电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2023091602014

姓 名 张烨涛

（实验） 课程名称 操作系统原理与实践

理论教师 刘瑶

实验教师 刘瑶

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：张烨涛 学号：2023091602014 指导教师：刘瑶**

**实验地点：信软楼304 实验时间：2025.05.12**

1. **实验名称：**

实验二：文件状态测试

1. **实验学时：**

3h

1. **实验目的：**

熟悉UNIX的基本SHELL程序设计方法，包括：

1. 命令行参数检测
2. 变量设置
3. 文件状态检测与特定信息读取
4. 程序运行控制。
5. **实验原理：**

**4.1 任务逻辑梳理**

本次实验主要通过获取指定文件的状态信息，并在一定时间间隔内持续监测文件的状态，判断文件大小是否发生变化。主要会用到 shell 命令和语句，如while 循环、条件语句、文件操作等，相关逻辑如下——

 程序首先提示用户输入目标文件名，并验证文件是否存在。

 然后开始进行监测，记录初始文件大小的状态信息，并在每次循环中获取当前文件的状态信息。

 如果当前文件大小与前一个状态不同，则统计文件大小变化的次数和持续检测无变化次数，并输出相应的提示信息。

 同时，程序会不断循环检测文件是否发生变化，直到达到特定的条件（如文件大小变化的次数或持续检测无变化次数），程序才会退出检测并结束。

**4.2 ls -l 指令**

本程序主要会用到如下一些常见 Shell 指令，典型的如 ls -l，这个命令的作用是获取指定文件的详细状态信息（包括文件类型、访问权限、所有者、大小、修改时间等）。该命令将文件的所有详细信息都列出来，并以列表的形式进行显示，每一行对应一个文件或目录的详细信息。

其中，第一列的第一个字符代表文件类型——以 - 表示普通文件，d 表示

目录文件，l 表示符号链接等。

第一列的后 12 个字符每三个一组分别代表文件作者、文件所属组、其他人对于该文件的文件权限信息。

第三、四列代表文件所有者的用户名以及文件所属用户组的名称。

第五列即是本次实验所将要使用的文件大小。文件大小以字节为单位。

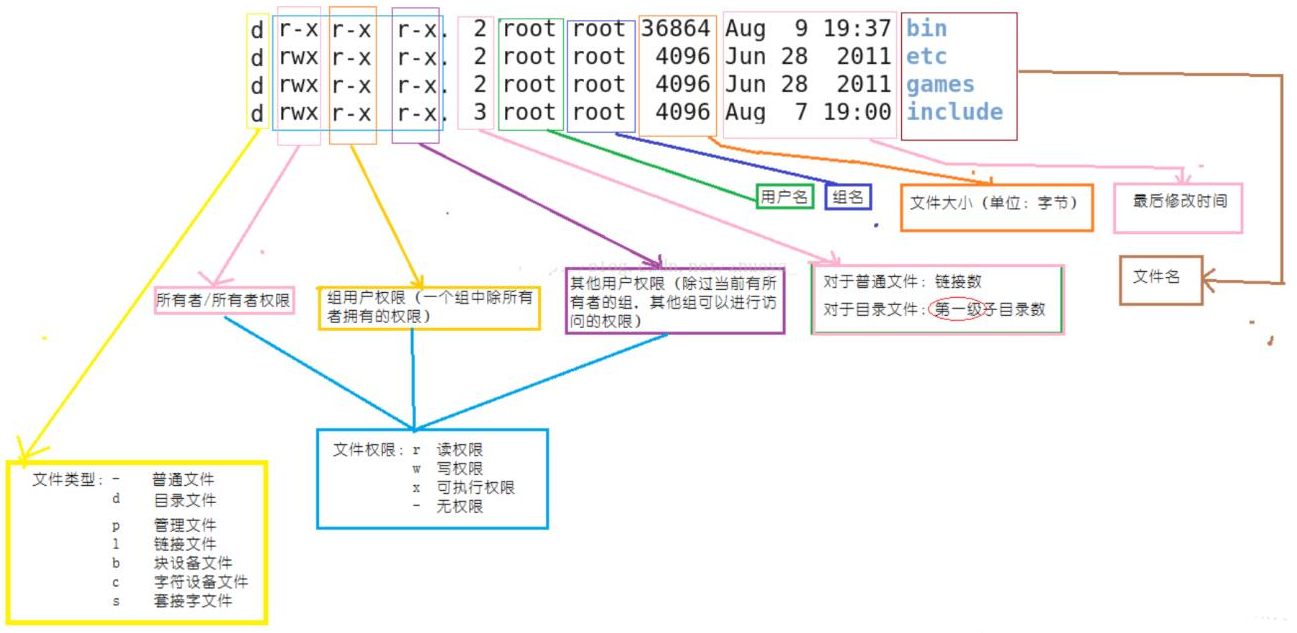


图 4-1 ls命令解析图

**4.3 test 指令**

test 指令是一条 Shell 命令，用于对某种条件进行判断和测试，并根据测试结果返回不同的状态值。在 Bash 中，test 命令也可以使用方括号 [] 或双括号[[]] 代替，它们的作用是相同的。

