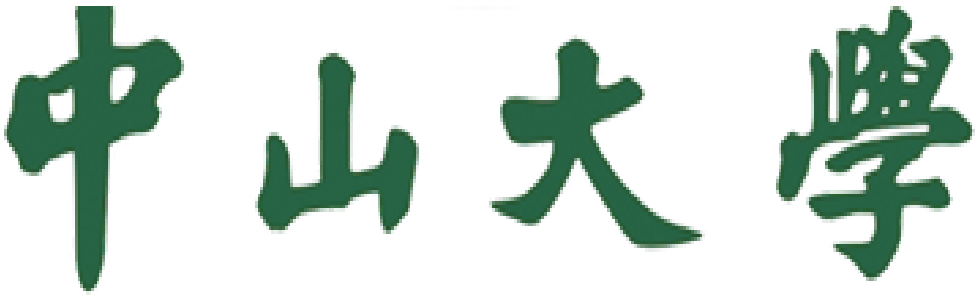


SUN YAT-SEN UNIVERSITY

# 本科生实验报告

实验课程:操作系统原理

实验名称:操作系统编程大作业

专业名称:计算机科学与技术(人工智能与大数据方向)

学生姓名:刘卓逸

学生学号:21307303

实验地点:东校园实验中心大楼D503

# 1.实验要求

针对下列问题提出解决方案，将相应的源代码和说明文件打包后提交。

- 题目一：课本p134，项目一“数独解决方案验证器”
- 题目二：课本p105，3.14“Collatz 猜想问题”

# 2.实验过程

## Assignment 1 数独解决方案验证器

### 实验内容

设计一个**多线程**应用程序，以判定数独谜题的解决方案是否有效

### 实验原理

使用C++语言，基于pthread.h库完成

主线程创建3个子线程分别判断所有的行、列、块是否有效，

3个子线程又分别创建9个子线程判断每个行、列、块是否有效

### 实验代码

Main函数：输入与输出

```
int main()
{
    int a[9][9];
    for (int i=0;i<9;i++){
        for (int j=0;j<9;j++)
            scanf("%d",&a[i][j]);
    }
    if (parallel_check(a)){
        printf("Outside call: True");
    }
    else{
        printf("Outside call: False");
    }
```

```
    return 0;
}
```

pthread只能传递给子线程一个 void*参数，考虑用一个 struct args来传递所有需要的参数

```
struct args{
    int (*a)[9]; // 数独
    bool *ans; // 要将结果返回的地方
    int rcb; // 当前任务是判断行还是列还是块
    int id; // 当前任务是判断第几个行列块
    args(int (*_a)[9]=nullptr,bool *_ans=nullptr,int _rcb=0,int
_id=0):a(_a),ans(_ans),id(_id),rcb(_rcb){}
};
const char *(RCB[3])={"Row","Column","Block"};
```

主判断函数,创建三个子线程，三个子线程都完成后将结果汇总

```
bool parallel_check(int a[][9]){
    // 参数准备
    bool ans[3]={0};
    args arg[3]={args(a,ans,0),args(a,ans+1,1),args(a,ans+2,2)}; //分别是行、列、
块，把结果返回到ans[3]中
    // 从创建线程，传递参数
    pthread_t c_cols, c_rows, c_blks;
    int iret_rows = pthread_create(&c_rows, NULL, check_rcbs, (void*)(&arg[0]));
    int iret_cols = pthread_create(&c_cols, NULL, check_rcbs, (void*)(&arg[1]));
    int iret_blks = pthread_create(&c_blks, NULL, check_rcbs, (void*)(&arg[2]));
    // 等待线程结束
    pthread_join(c_cols, NULL);
    pthread_join(c_rows, NULL);
    pthread_join(c_blks, NULL);
    // 整理结果
    bool ansX=!(ans[0]||ans[1]||ans[2]);
    printf("Checking %s\n",ansX?"Succeed":"Fail");
    return ansX;
}
```

