**Shell**

**需要提前设计智能化难度大 批量执行效率高 方便在后台运行**

**切换shell环境**

**通过usermod chsh更改登陆shell**

**cat /etc/shells**

**/bin/sh 多数Unix默认使用的shell /bin/bash 多数Linux默认使用的shell**

**在RHEL系统中 实际上sh是bash的符号连接**

**历史命令**

**保存位置： ~/.bash\_history 控制历史命令的数量 /etc/profile**

**Histroy -c :清空历史命令**

**标准输入：从该设备接收用户输入的数据 /dev/stdin 文件描述号：0 设备：键盘**

**标准输出:通过设备向用户输出数据 /dev/stdout 文件描述号:1 设备: 显示器**

**标准错误:通过该设备报告执行的出错信息 /dev/stderr 文件描述号;2 设备 显示器**

**重定向错误 2> 将命令行的执行出错信息保存到文件 2>>追加操作**

**混合重定向 &> 是>和2> 覆盖到同一个文件**

**变量 若指定的变量名已经存在 相当于为此变量重新赋值 等号两边不要有空格**

**变量名由字母/数字/下划线组成 区分大小写 变量名不能以数字开头 不要使用关键字和特殊字符**

**Unset取消变量**

**变量的种类、**

**存储类型 整数型 浮点型 双精度浮点型 字符型**

**使用类型**

|  |  |
| --- | --- |
| **类型** | **说明** |
| **环境变量** | **变量名通常都大写，由系统维护 用来设置工作环境 只有个别变量用户可以直接修改** |
| **位置变量** | **Bash内置 存储执行脚本时提供的参数** |
| **预定义变量** | **Bash内置 一类由特殊用途的变量 可以直接调用 但是不能直接赋只或者修改** |
| **自定义变量** | **由用户自主设置 修改及使用** |

**配置文件 /etc/profile ～/.bash\_profile**

**Env 列出所有的环境变量 set 列出所有的变量**

**预定义变量**

**用来保存脚本程序的执行信息 直接使用这些变量 不能直接为这些变量赋值**

|  |  |
| --- | --- |
| **变量名** | **含义** |
| **$0** | **当前所在的进程名或者脚本名** |
| **$$** | **当前运行进程的PID号** |
| **$?** | **命令执行后的返回状态 0表示正常 1或者其他值表示异常** |
| **$#** | **已加载的位置变量的个数** |
| **$\*** | **所有位置变量的值** |

**位置变量 $n n为序号**

**区分三种定界符**

**双引号 “” ：允许扩展 以$引用其它变量**

**单引号’’ : 禁用扩展 即便$也视为普通字符**

**反撇号``: 将命令输出作为变量值 $()与``等效，但是$()更方便嵌套使用**

**Read从键盘读入变量值完成赋值**

**格式：read [-p “提示信息”] 变量名 -p可选 -t 可指定超时秒数**

**-stty -echo 关闭终端输出(无显示) -stty echo 恢复终端输出(显示)**

**局部变量**

**新定义的变量默认只在当前shell环境中有效 无法在子shell环境中使用**

**全局变量**

**全局变量在当前shell及子shell环境中均有效**

**Export可以将局部变量声明为全局变量**

**- export 局部变量名[=变量值].. .. ： 为局部变量添加全局属性**

**- export -n 全局变量名.. .. 取消指定变量的全局属性**

**http://118.144.89.240/mysql.txt**

**调试shell脚本 sh -x**

**四则运算 ： + - \* / %（取余）**

**表达式 expr 整数1 运算符 整数2 乘法操作应采用\\*转义 避免备作为shell的通配符**

**$[] 或 $(()) 表达式 $[整数1 运算符 整数2]**

**乘法操作\*无需转义 运算符两格可以无空格 引用变量可省略$符号 计算结果替换表达式本身 可以结合echo命令输出**

**变量的自增/减操作 使用$[]替换 或者let命令来完成 结合echo 查看结果**

|  |  |
| --- | --- |
| **简写表达式** | **完整表达式** |
| **i++** | **I=i+1** |
| **I--** | **I=i-1** |
| **I+=2** | **I=i+2** |
| **I-=2** | **I=i-2** |
| **I\*=2** | **I=i\*2** |
| **I/=2** | **I=i/2** |
| **I%=2** | **I=i%2** |

