总结

写bash script，最常用的功能是遍历。

要么遍历文件夹中的文件，要么遍历文件中的内容。

1进行比对，比如查询特定的hash code文件

2进行brute force，将密码逐一输入，使用对应的指令，比如gpg，进行解密。

3对文本进行分割，比如利用IFS，配合while read -r key value。。。

q1:若指令，比如cut，要求输入的是file文件，但你要输入进去的其实就是一行内容，该怎么办?

利用echo 将一行内容，通过Pipe line给输进去。

q2: 要获得指令的内容该怎么获得?

使用**$()**，$(指令 <...>)，比如$(cat abc.txt)

q3: 字符串怎么转数字?

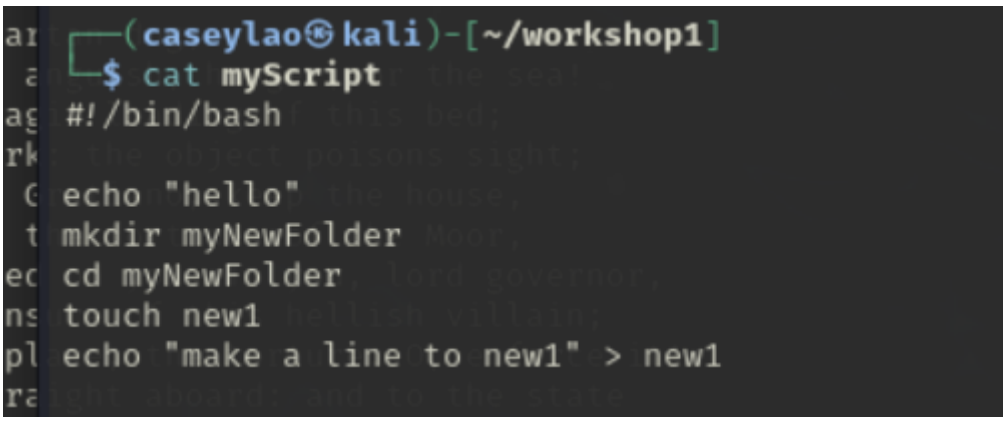
利用expr。$(expr $index + 0)。就能将index字符串转成数字。

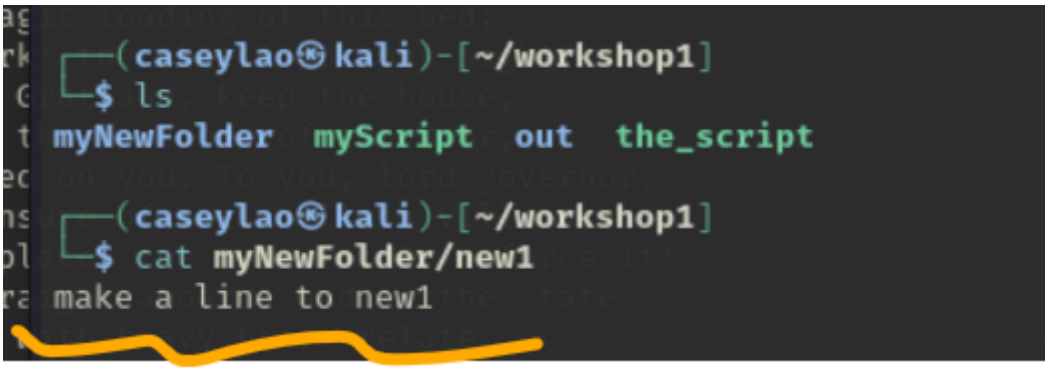
介绍

A bash script is a series of commands written in a file. These are read and executed by the bash program. The program executes line by line.