在本次实验中，将用 test 指令来判断程序是否存在且为普通的文件。我们会使用到-f 参数，这个参数是 test 命令的其中一个选项参数，用于检查一个文件是否为“普通文件”（regular file）。如果文件存在且是普通文件，则该选项返回真（0），否则返回假（非零整数）。使用格式如下：

test -f file\_path

[ -f file\_path ]

这里 test 命令的 -f 选项表示要检查的条件，“file\_path” 是需要被测试的文件路径。

**4.4 awk 指令**

在本次实验中，我们需要用到 awk 命令来截取状态信息中文件的大小并保存。

具体来讲，可以从 ls -l 命令的输出中，通过 awk 提取出第 5 列（也就是文件大小），以便在后续代码中进行处理和计算。

awk 实际上是一种流程处理器，它可以用来对文本文件进行处理。它是一种强大的文本解析工具，在 Linux/Unix 系统中广泛使用。通常，awk 可以用于文本分析、数据统计和排版中。

我们可以使用如下的语句完成输出——

awk '{print $1}'

在这里 print 是 awk 内置的命令，用单引号引起，用于在输出中打印文本或某些变量的值。$ 符号后面跟着的数字表示要操作的字段编号，而这个数字是从1 开始的。例如，{print $1} 表示打印第一个字段。字段以空格为分隔符。

1. **实验内容：**

编写一个SHELL程序，动态检测指定文件的状态信息，当文件的大小发生改变时，给出提示信息，并继续前进检测。当文件的大小的变化次数或持续检查无变化次数达到一定值时，退出检查，程序结束。

1. 程序运行时，如果找不到指定的文件，程序提示错误信息：

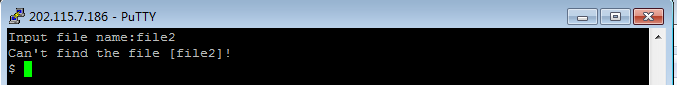


图 5-1 错误信息

1. 运行程序，输入要检查的文件名file1，在十次检查周期内，文件大小都一直不变的情况下，程序输出如下信息：

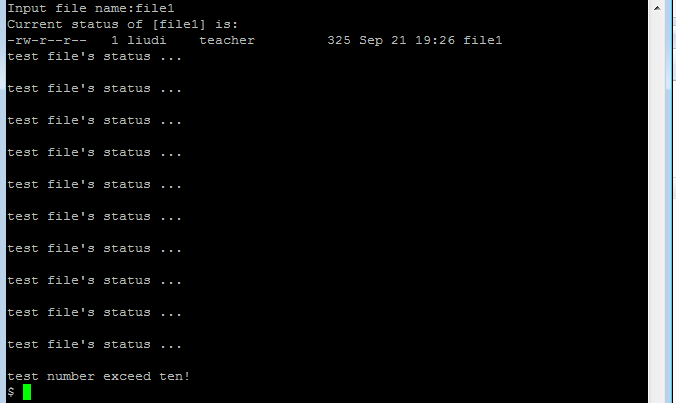


图 5-2 程序超时退出

1. 一旦当发现检查的文件大小改变时，程序显示如下信息，改变两次后停止检测：

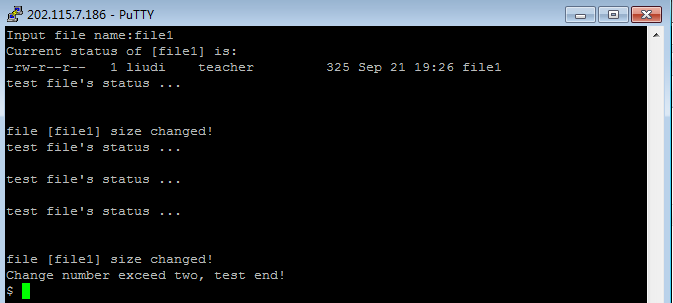


图 5-3 检测到大小改变

1. **实验器材（设备、元器件）：**

· 学生每人一台PC，安装WindowsXP/2000操作系统。

· 局域网络环境。

· 个人PC安装VMware虚拟机和Ubuntu系统。

1. **实验步骤：**

（1）清屏；

clear

（2）提示用户输入要检测其状态的文件名；

echo “Input name:”

read fname

（3）显示该文件的状态信息（提示：该状态信息可由命令ls –l 来得到），或找不到该文件时的错误提示；

用 if [ -f $name]

then

fi ls –l [ $fname]