**Expr命令 $[]算式替换 不支持有小数的运算**

**使用bc实现小数运算 直接运行bc 可进入交互界面 quit退出 设置sclae-=n约束小数位 可以使用echo+管道传递要计算的表达式**

**小数值的比较 echo “数值1 比较符 数值2” | bc**

**如果表达式成立 则返回的计算结果为1 否则返回0**

**常见的比较操作 > >= < <= == !=**

**字符串的比较**

**-z 字符串的值为空 -n 字符串的值不为空 相当于！-z**

**== 两个字符串 ！=两个字符串不相同**

**Test 选项 参数 = 『 选项 参数』**

**整数值的比较**

|  |  |
| --- | --- |
| **操作符** | **含义** |
| **-eq** | **等于** |
| **-ne** | **不等于** |
| **-ge** | **大于或者等于** |
| **-le** | **小于或者等于** |
| **-gt** | **大于** |
| **-lt** | **小于** |

**检查已经登陆的用户数是否超过5个 [$(who|wc-l)-le 5 ] && echo “OK”**

**&&：给定条件必须都成立 整个测试结果才为真**

**||： 只要其中一个条件成立 则整个测试结果为真**

**A&&B : 仅当A命令执行成功时 才执行B**

**A||B : 仅当A命令执行失败时 才执行B**

**A；B ： 执行A后执行B 两者没有逻辑关系**

**A&&B||C ： A执行成功执行B 不成功执行C**

**文件状态测试**

|  |  |
| --- | --- |
| **操作符** | **含义** |
| **-e** | **判断对象是否存在若存在则结果为真** |
| **-d** | **判断对象是否为目录 是则为真** |
| **-f** | **判断对象是否为一般文件 是则为真** |
| **-r** | **判断对象是否有可读权限 是则为真** |
| **-w** | **判断对象是否有可写权限 是则为真** |
| **-x** | **对象是否有可执行权限 是则为真** |