行、列、块的大判断函数，创建9个子线程

```
void *check_rcbs(void *_arg){
    // 参数传输
    args* arg=(args*)_arg;
    int (*a)[9]=arg->a;
    bool *ans=arg->ans;
    int rcb=arg->rcb;
    // 参数准备
```

```
    bool ans2[9]={0};
    args arg2[9];
    for (int i=0;i<9;i++)
        arg2[i]=args(a,ans2+i,rcb,i); //rcb将大任务传递给小任务,i切割小任务
    // 主模块
    pthread_t c_rcb[9];
    int iret_rcb[9];
    for (int i=0;i<9;i++)
        iret_rcb[i]=pthread_create(&c_rcb[i], NULL, check_rcb, (void*)(&arg2[i]));
    for (int i=0;i<9;i++)
        pthread_join(c_rcb[i], NULL);
    // 合并
    bool ansX=0;
    for (int i=0;i<9;i++)
        ansX|=ans2[i];
    if (ansX)
        printf("-- Checking %s Fail\n",RCB[rcb]);
    else
        printf("-- Checking %s Succeed\n",RCB[rcb]);
    *ans=ansX;
    return nullptr;
}
```

行、列、块的小判断函数，判断单独的行、列、块是否满足 由于三个函数结果几乎一致，只需在不同的地方用if来区分

```
void *check_rcb(void *_arg){
    // 参数传输
    args* arg=(args*)_arg;
    int (*a)[9]=arg->a;
    bool *ans=arg->ans;
    int rcb=arg->rcb;
    int i=arg->id;
    // 主模块
    bool hash[10]={0};
    for (int j=0;j<9;j++){
        //区分rcb
        int temp=(rcb==0)?a[i][j]:
                 (rcb==1)?a[j][i]:
                 a[(i/3)*3+j/3][(i%3)*3+j%3];
        if (temp<1||temp>9||hash[temp-1]){
            *ans=1;
            printf("---- checking %s %d Fail\n", RCB[rcb] , i);
            return nullptr;
        }
        else{
            hash[temp-1]=1;
        }
    }
    printf("---- checking %s %d Succeed\n",RCB[rcb],i);
    return nullptr;
}
```

