

## 14. Hints and Tips

```
[[ : [[ [[ [[ https://www.shellscript.sh/tips [[ [[ [. [[  
[[ [[ [[ [[ [[ . CGI [[ [[ [[ [[
```

在 Linux 中，我们通常使用 `ls` 命令来列出目录内容。但是，如果我们想要更详细的信息，比如文件的权限、所有者、大小等，我们可以使用 `ls -l`。此外，我们还可以使用 `find` 命令来查找文件。例如，`find / -type f` 会列出所有文件。

\*   "   "    , -  .

# CGI Scripting

```
CGI [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] . [REDACTED] CGI
[REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] . [REDACTED]
CGI [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] [REDACTED] CGI [REDACTED]
[REDACTED] [REDACTED] [REDACTED] . fortune.cgi [REDACTED] cookie.cgi
```

## Exit Codes

0 到 255 的整数，在 Unix 系统中，每个字节的大小是 8 位，因此 256 个字节的大小是 2048 位，即 256 字节。

□ □□□      □□    □□    □□    □□□    □□    □□    □    □    □□    □    □    □□    □□    □

□□□    . □    □□□    10□ , "□ - 2□ "□    □□□    □□□□    . □□□    □    □□    □□    □□    □

□ □ □□    □□□□□□    .

00 (**Success**) 00000 00 00 0000 , 00 (**Failure**) 00000 00 00 00 0000  
 00000 . 00 00 000 000 000 00 0000 . 00 00 GNU grep 0000 00 ,  
 0000 000 000 10 , 00 00 (00 00 , 0000 00 00 00 00 ) 0000 20 00000

```
#!/bin/sh

# First attempt at checking return codes

USERNAME=`grep "^${1}:" /etc/passwd|cut -d":" -f1`

if [ "$?" -ne "0" ]; then

    echo "Sorry, cannot find user ${1} in /etc/passwd"

    exit 1

fi

NAME=`grep "^${1}:" /etc/passwd|cut -d":" -f5`

HOMEDIR=`grep "^${1}:" /etc/passwd|cut -d":" -f6`


echo "USERNAME: $USERNAME"

echo "NAME: $NAME"

echo "HOMEDIR: $HOMEDIR"
```

USERNAME :  
NAME :  
HOMEDIR :

```
#!/bin/sh

# Second attempt at checking return codes
grep "^${1}:" /etc/passwd > /dev/null 2>&1

if [ "$?" -ne "0" ]; then

    echo "Sorry, cannot find user ${1} in /etc/passwd"

    exit 1

fi

USERNAME=`grep "^${1}:" /etc/passwd|cut -d":" -f1`
NAME=`grep "^${1}:" /etc/passwd|cut -d":" -f5`
HOMEDIR=`grep "^${1}:" /etc/passwd|cut -d":" -f6`
```

```
echo "USERNAME: $USERNAME"
echo "NAME: $NAME"
echo "HOMEDIR: $HOMEDIR"
```

Das ist ein Skript, das die Umgebungsvariablen USERNAME, NAME und HOMEDIR ausliest und sie auf der Standardausgabe ausgibt. Es ist ein einfaches Skript, das die Umgebungsvariablen USERNAME, NAME und HOMEDIR ausliest und sie auf der Standardausgabe ausgibt.

Das Skript ist in drei Zeilen unterteilt. Die erste Zeile gibt den Namen des Benutzers aus, die zweite Zeile den Namen und die dritte Zeile das Homeverzeichnis.

```
#!/bin/sh
# A Tidier approach

check_errs()
{
    # Function. Parameter 1 is the return code
    # Para. 2 is text to display on failure.
    if [ "${1}" -ne "0" ]; then
        echo "ERROR # ${1} : ${2}"
        # as a bonus, make our script exit with the right error code. exit ${1}
    fi
}

### main script starts here ###

grep "^${1}:" /etc/passwd > /dev/null 2>&1
check_errs $? "User ${1} not found in /etc/passwd"
USERNAME=`grep "^${1}:" /etc/passwd|cut -d":" -f1`
check_errs $? "Cut returned an error"
echo "USERNAME: $USERNAME"
check_errs $? "echo returned an error - very strange!"
```

Das Skript ist in drei Zeilen unterteilt. Die erste Zeile gibt den Namen des Benutzers aus, die zweite Zeile den Namen und die dritte Zeile das Homeverzeichnis.

