - 特殊除字符变为一般字符　跳脱字符 \
    - 变量值设置时需要使用其他指令时使用
      - 【例：version=`uname -r`】　反单引号 `
    - 变量值设置时需要调用其他指令时使用
      - 【例：version=$(uname -r)】　$(指令)
    - 变量名称=
      - $变量名称
    - 扩展变量值
      - ${变量名称}
    - 将自定变量转换为环境变量　export
    - 将环境变量转换为自定变量　declare
    - 取消变量的设置　unset
  - 环境变量
    - 历史命令记录的笔数　HISTSIZE
    - 语系数据　LANG　常见环境变量
    - 产生随机数的变量，取用范围为0~32767　RANDOM
    - 列出目前shell环境下所有环境变量及其值　env
    - 查看所有变量　set
  - 重要系统内置环境变量
    - 命令提示字符　PS1
    - 关于本shell的PID　$
    - 上个执行的指令所回传的值
      - 【指令执行成功回传0，失败则回传非0值】　?
    - 主机硬件与核心的等级　OSTYPE、HOSTTYPE、MACHTYPE
  - 语系变量
    - 列出目前支持的语系　locale
    - /usr/lib/locale/　语系文件放置位置
    - /etc/locale.conf　系统默认语系定义
  - 输入读取
    - read [-pt] 变量　用法　read
  - 声明类型
    - declare [-aiorl] 变量　用法
    - 选项与参数
      - 将变量定义为数组类型　-a
      - 将变量定义为整数类型　-i
      - 用法与export相同，将变量转换成环境变量　-x
      - 将变量设置为只读，不可被修改、unset　-r
      - 变量类型默认为字符串；
        bash环境中的数值计算只能精确到整数位；　特性
      - xor(Linux)<<content　设置方式　数组类型
      - ulimit [-SHacdfltu] [限制]　用法　ulimit　系统资源限制
      - 删除配置文件、禁止再登录　取消
  - 删除、取代与替换
    - 删除
      - $(变量#关键字)
      - $(变量##关键字)　用法　#
      - 若变量值从头开始的数据符合关键字，则删除符合的最短数据　#　功能
      - 若变量值从头开始的数据符合关键字，则删除符合的最长数据　##
      - $(变量%关键字)
      - $(变量%%关键字)　用法　%
      - 若变量值从尾向前的数据符合关键字，则删除符合的最短数据　%　功能
      - 若变量值从尾向前的数据符合关键字，则删除符合的最长数据　%%
      - $(变量/旧字符串/新字符串)
      - $(变量//旧字符串/新字符串)　用法　/
      - 若变量的内容符合旧字符串，则第一个旧字符串会被新字符串替换　替换
      - 若变量的内容符合旧字符串，则全部的旧字符串会被新字符串替换　功能
    - 变量设置方式表格

变量设置方式	str 没有设置	str 为空字串	str 已设置非为空字串
var=${str-expr}	var=expr	var=	var=str
var=${str:-expr}	var=expr	var=expr	var=str
var=${str+expr}	var=	var=expr	var=expr
var=${str:+expr}	var=	var=	var=expr
var=${str=expr}	str=expr var=expr	str 不变 var=	str 不变 var=str
var=${str:=expr}	str=expr var=expr	str=expr var=expr	str 不变 var=str
var=${str?expr}	expr 输出至 stderr	var=	var=str
var=${str:?expr}	expr 输出至 stderr	expr 输出至 stderr	var=str

    - 善用法则示　测试及内容替换
- 命令别名与历史命令
  - 命令别名
    - 设置　alias　alias 别名=命令 ''
    - 取消　unalias　unalias 别名
  - 历史命令　history
    - 选项与参数
      - n　列出最近的n笔命令列表
      - -c　清除目前shell中的所有history内容
      - -a　将目前新增的history指令新增入hisfiles中
        （未指定则默认写入~/.bash_history）
      - -r　将hisfiles的内容读入目前shell的history记忆中
      - -w　将目前的history记忆写入hisfiles中
    - 用法　执行命令
      - !number　执行第几笔指令。
      - !command　从最近的指令前向搜索command并执行
      - !!　执行上一个指令。
    - 多Bash记录　最后登出的bash才会将最后写入~/.bash_history；
      无法记录时间
- 数据流重导向