完整代码

```
#include <iostream>
#include <stdio.h>
#include <pthread.h>

struct args{
    int (*a)[9]; // 数独
    bool *ans; // 要将结果返回的地方
    int rcb; // 当前任务是判断行还是列还是块
    int id; // 当前任务是判断第几个行列块
    args(int (*_a)[9]=nullptr,bool *_ans=nullptr,int _rcb=0,int
_id=0):a(_a),ans(_ans),id(_id),rcb(_rcb){}
};
const char *(RCB[3])={"Row","Column","Block"};

void *check_rcb(void *_arg){
    // 参数传输
    args* arg=(args*)_arg;
    int (*a)[9]=arg->a;
    bool *ans=arg->ans;
    int rcb=arg->rcb;
    int i=arg->id;
    // 主模块
    bool hash[10]={0};
    for (int j=0;j<9;j++){
        int temp=(rcb==0)?a[i][j]:
                 (rcb==1)?a[j][i]:
                 a[(i/3)*3+j/3][(i%3)*3+j%3];
        if (temp<1||temp>9||hash[temp-1]){
            *ans=1;
            printf("---- checking %s %d Fail\n", RCB[rcb] , i);
            return nullptr;
        }
        else{
            hash[temp-1]=1;
        }
    }
    printf("---- checking %s %d Succeed\n",RCB[rcb],i);
    return nullptr;
}

void *check_rcbs(void *_arg){
    // 参数传输
    args* arg=(args*)_arg;
    int (*a)[9]=arg->a;
    bool *ans=arg->ans;
    int rcb=arg->rcb;
    // 参数准备
    bool ans2[9]={0};
    args arg2[9];
    for (int i=0;i<9;i++)
        arg2[i]=args(a,ans2+i,rcb,i);
    // 主模块
    pthread_t c_rcb[9];
```

```
    int iret_rcb[9];
    for (int i=0;i<9;i++)
        iret_rcb[i]=pthread_create(&c_rcb[i], NULL, check_rcb, (void*)(&arg2[i]));
    for (int i=0;i<9;i++)
        pthread_join(c_rcb[i], NULL);
    // 合并
    bool ansX=0;
    for (int i=0;i<9;i++)
        ansX|=ans2[i];
    if (ansX)
        printf("-- Checking %s Fail\n",RCB[rcb]);
    else
        printf("-- Checking %s Succeed\n",RCB[rcb]);
    *ans=ansX;
    return nullptr;
}

bool parallel_check(int a[][9]){
    // 参数准备
    bool ans[3]={0};
    args arg[3]={args(a,ans,0),args(a,ans+1,1),args(a,ans+2,2)};
    // 从创建线程，传递参数
    pthread_t c_cols, c_rows, c_blks;
    int iret_rows = pthread_create(&c_rows, NULL, check_rcbs, (void*)(&arg[0]));
    int iret_cols = pthread_create(&c_cols, NULL, check_rcbs, (void*)(&arg[1]));
    int iret_blks = pthread_create(&c_blks, NULL, check_rcbs, (void*)(&arg[2]));
    // 等待线程结束
    pthread_join(c_cols, NULL);
    pthread_join(c_rows, NULL);
    pthread_join(c_blks, NULL);
    // 整理结果
    bool ansX=!(ans[0]||ans[1]||ans[2]);
    printf("Checking %s\n",ansX?"Succeed":"Fail");
    return ansX;
}

int main()
{
    int a[9][9];
    for (int i=0;i<9;i++){
        for (int j=0;j<9;j++)
            scanf("%d",&a[i][j]);
    }
    if (parallel_check(a)){
        printf("Outside call: True");
    }
    else{
        printf("Outside call: False");
    }
    return 0;
}
```

实验结果

三组输入证明三个模块能正常工作吗，而且确实是并行的工作

```
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
---- checking Column 0 Fail
---- checking Block 5 Fail
---- checking Block 1 Fail
---- checking Column 2 Fail
---- checking Column 3 Fail
---- checking Row 5 Succeed
---- checking Row 0 Succeed
---- checking Column 1 Fail
---- checking Column 4 Fail
---- checking Row 2 Succeed
---- checking Block 2 Fail
---- checking Row 3 Succeed
---- checking Block 3 Fail
---- checking Column 5 Fail
---- checking Row 4 Succeed
---- checking Column 6 Fail
---- checking Block 4 Fail
---- checking Column 7 Fail
---- checking Block 0 Fail
---- checking Row 1 Succeed
---- checking Block 6 Fail
---- checking Row 6 Succeed
---- checking Block 7 Fail
---- checking Row 7 Succeed
---- checking Block 8 Fail
---- checking Row 8 Succeed
---- checking Column 8 Fail
-- Checking Block Fail
-- Checking Row Succeed
-- Checking Column Fail
Checking Fail
Outside call: False
请按任意键继续. . .
```

```
1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9
---- checking Column 0 Succeed
---- checking Column 6 Succeed
---- checking Block 1 Fail
---- checking Row 5 Fail
---- checking Column 2 Succeed
---- checking Block 4 Fail
---- checking Block 0 Fail
---- checking Row 2 Fail
---- checking Row 3 Fail
---- checking Column 4 Succeed
---- checking Row 4 Fail
---- checking Column 1 Succeed
---- checking Column 5 Succeed
---- checking Row 1 Fail
---- checking Block 2 Fail
---- checking Column 3 Succeed
---- checking Row 6 Fail
---- checking Column 7 Succeed
---- checking Block 3 Fail
---- checking Row 7 Fail
---- checking Column 8 Succeed
---- checking Block 5 Fail
---- checking Row 0 Fail
---- checking Row 8 Fail
---- checking Block 6 Fail
---- checking Block 7 Fail
---- checking Block 8 Fail
-- Checking Column Succeed
-- Checking Row Fail
-- Checking Block Fail
Checking Fail
Outside call: False
请按任意键继续. . .
```

```
1 2 3 1 2 3 1 2 3
4 5 6 4 5 6 4 5 6
7 8 9 7 8 9 7 8 9
1 2 3 1 2 3 1 2 3
4 5 6 4 5 6 4 5 6
7 8 9 7 8 9 7 8 9
1 2 3 1 2 3 1 2 3
4 5 6 4 5 6 4 5 6
7 8 9 7 8 9 7 8 9
---- checking Column 0 Fail
---- checking Row 6 Fail
---- checking Row 0 Fail
---- checking Block 4 Succeed
---- checking Column 2 Fail
---- checking Row 2 Fail
---- checking Column 3 Fail
---- checking Row 3 Fail
---- checking Block 0 Succeed
---- checking Column 4 Fail
---- checking Row 4 Fail
---- checking Block 1 Succeed
---- checking Column 5 Fail
---- checking Row 5 Fail
---- checking Column 6 Fail
---- checking Block 2 Succeed
---- checking Column 1 Fail
---- checking Column 7 Fail
---- checking Block 3 Succeed
---- checking Row 7 Fail
---- checking Column 8 Fail
---- checking Row 1 Fail
---- checking Row 8 Fail
---- checking Block 5 Succeed
---- checking Block 6 Succeed
---- checking Block 7 Succeed
---- checking Block 8 Succeed
-- Checking Column Fail
-- Checking Row Fail
-- Checking Block Succeed
Checking Fail
Outside call: False
请按任意键继续. . .
```

