## 文本处理工具

#### 1、grep工具

grep是行过滤工具，用于根据关键字对行进行过滤

语法和选项

语法：

# grep [选项] ‘关键字’ 文件名

常见选项：

OPTIONS

-i：不区分大小写

-v：查找不包含指定内容的行

-w：按单词进行搜索

-o：打印匹配关键字

-c： 统计匹配到的次数

-n：显示行号

-r：逐层遍历目录查找

-A：显示匹配行后面多少行

-B：显示匹配行前面多少行

-C：显示匹配行前后多少行

-l：只列出匹配的文件名

-L：列出不匹配的文件名

-e：使用正则表达式

-E：使用扩展正则表达式

^key: 以关键字开抬头

Key$: 以关键字结尾的

^$: 空白行

--color=auto： 找到关键字部分加上颜色显示

#### 2、cut工具

cut是列截取工具，用于列的截取

语法和选项：

语法：

cut 选项 文件名

常见选项：

-c 以字符为单位进行截取

-d 自定义分割符号，默认是制表符号-t

-f 与-d一起使用，指定截取那个区域

举例说明：

cut -d: -f1,7 /etc/passwd | head 截取第一行和第七行

cut -c5-11 /etc/passwd 截取第5个字符到第11个字符

cut -c5- /etc/passwd 截取第5个字符到最后一个字符

课堂联系：

用小工具列出当前系统的运行级别

runlevel | cut -d" " -f2

5

#### 3、sort工具

sort是用于排序，它将文件的每一行作为一个单位，从首字符向后依次按照ASCII码进行比较，最后将他们按照升序输出

语法选项：

-u 去除重复行

-r 降序排列，默认为升序排列

-o 将排序结果输出到文件中类似于重定向符号>

-n 以数值排序默认是按照字符排序

-t 分隔符

-k 第N列

-b 忽略前导空格

-R 随机排序，每次运行的结果均不同

举例说明：

sort -n -t: -k3 /etc/passwd | head 按照第3行UID数值由小到大顺序排序

#### 4、uniq工具(去重复)

uniq用于去除连续的重复行

常见选项：

-i 忽略大小写

-c 统计重复行的次数

-d 只显示重复行

举例说明：

uniq 1.txt

uniq -c 1.txt

uniq -cd 1.txt

uniq -icd 1.txt

#### 5、tee工具

从标准输入读取并写入到标准输出和文件，即双向覆盖重定向（屏幕输出|文本输入）

选项：

-a 双向追加重定向

echo hello word | tee 1.txt

echo hello python | tee -a 1.txt

#### **6、diff工具**

diff 工具用于逐行比较文件不同

注意：diff描述两个文件不同的方式是告诉我们怎样改变第一个文件之后与 第二个文件匹配

语法和选项

语法：

diff [选项] 文件1 文件2

选项：

|  |  |  |
| --- | --- | --- |
| 选项 | 含义 | 备注 |
| -b | 不检查空格 |  |
| -B | 不检查空白行 |  |
| -i | 不检查大小写 |  |
| -w | 忽略所有空格 |  |
| --normal | 正常格式显示（默认） |  |
| -c | 上下文格式显示 |  |
| -u | 合并格式显示 |  |
| -q | 比较目录的不同 |  |