  - 指令执行过程的数据传输情况
  - standard output
    （标准输出STDOUT）　含义　指令执行所回传的正确信息
  - standard error output
    （标准错误输出STDERR）　含义　指令执行失败后，所传回的错误信息
  - 分别传送
    - 含义　将数据流分开的机制
    - 所用特殊字符
      - 标准输入　(stdin)：代码为0，使用<或<<
      - 标准输出　(stdout)：代码为1，使用>或>>
      - 标准错误输出　(stderr)：代码为2，使用2>或2>>
    - 重定与复加
      - 重置　1> (>)　或2>
      - 重加　1>> (>>)　或2>>
  - 黑洞设备/dev/null　功能　接收所有传送至此设备的信息
  - &+或2>&1
    （将各自数据流写入黑洞设备时次序的差异）
  - 同一文件写入的特殊语法
  - standard input
    （标准输入STDIN）　<将<<
    - <　将原本需要由键盘输入的数据，改由文件内容来取代。
    - <<　结束的输入字符。
  - ;（分号）　不考虑指令相关性的连续指令下达。
  - 指令回传值　$?　若前一个指令执行结果正确，则会回传一个$?=0的值
  - 指令执行判断

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。

    - 根据$?值选择执行
  - &&或||
- 管线命令
  - 撰定符号 |　特性
    - 1.仅会处理stdin，对于stderr会忽略（使用数据流重导向即可以处理）
    - 2.必须要能够接受来自前一个指令的数据，变成stdin才能处理才行
  - 读取命令
    - cut
      - 功能　剪切文本数据分切（处理多空格信息的数据切割表）
      - 用法
        - 剪除部分指令的文本行　cut -d '分割字符' -f fields
        - 接到整天的文本　cut -c 字符区间
      - 选项与参数
        - -d　后接分隔字符。与 -f 一起使用
        - -f　根据 -d的分隔字符将文本分为数段，用 -f为取出第几段
        - -c　以字符的单位取出固定字符区间。
    - grep
      - 功能　多行信息处理
      - 用法　grep [-acinv] [--color=auto] 查找字符串 filename
      - 选项与参数
        - -a　将binary文件以text文件的方式查找数据。
        - -c　计算由中查找字符串的次数
        - -i　忽略大小写差异
        - -n　输出行号
        - -v　反向选择。
        - --color=auto　关键字部分高亮（CentOS7默认alias）
  - 排序命令
    - sort
      - 功能　根据不同的数据类型和语系编码对数据进行排序。
      - 用法　sort [-fbMnrtuk] [file or stdin]
      - 选项与参数
        - -f　忽略大小写差异。
        - -b　忽略最前的空白字符部分。
        - -M　以月份的英文名称来排序。
        - -n　使用 纯数字 进行排序。
        - -r　反向排序。
        - -u　即uniq，相同数据仅输出一行作为代表。
        - -t　分隔符号。默认为tab键。
        - -k　以哪个区间(field)来进行排序。
    - uniq
      - 功能　删除重复的行仅输出显示一行作为代表。
      - 用法　uniq [-ic]
      - 选项与参数
        - -i　忽略大小写。
        - -c　进行计数。
    - wc
      - 功能　计算输出信息的整体数据。
      - 用法　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。
      - 默认输出数字　分别代表行、字数、字符数。
  - 双向重定向　tee

    - 功能　复用前一段资讯　工作流程图
    - 用法　tee [-a] file
    - 选项与参数　-a　以累加（append）的方式，将数据加入file中（不加-a则以覆盖方式写文件）
  - 字符处理命令
    - tr
      - 功能　删除一段信息中的文字或是进行替换。
      - 用法　tr [-d-s] 字串1 .....
      - 选项与参数
        - -d　删除信息中的字串1
        - -s　取代掉重复的字符
    - col
      - 功能　将tab键取代成空白键。
      - 用法　col [-xb]
      - 选项与参数　-x　将tab键转换成对等的空白键
    - join
      - 功能　处理两个文件中有相同数据的一行（使用join前所需要处理的文件最好先经过排序，否则有可能会遭遇一些状况的困扰）
      - 用法　join [-t12] file1 file2
      - 选项与参数
        - join默认以空白字符分隔数据，且比对第一个字段的数据，若两文件相同，则将两笔数据联成一行，且第一个字段放在第一个
        - -t　指定分隔一个
        - -i　忽略大小写。
        - -1　第一个文件要使用那个字段来分析。
        - -2　第二个文件要使用那个字段来分析。
    - paste
      - 功能　将两行粘贴在一起，且中间以tab键隔开。
      - 用法　paste [-d] file1 file2
      - 选项与参数
        - -d　后可接分隔符，默认为tab键。
        - -　若file部分写为 - ，代表来自stdin的数据
    - expand
      - 功能　将tab键转换为空白键
      - 用法　expand [-t] file
      - 选项与参数　-t　后接数字，（一般一个tab键以8个空白键替换）
    - unexpand　功能　将空白键转为tab键
  - 切割命令　split
    - 功能　将文件根据文件大小或行数切割成小文件。
    - 用法　split [-bl] file PREFIX
    - 选项与参数
      - -b　可接要切割成的文件大小，可加单位，如b, k, m等
      - -l　以行数来进行切割
      - PREFIX　后置文件，可作为切割文件的前导文字。
  - 参数代换　xargs
    - 产生某个指令的参数
      （很多命令不支持管道命令，因此可以通过xargs来提供该命令引用stdin之用）
    - 功能
    - 用法　xargs [-0pen] command
    - 选项与参数
      - -0　若输入的stdin含有特殊字符，如 `、\、空白字符等时，可以将其原来一般字符（这参数可用于特殊状况）。
      - -e　接EOF，后可接一个字符串，当xargs分析到该字符串时，就会停止继续工作
      - -p　执行每个指令的参数时，都会向问用户。
      - -n　后接次数，每次command指令执行时，要使用几个参数。
    - 未指定指令　当xargs后未接任何指令时，默认以echo来进行输出
  - 减号（-）用法　某些命令需要用到文件名来进行处理时，该stdin与stdout可以用减号（-）来替代。
- 操作环境
  - 路径与指令搜寻序
    - 进站与欢迎信息
      - 进站欢迎信息
        文件内容可使用反斜线作为变量取用
        vi /etc
      - /etc/issue
      - 提供给telnet登录程序使用
        使用telnet连接时显示此文件内容　/etc/issue.net
      - 所所有用户在其登录后显示信息　/etc/motd
    - 由完整登录流程取得shell
      （由vi /etc登录并输入用户名和密码）

