回顾

case分支

case 变量名称 in

模式1)

执行指令;;

模式2)

执行指令;;

\*)

执行指令

esac

echo -e "\033[31mABCD\033[0m"

函数

函数名称(){

执行指令

}

循环的控制

exit 、 break 、 continue

字符串的处理

1. 字符串的截取 ${变量名称:}
2. 字符串的替换 ${变量名称/}
3. 字符串的删除

${变量名称#} 从左往右删除

a=abc 定义变量

echo ${a#a} 从左往右删除变量中的字符a，剩下的是bc

${变量名称%} 从右往左删除

a=abcdef 定义变量

echo ${a%c\*} 从右往左删除到c

编写可以批量修改文件扩展名的脚本

#!/bin/bash

for i in `ls \*.txt` 首先找到要修改的文件交给for循环

do

n=${i%.\*} 然后将扩展名去掉

mv $i $n.doc 将原文件改成doc结尾的

done

思考：如何修改脚本，能够方便的将任何扩展名的文件修改成其他扩展名

#!/bin/bash

for i in `ls \*.$1`

do

n=${i%.\*}

mv $i $n.$2

done

4，定义初值(备用值)

#!/bin/bash

read -p "请输入用户名" u

useradd $u

read -p "请输入密码" p

echo "${p:-123456}" | passwd --stdin $u 此处定义了变量p的初值，如果用户在输入密码时直接回车，那么变量p为空就会调用初值，如果变量p非空，则不会使用初值

${变量名:} 截取

${变量名/} 替换

${变量名#} 掐头，从左往右删除

${变量名%} 去尾，从右往左删除

${变量名:-} 定义初值

$( ) 获取命令执行的结果，或使用``反撇号

$[ ] 运算

练习二：正则表达式

正则表达式，由一些符号构成，可以配合某些工具实现对文本文档的过滤筛选等处理工作

head -5 /etc/passwd > user 创建素材

grep bin user 找有bin的行

grep ^bin user 找以bin开头的行

grep n user 找有n的行

grep n$ user 找以n结尾的行

grep "[bin]" user 找b或者i或者n

grep "[^bin]" user 取反，找b或者i或者n之外的字符

grep "[a-z]" user 找所有小写字母

grep "[A-Z]" user 找所有大写字母

grep "[a-Z]" user 找所有字母

grep "[^a-Z]" user 找所有字母之外的字符

grep "[0-9]" user 找所有数字

grep "roo." user 找roo开头后面有任意一个字符的字串

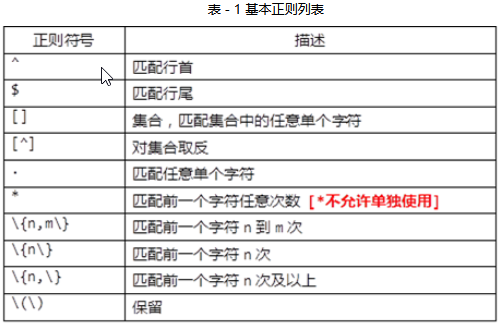
grep "ro\*t" user 找rt，中间可以有任意个o

grep ".\*" user 找任意

grep "ro\{2,3\}t" user 找rt，中间可以有2~3个o

grep "ro\{2,\}t" user 找rt，中间可以出现2次以及2次以上的o

grep "ro\{2\}t" user 找rt，中间必须是2个o



grep "ro\{2,3\}t" user 基本正则

grep -E "ro{2,3}t" user 扩展正则，搜索效果同上

egrep "ro{2,3}t" user 扩展正则，搜索效果同上

egrep "ro{2,}t" user 找rt，中间可以出现2次以及2次以上的o

egrep "ro{2}t" user 找rt，中间必须是2个o

egrep "ro{1,}t" user 找rt，中间可以出现1次以及1次以上的o

egrep "ro+t" user 效果同上

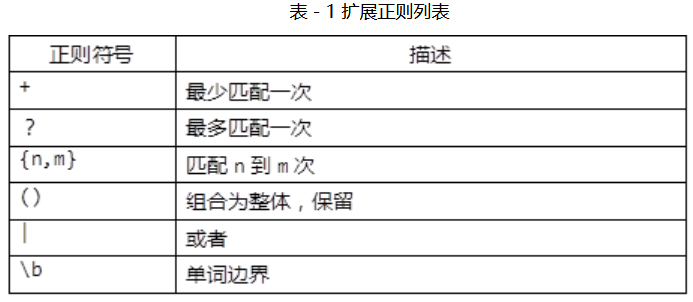
egrep "do{0,1}a" user 找da，中可以出现0~1个o

egrep "do?a" user 效果同上

egrep "^root|^bin" user 找root开头或者bin开头的行

egrep "^(root|bin)" user 效果同上

egrep "\bthe\b" abc 搜索the，两边不许出现数字，字母，下划线



练习三：

sed 流式编辑器 ，对文档非交互式增删改查，逐行处理

使用方法：

1. 前置指令 | sed 选项 定址符 指令
2. sed 选项 定址符 指令 被处理的文档

选项 -n 屏蔽默认输出 -r 支持扩展正则 -i 写入文件

指令 p 输出 d 删除 s替换

sed -n 'p' user 查看所有行

sed -n '3p' user 查看第3行

sed -n '4p' user 查看第4行

sed -n '1,3p' user 查看1~3行

sed -n '4,5p' user 查看4~5行

sed -n '2p;4p' user 查看第2行与第4行

sed -n '2,+1p' user 查看第2行以及后面1行

sed -n '1~2p' user 查看奇数行

sed -n '2~2p' user 查看偶数行

sed -n '=' user 查看行号

sed -n '$=' user 查看最后一行的行号

sed -n '$p' user 查看最后一行

sed -n '/^root/p' user 使用正则表达式输出以root开头的行

sed -n '/^root/!p' user 取反

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 '/^install/d' a.txt 删除以install开头的行

sed '$d' a.txt 删除文件的最后一行

sed '/^$/d' a.txt 删除空行

sed 's/2017/xxxx/' test 将所有行的第1个2017替换成x

sed 's/2017/xxxx/2' test 将所有行的第2个2017替换成x

sed 's/2017/xxxx/g' test 将所有行的所有2017替换成x

sed '1s/2017/xxxx/' test 将第1行的第1个2017替换成x

sed '2,3s/2017/xxxx/' test 将2~3行的第1个2017替换成x

sed '/2024/s/2017/xxxx/' test 将有2024的行的第1个2017替换成x

sed '1s/2017/xxxx/;3s/2017/xxxx/' test 分辨替换第1行与第3行

/bin/bash 改成 /sbin/sh

sed '1s//bin/bash//sbin/sh/' user 常规方式，替换符号是斜杠，被替换内容也有斜杠，发生冲突报错

sed '1s/\/bin\/bash/\/sbin\/sh/' user 将被替换内容的斜杠用转义符号区分，可以成功替换，但不容易识别

sed '1s#/bin/bash#/sbin/sh#' user 更改s的替换符为#，最佳方案