相当于，你为了实现某个目标的所要执行的每一行指令，提前写好在一个scipt中，又要实现该目标，只要运行该scipt就好。

比如创建文件夹，创建文件，创建内容，执行等一系列的操作。

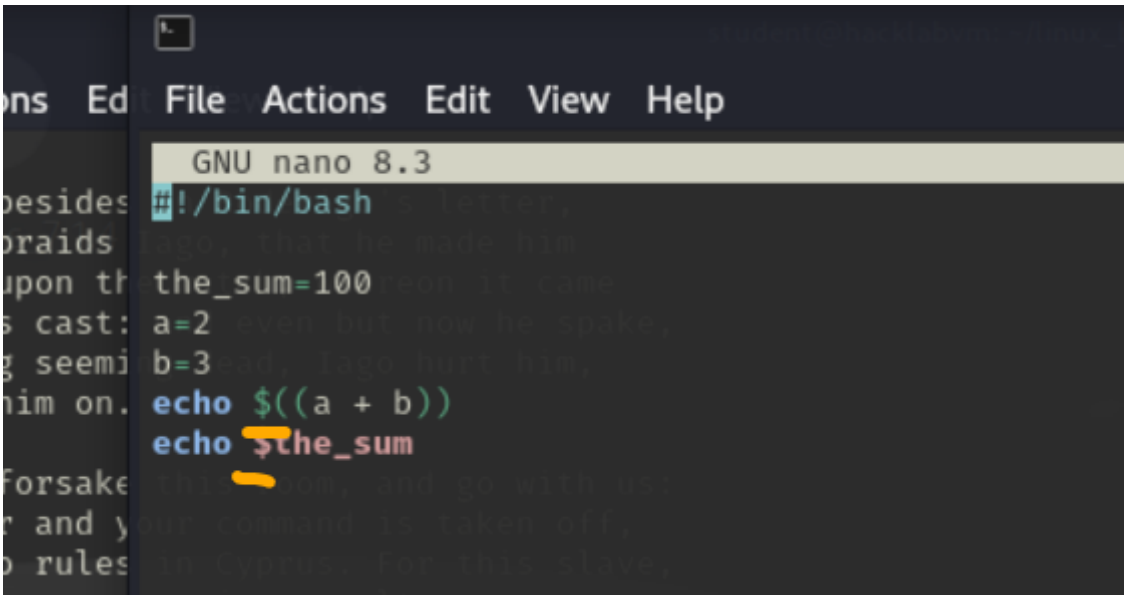




基本使用

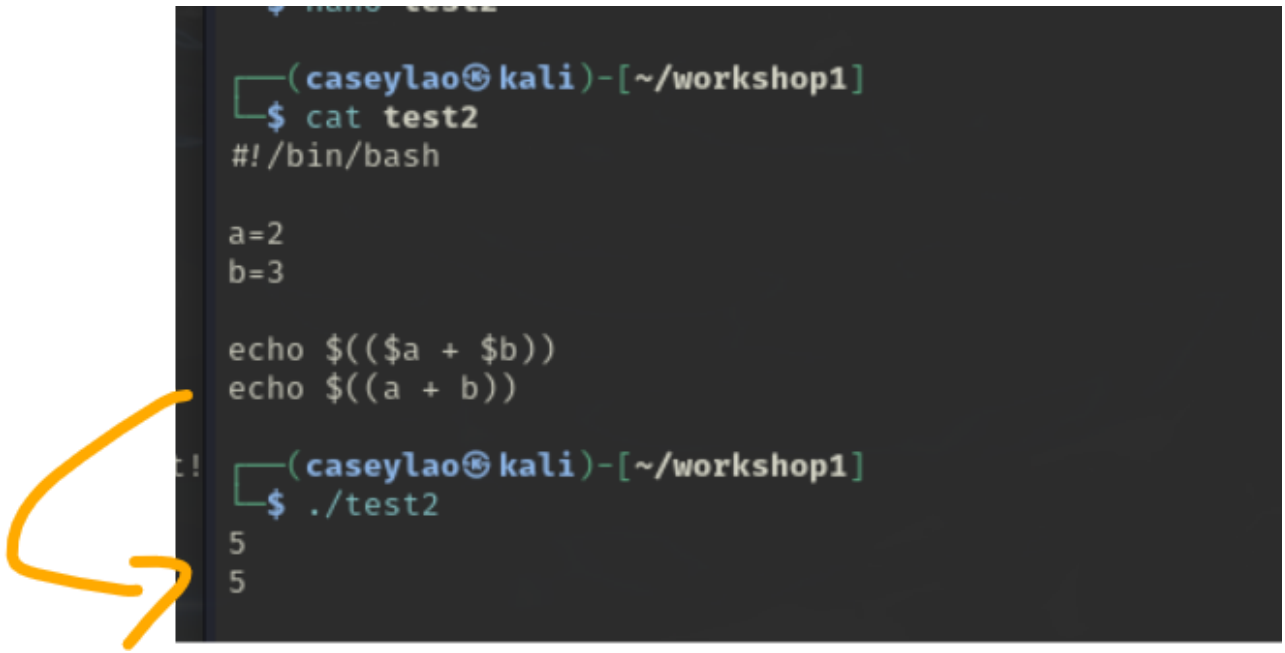
设置变量就直接写=，但是注意不要空格，比如 a = 3 No。

获取变量，要加$，$变量名



数学运算

要加$(( ))



比较运算符

算术运算符

下表列出了常用的算术运算符，假定变量 a 为 10，变量 b 为 20：

运算符	说明	举例
+	加法	`expr $a + $b` 结果为 30。
-	减法	`expr $a - $b` 结果为 -10。
*	乘法	`expr $a \* $b` 结果为 200。
/	除法	`expr $b / $a` 结果为 2。
%	取余	`expr $b % $a` 结果为 0。
=	赋值	a=$b 把变量 b 的值赋给 a。
==	相等。用于比较两个数字，相同则返回 true。	[$a == $b] 返回 false。
!=	不相等。用于比较两个数字，不相同则返回 true。	[$a != $b] 返回 true。

1243

For string equality comparison, use:

```
if [[ "$s1" == "$s2" ]]
```

For string does NOT equal comparison, use:

```
if [[ "$s1" != "$s2" ]]
```

For the `a` contains `b`, use:

```
if [[ $s1 == *"$s2"* ]]
```

(and make sure to add spaces between the symbols):

Bad:

```
if [["$s1" == "$s2"]]
```

Good:

```
if [[ "$s1" == "$s2" ]]
```

关系运算符

只支持数字。

关系运算符

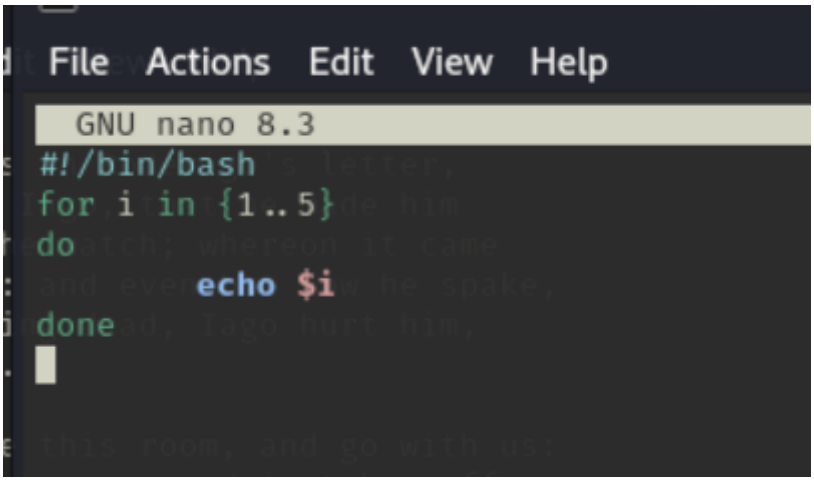
关系运算符只支持数字，不支持字符串，除非字符串的值是数字。

下表列出了常用的关系运算符，假定变量 a 为 10，变量 b 为 20：

运算符	说明	举例
-eq	检测两个数是否相等，相等返回 true。	[$a -eq $b] 返回 false。
-ne	检测两个数是否不相等，不相等返回 true。	[$a -ne $b] 返回 true。
-gt	检测左边的数是否大于右边的，如果是，则返回 true。	[$a -gt $b] 返回 false。
-lt	检测左边的数是否小于右边的，如果是，则返回 true。	[$a -lt $b] 返回 true。
-ge	检测左边的数是否大于等于右边的，如果是，则返回 true。	[$a -ge $b] 返回 false。
-le	检测左边的数是否小于等于右边的，如果是，则返回 true。	[$a -le $b] 返回 true。

for loop 实现

1遍历数字



For 循环赋值操作

```
┌──(caseylao㉿kali)-[~/workshop1]
└─$ cat myScript1
#!/bin/bash

value=0
for i in {1..6}
do
        value=$i
        echo "my value is:" $value
done

┌──(caseylao㉿kali)-[~/workshop1]
└─$ ./myScript1
my value is: 1
my value is: 2
my value is: 3
my value is: 4
my value is: 5
my value is: 6
```