举例说明：

$ diff -u 1.txt 2.txt

--- 1.txt 2021-01-06 19:24:32.068908400 +0800

+++ 2.txt 2021-01-06 19:24:56.465182264 +0800

@@ -1,9 +1,8 @@

-hello word 删除掉本行

hello python

-cndsncdsncds 删除掉本行

+cncdsncds 添加本行

aaaa

aaaaa

-bbbbb 删掉本行

+bbbb 添加本行

cccc

dddd

eeeee

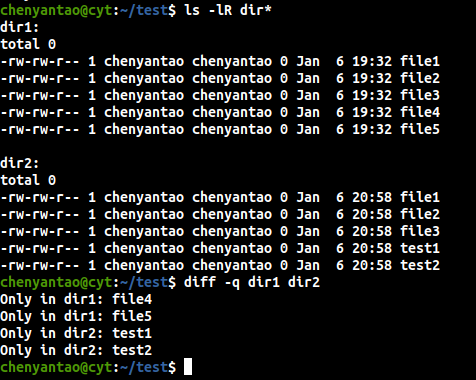
按照如上步骤执行后就和2.txt一致了

$ diff -u 1.txt 2.txt

$

没有返回说明两个文件一致了

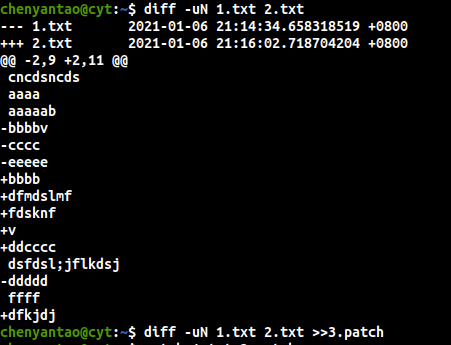
目录比较：



其他技巧：

有时候我们需要以一个文件为标准，去修改其他文件，并且修改的地方较多时候，我们可以通过打补丁的方式完成，实现在线升级，不中断服务的情况。

1 先找出两个文件的不同，然后输出到一个文件



2 将不同的内容打补丁到文件

patch 1.txt 3.patch



3 测试验证两个文件是否相同

diff -uN 1.txt 2.txt



#### **7、paste工具**

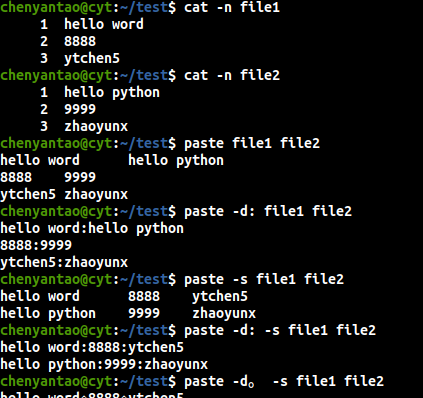
paste工具用于合并文件行

常用选项：

-d 自定义间隔符，默认是tab

-s 串行处理，非并行

举例说明



#### **8、tr工具**

Tr用于字符的转换，替换和删除；主要用于删除文件中控制字符和进行字符转换

语法和选项：

语法：

用法1 命令执行结果交给tr处理，其中string1用于查询，string2用于转换处理

# commands | tr 'string1' 'string2'

用法2 tr处理的内容来自文件记住要使用 < 标准输入

# tr 'string1' 'string2' < filename

用法3 匹配string1 进行相应操作 如删除操作

# tr options 'string1' < filename

选项：

-d 删除字符窜中所有输入字符

-s 删除所有重复出现的字符序列，只保留第一个，既将重复出现的字符串压缩成一个字符串。

常匹配字符串：

|  |  |  |
| --- | --- | --- |
| 字符串 | 含义 | 备注 |
| a-z或 [:lower:] | 匹配所有小写字母 | [a-zA-Z0-9] |
| A-Z或[:upper:] | 匹配所有大写字母 |  |
| 0-9或[:digit:] | 匹配所有数字 |  |
| [:alnum:] | 匹配所有字母和数字 |  |
| [:alpha:] | 匹配所有字母 |  |
| [:blank:] | 所有水平空白 |  |
| [:punct:] | 匹配所有标点符号 |  |
| [:space:] | 所有水平或垂直的空格 |  |
| [:cntrl:] | 所有控制字符 | \f Ctrl+l 走行换页  \n Ctrl+J 换行  \r Ctrl+M 回车  \t Ctrl+l tab键 |
|  |  |  |

举例子说明:

# 将test.txt 文件内部的所有小写字母替换成大写字母

$ tr '[:lower:]' '[:upper:]' <test.txt

ROOT:X:0:0:ROOT:/ROOT:/BIN/BASH

DAEMON:X:1:1:DAEMON:/USR/SBIN:/USR/SBIN/NOLOGIN

BIN:X:2:2:BIN:/BIN:/USR/SBIN/NOLOGIN

SYS:X:3:3:SYS:/DEV:/USR/SBIN/NOLOGIN

SYNC:X:4:65534:SYNC:/BIN:/BIN/SYNC

GAMES:X:5:60:GAMES:/USR/GAMES:/USR/SBIN/NOLOGIN

MAN:X:6:12:MAN:/VAR/CACHE/MAN:/USR/SBIN/NOLOGIN

LP:X:7:7:LP:/VAR/SPOOL/LPD:/USR/SBIN/NOLOGIN

MAIL:X:8:8:MAIL:/VAR/MAIL:/USR/SBIN/NOLOGIN

NEWS:X:9:9:NEWS:/VAR/SPOOL/NEWS:/USR/SBIN/NOLOGIN

# 将这个文件内部的：和/线都替换成#

$ tr '[:punct:]' '#' <test.txt

root#x#0#0#root##root##bin#bash

daemon#x#1#1#daemon##usr#sbin##usr#sbin#nologin

bin#x#2#2#bin##bin##usr#sbin#nologin

sys#x#3#3#sys##dev##usr#sbin#nologin

sync#x#4#65534#sync##bin##bin#sync

games#x#5#60#games##usr#games##usr#sbin#nologin

man#x#6#12#man##var#cache#man##usr#sbin#nologin

lp#x#7#7#lp##var#spool#lpd##usr#sbin#nologin

mail#x#8#8#mail##var#mail##usr#sbin#nologin

news#x#9#9#news##var#spool#news##usr#sbin#nologin

也可以使用：

# tr ':/' '#' <test.txt

# 删除这个文件内部的所有的字母

$ tr -d 'a-z' <test.txt

::0:0::/://

::1:1:://:///

::2:2::/:///

::3:3::/:///

::4:65534::/://

::5:60:://:///

::6:12::///:///

::7:7::///:///

::8:8:://:///

::9:9::///:///

# 删除这个文件中除了字母以外的所有字符

$ tr -d '[0-9][:punct:]' <test.txt

rootxrootrootbinbash

daemonxdaemonusrsbinusrsbinnologin

binxbinbinusrsbinnologin

sysxsysdevusrsbinnologin

syncxsyncbinbinsync

gamesxgamesusrgamesusrsbinnologin

manxmanvarcachemanusrsbinnologin

lpxlpvarspoollpdusrsbinnologin

mailxmailvarmailusrsbinnologin

Newsxnewsvarspoolnewsusrsbinnologin

#### **9、常用案例**

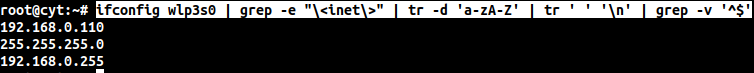
1. 提取ip地址

ifconfig wlp3s0 | grep -e "\<inet\>" | awk -F"[ \t]+" '{print $3}'

ifconfig wlp3s0 | grep -e "\<inet\>" | tr -s ' ' | cut -d" " -f3

1. 一次性截取ip地址，掩码，广播地址

ifconfig wlp3s0 | grep -e "\<inet\>" | tr -d 'a-zA-Z' | tr ' ' '\n' | grep -v '^$'



ifconfig wlp3s0 | grep -e "\<inet\>" | tr -s ' ' | cut -d" " -f3,5,7 | tr ' ' '\n'

1. 截取mac地址

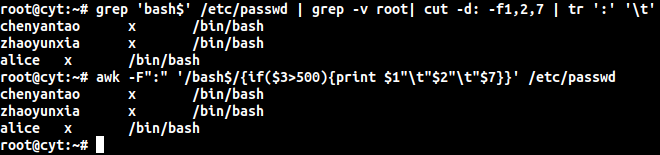
ifconfig wlp3s0 | grep ether | tr -s ' ' | cut -d" " -f3



4 将系统中所有普通用户的用户名密码默认shell保存到一个文件中，要求用户名密码默认shell之间使用tab分割

grep 'bash$' /etc/passwd | grep -v root| cut -d: -f1,2,7 | tr ':' '\t'

awk -F":" '/bash$/{if($3>500){print $1"\t"$2"\t"$7}}' /etc/passwd



## BASH特性

### **1、命令和文件自动补全**

Tab只能补全命令和文件（RHEL6/Centos6）

### 常见的快捷键

^c 终止前台运行的程序

^z 将前台运行的程序挂到后台

^d 退出等价于exit

^l 清屏

^a 光标移动到命令行最前端

^e 光标移动到命令行最后端

^u 删除光标前所有字符

^k 删除光标后所有字符

^r 搜索历史命令

### 常用的通配符（重点）

\* 匹配0个或者多个字符

？ 匹配任意单个字符

[list] 匹配list中任意单个字符或者一组字符 例如[a-z]