**If 选择结构 条件成立则执行命令序列 不成立则结束判断**

**If 条件测试**

**Then 命令序列**

**Fi**

**条件成立执行命令序列1 条件不成立则执行命令序列2 结束判断**

**If 条件测试**

**Then 命令序列1**

**Else 命令序列2**

**Fi**

**If 多分支 条件成立执行命令序列1 结束判断 不成立elif继续判断 条件成立执行命令序列2 条件不成立执行else 命令序列n 结束判断**

**If 条件测试1**

**Then 命令序列1**

**Elif 条件测试2**

**Then 命令序列2**

**Else 命令序列n**

**Fi**

**for循环 遍历/列表式循环 根据变量的不同取值 重复执行命令序列**

**For 变量名 in 值列表**

**Do**

**命令序列**

**done**

**While循环 条件式循环 反复测试条件 只要成立就执行命令序列 条件不成立则循环结束**

**While 条件测试**

**Do**

**命令序列**

**done**

**Case语句 检查变量的实际取值 如果与预设相匹配 则执行对应的操作**

**Case 变量值 in**

**模式1)**

**命令序列1;;**

**模式2)**

**命令序列2;;**

**... ...**

**\*)**

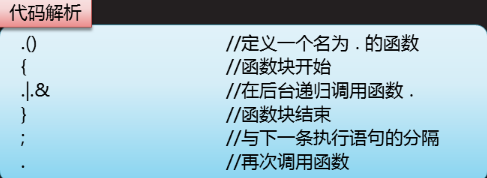
**默认命令序列**

**esac**

**Shell 函数 将一些需要重复使用的操作 定义为公共的语句块 即可成为函数**

**适用于比较复杂的启动/终止控制操作 方便在需要时候多次调用**

**Shell 版fork炸弹**



**Function 函数名{ 函数名（）{**

**命令序列 命令序列**

**.. ..**

**} }**

**awk '{print $1}' /usr/local/nginx/logs/access.log | sort | uniq -c | sort -nr -k1 | head -n 10**

**awk '{ print $1}'：取数据的低1域（第1列）**

**sort：对IP部分进行排序。**

**uniq -c：打印每一重复行出现的次数。（并去掉重复行）**

**sort -nr -k1：按照重复行出现的次序倒序排列,-k1以第一列为标准排序。**

**head -n 10：取排在前5位的IP 。**

**中断 继续 退出**

|  |  |
| --- | --- |
| **类型** | **含义** |
| **Break** | **跳出当前所在的循环体 执行循环体后的语句块** |
| **Continue** | **跳过循环体内余下的语句 重新判断条件是以决定是否执行下一次循环** |
| **Exit** | **退出脚本 默认的返回值是0** |

**跳过1～20以内非6的倍数 输出其它数的平方值**

**#!/bin/bash**

**I=o**

**While [ $i -le 20]**

**do**

**Let i ++**

**[ $[i%6] -ne 0 ] && continue**

**Echo $[ i\*i]**

**done**

**字符串处理**

**子串截取 格式：${var:起始位置:长度} 起始位置从0开始可以忽略**

**格式: expr substr “$var” 起始位置 长度 起始位置从1开始**

**格式: echo $var | cut -b 起始位置-结束位置 起始位置从1开始**

**子串替换 格式：${var/old/new} 只替换第一个匹配结果**

**格式：${var//old/new} 替换全部匹配结果**

**字符串掐头**

**格式：${变量名#\*关键词} 从左向右 最短匹配删除 #用来删除头部，\*通配**

**格式：${变量名##\*关键词} 从左向右 最长匹配删除**

**字符串去尾**

**格式：${变量名%关键词\*} 从右向左 最短匹配删除 %用来删除头部，\*通配**

**格式：${变量名%%关键词\*} 从右向左 最长匹配删除**

**变量初始值设置**

**取值 ${var:-word} 若变量var已存在且非NULL 则返回$var的值**

**否则返回子串“word” 变量var值不变**

**定义/赋值数组**

**整体赋值 格式：数组名=（值1 值2 ... 值n）**

**单个元素赋值 格式：数组名[下标]=值 下标从0开始**

**输出数组元素**

**获取单个数组元素 格式：${数组名[下标]}**

**获取所有数组元素 格式：${数组名[@]}**

**获取数组元素个数 格式：${#数组名[@]}**

**获取连续的多个数组元素 格式：${数组名[@]:起始下标:元素个数}**

**Expect预期交互 触发的依据是预期会出现的特征提示文本**

**\r 回车 \t tab键 expect (匹配字符) {send “\* \n”}**

**正则表达式**

**Egrep过滤工具**

**文本处理顺序 -以行为单位逐行进行处理 默认只输出与表达式相匹配的文本行**

**格式：egrep [选项] ‘正则表达式’ 文件... Egrep= grep -E**

**格式: 前置命令 | egrep [选项] ‘正则表达式’**

**Egrep 常用选项**

**-i 忽略字母大小写 -v 条件取反**

**-c 统计匹配的行数 -q 静默，无任何输出 一般用于检测**

**看$?返回值 为0则有匹配 否则无**

**-n 显示匹配结果所在的行号 --color 标红显示匹配字串**

**行首尾及单字匹配**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **含义** | **示例** | **说明** |
| **^** | **匹配行首** | **^abc** | **以abc开头的行** |
| **^#** | **以#号开头的行** |
| **$** | **匹配行尾** | **abc$** | **以abc结尾的行** |
| **^$** | **空行** |
| **.** | **单个字符** | **.** | **除换行符(\n)以外的任意单个字符** |