2for 遍历文件夹

表示要遍历out文件夹下所有的文件。*是通配符。

```
┌──(caseylao㉿kali)-[~/workshop1]
└─$ cat myScript
#!/bin/bash
for i in out/*
do
        echo $i
done

┌──(caseylao㉿kali)-[~/workshop1]
└─$ ls
myNewFolder  myScript  out  the_script

┌──(caseylao㉿kali)-[~/workshop1]
└─$ ./myScript | head
out/00001
out/00002
out/00003
out/00004
out/00005
out/00006
out/00007
out/00008
out/00009
out/00010
```

3遍历文件内容

通过$()获得指令的内容，传入到for中。

```
student@hacklabvm:~/mya1scripts$ cat q4t
#!/bin/bash

cd /home/student/linux_basics/q04

i33t_word=""
for i in $(cat words.txt)
do
        echo $i
done
student@hacklabvm:~/mya1scripts$ ./q4t | head
mark
suppose
acceptable
business
earsplitting
level
lackadaisical
quill
evanescent
grateful
student@hacklabvm:~/mya1scripts$
```

若你想要将某指令的输出，传入变量，则要加$()

```
┌──(caseylao㉿kali)-[~/workshop1]
└─$ cat myScript1
#!/bin/bash

sum_value=0
for i in out/*
do
        sum_value=$(awk '{sum+=$1} END {print(sum)}' $i)
        echo $sum_value
done

┌──(caseylao㉿kali)-[~/workshop1]
└─$ ./myScript1 | head
1358
2037
1755
3419
1813
3460
1043
2284
2378
2673
```

if语句

✅ 基本语法：

```bash
if [ 条件 ]; then
    # 条件为真时执行的语句
fi
```

✅ 示例1：判断文件是否存在

```bash
#!/bin/bash

if [ -f "myfile.txt" ]; then
    echo "文件存在"
else
    echo "文件不存在"
fi
```

✅ 示例2：判断两个数字大小

```bash
#!/bin/bash

a=10
b=20

if [ "$a" -lt "$b" ]; then
    echo "$a 小于 $b"
fi
```

条件 - 关系运算符

✅ 支持的条件操作符：

条件	含义
`-f FILE`	FILE 存在且是普通文件
`-d DIR`	DIR 存在且是目录
`-e FILE`	FILE 存在（不管是文件还是目录）
`STRING1 = STRING2`	字符串相等
`STRING1 != STRING2`	字符串不相等
`-n STRING`	字符串长度非零
`-z STRING`	字符串长度为零
`NUM1 -eq NUM2`	数值相等
`NUM1 -ne NUM2`	数值不相等
`NUM1 -gt NUM2`	NUM1 大于 NUM2
`NUM1 -lt NUM2`	NUM1 小于 NUM2

实操

一定要空开两个格，否则报错。

总结

设置变量，直接写变量名。

获取变量内容，加$.

数学运算,$(( ))，本质上是对变量中的内容，进行运算，所以要加$。

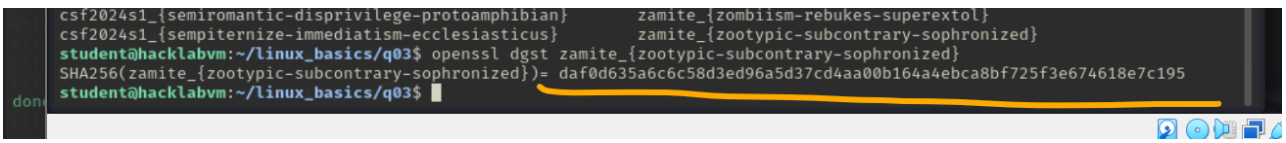
if语句，for loop要记住。格式是严格的，该空格就要空格。

```
┌──(caseylao㉿kali)-[~/workshop1]
└─$ cat myScript1
#!/bin/bash

sum_value=0
for i in out/*
do
        sum_value=$(awk '{sum+=$1} END {print(sum)}' $i)
        if [ $sum_value -gt 2000 ];then
                echo $sum_value
                echo "file is: " $i
        fi
done

┌──(caseylao㉿kali)-[~/workshop1]
└─$ ./myScript1 | head
2037
file is:  out/00002
3419
file is:  out/00004
3460
file is:  out/00006
2284
file is:  out/00008
2378
file is:  out/00009
```

实操 —— 匹配文件的hash code.

匹配一堆文件的hash code.