（4）用cut命令，或用sed或awk命令来截取状态信息中文件的大小并保存；

ls –l [ $fname] | cut –c 36-46

oldstat=` ` #注意是单撇号

（5）每隔5秒钟检测一次该文件大小的信息，并与保存的文件原来的大小相比较；

while [ xxx -ne 2 ]

do

……

done

（6）如果文件大小未改变，则屏幕显示不变，并继续每隔5秒钟检测一次；

（7）如果文件大小已改变，则保存新的文件大小，并在屏幕上显示：

file [ filename ] size changed

（括号中的filename为本程序运行时用户输入的被检测的文件名）。程序继续每隔5秒钟检测一次文件的大小；

（8）程序循环执行5~7步的操作。当被检测的文件或者已累计改变了两次大小，或者已连续被检测了十次还未改变大小时，给出相应提示，然后清屏退出。

计数： test\_count=`expr $test\_count + 1`

change\_count=`expr $change\_count + 1`.

#注意，数学运算命令expr，两边用单撇号

1. **实验结果与分析（含重要数据结果分析或核心代码流程分析）**

本次实验主要实现了一个具备文件变化监测的功能的 shell 程序，通过读取文件信息，不断与上一次信息比较，完成文件大小变化的判断，并用 echo 语句输出相关内容。

为此，可以对代码流程进行如下的梳理——

**1.** 清屏，提示用户输入目标文件名，并验证该文件是否存在；

**2.** 若文件不存在，提示用户文件无法找到，继续循环使得用户输入合法文件名；

**3.** 若文件存在，转下一步；

**4.** 初始化监测信息，包括初始文件大小、检测次数、更改次数、未更改次数等；

**5.** 进入一个循环，不停地监听目标文件是否有变化；

**6.** 如果文件大小和上一次检测时的大小相同，则表示文件未发生更改，输出相关提示语句，检测次数、未更改次数自增；

**7.** 若文件大小和上次检测时的大小已经不同，那么表示文件已发生更改，输出相关提示语句，检测次数、更改次数自增，未更改次数清零；

**8.** 检测后，暂停运行（sleep）5s。

**9.**如果文件连续 10 次未发生更改，则表示文件已经稳定，结束监测；

**10.** 如果文件累计发生了 2 次更改，结束监测，退出循环。

源代码如下：

clear  
echo "Input file name:"  
read file\_name  
  
if [ -f "$file\_name" ]  
    then  
        echo "Starting status observer..."  
    else  
        echo "File \"$file\_name\" does not exist"  
        exit 1  
fi  
  
current\_size=`ls -l $file\_name | awk '{print $5}'`  
echo "Current file size: $current\_size"  
checkTime=0  
changeTime=0  
unchangeTime=0  
  
while true  
do  
    new\_size=`ls -l $file\_name | awk '{print $5}'`  
    checkTime=`expr $checkTime + 1`  
    if [ $new\_size -ne $current\_size ]  
        then  
            echo "File size changed from $current\_size to $new\_size at $checkTime th check"  
            changeTime=`expr $changeTime + 1`  
            unchangeTime=0  
            current\_size=$new\_size  
        else  
            echo "File size has not changed at $checkTime th check"  
            unchangeTime=`expr $unchangeTime + 1`  
    fi  
  
    if [ $unchangeTime -eq 5 ] || [ $changeTime -eq 2 ]  
        then  
            echo "Observer exited after $checkTime checks with file changed $changeTime times"  
            break  
    fi  
  
