文件权限进行判断
1. -r 有读的权限（read）
2. -w 有写的权限（write）
3. -x 有执行的权限（execute）
3. 按照文件类型进行判断
1. -f 文件存在并且是一个常规的文件（file）
2. -e 文件存在（existence）
3. -d 文件存在并是一个目录（directory）

多条件判断使用`&&`和`||`来连接：

* `&&` 表示前一条命令执行成功时，才执行后一条命令
* `||` 表示上一条命令执行失败后，才执行下一条命令

接下来进行实操

1. 创建一个`条件判断.sh`,粘贴以下内容

```
#! /bin/bash

# 注意[]两边要加空格

# 判断输入参数是否大于2，如果大于就echo输出一句话
[ $# -gt 2 ] && echo "参数个数大于2" || echo 参数个数少于等于2个""

[ -e /home/meifannao/ping.txt ] && echo "文件存在" || echo "文件不存在"
```

2. 打开一个终端窗口
3. 输入`chmod +x 条件判断.sh`，确保`ll`命令查看到`条件判断.sh`有可执行权限
4. `./条件判断.sh a1 a2 a3`执行脚本,后面的参数可以自己定义(包括参数个数和参数名称)
5. 查看执行结果

if条件判断即案例

在bash中需要遵顼以下格式编写if语句

```
if [ 条件判断式 ];then
    程序
fi
```

或者

```
if [ 条件判断式 ]
    then
        程序
fi
```

接下来我们编写一个案例，

* 如果年龄小于18输出”未成年“
* 如果年龄大于18并且小于35，输出”青年”
* 如果年龄大于35输出”中老年“

可以先自己尝试编写shell脚本.(同时判断两个条件用`-a`连接)

参考脚本:

```
#! /bin/bash

if [ $1 -lt 18 ]; then
    echo ”未成年”
elif [ $1 -ge 18 -a $1 -lt 35 ]; then
    echo ”青年”
else
    echo ”中老年”
fi
```

```

流程控制(for,while,case)
==========

case
----

在 Bash 中，我们使用 `case` 语句来进行多条件匹配，其格式如下：

```
case $变量名 in
    ”值1”)
        如果变量的值等于值1，则执行程序1
    ;;
    ”值2”)
        如果变量的值等于值2，则执行程序2
    ;;
    …省略其他分支…
    *)
        如果变量的值都不是以上的值，则执行此程序
    ;;
esac
```

* case行尾必须为单词`“in”`，每一个模式匹配必须以右括号`“）”`结束。
* 双分号`“;;”`表示命令序列结束，相当于C++中的break。
* 最后的`“*）”`表示默认模式，相当于C++中的default。

当我们需要匹配输入参数是否符合预期时，可以使用 `case` 语句。以下是示例代码：

```
#! /bin/bash

case $1 in
    ”start”)
        echo ”输入了start”
    ;;
```
```

```
    ”end”)
        echo ”输入了end”
    ;;
    *)
        echo ”输入了未知”
    ;;
esac
```

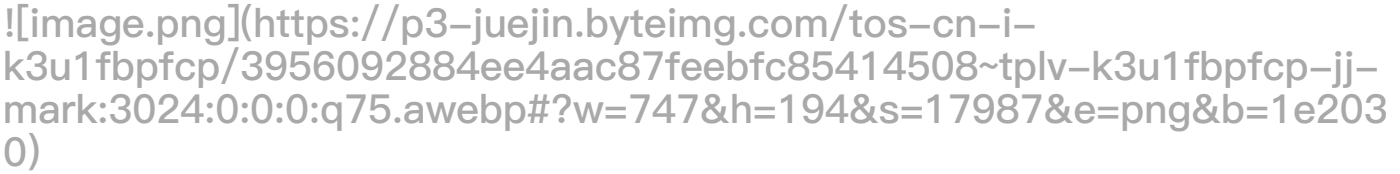
这个脚本根据输入参数的值进行不同的操作，如果输入参数为 `”start”`，则输出 `”输入了start”`；如果输入参数为 `”end”`，则输出 `”输入了end”`；如果输入参数为其他值，则输出 `”输入了未知”`。

使用示例

1. 将“mei nv”这个单词插入到wenben.txt第二行下，打印。

```
sed ’2a mei nv’ wenben.txt
```

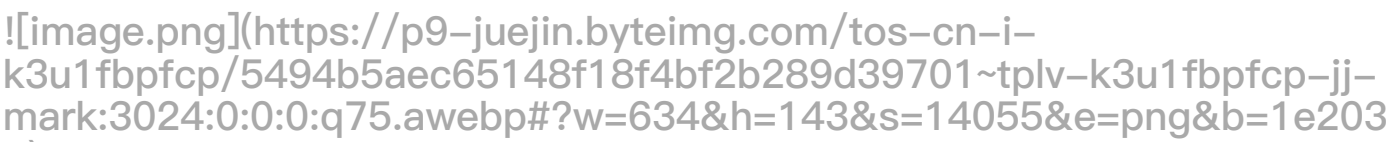
2代表加入到第二行的结尾(也就是第三行的开头),a代表新增



2. 删除wenben.txt文件所有包含xm的行

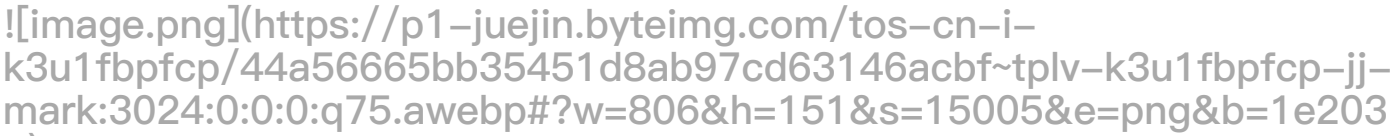
```
sed ’/xm/d’ wenben.txt
```

```



3. 将wenben.txt文件中的所有ping换成pang

```
sed 's/ping/pong/g' wenben.txt
```



awk
---

AWK的工作方式是逐行扫描输入文本文件，将每行分割成字段，并执行用户定义的操作。通常情况下，AWK的操作可以由用户通过命令行参数或者在一个单独的脚本文件中指定。

使用语法

```
awk options 'pattern {action}' file
```

* `options`：是一些选项，用于控制 `awk` 的行为。
* `pattern`：是用于匹配输入数据的模式。如果省略，则 `awk` 将对所有行进行操作。
* `{action}`：是在匹配到模式的行上执行的动作。如果省略，则默认动作是打印整行。
```

内建变量

1. FS：列分割符。指定每行文本的字段分隔符，默认为空格或制表位。与”-F”作用相同
2. NF：当前处理的行的字段个数。
3. NR：当前处理的行的行号（序数）。
4. $0：当前处理的行的整行内容。
5. $n：当前处理行的第n个字段（第n列）。
6. FILENAME：被处理的文件名。

以下是AWK的命令示例

```
awk ’{print}’ wenben.txt#输出所有内容
awk ’{print $0}’ wenben.txt#输出所有内容
```

```
awk ’NR==1,NR==3 {print}’ wenben.txt#输出第 1~3 行内容
awk ’NR==1;NR==3 {print}’ wenben.txt#输出第 1和第3 行内容
awk ’(NR>=1)&&(NR<=3) {print}’ wenben.txt#输出第 1~3 行内容
```

```
awk ’(NR%2)==1{print}’ wenben.txt #输出所有奇数行的内容
awk ’(NR%2)==0{print}’ wenben.txt#输出所有偶数行的内容
```

原文链接: https://juejin.cn/post/7359541702049251343