Das Skript ist in drei Zeilen unterteilt. Die erste Zeile gibt den Namen des Benutzers aus, die zweite Zeile den Namen und die dritte Zeile das Homeverzeichnis.

```
#!/bin/sh
cd /usr/src/linux && \
    make dep && make bzImage && make modules && \
    make modules_install && \
    cp arch/i386/boot/bzImage /boot/my-new-kernel && \ cp System.map /boot && \
    echo "Your new kernel awaits, m'lord."
```

`if` `(` `int` `a` `=` `0` `)` `{` `int` `b` `=` `1` `;}`

```
#!/bin/sh
cd /usr/src/linux
if [ "$?" -eq "0" ]; then
    make dep
    if [ "$?" -eq "0" ]; then
        make bzImage
        if [ "$?" -eq "0" ]; then
            make modules
            if [ "$?" -eq "0" ]; then
                make modules_install
                if [ "$?" -eq "0" ]; then
                    cp arch/i386/boot/bzImage /boot/my-new-kernel
                    if [ "$?" -eq "0" ]; then
                        cp System.map /boot/
                        if [ "$?" -eq "0" ]; then
                            echo "Your new kernel awaits, m'lord."
                        fi
                    fi
                fi
            fi
        fi
    fi
fi
```

...      .

☐☐ && ☐ ||☐ ☐☐ AND ☐ OR ☐☐☐☐ ☐☐☐☐☐☐☐ .☐ ☐ ☐ ☐☐ ☐ ☐,  
☐ :

```
#!/bin/sh
cp /foo /bar && echo Success || echo Failed
```

❏   ❏   ❏   echo❏   .

Success

❏

Failed

❏   cp❏   ❏❏❏   ❏❏❏❏   ❏   ❏❏❏   .❏   ❏❏   ❏❏   ❏❏   :

```
command && command-to-execute-on-success \
|| command-to-execute-on-failure
```

❏   ❏❏❏   ❏❏   ❏❏   ❏❏   ❏   ❏❏❏   .❏   ❏❏❏   ❏❏   ❏   /❏   ❏❏❏❏❏   ❏❏❏❏   ,❏  
❏   ❏❏❏   ❏❏   ❏❏❏❏   ❏   ❏   &&❏   ||❏   ❏   ❏❏   ❏❏❏❏   ❏   ❏❏❏   ❏❏❏   .❏  
❏   ❏❏❏   ❏   ❏❏❏❏   .❏❏❏   ❏   ❏❏❏   ❏❏   ❏   ❏❏❏❏   ❏❏❏❏   ❏   ❏❏❏   .

❏   ❏❏❏❏   cp❏❏❏   ❏   ❏   ❏   ❏❏   ❏   ❏❏❏❏   ❏❏❏❏   ❏   ❏❏❏   ❏❏❏   ❏   ❏❏❏  
❏❏❏   ❏   ❏❏❏❏   :

```
cp /foo /bar && \
( echo Success ; echo Success part II; ) || \
( echo Failed ; echo Failed part II )
```

❏❏❏   ❏❏❏   Marcel❏   ❏❏❏   ❏❏❏❏   ❏❏❏❏   ❏   ❏❏❏❏❏❏   .❏❏❏❏   ❏❏❏   ❏❏❏  
❏❏❏❏   :

```
( command1 ; command2; command3 )
```

❏❏❏❏   ❏   ❏❏❏   ❏   ❏   (❏   ❏❏❏❏   command3)❏   ❏   ❏❏❏❏❏   .❏   ❏   ❏❏❏   ❏   ❏❏❏  
❏❏❏   ❏❏❏❏   .❏❏❏❏   ❏   ❏❏❏❏❏   ❏❏❏   ❏❏❏   ❏❏❏❏   :

```
cp /foo /bar && \
( echo Success ; echo Success part II; /bin/false ) ||\
( echo Failed ; echo Failed part II )
```

cp❏   ❏❏❏❏   ❏❏❏   ❏   ❏❏❏   ❏❏❏❏   , /bin/false❏   ❏❏❏   ❏❏❏❏❏   ❏❏❏   ❏   ❏❏❏   ❏❏❏❏❏  
❏❏❏❏   :

```
Success
Success part II
Failed
Failed part II
```

if, then, else  
.