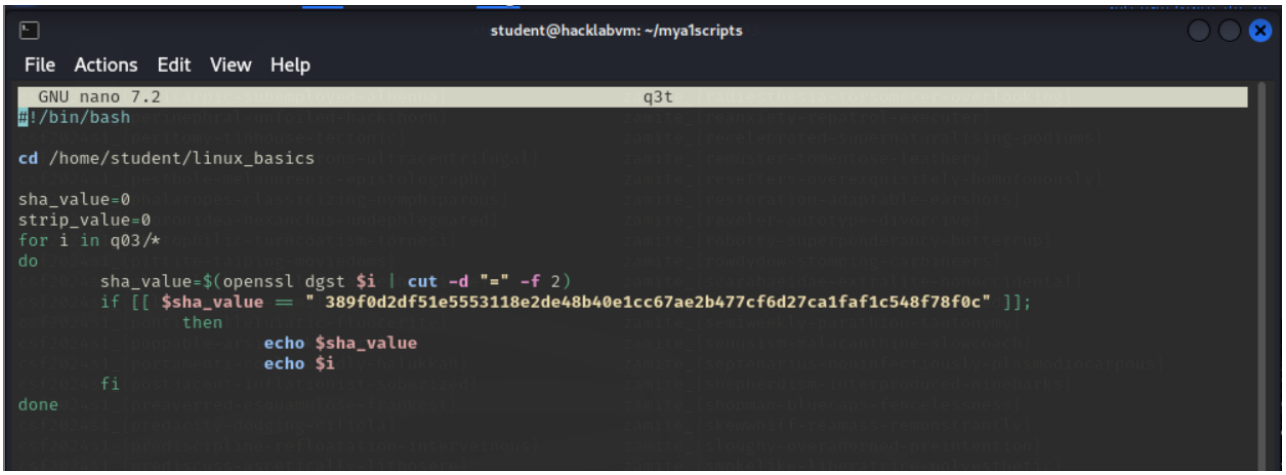
openssl dgst 文件，会得到hash code

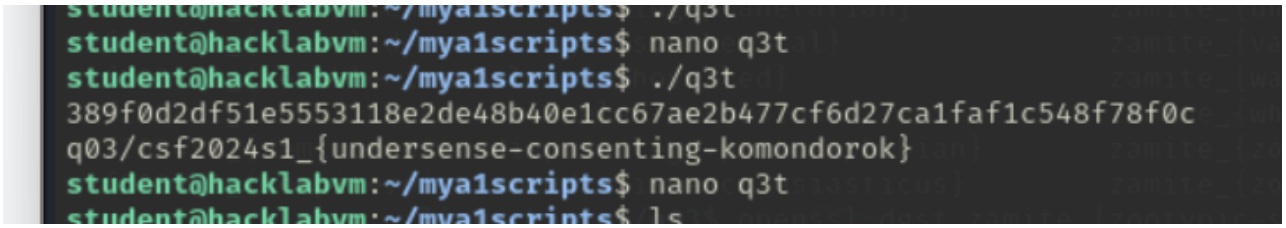


之所以用cut指令，是为了从输出截取出对应的hash code

因为是字符串，所以用==来做比较运算符。

若等于，则输出结果。





进阶

解惑了我3个疑问

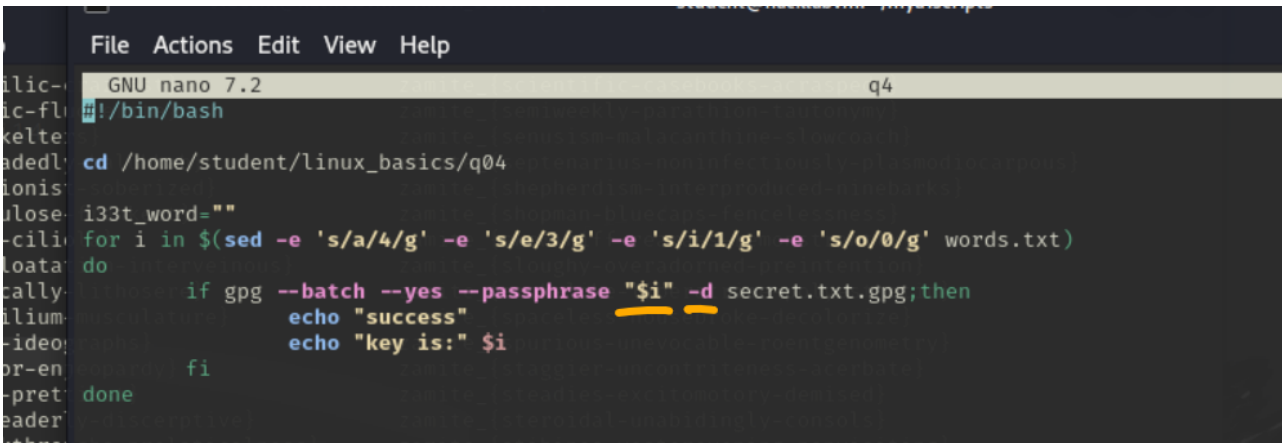
1一个指令如果有两个输入，一个是要输入的密码，另一个是对应的文件，该怎么写?

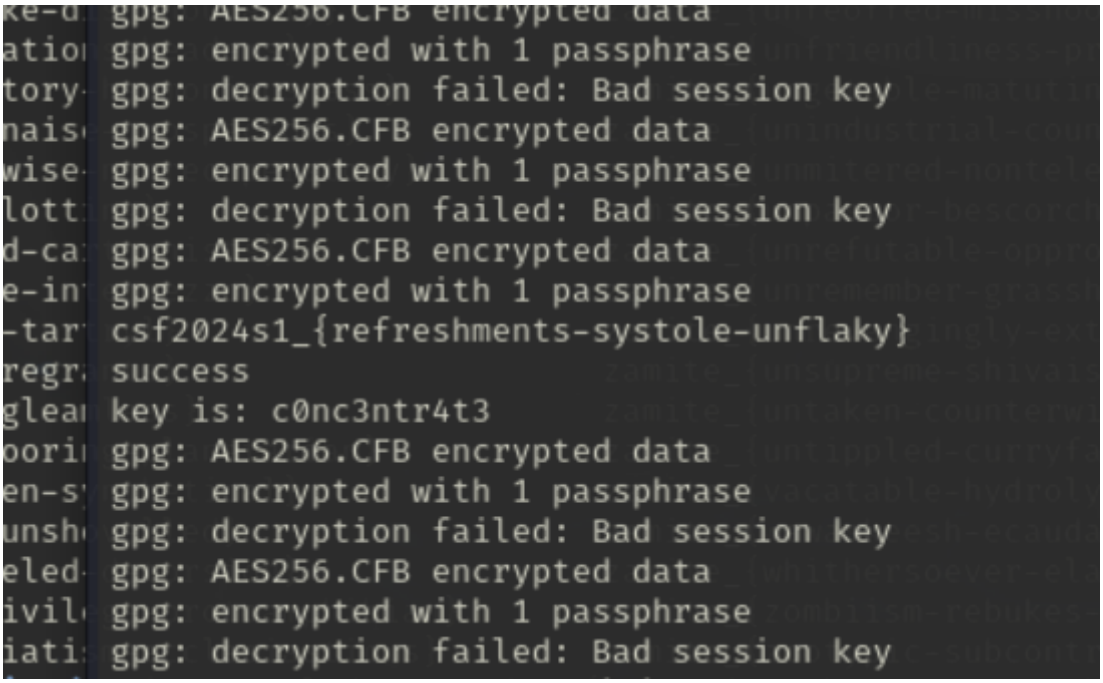
很简单，在对应的标识符后面，写上对应的输入。通常，指令的最后输入，是file名。

2 原来if不是非得写[ ]，直接是一个指令也可以。如果该指令成功，就会执行该if 语句。

3.如果我要用for**遍历一个文件的内容**，该怎么做?