[!list] 匹配除了list中任意单个字符 取反

{string1，string2，..} 匹配string1，string2 或更多字符串

### Bash中的引号（重点）

* 双引号 ”” 会把引号的内容当成整体看待，允许通过$符号引用其他变量值
* 单引号 ’’ 会把引号内的内容当做整体看待，禁止引用其他变量，shell中特殊符号都被视为普通字符
* 反撇号 `` 反撇号和$()一样引号或者括号里面的内容会被优先执行，如果存在嵌套反撇号不能用

$ echo "$HOSTNAME"

cyt

$ echo '$HOSTNAME'

$HOSTNAME

举例：不支持嵌套

$ echo `echo "`date +%F`"`

bash: command substitution: 行 1: 寻找匹配的“"”时遇到了未预期的文件结束符

bash: command substitution: 行 2: 语法错误: 未预期的文件结尾

bash: command substitution: 行 1: 寻找匹配的“"”时遇到了未预期的文件结束符

bash: command substitution: 行 2: 语法错误: 未预期的文件结尾

date +%F

$ echo $(echo "`date +%F`")

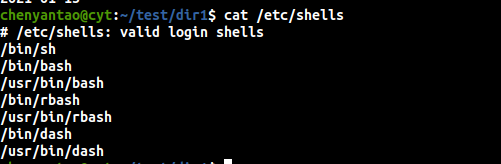
2021-01-13

## shell编程模块课程目标

1. shell的基本语法结构
2. 基本正则表达式的运用
3. 文件处理3剑客 grep sed awk工具的使用
4. 使用shell脚本完成一些复杂的任务，如服务搭建，批量处理等

说明：以上内容仅仅是基本要求，还有很多更深更难的语法需要扩充学习

### shell介绍



思考： 终端和shell有什么关系

#### 1、Shell脚本

1. 什么是shell脚本

一句话概括

简单来说就是将执行的命令保存在文本中，按照顺序执行，它是介绍型的意味不需要编译

准确描述

若干命令+脚本的基本格式+脚本特定语法+思想=shell脚本

1. 什么时候用shell脚本

重复化：复杂的工作，通过把工作的命令写成脚本，以后仅仅需要执行脚本就能完成这些工作

1. Shell脚本能干啥？
2. 自动化部署 LAMP/LNMP/Tomcat
3. 自动化管理 系统初始化脚本，批量更改主机密码，推送公钥
4. 自动化分析处理 统计网站访问量
5. 自动化备份 数据库备份 日志转储
6. 自动化监控脚本
7. 如何学习shell脚本

1 尽量记忆更多的命令

2 掌握脚本的标准格式（指定魔法字节，使用标准的运行的方式运行脚本）

3 必须熟练掌握脚本的基本语法（重点）

1. 学习shell脚本的秘诀
2. Shell脚本的基本写法

1 脚本第一行 魔法字符#!指定解释器 必须写

注意：

如果直接将解释器路径写死在脚本里面，可能在某些系统，就会存在找不到解释器的兼容性问题，所以可以使用#!/bin/env 解释器

2 脚本第二部分 注释 （#）说明，对脚本的基本信息进行描述【可选】

#!/bin/env bash

# 以下内容是脚本的基本信息的描述

# name：名字

# desc：描述describe

# Path：存放路径

# Usage：用法

# Update：更新时间

3 脚本第三部分 脚本要实现的具体代码内容

1. Shell脚本的执行方法

1 标准脚本执行方法（推荐）

注意：需要脚本有执行权限

绝对路径执行方法

$ /home/chenyantao/scripts/first\_shell.sh

hello word

hello word

hello word

hello word

相对路径执行方法

$ cd /home/chenyantao/scripts/

$ ./first\_shell.sh

hello word

hello word

hello word

hello word

2 非标准脚本执行方法

注意：基本可以没有执行权限

$ ll first\_shell.sh

-rw-rw-r-- 1 chenyantao chenyantao 302 1月 19 20:05 first\_shell.sh

$ bash first\_shell.sh # 直接使用bash 执行脚本

$ bash -vx first\_shell.sh # -vx 查看脚本的执行过程

$ bash -n first\_shell.sh # -n 检查脚本的语法是否有错误

#### 2、小试牛刀1

要求：

1 删除/tmp/目录下面的所有文件

2 然后在tmp目录下面创建三个目录，分别试dir1-dir3

3 拷贝/etc/hosts文件到刚创建的dir1目录里

4 最后打印”报告首长，任务已于当前系统时间完成”内容

#!/bin/env bash

# 以下内容是脚本的基本信息的描述

# name: two-shell.sh

# desc：two shell by ytchen5

# Path：/home/chenyantao/scripts

# Usage: /home/chenyantao/scripts/two-shell.sh

# Update：2021-01-19-20:36

#1 删除/tmp 目录下面的所有文件

echo "删除/tmp目录下面的的所有文件"

