电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2021090922022

姓 名 韩政君

（实验） 课程名称 **操作系统原理与实践实验**

理论教师 赵太银

实验教师 冯月

**电子科技大学教务处制表**

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：韩政君 学号：2021090922022 指导教师：冯月**

**实验地点：信软楼西304 实验时间：2023.05.24**

**一、实验名称：**文件状态测试

**二、实验学时：**4学时

**三、实验目的：**

熟悉UNIX的基本SHELL程序设计方法，包括：

命令行参数检测

变量设置

文件状态检测与特定信息读取

程序运行控制

**四、实验原理：**

编写一个SHELL程序，动态检测指定文件的状态信息，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。

**五、实验内容：**

（1）清屏；

（2）提示用户输入要检测其状态的文件名；

（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；

（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；

（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；

（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。

**六、实验器材（设备、元器件）：**

PC计算机，操作系统：Ubuntu

**七、实验步骤：**

1. 使用了Windows的Linux子系统（Ubuntu内核）。

2.使用vscode远程连接wsl进行实验操作

3.程序实现

4. 编译、调试、运行

5. 获取实验结果并分析

**八、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）**

程序代码

#!/bin/bash

clear

echo "Please input the filename to be detected:"

read filename

if [ ! -e $filename ]; then

    echo "File ${filename} not found!"

    exit 1

fi

echo "Current status of ${filename}:"

ls -l $filename

filesize=`ls -l $filename | cut -d " " -f 5`

unchanged=0

changed=0

while true; do

    sleep 5

    if [ $filesize -eq `ls -l $filename | cut -d " " -f 5` ]; then

        unchanged=`expr $unchanged + 1`

        echo "File ${filename} size unchanged."

        if [ $unchanged -eq 10 ]; then

            echo "The file has not been changed for 10 times."

            break

        fi

    else

        filesize=`ls -l $filename | cut -d " " -f 5`

        echo "File ${filename} size changed."

        changed=`expr $changed + 1`

        unchanged=0

        if [ $changed -eq 2 ]; then

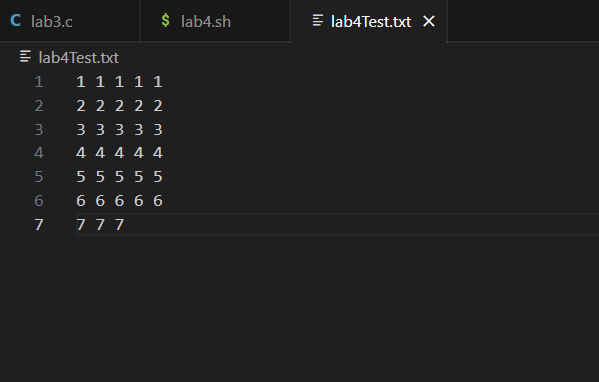
            echo "The file has been changed for 2 times."

            break

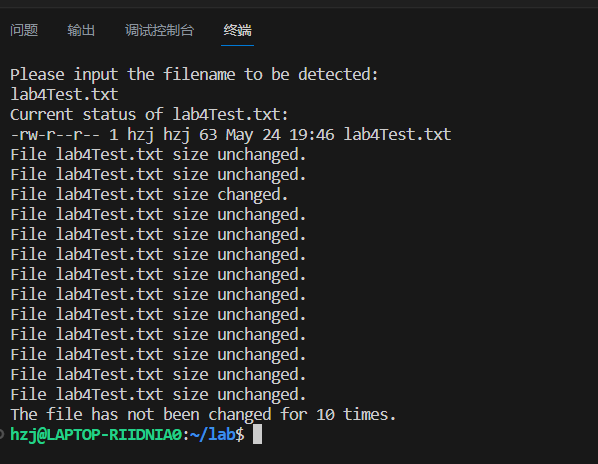
        fi

    fi

done

 测试文件

测试结果



**九、总结及心得体会：**

通过实现文件状态检测的程序，我深刻理解了文件属性的测试原理和方法，学会了如何使用相关的函数实现文件状态检测。我还学会了如何使用Linux系统提供的文件操作函数来打开、读取和关闭文件，以及如何使用文件状态检测来判断文件的类型、大小等属性。这对我以后的程序设计和开发都非常有帮助。

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：韩政君 学号：2021090922022 指导教师：冯月**

**实验地点：信软楼西304 实验时间：2023.05.31**

**一、实验名称：**用户登陆动态检测

**二、实验学时：**4学时

**三、实验目的：**

使学生熟悉一种简单的指定用户的监测方法，熟悉UNIX的基

本SHELL程序设计方法和使用技巧，包括：

① 命令行参数检测

② 用户变量

③ while循环控制

④ 暂停进程（sleep）

**四、实验原理**

编写一个用户监测程序usr\_monitor, 其运行格式为：

usr\_monitor username

其中username是用户指定的任意一个用户名。程序运行时首先列出当前系统中的已登录用户的名单，再检查指定用户是否已登录。如果已登录，则显示相应信息；如果未登录，则等待该用户登录，直到指定用户登录进入系统为止。