很简单，直接$(cat 文件)，这样通过cat获得文件内容，形成数组，传入到for语句中。





关键 IFS

Internal Field Separator

IFS是bash内建的参数。

✅ 它的作用是：

`IFS` 控制 `read` **命令在分隔输入字段时用什么符号。**

举例来说：

```bash
IFS=":" read -r key value
```

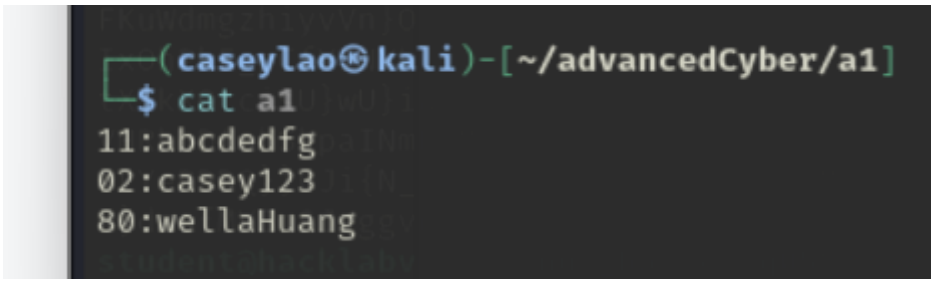
这行的意思是：

- 把一行按 **冒号** `:` 分成两部分
- 存入变量 `key` 和 `value` 中

自动将一行以 IFS来分割成两部分，分别存入key 和 value中。

while loop —— 配合IFS

利用read指令，读取内容，进入while循环，直到读完内容位置。



```
┌──(caseylao㉿kali)-[~/advancedCyber/a1]
└─$ cat a1 | ./test2
11
abcdedfg
02
casey123
80
wellaHuang

┌──(caseylao㉿kali)-[~/advancedCyber/a1]
└─$ cat test2
#!/bin/bash

IFS=":"

while read -r index content;
do
        echo $index
        echo $content

done
```

进阶使用2 —— 根据前面的index，来筛选后内容的字母

1.获得指令执行后的内容，自然要$()，若指令中含有先前的变量，可以"$()"来作为识别。

比如char=$(echo "$content" | cut -c "$index")中的content和index。但也可以不加" "

2.cut指令的输入内容要求，其实是file，但若是一行输入该怎么办?

利用echo来作为输入，内容将作为一个file，来进行处理。

```
┌──(caseylao㉿kali)-[~/advancedCyber/a1]
└─$ cat a1
05:abcdedfg
02:casey123
01:wellaHuang
```

```
File  Actions  Edit  View  Help
  GNU nano 8.3                                  test2
#!/bin/bash

IFS=":"
char=""
while read -r index content;
do
        echo $index
        echo $content
        char=$(echo "$content" | cut -c "$index")
        echo "char is:" $char
done
```

```
┌──(caseylao㉿kali)-[~/advancedCyber/a1]
└─$ cat a1 | ./test2
05
abcdedfg
char is: e
02
casey123
char is: a
01
wellaHuang
char is: w
```

情况2 —— cut的使用，echo 和无echo.

这是正常的指令使用，但是会无作用。

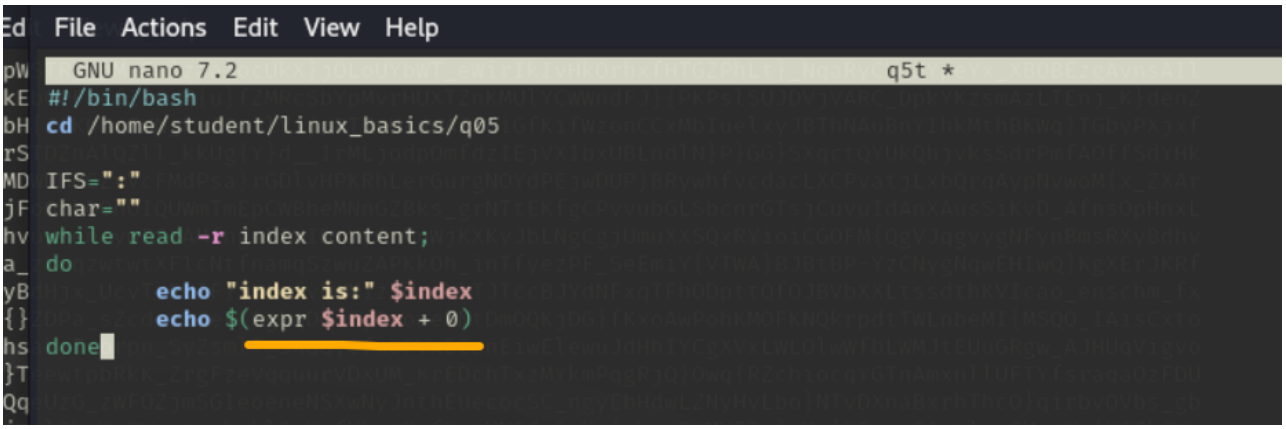
```
 GNU nano 8.3                                                te
#!/bin/bash

IFS=":"
char=""
while read -r index content;
do
        echo $index
        echo $content
        char=$(cut -c "$index" "$content")
        echo "char is:" $char
done
```