## Simple Expect Replacement

expect  
Sun Microsystems Explorer  
.

expect.txt

```
S command E[delay] expected_text
```

"S"(Send  
"E"  
:"E10 \$" 10  
1  
MAX\_WAITS  
"E \$"

**MAX\_WAITS=5** 5 1+2+3+4+5=15

```
#!/bin/sh
# expect.sh | telnet > file1
host=127.0.0.1
port=23
file=file1
MAX_WAITS=5

echo open ${host} ${port}

while read l
do
c=`echo ${l}|cut -c1`
if [ "${c}" = "E" ]; then
    expected=`echo ${l}|cut -d" " -f2-`
```

```

delay=`echo ${l}|cut -d" " -f1|cut -c2-`
if [ -z "${delay}" ]; then
    sleep ${delay}
fi
res=1
i=0
while [ "${res}" -ne "0" ]
do
    tail -1 "${file}" 2>/dev/null | grep "${expected}" > /dev/null
    res=$?
    sleep $i
    i=`expr $i + 1`
    if [ "${i}" -gt "${MAX_WAITS}" ]; then
        echo "ERROR : Waiting for ${expected}" >> ${file}
        exit 1
    fi
done
else
    echo ${l} |cut -d" " -f2-
fi
done < expect.txt

```

### ##### :

```
$ expect.sh | telnet > file1
```

### ## ##### file1### ## . ## ## , /tmp ## ls, cal  
### ## . ## :

```

telnet> Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.

declan login: steve
Password:
Last login: Thu May 30 23:52:50 +0100 2002 on pts/3 from localhost.
No mail.
steve:~$ ls /tmp
API.txt                cgihtml-1.69.tar.gz    orbit-root
cal

```

```

a.txt          cmd.txt          orbit-steve
apache_1.3.23.tar.gz  defaults.cgi  parser.c
b.txt          diary.c          patchdiag.xref
background.jpg drops.jpg       sh-thd-1013541438
blocks.jpg     fortune-mod-9708.tar.gz  stone-dark.jpg
blue3.jpg     grey2.jpg          water.jpg
c.txt          jpsock.131.1249

steve:~$ cal

      May 2002
Su Mo Tu We Th Fr Sa
                1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31

steve:~$ exit

logout

```

# Trap

```

Trap  00000000  00  000  00000000  . 00  00000  00  0000  FOO  BAR  000  0
000  00000  00  00000  00  000  0000  00  , 00000  000  0 /tmp  000000
000  00000  000  0000  /tmp  000  0000  0  0000  :

```

```

#!/bin/sh

trap cleanup 1 2 3 6

cleanup()
{
    echo "Caught Signal ... cleaning up."
    rm -rf /tmp/temp_*. $$
    echo "Done cleanup ... quitting."
    exit 1
}

### main script
for i in *

```



```
do
    sed s/F00/BAR/g $i > /tmp/temp_${i}.$$ && mv /tmp/temp_${i}.$$ $i
done
```

```
trap '' SIGINT signal 1, 2, 3 6 cleanup()
(CTRL-C) signal 2(SIGINT)
```

```
#!/bin/sh

trap 'increment' 2

increment()
{
    echo "Caught SIGINT ..."
    X=`expr ${X} + 500`
    if [ "${X}" -gt "2000" ]
    then
        echo "Okay, I'll quit ..."
        exit 1
    fi
}

### main script
X=0
while :
do
    echo "X=$X"
    X=`expr ${X} + 1`
    sleep 1
done
```

```

. CTRL-C
.
4
( 2000 )
kill -9 <PID>
:
:
```

Number	SIG	
0	0	
1	SIGHUP	

2	SIGINT	终端
3	SIGQUIT	终端 (Quit)
6	SIGABRT	终端 (Abort)
9	SIGKILL	终端 Kill (强制终止)
14	SIGALRM	闹钟
15	SIGTERM	终端 (Terminate)

在 Linux 中，使用 `nohup` 命令可以防止进程在终端关闭后被杀死。使用 `nohup` 命令时，通常会将输出重定向到一个文件中，以便查看程序的运行结果。