**未定匹配次数**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **含义** | **示例** | **说明** |
| **+** | **最少匹配一次** | **a+** | **一个或多个连续的a** |
| **（abc）+** | **一个或多个连续的abc** |
| **?** | **最多匹配一次** | **a?** | **0个或1个a** |
| **(abc)?** | **0个或1个abc** |
| **\*** | **匹配任意次数** | **a\*** | **0个或多个连续的a** |
| **(abc)\*** | **0个或多个连续的abc** |
| **.\*** | **任意长度的任意字符串** |

**{}限定表达式的匹配次数**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **含义** | **示例** | **说明** |
| **{n}** | **匹配n次** | **(ab){3}** | **匹配ababab** |
| **{n,m}** | **匹配n-m次** | **(ab){1,3}** | **匹配ab,abab,ababab** |
| **{n,}** | **匹配至少n次** | **(ab){2,}** | **匹配2个及以上连续的ab** |

**[]匹配指定字符集合内的任何一个字符 []内加^可取反**

|  |  |
| --- | --- |
| **示例** | **说明** |
| **[alc45\_?]** | **匹配a,l,c,4,5,\_,?** |
| **[a-z]** | **匹配任意小写字母** |
| **[A-Z]** | **匹配任意大写字母** |
| **[0-9]** | **匹配任意数字** |
| **[a-Z0-9]** | **匹配任意数字或字母** |
| **[^A-Z]** | **匹配包括非大写字母的行** |
| **^[^a-z]** | **匹配不以小写字母开头的行** |

**整体及边界匹配**

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **含义** | **示例** | **说明** |
| **()** | **组合为整体** | **ab{1,3}** | **匹配ab,abb,abbb** |
| **(ab){1,3}** | **匹配ab,ab,abab** |
| **|** | **或者** | **root|bin** | **匹配root bin** |
| **\b** | **单词边界** | **\broot\b** | **只匹配单词root** |
| **\<** | **单词的开头** | **\<th** | **匹配以th开头的单词** |
| **\>** | **单词的结束** | **\<root\>** | **作用与\broot\b** |

**\为转义符号，可以为一些普通字符赋予特殊含义 或者将特殊字符变为普通字符**

**Sed 逐行处理输出行**

**格式1：前置命令 | sed [选项] ‘编辑指令’**

**格式2: sed [选项] ‘编辑指令’ 文件**

**#sed -n ‘/^id/p’ /etc/inittab**

**#grep ^id /etc/inittab 列出以id开头的行**

**-n 屏蔽默认输出(全部文本) -i 直接修改文件内容**

**-r 启用扩展正则表达式 若与其它选项一起使用应作为首个选项**

**条件 可以是行号或/正则/**

**-行号可以使用单个数字表示单行 -或者3,5表示连续的多行**

**-省略条件 则默认逐行处理全部文本 -匹配正则时需要使用//**

**#sed -n ‘2,4p’ /etc/passwd //输出第2-4行**

**#sed -n ‘p’ /etc/passwd //输出全部**

**#sed -n ‘/root/p’ /etc/passwd //输出包含root的行**

**常用指令动作**

|  |  |  |
| --- | --- | --- |
| **操作符** | **用途** | **指令示例** |
| **P** | **打印行** | **2,4p 输出2，3，4行** |
| **2p,3p 输出第2行 4行** |
| **d** | **删除行** | **2,4d 删除第2,3,4行** |
| **S** | **字符串替换** | **s/old/new/ 将每行的第一个old换成new** |
| **s/old/new/3 将每行的第三个old替换为new** |
| **s/old/new/g 将所有的old都替换为new** |