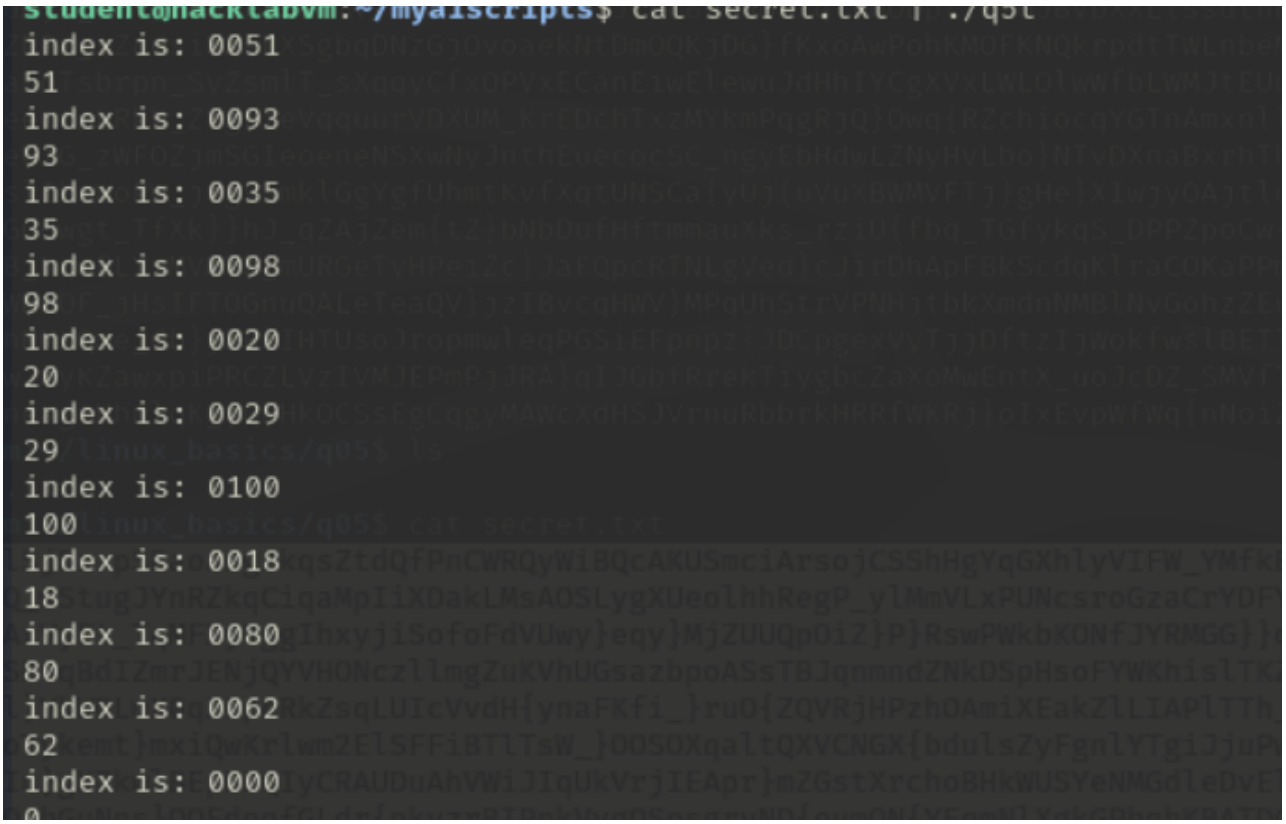
```
┌──(caseylao㉿kali)-[~/advancedCyber/a1]
└─$ cat a1 | ./test3
05
abcdedfg
cut: abcdedfg: No such file or directory
char is:
02
casey123
cut: casey123: No such file or directory
char is:
01
wellaHuang
cut: wellaHuang: No such file or directory
char is:
```

