1. 企业级Shell应用
   1. 命令知多少
      1. 内部命令
2. 集成在bash的命令是内部命令。内部命令依赖于shell类型。内部命令依赖于shell类型。这些命令由shell程序识别并在shell程序内部完成运行。通常在linux系统加载运行时shell就被加载并驻留在系统内存中。内部命令是写在bash源码中的,其执行速度比外部命令快, 因为解析内部命令shell不需要创建子进程。
3. enable管理内部命令

· 显示内部命令: enable

· 显示全部内部命令: enable -a

· 禁用内部命令: enable -n cd

· 启用内部命令: enable cd

1. 系统先检查内部命令状态, 如果打开, 使用内部命令, 如果未打开, 使用外部命令
2. 查看内部命令相关参数: help
   * 1. 外部命令
3. 外部命令是在bash之外额外安装的, 在文件系统路径$PATH有对应的可执行程序文件, 就是外部命令。在系统加载时并不随系统一起被加载到内存中, 而是在需要时才将其调用内存。
   * 1. 命令别名

· 在管理和维护Linux系统的过程中, 将会使用大量命令, 有一些很长的命令或用法经常被调用到, 重复而频繁地输入某个很长的命令或用法是不可取的。这时可以使用命令别名功能将这个过程简单化。

* + 1. hash

· 系统初始hash表为空, 当外部命令执行时, 默认会从PATH路径下寻找该命令, 找到后会将这条命令记录到hash表中, 再使用该命令时, shell解释器会首先查看hash表, 存在则执行, 不存在则去PATH路径下寻找,利用hash缓存表可大大提高命令的调用效率。

* + 1. 函数

· function: 函数是存在内存里的一组代码的命名元素。 函数创建于脚本运行环境之中, 并且可以执行。

* + 1. 复合命令

· 在shell中指循环, 判断, 分支, 选择的表达式命令

* 1. 命令优先级
     1. 获取一个命令的优先级别

alias

compound commands

function

build\_in

hash

$PATH

error: command not found

* + 1. hash操作

[root@ansible ~]# hash -p /usr/bin/ls echo

[root@ansible ~]# enable -n echo

[root@ansible ~]# echo

ansible.cfg hadoop.yml register.yml with\_items.yml

ansible\_soft.tar.xz include.yml useradd.retry with\_nested.retry

apache.yml ping.yml useradd.yml with\_nested.yml

hadoop plj.yml with\_items.retry xxoo

* + 1. 命令的解析顺序

1. 分隔成记号: 双引号 ==> 6, 单引号 ==> 11
2. 检测第一个非关键字: 开放的关键字 | 其他关键字
3. 检测第一个记号是不是别名: 是 ==> 扩展别名
4. 大括号扩展
5. 符号扩展
6. 参数扩展
7. 命令替换(嵌套命令行处理)
8. 算术扩展
9. 单词分割
10. 路径名扩展
11. 命令查询: 函数, 内置命令, 可执行文件
12. 运行命令: eval ==> 将参数带入下一个命令
    1. 命令行步骤
       1. 概述

Shell从标准输入或脚本中读取的每一行称为一个管道行, 它包含一个或多个由0个或多个管道字符(|)分隔的命令。对每个管道行, 进行12个步骤的处理。

* + 1. 步骤

1. 将命令行分成由固定元字符集分隔的记号:

SPACE, TAB, NEWLINE, ;, (, ), <, >, | ,&

记号类型包括单词, 关键字, I/O重定向符号和分号

1. 检测每个命令的第一个记号, 查看是否为不带引号或反斜线的关键字。如果是一个开放的关键字, 如 if和其他控制结构起始字符串, function, {或(, 则命令实际上为一复合命令。shell在内部对复合命令进行处理, 读取下一个命令, 并重复这一过程。如果关键字不是符合命令起始字符串(如then等一个控制结构中间出现的关键字), 则给出语法错误信号。
2. 依据别名列表检查每个命令的第一个关键字。如果找到相应匹配, 则替换其别名定义, 并退回第一步, 否则进入第4步。该策略允许递归别名, 还允许定义关键字别名。如alias procedure=function.
3. 执行大括号扩展, 例如a{b,c}编程ab ac
4. 如果~位于单词开头, 用$HOME替换~。使用user的主目录替换~user。
5. 对任何以符号$开头的表达式执行参数(变量)的替换
6. 对形式$(string)的表达式进行命令替换

这里是嵌套的命令行处理

1. 计算形式为$((string))的算术表达式
2. 把行的参数, 命令和算是替换部分再次分成单词, 这次它使用$IFS中的字符做成分隔符而不是步骤1的元字符集
3. 对出现\* , ? ,[ , / ,]对执行路径名扩展, 也称为通配符扩展
4. 按命令优先级表(跳过别名), 进行命令查询
5. 设置完I/O重定向和其他操作后执行该命令
   * 1. 关于引用
6. 单引号跳过了前10个步骤, 不能在单引号里放单引号
7. 双引号跳过了步骤1-5, 步骤9-10, 也就是说, 只处理6-8个步骤。也就是说, 双引号忽略了管道符, 别名, ~替换, 通配符扩展, 和通过分隔符分裂成单词

双引号里的单引号没有作用, 但双引号允许参数替换, 命令替换和算术表达式求值。可以在双引号里包含双引号, 方式是加上转义符"\", 还不许转移$, `, /。

1. <<<把后面的字符串读入
   * 1. Web备份脚本