    - 配置文件读取流程
    - login shell
    - 取得bash接口的方法不需要重复登录
      （1.3 Windows模拟以GUI方式启动Terminal
      2.在bash环境下再次下达bash，那才相当）　non-login shell
    - login与non-login
    - 环境配置文件
      - /etc/profile
        - 系统整体设置，所有用户取得bash时一定会读取此　功能
        - 根据UID决定PATH变量是否要包含sbin的系统指令目录　PATH
        - 根据账号设置用户的mailbox到/var/spool/mail/用户名　MAIL
        - 根据使用者的账号设置此变量内容　USER
        - 根据主机的hostname指令决定此一变量内容　HOSTNAME
        - 历史命令记录笔数　HISTSIZE
        - root默认为022，一般用户为002等　umask
        - 设置变量
        - 文件名：
          只要在/etc/profile.d此目录内且扩展名为.sh，使用者具有-r权限，就会被调用。
          相关了bash操作相关的颜色、语系和指令别名等；　/etc/profile.d/*.sh
        - 决定bash默认使用哪种语系
          （由/etc/profile.d/lang.sh调用）　/etc/locale.conf
        - 调用外部数据
        - 补充功能（命令、文件名和指令的选项、参数等）　/usr/share/bash-completion/completions/*
        - 读取使用者的个人配置文件
          （1.立即执行此项目所有主文件夹下的~/bin/为额外的可执行检查目录
          2.指定~/.bashrc）　功能
        - ~/.bash_profile
        - ~/.bash_login
        - ~/.profile
        - 是三个文件按顺序读取，读取一个后其余不再读取　照义读取
        - ~/.bash_profile
      - 登入环境配置文件
        （每时生效，无需注销再登录）　功能
      - source 配置文件名
        配置文件名　用法　source指令/. 指令（小数点）
      - 仅non-login shell读入
        - 指定bash的环境
        - 指定/etc/bashrc　功能　~/.bashrc
        - 根据不同的UID指定umask值
        - 根据不同的UID指定出提示字符（PS1变量）；　/etc/bashrc
        - 调用/etc/profile.d/*.sh的设置　(Red Hat系特有)
        - 指定使用man时，man page路径去哪里寻找　/etc/man_db.conf
        - 记录历史命令　~/.bash_history
        - 记录登出后系统需要执行的操作　~/.bash_logout
    - 终端机环境设置
      - 列出所有的按键与按键内容　功能
      - 发送一个中断信号给正在运行的程序（即时止）　intr
      - 发送一个quit信号给正在运行的程序　quit
      - 向后删除字符　erase
      - 删除在目前命令行上的所有文字　kill
      - End of file，结束输入　eof
      - 在某程序停止后，重启输output　start
      - 停止目前所有的输出　stop
      - 发送一个terminal stop信号给正在运行的程序　susp
      - stty
      - 参数含义
      - 设置整个指令输出/输入的环境　set
    - 通配符/特殊符号
    - 组合按键与执行结果

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

bash默认组合键

符号	意义
*	代表“ o 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 的特殊字符就接受的意思。

常用通配符及其意义

符号	内容
#	注解符号：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“特殊字符或万用字符”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号：连续性命令的界定 （注意！与管线命令并不相同）
~	使用者的主文件夹
$	取用变量前置字符：亦即是变量之前需要加以变量取代值
&	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”之意思！
/	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（$ 查询该文本）
" "	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！

常用特殊符号及其意义

试用版 XMind:ZEN 试用版