字符串转数字

bash script，**本质上所有变量，都是字符串。**只有通过特殊符号或转换。

利用expr ，然后再加0，即可转换。





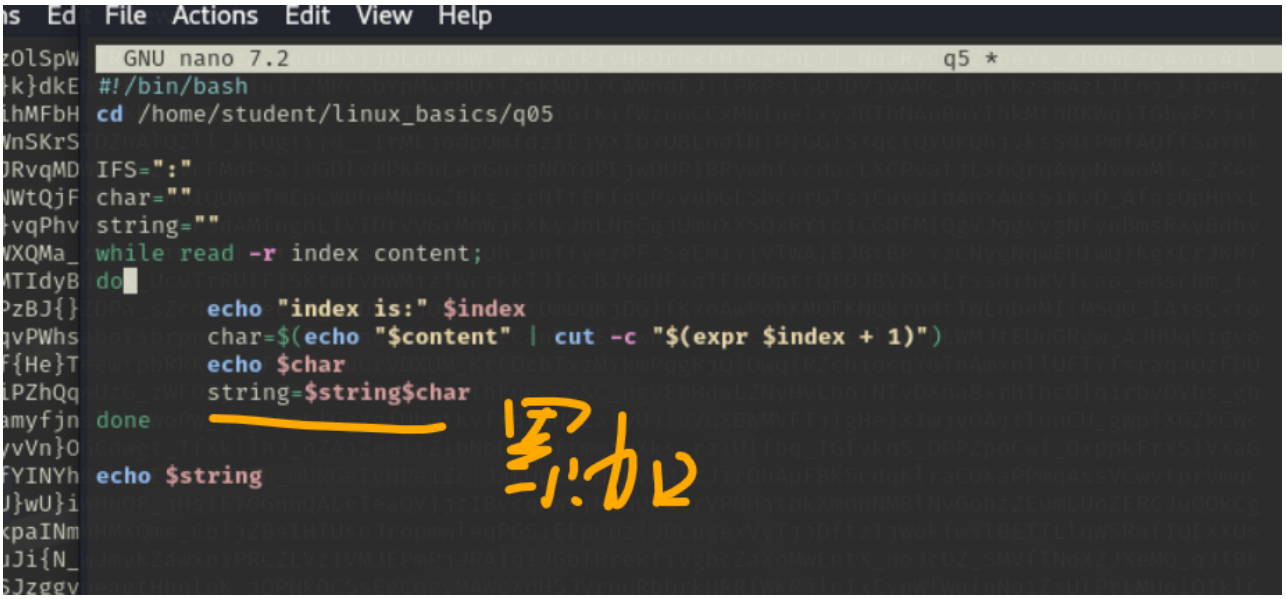
进阶3 —— 如何累加字符串

该代码的思想：利用IFS，分割每一行的输入，该题是":"。利用前面的index数，获取后面字符中的第index个字符。

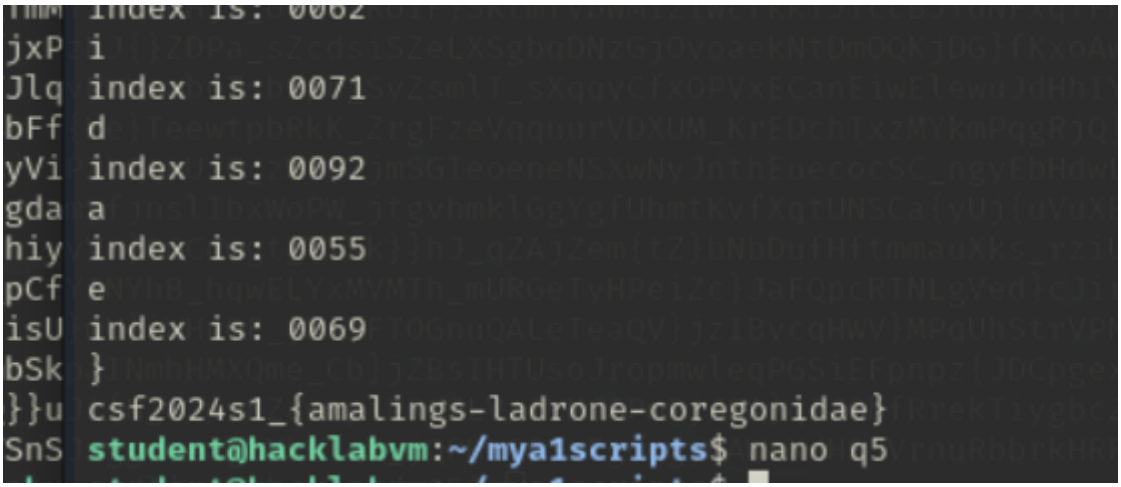
利用expr 进行字符串转数字，并且+1.这是因为cut是从1开始计算。

利用string整合累加char字符串，最终得到flag.

注意 string=$string$char，第一个string是没有$的。


File Actions Edit View Help
GNU nano 7.2 q5 *
#!/bin/bash
cd /home/student/linux_basics/q05
IFS=":"
char=""
string=""
while read -r index content;
do
echo "index is:" $index
char=$(echo "$content" | cut -c "$(expr $index + 1)")
echo $char
string=$string$char
done
echo $string
累加