#2 然后在tmp目录下面创建三个目录，分别试dir1-dir3

mkdir -p /tmp/dir{1..3}

#3 拷贝/etc/hosts文件到刚创建的dir1目录里

cp /etc/hosts /tmp/dir1

#4 最后打印”报告首长，任务已于当前系统时间完成”内容

echo "报告首长，任务已于$(date +%F-%T)完成"

## 变量的定义

### 1 变量是什么

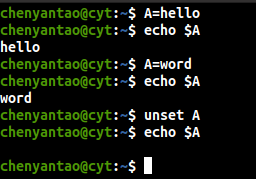
一句话概括：变量就是用来临时保存数据的，该数据是可变化的数据

### 2 什么时候需要定义变量

* 如果某个内容需要多次使用，并且在代码中重复出现，那么可以用变量代表该内容，这样在修改内容的时候仅仅需要修改变量的值即可
* 在代码运行过程中，可能会把某些命令的执行结果保存起来，后续代码如果需要使用这些结果，可以直接使用这个变量

### 3 变量如何定义

变量名=变量值



### **4 变量的定义规则**

虽然可以给变量（变量名）赋予任何值；但是对于变量名也是有要求的！

1. 变量区分大小写
2. 变量名不能有特殊符号
3. 变量名不能以数字开头
4. 等号两边不能有任何空格
5. 变量名要做到见名知意思

注意： 一般变量名字一般使用大写字母

### **5 变量的定义方式**

1. 基本方式

$ A=1234567

$ echo $A

1234567

$ echo ${A:2:3} # 变量的切片功能

345

1. 命令执行结果赋值给变量

$ A=`hostname`

$ echo $A

cyt

目的：让用户自己给变量赋值，比较灵活

语法：read 【选项】 变量名

常见选项：

|  |  |
| --- | --- |
| 选项 | 释义 |
| -p | 定义提示用户的信息 |
| -n | 定义字符数（限制变量值的长度） |
| -s | 不显示（不显示用户输入的内容） |
| -t | 定义超时时间，默认单位为秒（限制用户输入变量值的超时时间） |

举例说明：

1 用户自己定义变量值

$ read name

harry

$ echo $name

harry

$ read -p "Input you name: " name

Input you name: harry

$ echo $name

harry

$ read -s -p "Input your password: " pass

Input your password: 这里不显示输入内容

$ echo $pass

123456

2 变量值来自于一个文件

$ cat 1.txt

1.1.1.100

$ read -p "Input your ip address: " ipadd < 1.txt

$ echo $ipadd

1.1.1.100

$

$ cat 1.txt

192.168.0.100 255.255.255.0

$ read IP MASK < 1.txt

$ echo $IP

192.168.0.100

$ echo $MASK

255.255.255.0

$

[root@vm1 shell]# cat ip.txt

192.168.0.1 255.255.255.0 192.168.0.255

[root@vm1 shell]# read IP MASK BROADCAST < ip.txt

[root@vm1 shell]#

[root@vm1 shell]#

[root@vm1 shell]# echo $IP

192.168.0.1

[root@vm1 shell]# echo $MASK

255.255.255.0

[root@vm1 shell]# echo $BROADCAST

192.168.0.255

[root@vm1 shell]#

定义有类型的变量（declare）

目的：给变量做一些限制，固定变量的类型 比如：整型 只读

用法：declare 选项 变量名=变量值

常用选项：

|  |  |  |
| --- | --- | --- |
| 选项 | 释义 | 举例 |
| -i | 将变量看成整数 | declare -i A=123 |
| -r | 定义只读变量 | declare -r B=hello |
| -a | 定义普通数值，查看普通数组 |  |
| -A | 定义关联数组，查看关联数组 |  |
| -X | 将变量通过环境导出 | declare -X AAA=123456  等同于  export AAA=123456 |
|  |  |  |

举例说明：

$ AAAA=itcast

$ env | grep AAAA

$

$ export AAAA

$ env | grep AAAA

AAAA=itcast

$ declare -x BBBB=heima

$ env | grep BBBB

BBBB=heima

### **6 变量的分类**