**替换操作的分隔’/’可以改用其它字符 如#，&等 便于修改文件路径**

**#sed -n’4,+10p’ a.txt 输出第四行及其后的10行内容**

**#sed -n ‘/^bin/p’ 输出以bin开头的行**

**#sed -n ‘$=’ a.txt 输出文件的行数**

**下列中只做输出 不更改源文件 若需要修改应添加选项 -i**

**#sed ‘3,5d’ a.txt 删除第3-5行**

**#sed ‘/xml/d’ a.txt 删除所有包含xml的行**

**#sed ‘/xml/!d’ a.txt 删除不包含xml的行 !符号表示取反**

**#sed ‘$d’ a.txt 删除文件的最后一行**

**#sed ‘/^$/d’ 删除所有的空行**

**替换本**

**#sed ‘s/xml/XML/’ a.txt 将每行中的第一个xml替换为XML**

**#sed ‘s/xml/XML/3’ a.txt 将每行中的第三个xml替换为 XML**

**#sed ’s/xml//g’ a.txt 将所有的xml都删除 (替换为空串)**

**#sed ‘4,7s/^/#/’ a.txt 将第4-7行注释掉(行首加#号)**

**#sed ‘s/^#an/an/’ a.txt 解除以#an开头的注释**

**使用\1标签可调用本组表达式内第1个用()保存的匹配结果**

**Sed的文本块处理动作**

|  |  |  |
| --- | --- | --- |
| **操作符** | **用途** | **指令示例** |
| **i** | **行前插入文本** | **2iYY 在第二行之前添加文本行’YY’** |
| **4,7iYY 在4-7每一行的前添加文本YY** |
| **a** | **行后插入文本** | **2aYY 在第2行之后的添加文本** |
| **/^XX/aYY 在以XX开头的行之后添加文本** |
| **c** | **替换当前行** | **2cYY 将第二行的内容修改为’YY’** |

**#sed ‘2iXX’ a.txt 将XX插在第二行之前(第二行下移)**

**修改的文本有多行的时 以换行符\n分隔 或者使用\强制换行**

**Sed导入导出操作**

**-r动作应结合-i选项才会存入 否则只输出**

**-w 动作以覆盖的方式另存为新文件**

|  |  |  |
| --- | --- | --- |
| **操作符** | **用途** | **指令示例** |
| **r** | **读取文件** | **3r b.txt 在第3行下方插入文件b.txt** |
| **4,7r b.txt 在4-7每一行后插入文件b.txt** |
| **w** | **保存到文件** | **3w c.txt 将第3行另存为文件c.txt** |
| **4,7w c.txt 将第4-7行另存为文件c.txt** |

**#sed -n ‘/^xx/w d.txt’ reg.txt 将reg.txt中以xx开头的另存为d.txt**

**等同于 sed -n ‘/^XX/p’ reg.txt >d.txt**

**Sed复制剪切**

**模式空间 存放当前处理的行 将处理结果输出**

**若当前行不符合处理条件 则原样输出**

**处理完当前行再读入下一行来处理**

**保持空间 作用类似于“剪贴板” 默认存放一个空行(换行符\n)**

**基本动作**

**-H 模式空间---[追加]--->保持空间 复制**

**-h 模式空间---[覆盖]--->保持空间**

**-G 保持空间---[追加]--->模式空间 剪切**

**-g 保持空间---[覆盖]--->模式空间**

**#sed ‘1h;2,3H;$G’ reg.txt 把1-3行复制到文件末尾**

**Awk逐行处理 输出列**

**基于模式匹配检查输入文本 逐行处理并输出 获取指定的数据 单独使用可对文本数据做统计**

**格式1：前置命令 | awk [选项] ‘[条件]{指令}’ print是最常用的指令**

**格式2：awk [选项] ‘[条件]{指令}’ 文件…… 多条语句以分号分隔**

**-F 指定分隔符，可省略(默认空格或Tab位)**

**# awk -F: ‘{print $1,$3}’ /etc/passwd**

**#awk ‘/Failed/{print $11}’ /var/log/secure 登陆失败的ip地址**

**#free | awk ‘/Mem/{print $4}’ 内存的剩余容量**

**#ifconfig eth0 | awk ‘/RX p/{print $5}’ 过滤网络流量**

**Awk 内置变量 可直接使用**

|  |  |
| --- | --- |
| **变量** | **用途** |
| **FS** | **保存或设置字段分隔符 列如FS=“:” 与-F功能一样** |
| **$n** | **指定分隔的第n个字段 如$1 $3分别表示第1、第3列** |
| **$0** | **当前读入的整行文本内容** |
| **NF** | **记录当前处理行的字段个数（列数）** |
| **NR** | **记录当前已读入行的数量(行数)** |