## echo: -n vs \c

在 Linux 中，`echo` 命令用于在终端上输出文本。默认情况下，`echo` 会在输出文本的末尾添加一个换行符。使用 `-n` 选项可以抑制换行符的输出。使用 `\c` 选项可以抑制换行符的输出，并在输出文本的末尾添加一个换行符。

在 Unix 中，使用 `echo -n message` 可以在终端上输出消息，而不会在消息末尾添加换行符。使用 `echo message \c` 可以在终端上输出消息，并在消息末尾添加换行符。

```
echo -n "Enter your name:"
read name
echo "Hello, $name"
```

在终端中运行上述代码，输入名称后，输出结果如下：

```
Enter your name: Steve
Hello, Steve
```

在 Linux 中，使用 `echo "Enter your name: \c"` 可以在终端上输出消息，并在消息末尾添加换行符。

```
echo "Enter your name: \c"
read name
echo "Hello, $name"
```

在终端中运行上述代码，输入名称后，输出结果如下：

在 Linux 中，使用 `if [ "`echo -n`" = "-n" ]; then` 可以在终端上输出消息，并在消息末尾添加换行符。

```
if [ "`echo -n`" = "-n" ]; then
    n=""
```

```

c="\c"
else
n="-n"
c=""
fi

echo $n Enter your name: $c
read name
echo "Hello, $name"

```

echo -n 名字 姓 -n 名字 姓 echo 名字 , 姓 \$n 名字 姓  
\$c \c 名字 . 名字 姓 名字 名字 \$n -n 名字 \$c 名字 名字  
名字 .

名字 名字 名字 名字 cut 名字 名字 名字 . 名字 名字 名字 名字  
名字 名字 名字 名字 名字 名字 名字 .

grep 名字 名字 名字 名字 . grep 名字 名字 名字 :

```

#!/bin/sh
steves=`grep -i steve /etc/passwd | cut -d: -f1`
echo "All users with the word \"steve\" in their passwd"
echo "Entries are: $steves"

```

名字 名字 名字 名字 名字 名字 名字 名字 . 名字 /etc/passwd 名字  
名字 名字 名字 名字 名字 名字 名字 :

```

$> grep -i steve /etc/passwd
steve:x:5062:509:Steve Parker:/home/steve:/bin/bash
fred:x:5068:512:Fred Stevens:/home/fred:/bin/bash
$> grep -i steve /etc/passwd |cut -d: -f1
steve
fred

```

名字 名字 名字 :

Entries are: steve fred

名字 名字 名字 NEWLINE 名字 名字 . sh 名字 名字 \$IFS 名字 名字  
名字 名字 名字 名字 . IFS 名字 <space><tab><cr>名字 . 名字 名字  
NEWLINE 名字 名字 名字 : 名字 NEWLINEs.... 名字 名字 名字 名字 . 名字 tr  
名字 名字 名字 :

```
#!/bin/sh
steves=`grep -i steve /etc/passwd | cut -d: -f1`
echo "All users with the word \"steve\" in their passwd"
echo "Entries are: "
echo "$steves" | tr ' ' '\012'
```

tr 命令 8 个 012(NEWLINE) 字符 替换 所有 空格 字符 . tr 命令 将 所有 小写字母 替换 为大写字母 . 命令 如下 :

```
#!/bin/sh
steves=`grep -i steve /etc/passwd | cut -d: -f1`
echo "All users with the word \"steve\" in their passwd"
echo "Entries are: "
echo "$steves" | tr ' ' '\012' | tr '[a-z]' '[A-Z]'
```

命令 [a-z] [A-Z] 替换 所有 小写字母 为大写字母 . a-z 替换 A-Z 命令 将 所有 小写字母 替换 为大写字母 . 命令 如下 :  
 命令 ASCII 码 a-z 命令 将 所有 小写字母 替换 为大写字母 . 命令 如下 :  
 命令 . tr 命令 将 所有 小写字母 替换 为大写字母 . tr [:lower:] [:upper:] 命令 将 所有 小写字母 替换 为大写字母 .  
 命令 . 命令 将 所有 小写字母 替换 为大写字母 . tr 命令 将 所有 小写字母 替换 为大写字母 .

