**武汉大学国家网络安全学院**

**实验报告**

**课程名称 操作系统**

**专业年级 信安21级**

**姓名学号 赵伯俣2021302181156**

**协 作 者 无**

**实验学期 2022-2023　 学年 第二 学期**

**课堂时数 32 课外时数**

**填写时间 2023 年 4 月 23 日**

一、实验目的

设计一个多线程应用程序，以判定数度谜题的解决是否有效

二、实验环境

vscode编程软件用于实验代码的编写和调试。

三、实验源代码

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

void \*column\_checker(void \*param);

void \*row\_checker(void \*param);

void \*block\_checker(void \*param);

//示例数独

int example[9][9] = {

    { 6, 2, 4, 5, 3, 9, 1, 8, 7},

    { 5, 1, 9, 7, 2, 8, 6, 3, 4},

    { 8, 3, 7, 6, 1, 4, 2, 9, 5},

    { 1, 4, 3, 8, 6, 5, 7, 2, 9},

    { 9, 5, 8, 2, 4, 7, 3, 6, 1},

    { 7, 6, 2, 3, 9, 1, 4, 5, 8},

    { 3, 7, 1, 9, 5, 6, 8, 4, 2},

    { 4, 9, 6, 1, 8, 2, 3, 3, 3},

    { 2, 8, 5, 4, 7, 3, 9, 1, 3}};

//判断数独是否有效数组，初始值为零，每个线程控制一位，为零则对应的线程区域无效

int judge[11] = {1,1,1,1,1,1,1,1,1,1,1};

typedef struct

{

    int row;

    int column;

} parameters;

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

{

    parameters \*data = (parameters \*)malloc(sizeof(parameters));

    data->row = 1;

    data->column = 1;

    //创建线程id

    pthread\_t tid\_row;

    pthread\_t tid\_column;

    pthread\_t tid\_subfield;

    //创建线程

    pthread\_create(&tid\_row, NULL, row\_checker, (void \*)data);

    pthread\_create(&tid\_column, NULL, column\_checker, (void \*)data);

    pthread\_create(&tid\_subfield, NULL, block\_checker, (void \*)data);

    //线程阻塞

    pthread\_join(tid\_row, NULL);

    pthread\_join(tid\_column, NULL);

    pthread\_join(tid\_subfield, NULL);

    printf("本次测试的数独为:\n");

    for (int i = 0; i < 9;i++)

    {

        for (int j = 0; j < 9;j++)

        {

            printf("%d ", example[i][j]);

        }

        printf("\n");

    }

    int result = 1;       //有效与否的总结果

    //对数组中的每个值求乘积不为零则所有检查均有效

    for (int k = 0; k <= 10;k++)

    {

        //printf("%d", judge[k]);

        result \*= judge[k];

    }

    if (result == 1)

    {

        printf("数独猜谜有效\n");

    }

    else

    {

        printf("数独猜谜无效\n");

    }

    return 0;

}

//列检查函数

void \*row\_checker(void \*params)

{

    parameters \*temp\_row;

    temp\_row = (parameters \*)params;

    for (int i = temp\_row->row; i <= 9; i++)

    {

        int flag[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0,0};  //每检查一个数字就在flag数组对应位置填入1，若重复填写则该区域发生错误

        for (int j = temp\_row->column; j <= 9; j++)

        {

            if (flag[example[i-1][j-1]] == 0)  //检查正确

            {

                flag[example[i-1][j-1]] = 1;

            }

            else if (flag[example[i-1][j-1]] == 1)//出现错误

            {

                printf("第%d行不满足\n", i);//输出发生错误对应的部分

                judge[0]=0;

                break;

            }

        }

    }

    pthread\_exit(0);

}

//行检查函数

void \*column\_checker(void \*params)

{

    parameters \*temp;

    temp = (parameters \*)params;

    for (int i = temp->column; i <= 9; i++)

    {

        int flag[10] = {0,0, 0, 0, 0, 0, 0, 0, 0, 0};    //每检查一个数字就在flag数组对应位置填入1，若重复填写则该区域发生错误

        for (int j = temp->row; j <= 9; j++)

        {

            if (flag[example[j-1][i-1]] == 0)//检查正确

            {

                flag[example[j-1][i-1]] = 1;

            }

            else if (flag[example[i-1][j-1]] == 1)//出现错误

            {

                printf("第%d列不满足\n", i);//输出发生错误对应的部分

                judge[1]=0;

                break;

            }

        }

    }

    pthread\_exit(0);

}

//块检查函数

void \*block\_checker(void \*params)

{

    parameters \*temp;

    temp = (parameters \*)params;

    int block = 1;    //当前正在检查的块编号

    for (int i = temp->row; i <= 9; i += 3)

    {

        for (int j = temp->column; j <= 9; j += 3)  //确定要检查哪一块

        {

            int exit = 0;

            int flag[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};//每检查一个数字就在flag数组对应位置填入1，若重复填写则该区域发生错误

            for (int a = 0; a < 3; a++)

            {

                for (int b = 0; b < 3; b++)  //确定检查该块的哪一个位置

                {

                    if (flag[example[i + a-1][j + b-1]] == 0)//检查正确

                    {

                        flag[example[i + a-1][j + b-1]] = 1;

                    }

                    else if (flag[example[i + a-1][j + b-1]] == 1)//检查错误

                    {

                        judge[block+1]=0;

                        printf("第%d块不满足\n", block);//输出发生错误的部位

                         exit = 1;

                        break ;

                    }

                }

                if(exit==1)

                {

                    break;

                }

            }

            block++;

        }

    }

    pthread\_exit(0);