**五、实验内容：**

（1） 建立shell程序usr\_monitor

（2） 根据变量$#的值检查命令行上是否有一个用户名，如果有则继续运行；如果没有或多于一个用户名则提示运行格式：

Usage: usr\_monitor username

（3） 运行who命令，并将运行结果中的用户名字段截取下来，保存到一个用户变量中。

（4） echo命令显示该变量的值，即当前系统中的用户名单。

（5） 判断用户名单中是否包含命令行上指定的用户，如果有则显示：

user [username] is logon

并退出程序。

（6） 如果用户名单中没有包含命令行上指定的用户，则显示：

waiting user [username] …

光标停留在省略号后面。

（7） 使用while循环，用sleep命令每隔5秒钟检查指定用户是否已登录系统。检测方法与步骤3相同。

（8） 如果指定用户未登录，则一直等待下去；如果指定用户已登录，则在省略号后面接着显示：

[username] is log on

退出循环，并结束程序。

**六、实验器材（设备、元器件）：**

PC计算机，操作系统：Ubuntu

**七、实验步骤：**

1. 使用了Windows的Linux子系统（Ubuntu内核）。

2.使用vscode远程连接wsl进行实验操作

3.程序实现

4. 编译、调试、运行

5. 获取实验结果并分析

**八、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）**

程序代码

#!/bin/bash

# 检查命令行参数

if [ $# -ne 1 ]; then

  echo "Usage: $0 username"

  exit 1

fi

# 获取当前已登录用户列表

user\_list=$(who | awk '{print $1}')

# 显示当前已登录用户列表

echo "Current users: $user\_list"

# 检查指定用户是否已登录

if echo "$user\_list" | grep -q "$1"; then

  echo "User $1 is logged in"

  exit 0

else

  echo "Waiting for user $1 ..."

fi

# 持续检查指定用户是否已登录

while true; do

  sleep 5

  user\_list=$(who | awk '{print $1}')

  if echo "$user\_list" | grep -q "$1"; then

    echo "User $1 is logged in"

    exit 0

  else

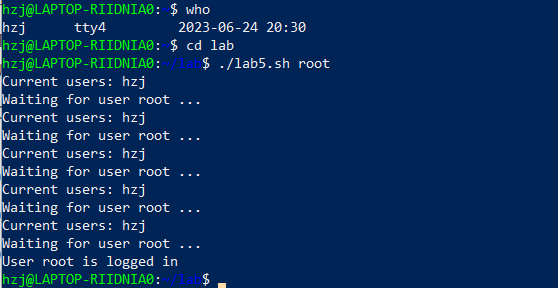
    echo "Current users: $user\_list"

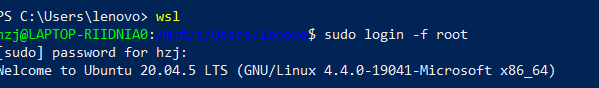
    echo "Waiting for user $1 ..."

  fi

done

运行结果





**九、总结及心得体会**

在实现用户登陆动态检测的程序过程中，我深刻理解了Linux系统中用户登陆的原理和方法，以及如何使用相关的函数和工具来实现用户登陆的动态检测。

通过实现用户登陆动态检测的程序，我深刻理解了Linux系统中用户登陆的原理和方法，以及如何使用相关的函数和工具来实现用户登陆的动态检测。我还学会了如何使用C语言编写程序来实现用户登陆动态检测，并且学会了如何使用Linux系统提供的命令行工具来测试和调试程序。

**电 子 科 技 大 学**

**实 验 报 告**

**学生姓名：韩政君 学号：2021090922022 指导教师：冯月**

**实验地点：信软楼西304 实验时间：2023.06.07**

**一、实验名称：**系统命令的实现

**二、实验学时：**4学时

**三、实验目的：**

掌握Linux目录操作方法，包括打开目录、关闭目录、读取

目录文件

② 掌握Linux文件属性获取方法，包括三个获取Linux文件属性

的函数、文件属性解析相关的宏

③ 掌握POSIX与ANSI C文件I/O操作方法，包括打开文件、关闭

文件、创建文件、读写文件、定位文件

**四、实验原理**

UNIX/Linux中C语言应用程序的启动函数是main

操作系统通过C启动例程来启动C程序，启动例程会从标准输入获取应用程序的命令行参数，并且将这些参数传递给main函数

**五、实验内容：**

编写Linux应用程序，利用POSIX API在Linux系统上编写应用程序，实现cp命令的部分功能，主要实现文件到文件的复制功能（执行“cp 源文件路径 目标文件路径”实现的功能）。