## Cheating

命令 将 所有 小写字母 替换 为大写字母

命令 将 所有 小写字母 替换 为大写字母 ! 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 . sed 命令 将 所有 小写字母 替换 为大写字母 .  
 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 .  
 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 .

命令 将 所有 小写字母 替换 为大写字母 (sed 命令 52k, awk 命令 110k) 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 .  
 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 .  
 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 .

## Cheating with awk

命令 将 所有 小写字母 替换 为大写字母 , 命令 将 所有 小写字母 替换 为大写字母 . wc 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 :

```
$ wc hex2env.c
    102    189   2306   hex2env.c
```

命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 . 命令 将 所有 小写字母 替换 为大写字母 .

```
NO_LINES=`wc -l file`
```

```
0 0000    00 00 00 0 0000 .000 000 0000    00 000 00 1020 0000  
0000    00 0 0000 .00 ,awk C scanf 0000    0000    000 00 00  
000 00000 .00 $1 $2 $3 00 000 0000 .000 0 000 00000 :
```

```
NO_LINES=`wc -l file | awk '{ print $1 }'`
```

```

00 NO_LINES 000 102000 .

```

# Cheating with sed

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

```
sed s/eth0/eth1/g file1 > file2
```

```

1 # eth0 2 eth1 . 3 tr 4 .tr 5 :

```

```
echo ${SOMETHING} | sed s/"bad word"/g
```

```

[[[ ${SOMETHING} [bad word]" [ [ . [ [ [
[
[
"[ grep [ [ [ !" [ [ [ [ [ [ [ - grep [ [ [ [ [
[ [ [ :






























































































```

```
This line is okay.  
This line contains a bad word. Treat with care.  
This line is fine, too.
```

grep           , sed     :

```
This line is okay.  
This line contains a . Treat with care.  
This line is fine, too.
```

## Telnet hint






























































```
$ ./telnet1.sh | telnet
```

在 `grep` 命令中，`-q` 选项用于静默模式，即只返回匹配结果，不显示匹配内容。GNU `grep` 命令的 `-q` 选项与 `grep` 命令的 `-q` 选项类似，但 `grep` 命令的 `-q` 选项是用于指定输出文件的路径，如 `/dev/null`。

```
#!/bin/sh
host=127.0.0.1
port=23
login=steve
passwd=hellothere
cmd="ls /tmp"

echo open ${host} ${port}
sleep 1
echo ${login}
sleep 1
echo ${passwd}
sleep 1
echo ${cmd}
sleep 1
echo exit
```

[illegible]

```
$ ./telnet2.sh | telnet > file1
```

```
#!/bin/sh
# telnet2.sh | telnet > FILE1
host=127.0.0.1
port=23
login=steve
passwd=hellothere
cmd="ls /tmp"
timeout=3
file=file1
prompt="$ "

echo open ${host} ${port}
```

```
sleep 1
tout=${timeout}
while [ "${tout}" -ge 0 ]
do
    if tail -1 "${file}" 2>/dev/null | \
        egrep -e "login:" > /dev/null
    then
        echo "${login}"
        sleep 1
        tout=-5
        continue
    else
        sleep 1
        tout=`expr ${tout} - 1`
    fi
done

if [ "${tout}" -ne "-5" ]; then
    exit 1
fi

tout=${timeout}
while [ "${tout}" -ge 0 ]
do
    if tail -1 "${file}" 2>/dev/null | \
        egrep -e "Password:" > /dev/null
    then
        echo "${passwd}"
        sleep 1
        tout=-5
        continue
    else
        if tail -1 "${file}" 2>/dev/null | \
            egrep -e "${prompt}" > /dev/null
        then
            tout=-5
        else
            sleep 1
            tout=`expr ${tout} - 1`
        fi
    fi
done
```

```
> file1, , , , .  
"> ${file}" , , , , .
```

Updated 17 January 2024 04:16:56 by     (MeatDumpling)