    sleep 2  
done

Shell 程序运行无需编译，可以直接运行。在此之前，我们使用 chmod 语句

为其给予其执行权限——

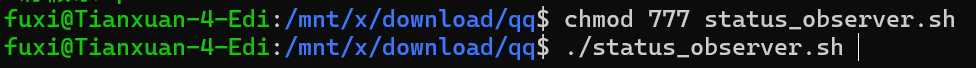


图 8-1 修改权限及运行程序

然后执行该程序，首先需要输入要监测的文件名，若该文件不存在，会提示用户并退出，否则，正常执行下一步。

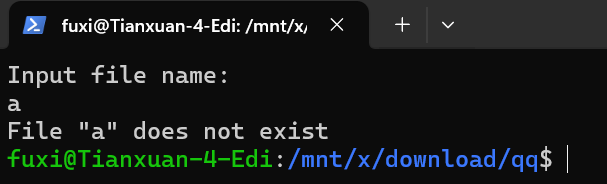


图 8-2 文件不存在

按要求，连续10次未检测到大小变化就退出，在此我为了方便，将其设置为5次。

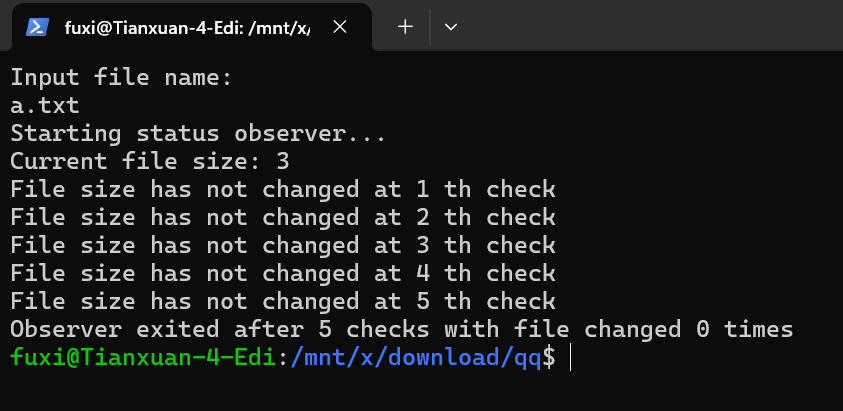


图 8-3 程序超时退出

当程序检测到文件大小发生了变动，会有相应的信息提示，若改变了两次，也退出。

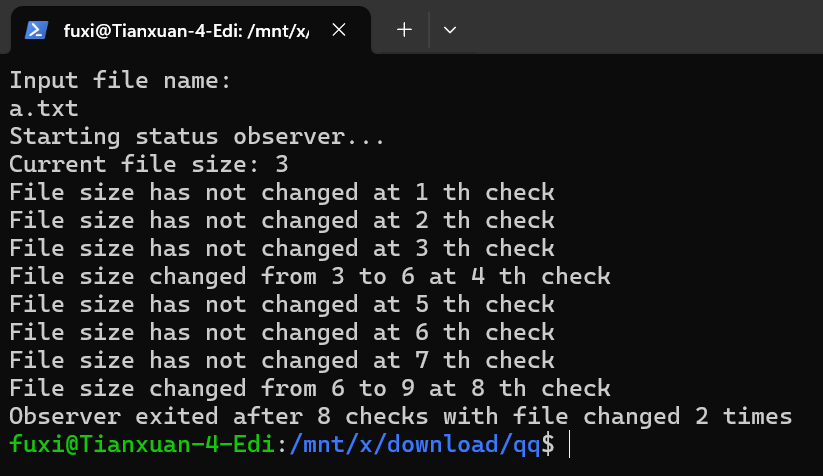


图 8-4 文件被检测到改变了两次

1. **总结及心得体会：**

本次实验主要围绕Linux系统下的文件监控脚本开发展开，通过编写能够自动检测文件尺寸变化并智能终止的Shell程序，使我对系统级编程有了新的认识。在具体实现过程中，首先利用系统内置命令构建了文件状态获取机制，通过定时轮询的方式持续采集目标文件元数据。每当检测周期到来时，程序会提取当前文件体积数值，并与历史记录进行比对分析，根据变化结果动态更新计数器。

在编程实践中，命令行工具的组合运用给我留下了深刻印象。原本分散的ls、cut等指令通过管道连接后，形成了高效的数据处理流水线，这种模块化思维极大提升了脚本的执行效率。特别是在解析文件详细信息时，发现不同解析方式对结果精度的影响，例如使用awk直接提取字段比层层截取更可靠。调试过程中遇到的变量作用域问题也让我意识到，Shell脚本虽然灵活，但对代码结构的严谨性要求并不低于其他编程语言。

此次实验不仅巩固了Shell编程基础，更培养了系统层面的问题分析能力。衷心感谢老师在实验过程中给予的关键性指导，那些关于信号捕获和进程管理的建议，为我打开了Linux系统编程的新视野，这份经验将持续影响未来的技术学习路径。

1. **对本实验过程及方法、手段的改进建议：**

**报告评分：**

**指导教师签字：**