}

四、实验原理

1.pthread.h库函数的应用。

在本次实验中使用pthread库函数中的函数完成了多线程应用程序的设计，以下对于使用到的函数进行说明。

（1）pthread\_creat函数。该函数的原型为函数原型：int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*start\_routine)(void \*), void \*arg)

其中thread变量代表线程的句柄，在使用之前需要先定义为pthread\_t类型的变量，将变量的地址传递进入函数中去进而得到系统创建好的线程句柄。

其中attr变量代表线程的属性，在本次实验中使用默认的属性所以直接传递为NULL。

start\_routine变量代表线程中所执行的函数，将其所要运行的函数名传递入函数即可。

arg变量代表线程的参数，即为主线程传递给子线程的参数，在本次实验中主线程将parameters类型的变量data传入子线程从而进行数独谜题的判断。

（2）pthread\_join函数。该函数的原型为：

int pthread\_join(pthread\_t thread,void \*\*retval);

其中的参数thread代表着等待线程的线程ID，retval代表该线程的退出状态，在本次实验中退出状态设置为NULL，该函数的功能为阻塞等待线程退出并可以获得线程的退出状态。

（3）pthread\_exit函数。该函数的原型为：

void pthread\_exit(void \*retval);该函数的功能为将单个的线程进行退出函数中传递的参数为该线程的返回状态。在本次实验中将线程的返回状态设置为0。

2.行（列）检查函数row\_checker和column\_checker。

在该函数中首先接受主线程传递过来的data变量将其存入temp变量中，设置一个数组flag然后通过调用两轮循环便历每一行（列），将遍历到的数字的对应位置flag设置为1，如果在该行（列）遍历过程中发生了数字的重复即两次遍历到了重复的数字则输出发生错误的行（列）号并将judge数组对应线程的值设置为0，表示该线程判定该区域无效。并且使用break跳出循环。

3.块检查函数block\_checker。

在块检查函数中所用到的思路与行列检查函数基本相同，不同点为在函数中i，j变量表示当前检查到的块的第一个数字的位置坐标。a，b表示具体检查哪一个数字。变量block代表当前检查到的块标号。在一个块中出现数字重复后会输出出现错误的块号并break退出循环，进行下一块的检查。

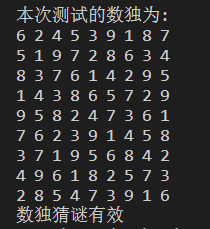
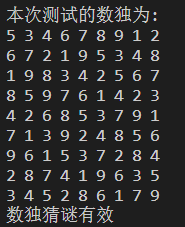
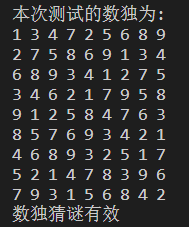
4.main函数。

在该程序中定义了数组judge，该数组的值对于每一个线程都是可见的，这个数组中的每一个索引i都对应着第i个工作线程的结果。如果对应线程对其设置为1，则该线程所检查的区域是合法的，反之则将对应的区域设置为0。

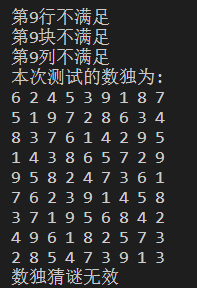
在程序末尾将judge数组中的各位进行相乘，当且仅当各位不为零的情况下才会输出数度谜题有效，反之输出数独谜题无效。实现对数独谜题的判断。

五、实验结果

1.判断课本中给出的实例和其他实例数独结果如下：

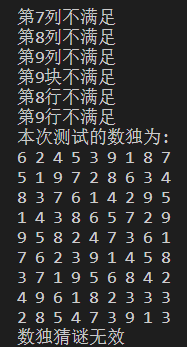
  

2.在修改正确数独解密一个数字之后观察报错。



将正确的数独解(8，8)位置的数字6改为3后程序报错第9行第9列和第9块不满足，最终输出数独猜谜无效。

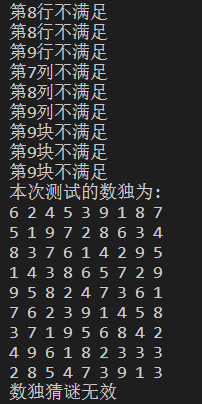
3.修改更多数字之后观察输出变化。



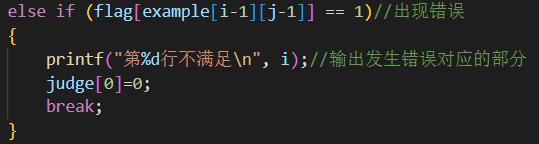
在将数组中的若干位置的数字修改之后，程序仍然会列出所有发生错误的区域并且输出该数独猜谜无效。

六、实验中出现的bug及其修改方式

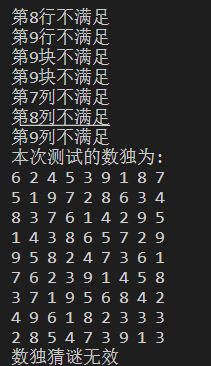
1.在输出不满足报错时会出现多个重复报错。输出结果如下所示。在产生数字重复错误时并不会退出循环，只会继续运行。导致若出现多于两次的数字重复时会出现重复报错。



解决方法：在输出不满足结果之后加入语句break跳出当前的循环即可消除重复输出。如下图所示



2.在输出块不满足报错时会输出多个重复的报错。如下所示在程序输出时出现了两个“第9块不满足”的输出。在break跳出循环时仅能跳出内层循环，使得在判断下一行时如果继续遇到数字重复会重复输出。



解决方法：将for循环中的代码修改为如下形式即可解决问题。设置标志exit使得能够在跳出时能够跳出双层循环。