在实现基本功能实现的基础上，针对各种特殊情况和边界条件等进行流程的完善与优化，包括：

a)目标文件存在时给出“是否覆盖”的提示信息,用户可以选择覆盖目标文件或者将已经存在的目标文件和源文件实现合并（在目标文件的尾部实现追加写入）

b)源文件不存在时给出出错提示信息

c)源文件是目录文件时给出错误提示信息

**六、实验器材（设备、元器件）：**

PC计算机，操作系统：Ubuntu

**七、实验步骤：**

1. 使用了Windows的Linux子系统（Ubuntu内核）。

2.使用vscode远程连接wsl进行实验操作

3.程序实现

4. 编译、调试、运行

5. 获取实验结果并分析

**八、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）**

程序代码

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <dirent.h>

#include <fcntl.h>

#include <string.h>

#include <sys/stat.h>

#include <sys/time.h>

// 首先实现文件到文件的拷贝

int copyfile(char \*spath, char \*dpath)

{

    int sfd, dfd;

    if ((sfd = open(spath, O\_RDONLY)) == -1) // 只读

    {

        printf("Source file does not exist!\n");

        return -1;

    }

    if (dfd = open(dpath, O\_WRONLY) == -1) // 目标文件不存在则创建

    {

        if ((dfd = open(dpath, O\_WRONLY | O\_CREAT, 0644)) == -1) // 只写，不存在则创建

        {

            printf("Destination file create error!\n");

            return -1;

        }

    }

    else

    {

        printf("File %s already exists, overwrite it(1) or skip it(else)?\n", dpath);

        char choice;

        while ((choice = getchar()) == '\n')

            ;

        if (choice == '1')

        {

            if ((dfd = open(dpath, O\_WRONLY | O\_TRUNC, 0644)) == -1) // 覆盖文件

            {

                printf("Destination file create error!\n");

                return -1;

            }

        }

        else

            return 0; // 跳过

    }

    char buf[1024]; // 缓冲区

    int n;

    while ((n = read(sfd, buf, 1024)) > 0) // 读取源文件,到达文件尾部时返回0

    {

        if (n == -1 || (write(dfd, buf, n)) == -1) // 写入目标文件

        {

            printf("Error copying file from %s to %s\n", spath, dpath);

            return -1;

        }

    }

    close(sfd);

    close(dfd);

    return 0;

}

// 实现文件属性的判断

int isdir(char \*path)

{

    struct stat buf;

    if (stat(path, &buf) == -1)

    {

        printf("Error getting file status!\n");

        return -1;

    }

    return S\_ISDIR(buf.st\_mode);

}

// 然后实现目录的遍历及复制

int walkdir(char \*spath, char \*dpath)

{

    DIR \*sdir;

    struct dirent \*sdp;

    if ((sdir = opendir(spath)) == NULL)

    {

        printf("Source directory does not exist!\n");

        return -1;

    }

    while ((sdp = readdir(sdir)) != NULL)

    {

        if (strcmp(sdp->d\_name, ".") == 0 || strcmp(sdp->d\_name, "..") == 0) // 忽略.和..

            continue;

        char sfile[1024], dfile[1024];

        // 拼接源文件和目标文件的路径

        strcpy(sfile, spath);

        strcat(sfile, "/");

        strcat(sfile, sdp->d\_name);

        strcpy(dfile, dpath);

        strcat(dfile, "/");

        strcat(dfile, sdp->d\_name);

        if (isdir(sfile))

        {

            // 如果目录不存在则创建

            DIR \*temp = opendir(dfile);

            if (temp != NULL)

                closedir(temp);

            else if (mkdir(dfile, 0777) == -1) // 创建目录

            {

                printf("Error creating directory %s\n", dfile);

                return -1;

            }

            if (walkdir(sfile, dfile) == -1) // 递归遍历子目录

                return -1;

        }

        else

        {

            if (copyfile(sfile, dfile) == -1) // 拷贝文件

                return -1;

        }

    }

    closedir(sdir);

    return 0;

}

// 主函数实现参数读取

int main(int argc, char \*argv[])

{

    if (argc != 3)

    {

        printf("Usage: %s source\_path destination\_path\n", argv[0]);

        return -1;

    }

    // 计算拷贝时间

    struct timeval start,end;

    gettimeofday(&start, NULL );

    if ((walkdir(argv[1], argv[2])) == 0)

        printf("Copy successfully!\n");

    else

    {

        printf("Copy failed!\n");

        return -1;

    }

    gettimeofday(&end, NULL );

    long timeuse = 1000000 \* ( end.tv\_sec - start.tv\_sec ) + end.tv\_usec - start.tv\_usec;

    printf("time = %ld us\n", timeuse);

    return 0;

}

运行结果



**九、总结及心得体会**

在系统命令实现部分,我熟练掌握了Linux系统中文件和目录的操作API,包括打开、关闭、创建、移动、复制等功能。我通过打开文件读取源文件内容并写入目标文件实现了文件拷贝,通过递归遍历目录结构实现了目录拷贝。同时,我对于C语言在Linux下的编译运行调试更加熟练了。

**报告评分：**

**指导教师签字：**