shell day 05

================================================

**sed中的替换，使用 s/旧内容/新内容/**

vim test.txt //准备素材,写入下列内容

2017 2011 2018

2017 2017 2024

2017 2017 2017

sed 's/2017/6666/' test.txt //替换所有行第1个2017

sed '1s/2017/6666/' test.txt //替换第1行第1个2017

sed '2s/2017/6666/' test.txt //替换第2行第1个2017

sed '2s/2017/6666/2' test.txt //替换第2行第2个2017

sed 's/2017/6666/g' test.txt //替换所有行所有2017

sed '/2024$/s/2017/6666/' test.txt //替换以2024结尾的行的

第1个2017

**如何把 /bin/bash 替换成 /sbin/sh ？**

sed 's//bin/bash//sbin/sh/' user //直接替换，报错

sed 's/\/bin\/bash/\/sbin\/sh/' user //使用转义符号，可以成功，但读

起来难识别

sed 's#/bin/bash#/sbin/sh#' user //更换替换符号，最佳方案

------------------------------------------------------------------------

**sed中替换的高级应用**

vim nssw //创建素材，写入下列两行文本

Hello the world

ni hao ma beijing

**1）删除文档中每行的第2个、最后1个字符**

sed 's/.//2' nssw //替换所有行的第2个任意字符为空

sed 's/.$//' nssw //替换所有行的最后1个任意字符为空

sed 's/.//2;s/.$//' nssw //连在一起写中间加分号隔开

**2）给文档中所有的大写字母加个括号**

( ) 保留，相当于复制 \1 代表粘贴

sed -r 's/([A-Z])/(\1)/g' nssw //找到文档中所有行的所有

大写字母，每找到一个就保留(复制)，然后替换的时候粘贴

该字母并套上个括号

**3）将文本中所有数字删除**

sed 's/[0-9]//g' nssw //使用正则表达式找到任意数字，然后

都替换为空

4）**将文档中每行的第1个字符和最后1个字符互换**

echo abc > a //准备素材

sed -r 's/(a)(b)(c)/\1\2\3/' a //替换时多用几个括号，按顺序粘贴内容不变

sed -r 's/(a)(b)(c)/\3\2\1/' a //粘贴时调换顺序可以实现首尾字符互换

echo xyz >> a //追加一行xyz

sed -r 's/(a)(b)(c)/\3\2\1/' a //这样替换就不灵了，因为只匹配abc的行

sed -r 's/(.)(.)(.)/\3\2\1/' a //换成点，可以匹配任意三个字符的行

echo abcd >> a //再追加abcd

sed -r 's/(.)(.)(.)/\3\2\1/' a //这样替换就不灵了，因为匹配任意三个字符的行

sed -r 's/^(.)(.\*)(.)$/\3\2\1/' a //中间加.\*代表匹配任意即可

sed -r 's/^(.)(.\*)(.)$/\3\2\1/' nssw //可以匹配任意文档实现首位互换

**编写脚本，安装httpd，启动时使用82号端口**

#!/bin/bash

yum -y install httpd

sed -i '/Listen 80/s/0/2/' /etc/httpd/conf/httpd.conf //找到有Listen 80的行，将0换成2

systemctl restart httpd

getenforce 0 //运行脚本前要关闭selinux

systemctl stop firewalld //关闭防火墙

使用火狐浏览器访问192.168.4.7:82

p d s

a行下追加 i行上添加 c替换整行

sed 'a 666' user //所有行的下面追加666

sed '1a 666' user //第1行的下面追加666

sed '/root/a 666' user //有root的行的下面追加666

sed 'i 666' user //所有行的上面添加666

sed '1i 666' user //第1行的上面添加666

sed '2i 666' user //第2行的上面添加666

sed '/root/i 666' user //有root的行的上面添加666

sed 'c 666' user //替换所有行为666

sed '1c 666' user //替换第1行为666

sed '/^bin/c 666' user //替换以bin开头的行为666

找到使用bash的用户，然后按照"用户名 --> 密码"的格式存储在

一个文件中。

#!/bin/bash

u=$(sed -n '/bash$/s/:.\*//p' /etc/passwd) //找到使用bash的用户名

for i in $u //把上一步找到的用户放入循环

do

n=$(grep $i: /etc/shadow) //将每个人的密码信息找出

n2=${n#\*:} //掐头

p=${n2%%:\*} //去尾，得到密码

echo "$i --> $p" //按 "用户-->密码" 格式喊出(用追加重定向可以

输出到文件中)

done

-n -r -i

p d s a i c

grep 易于使用，搜索

sed 非交互式增删改查

awk 精确搜索 逐行处理

使用方法：

1. 前置指令 | awk 选项 条件 指令

2，awk 选项 条件 指令 被处理文件

选项 -F 定义分隔符

指令 print

内置变量 $1第一列 $2第二列 $3 …. $0所有列

NR 行号 NF列号

awk '{print}' test.txt //输出所有行的内容

awk '/beijing/{print}' test.txt //输出有beijing的行的内容

awk '/beijing$/{print}' test.txt //输出以beijing结尾的行的内容

awk '{print $1}' test.txt //输出所有行的第1列

awk '{print $3}' test.txt //输出所有行的第3列

awk '/the/{print $3}' test.txt //输出有the的行的第3列

awk '/the/{print NR}' test.txt //输出有the的行的行号

awk '{print NR}' test.txt //输出所有行的行号

awk '{print NF}' test.txt //输出所有行的列号

awk '{print $0,NR}' test.txt //输出所有列然后空格输出行号

awk -F: '{print $1}' user //使用-F:修改分隔符为冒号，输出第1列

awk -F: '{print $3}' user

awk -F: '{print $1" 的解释器是 "$7}' user //输出常量时使用双引号

**利用awk提取本机的网络流量信息**

ifconfig eth0 | awk '/RX p/{print "eth0网卡的接收流量是"$5"字节"}'

ifconfig eth0 | awk '/TX p/{print "eth0网卡的发送流量是"$5"字节"}'

**利用awk提取本机的根分区剩余容量**

df -h | awk '/\/$/{print "根分区剩余容量是"$4}'

下课休息 17:25回