shell day 04

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

**使用循环嵌套，得到100个8位长度的随机字符存在pass.txt文件中**

#!/bin/bash

x=abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

for j in {1..100}

do

for i in {1..8}

do

n=$[RANDOM%62]

a=${x:n:1}

pass=$pass$a

done

echo "$pass" >> pass.txt

pass=

done

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

**2,字符串的替换**

**${变量名/被替换内容/新内容}**

a=1234 //定义素材

echo ${a/3/6} //把3换成6

a=11223344 //定义素材

echo ${a/3/6} //把3换成6,默认只换一个

echo ${a//3/6} //使用两个/可以实现把所有3换成6

echo ${a/3/} //把3换成空，相当于删除

echo $a //注意，变量只要不重新赋值，原始内容不会变

a=666888 //定义素材

echo ${a/6/9} //替换1个6

echo ${a//6/9} //替换所有6

echo ${a/66/99} //替换2个6

a=${a/6/9} //替换的同时对变量重新赋值

echo $a //变量内容发生改变

**3，字符串的删除**

**${变量名#被删除内容} 掐头 从左往右删除**

**${变量名%被删除内容} 去尾 从右往左删除**

a=abcdefg //定义素材

echo ${a#abc} //从左往右删除到c

echo ${a#abcd} //从左往右删除到d

echo ${a#abcdef} //从左往右删除到f

echo ${a#\*f} //效果同上，更精简

echo ${a%efg} //从右往左删除到e

echo ${a%defg} //从右往左删除到d

echo ${a%bcdefg} //从右往左删除到b

echo ${a%b\*} //效果同上，更精简

a=abcabc

echo ${a#ab} //从左往右删除到b

echo ${a#abcab} //从左往右删除到b,比上述方式多删除一些

echo ${a#\*b} //从左往右删除到b，只删除到第一个b

echo ${a##\*b} //从左往右删除到b，删除到最后一个b

echo ${a%bc} //从右往左删除到b

echo ${a%b\*} //从右往左删除到b，只删除到第一个b

echo ${a%%b\*} //从右往左删除到b，删除到最后一个b

**编写脚本，可以实现批量修改文件扩展名**

touch abc{01..10}.txt //创建10个文件作为素材

#!/bin/bash

for i in $(ls \*.txt) //找到所有txt文件交给for循环

do

n=${i%.txt} //先用去尾删除扩展名,n存储了纯粹的文件名

mv $i $n.doc //再将原文件修改为doc扩展名

done

**改良版：**

#!/bin/bash

for i in $(ls \*.$1) //使用位置变量$1决定要改的文件扩展名

do

n=${i%.$1}

mv $i $n.$2 //使用$2决定要修改成什么扩展名

done

**变量初值（备用值）的定义**

${变量名:-初值} //如果变量有值则使用本身的值，如果变

量为空，则使用初值

编写脚本，可以创建用户与配置密码，密码可以自定义，也

可以使用默认的123456

#!/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 $n //当用户没有输入密码时

密码就是123456

**总结**

**${变量名:} 截取**

**${变量名/} 替换**

**${变量名#} 掐头**

**${变量名%} 去尾**

**${变量名:-} 定义变量初值(备用值)**

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

**正则表达式，使用若干符号配合某工具对文档实现过滤、查找**

**、修改等功能**

head -5 /etc/passwd > user //准备素材

grep bin user //找有bin的行

grep ^bin user //找以bin开头的行

grep bash user //找有bash的行

grep bash$ user //找以bash结尾的行

vim user //编辑文档添加空行

grep -n ^$ user //找空行，加n选项可以显示行号

grep "root" user //找root

grep "[root]" user //找rot任意一个字符

grep "[rot]" user //效果同上

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 "[0-9]" user //找所有数字

grep "[^0-9]" user //找除了数字的内容，集合中的^代表取反

grep "[^a-Z0-9]" user //找特殊符号

grep "[a-Z0-9]" user //找所有字母和数字

grep "r..t" user //找rt之间有2个任意字符的行

grep "r...t" user //找rt之间有3个任意字符的行，没有匹配内容，就无输出

grep "\*" user //错误用法，\*号是匹配前一个字符任意次，不能单独使用

grep "ro\*t" user //找rt，中间的o有没有都行，有几次都行

grep "." user //找任意单个字符，文档中每个字符都可以理解为任意字符

grep ".\*" user //找任意，包括空行 .与\*的组合在正则中相当于通配符的效果

grep "ro\{1,2\}t" user //找rt，中间的o可以有1~2个

grep "ro\{2,4\}t" user //找rt，中间的o可以有2~4个

grep "ro\{4,6\}t" user //找rt，中间的o可以有4~6个

grep "ro\{2\}t" user //找rt，中间的o可以有2个

grep "ro\{3\}t" user //找rt，中间的o可以有3个

grep "ro\{2,\}t" user //找rt，中间的o可以有2个以及2个以上

grep "ro\{3,\}t" user //找rt，中间的o可以有3个以及3个以上

grep "ro\{1,\}t" user //找rt，中间的o可以有1个以及1个以上

grep "ro\{1,\}t" user //找1次以及1次以上的o

egrep "ro{1,}t" user //效果同上,egrep支持扩展正则，或者grep -E

egrep "ro+t" user //效果同上，最精简

grep "roo\{0,1\}t" user //第二个o要出现0~1次

egrep "roo{0,1}t" user //效果同上

egrep "roo?t" user //效果同上，最精简

grep "ro\{2\}t" user //找o出现2次的

egrep "ro{2}t" user //效果同上

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

egrep "^(root|bin)" user //效果同上

下课休息 16：25回