其他输入

```
1 2 3 4 5 6 7 8 9
4 5 6 7 8 9 1 2 3
7 8 9 1 2 3 4 5 6
2 3 4 5 6 7 8 9 1
5 6 7 8 9 1 2 3 4
8 9 1 2 3 4 5 6 7
3 4 5 6 7 8 9 1 2
6 7 8 9 1 2 3 4 5
9 1 2 3 4 5 6 7 8
---- checking Row 3 Succeed
---- checking Column 7 Succeed
---- checking Row 1 Succeed
---- checking Block 5 Succeed
---- checking Block 7 Succeed
---- checking Block 8 Succeed
---- checking Row 0 Succeed
---- checking Column 3 Succeed
---- checking Row 4 Succeed
---- checking Row 5 Succeed
---- checking Column 4 Succeed
---- checking Block 0 Succeed
---- checking Column 5 Succeed
---- checking Block 1 Succeed
---- checking Row 7 Succeed
---- checking Column 6 Succeed
---- checking Block 2 Succeed
---- checking Row 8 Succeed
---- checking Column 1 Succeed
---- checking Block 4 Succeed
---- checking Row 6 Succeed
---- checking Row 2 Succeed
---- checking Block 3 Succeed
---- checking Column 8 Succeed
---- checking Block 6 Succeed
---- checking Column 0 Succeed
---- checking Column 2 Succeed
-- Checking Row Succeed
-- Checking Block Succeed
-- Checking Column Succeed
Checking Succeed
Outside call: True
请按任意键继续. . .
```

```
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 5 7 3
2 8 5 4 7 3 9 1 6
---- checking Column 0 Succeed
---- checking Block 1 Succeed
---- checking Column 1 Succeed
---- checking Row 0 Succeed
---- checking Block 0 Succeed
---- checking Row 1 Succeed
---- checking Column 2 Succeed
---- checking Block 8 Succeed
---- checking Row 2 Succeed
---- checking Block 3 Succeed
---- checking Column 3 Succeed
---- checking Row 3 Succeed
---- checking Column 4 Succeed
---- checking Block 4 Succeed
---- checking Row 4 Succeed
---- checking Column 5 Succeed
---- checking Row 5 Succeed
---- checking Column 6 Succeed
---- checking Block 5 Succeed
---- checking Block 6 Succeed
---- checking Column 7 Succeed
---- checking Row 7 Succeed
---- checking Block 7 Succeed
---- checking Block 2 Succeed
---- checking Column 8 Succeed
---- checking Row 8 Succeed
---- checking Row 6 Succeed
-- Checking Block Succeed
-- Checking Column Succeed
-- Checking Row Succeed
Checking Succeed
Outside call: True
请按任意键继续. . .
```

```
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
3 7 1 9 5 6 8 4 2
7 6 2 3 9 1 4 5 8
4 9 6 1 8 2 5 7 3
2 8 5 4 7 3 9 1 6
---- checking Column 1 Succeed
---- checking Row 6 Succeed
---- checking Column 3 Succeed
---- checking Row 1 Succeed
---- checking Block 3 Fail
---- checking Column 7 Succeed
---- checking Row 3 Succeed
---- checking Block 0 Succeed
---- checking Row 0 Succeed
---- checking Row 4 Succeed
---- checking Block 1 Succeed
---- checking Column 4 Succeed
---- checking Block 2 Succeed
---- checking Row 5 Succeed
---- checking Column 5 Succeed
---- checking Column 2 Succeed
---- checking Column 6 Succeed
---- checking Column 0 Succeed
---- checking Block 4 Fail
---- checking Row 7 Succeed
---- checking Row 2 Succeed
---- checking Row 8 Succeed
---- checking Block 5 Fail
---- checking Column 8 Succeed
---- checking Block 6 Fail
---- checking Block 7 Fail
---- checking Block 8 Fail
-- Checking Row Succeed
-- Checking Column Succeed
-- Checking Block Fail
Checking Fail
Outside call: False
请按任意键继续. . .
```