student@hacklabvm:~/linux_basics/q05$ cat secret.txt
0051:srQWlOSQMWNlSjDzxpSDaoangzkqsZtdQfPnCWRQyWiBQcAKUSmciArsojCSShHgYqGXhlyVIFW_YMfkBEDirIRtYaechx{__fZJN
0093:bXnTf}xoDRnQ_wStugJYnRZkqCiqaMpIiXDakLMsAOSLygXUeolhhRegP_ylMmVLxPUNcsroGzaCrYDFYmWGJrTIQP{PAsLxCUNPT
0035:BQxMovdLgqRAxVyFt_TqMFXyYggIhxyjiSofoFdVUwy}eqy}MjZUUQpOiZ}P}RswPWkbKONfJYRMGG}}djvkLptmLFbkAXtYcUo{B
0098:IzMBAdyazJASDvqBdIZmrJENjQYVHONczllmgZuKVhUGsazbpoASsTBJqnmndZNkDSpHsoFYWKhislTKZSFPaGoW_LCnCkPtLg2qM
0020:suFmRhmtkJDl}VkWDLmX0qZFpcRkZsqLUIcVvdH{ynaFKfi_}ruO{ZQVRjHPzhOAmiXEakZlLIAPlTTh_bpyYSkKEcZTLknNmtfux
0029:H{jHX_KpzHroTykemt}mxiQwKrlwm2ElSFFiBTlTsW_}OOSOXqaltQXVCNGX{bdulsZyFgnlYTgiJjuPwyCfwcYfHywI{{NfnIPTN
0100:CqtGBobgjlKIu}gXZkwlBEpHZQIyCRAUDuAhVWiJIqUkVrjIEApr}mZGstXrchoBHkWUSYeNMGdleDvElvEKoPS_wkFZOqMkEfSm4
0018:jZjr}m}QkhiOzbGuNns}OOFdqofGLdr{pkvzrBIPnkVvgQSosgrvND{ovmQN{YEomNlXgkGPbqhKBATDCokSHKHYj}X}IVCYBCUox
0080:yzVyCMqZFSSt{faLQjosSsYZtabgzrktygzMtomhvN_TErHMUgVcPQIRNCbbCkpOzwSgRNz{stAUENuq1m_XULLbzVnDk}nNsAqBh
0062:LAOjIPXD_SnNwMktXiTRkTYvJERRKyRjNQNmTnpLFUXduXuTGDQ}kTyKHIGbOB_nkiv}TtQgKHrmzpOpJfmUOzLWvOMazcxwSslWE
0000:{DQ{IWIhabtyhFviGJSHHHMKpILmQce_xufg}fFUmvMcGov_BhEhlezr}eDoiSAVijqwUdXeTaOEKMSZllCJfRTIZfvWXl}}OIrvq
0063:REmSEleNHmqRPbdMOcopNGZPcxuUBosP}uf_kIMJmheXDnBqejhNyTEy}StypjtaVCvmeRjWgzsVWaZVxFv_QwimA_xpWGnGSw{dL
0055:Lu_XM{pYceiqXpuTgukywnCtvtSkcnCbSTPOidfaiotljbmDzrrrITlmzwSDyFjgOkCmIzaIUHhCEKoXPCOIKLWzlNOQsCvmSsTaZ
0028:ncjSyGqixXCNZtoYcneyRmAPWcjIadJhS{OAVZRnHTzqqhVtKfhBuMJWMhCtDaIkGEf}TAcPWlMG_W_IrJqajAYaMatZ{oSdARzYK
0053:ymAZOGUeijpfRyxOgSZvHwEncTgSDRTwolTqJHzLBMtJrK_SBphCnlcHeMnnYFiudPctyfAQpBFqcQggTQjjNVuYILySf}uNIzYZR
0066:DmyDFcgsUGemtsvHCvcNw_kPGqzY}wylV}VwBIRXTUUuCjNmAQFhJF{x}PgITtJlDYiZTOTR{PykDoPtSmpHx_WFweFcOBonwsnGf
0027:awfwcHTKVRFDBRnxvFZqBBqjXIgntbzalfmKAQjBG{zzIN}UQTbicTGThTrYiIbQ_rfp__kgwZGiEZarLUwxpBOyCxPaVhTSjGdnH
0093:ADQWzrGfeNLRpLdeCorNBWzQpTBDcJRiCGAWOJR{DWtDFX_rZNrXN{qPHA}DzKVSkena{XrsYLJVkmODdUELHdqaJ_jUeg_{WOzxS
0023:meVrasrRqmLCVnEsbfrFcddsSMmRyAzlhMrSP_ydGhA_eVtBoccsBhRinOjYehSmGrHBLQNeKvXBZxYMzGDrUrmPgNysOI}aYCeQd
0049:yV}FTIWtyFhWy_TizZeNCBPAL}XfJPty{IJAvrDQuG_wxYGMf-uDsLbJwLbyvlGjYzBJLBzguS{lsm{E_KLcCYkwuOjoaPFdqkPQd
0012:KM}aEKqVVdzOlSpWSTRyvpMnEwizaaiocUkX}jOLoUYbWT_eWirIkIvHkOrhxfHTGZPhLt}_NqaRyChQZYpeYx_XBOBEzgAvnsAIl
0020:GIaYsoEV{c}k}dkEbOPFacfZEGsM{u}fZMRcSbYpMvrHUXTZnKMUlYCWWndFJ}{PKPslSUJDVjVARC_DpkYKzsmAzLTEnj_K}denZ
0044:ZbVYGHET{{ihMFbH{ZIKbMqFVN_kyJOoJlYijpPTJvU{dWAZxdiGfKifWzonCCxMbIuelxyJBThNAuBnYIhkMthBKWq}TGbyPXjxf
0088:aAZMEJuIFsWnSKrSTDZnAlQZll_kkUg{Y}d__IrMLjodpOmfdzIEjVXIbxUBLndlN}P}GG}SXqctQYUkQhjvksSdrPmfAOffSdYHk
0092:yARZ_cMxR_JRvqMDWUMoGZvVcFMdPsa}rGDlvHPKRhLerGurgNOYdPEjwDUP}BRywhfvcdacLXCPvatjLxbQrqAypNvwoM{x_ZXAr
0067:VCaEUWvXUBNWtQjFoQobDOnOIQUWmTmEpCWBheMNnGZBks_grNTtEKfgCPvvubGLSbcnrGTsjCuvuIdAnXAusSiKvD_AfnsOpHnxL
0021:rzUNUGrZPN}vqPhvuyreneywFSdAMfngnLIVIDtVyGrMnWjKXKyJbLNgCgjUmuXXSQxRYioiCGOFM{QgVJqgvygNFynBmsRXyBdhv
0076:kvMd_DJHdNWXQMa_zbuqzwtwtXFlcNtfnamqSzwuZAPKkOh_inTfyezPF_SeEmiY{VTWA}BJBtBP-YzCNygNqwEHIwQ}KgXErJKRf
0088:FRyCNEngTmMTIdyBdHjx_UcvTrRUIF}SKtmfvbWMizIWcrkkTJTccBJYdNFxqTFhODpttQfOJBVbXXLtssdthKVIcao_enschm_fx
0061:sFrOtB{fjxPzBJ{}ZDPa_sZcdsiSZeLXSgbqDNzGjOvoaekNtDmOQKjDG}fKxoAwPohKMOFKNQkrpdtTWLnbeMI{MSQO_IAisCxto
0022:AyoKlWZTJlqvPWhsaboTsbrpn_SyZsmlT_sXqqyCfxOPVxECanEiwElewuJdHhIYCgXVxLWLOlwWfbLWMJtEUuGRgw_AJHUqVigvo
0013:wQkBqQgBbFf{He}TeewtpbRkK_ZrgFzeVqquurVDXUM_KrEDchTxzMYkmPqgRjQ}Owq{RZchiocqYGTnAmxnllUFTYfsraqaOzFDU
0056:FTXAGGDCyViPZhQqeUzG_zWFOZjmSGIeoeneNSXwNyJnthEuecocSC_ngyEbHdwLZNyHvLbo}NTvDXnaBxrhThcO}qirbvOVbs_gb
0022:_a}EvFicgdamyfjnslIbxWoPW_jtgvhmklGgYgfUhmtKvfXqtUNSCa{yUj{uVuXBWMVFTj}gHe}XIwjyOAjtlunCU_gWp{XGZkCWc
0013:FKuWdmgzhiyvVn}OGCdwgt_TfXk}}hJ_qZAjZem{tZ}bNbDufHftmmauXks_rziU{fbq_TGfykqS_DPPZpoCw{_OxppkFrxS}VXaG
0062:IxOhPkE_pCfYINYhB_hqwELYxMVMTh_mURGeTyHPeiZc}JaFQpcRTNLgVed}cJirDhApFBkScdqKlraCOKaPPmqAssVCwvtprVmqC
0071:tXTkXOtcisU}wU}iWHuOF_jHsIFTOGnuQALeTeaQV}jzIBvcqHWV}MPqUhStrVPNHjtbkXmdnNMBlNvGohzZEumLUoZERCJuQOkCg
0092:VHRkoDfmbSkpaINmhHMXQme_Cb}jZBsIHTUsoJropmwleqPGSiEFpnpz{JDCpgexVyTjjDftzIjWokfwslBETTLlqWSRafIQEXXUs
0055:ULYeRDDQ}}uJi{N_wJmyKZawxpiPRCZLVzIVMJEPmPjJRA}qIJGbfRrekTiygbcZaXoMwEntX_uoJcDZ_SMVfTNoXZJXeMQ_qJTBk
0069:R}daefMVSnSJzggvneagtHhnlqK_jQPHkOCSsEgCqgyMAWcXdHSJVrnuRbbrkHRRfWKRj}oIxEvpWfWq{nNoiZsUlPrLMUoIQIklC
student@hacklabvm:~/linux_basics/q05$


index is: 0062
i
index is: 0071
d
index is: 0092
a
index is: 0055
e
index is: 0069
}
csf2024s1_{amalings-ladrone-coregonidae}
student@hacklabvm:~/mya1scripts$ nano q5