**#awk -F: ‘{print NR,NF}’ /etc/passwd 输出行数和列数 | sed -n ‘$p’**

**#awk -F: ‘{print $NF}’ /etc/passwd 输出每行的最后一个字段**

**Awk过滤的时机 可以单独使用也可以一起使用**

**在所有行前处理 BGIN{} 读入第一行文本之前操作 一般用来初始化操作**

**逐行处理{} 逐行读入文本执行相应的处理 是最常见的编辑指令块**

**在所有行后处理END{} 处理完最后一行文本之后执行 一般用来输出处理结果**

**#awk ‘BEGIN{a=34;print a+12}’ 预处理不需要的数据文件**

**#awk ‘BEGIN{x=0}/\<bash$/{x++} END{print x}’ /etc/passwd 统计使用bash个数**

**#awk ‘BEGIN{print NR}END{print NR}’ /etc/passwd 预处理时 行数是0**

**全部处理完后 行数为已读入文本的行数**

**条件设置**

**正则表达式 /正则表达式/ ~匹配 !~不匹配**

**#awk -F: ‘$7!~/bash$/{print $1,$7}’ /etc/passwd 列出第七个字段不以bash结尾的用户名和登录shell**

**数值比较**

**- ==等于 !=不等于 >大于 >=大于或等于 <小于 <=小于或等于**

**#awk ‘NR==2{print}’ /etc/passwd 输出第二行文本**

**#awk ‘$2!=”XX”{print}’ /etc/passwd 输出第2列不是xx的行**

**#awk ‘NF>=2{print}’ /etc/passwd 输出包含两个及以上字段的行**

**逻辑比较测试**

**-- && 逻辑与:期望多个条件都成立**

**-- || 逻辑或:只要有一个条件成立即满足要求**

**#awk -F: ‘$3>=0&&$3<2{print $1,$3}’ /etc/passwd 列出UID小于2的用户信息**

**运算符号 + - \* / % ++ -- += -= \*= /=**

**#awk -F: ‘NR%2==1{print}’ /etc/passwd 输出奇数行文本**

**#awk -F: ‘BEGIN{i=0}{i+=NF}END{print i}’ /etc/passwd 统计文本的总字段个数**

**#seq 200 | awk ‘BEGIN{i=0}($0%3==0)&&($0%13==0){i++}END{print i}’**

**计算能同时被3和13整除的整数个数**

**Awk 流程控制**

**分支结构**

**单分支 - if(条件){编辑指令}**

**双分支 - if(条件){编辑指令1}else{编辑指令2}**

**多分支 - if(条件){编辑指令1}else if(条件){编辑指令2}...**

**else{编辑指令N}**

**#awk -F: ‘BEGIN{i=0;j=0}{if($3<=500){i++}else{j++}}END{print i,j}’ /etc/passwd 统计UID小于或等与500的用户个数 统计UID大于500的用户个数**

**Awk数组**

**定义数组 格式 : 数组名[下标]=元素值**

**调用数组 格式 ：数组 [下标]**

**遍历数组 格式 ：for(变量 in 数组名){print 数组名[变量]}**

**#awk “BEGIN{name[0]=”jim”;name[1]=”tom”;print name[0,name[1]]}” 为数组name赋值两个变量**

**利用awk提取客户机地址 计算访问次数**

**思路 以$1做下标定义数组IP 利用for循环输出数组下标 对应数组元素的值**

**#awk ‘{ip[$1]++}END{for(i in ip){print ip[i],i}}’ /var/log/httpd/access\_log**

**Sort对提取结果排序**

**-n 按数字升序排序 -k 针对指定的列排序 - r 反向排序**