证明验证器正常工作

# Assignment 2 Collatz猜想问题

### 实验内容

采用系统调用fork(),编写一个C程序，以便在子进程中生成一个Collatz序列，从命令行提供启动数

父进程调用wait()，以便在推出之前确保子进程已经完成，执行必要的错误检查以确保：一个正整数由命令行来传递

### 实验原理

fork()的返回值，对于父进程是子进程的pid，对于子进程是0,用个if语句即可区分父进程与子进程

## 实验代码

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main(int argc, char *argv[]) {
    pid_t pid;
    //错误检测
    int n = atoi(argv[1]);
    if (argc != 2 || n <= 0) {
        printf("Usage: %s positive_integer\n", argv[0]);
        return 1;
    }
    pid = fork();
    if (pid == 0) {
        printf("%d", n);
        while (n != 1) {
            if (n % 2 == 0) {
                n /= 2;
            } else {
                n = n * 3 + 1;
            }
            printf(" %d", n);
        }
        printf("\n");
    } else if (pid > 0) {
        wait(NULL);
    } else {
        printf("fork error\n");
        return 1;
    }
    return 0;
}
```

## 实验结果

有错误检测以确保输入的是正整数

```
donaldzy@DonaLdZY-Y9000P:~/Documents$ ./a 114514
114514 57257 171772 85886 42943 128830 64415 193246 96623 289870 144935 434806 217403 652210 326105 978316 489158 244579 733738 366869 1100608 550304 275152 137576 68
788 34394 17197 51592 25796 12898 6449 19348 9674 4837 14512 7256 3628 1814 907 2722 1361 4084 2042 1021 3064 1532 766 383 1150 575 1726 863 2590 1295 3886 1943 5830
2915 8746 4373 13120 6560 3280 1640 820 410 205 616 308 154 77 232 116 58 29 88 44 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
donaldzy@DonaLdZY-Y9000P:~/Documents$ ./a 35
35 106 53 160 80 40 20 10 5 16 8 4 2 1
donaldzy@DonaLdZY-Y9000P:~/Documents$ ./a -1
Usage: ./a positive_integer
donaldzy@DonaLdZY-Y9000P:~/Documents$
```