操作系统课程设计报告

题目：线程安全型双向链表的实现

专　　业：网络工程

班　　级：网络 121

学　　号：201210314005

姓　　名：朱正杰

上海海事大学信息工程学院

2014 年　12 月　　15 日

操作系统课程设计报告(网络 121 朱正杰)

目录

操作系统课程设计报告(网络 121 朱正杰)

1.课程设计任务描述与要求

1.1 任务描述

编写一个线程安全的双向链表，所谓线程安全，就是该链表能够实现多个线程同时正确的增删改链表结点，也就是能够实现对链表这个临界资源的保护。

1.2 任务要求

需要实现的函数包括：

（1） InitList　函数：初始化一个空的双向链表，并初始化各个用于保护链表的信号量。

（2） Insert　函数：向链表指定位置插入一个结点。

（3） Erase　函数：删除指定位置的结点。

（4） Clear　函数：删除链表中的所有结点。

（5） Find　函数：查找链表中是否有指定的元素，若有，返回能够访问该结点的指针；若无，返回 NULL

（6） Print　函数：打印当前链表中的所有元素。

完成该链表后，自己编写一个测试程序，生成多个线程同时读写该链表，验证链表执行是否正确，并给出测试报告。

2.系统总体结构描述与主要数据结构说明

2.1 系统总体结构描述

系统总体结构设计的任务，是根据系统分析的逻辑模型设计应用软件系统的物理结构。系统物理模型必须符合逻辑模型，能够完成逻辑模型所规定的信息处理功能。这是物理设计的基本要求。

系统应具有可修改性，即易读，易于进行查错、改错、可以根据环境的变化和用户的要求

进行各种的改变和改进。系统是否具有可修改性，对于系统开发和维护影响极大。据统计，在系统生命周期中各阶段的应用软件费用及人力投入大体分布如下：系统开发：20%；系统维护：80%。

由于程序功能简单，未用数据库辅助存储技术，本程序只供实现对双向链表的插入，删除，查找和打印等功能。

2.2 主要数据结构说明

宏定义部分：

```
#define random(x)(rand()%x)  //        产生随机数
#define cr 1  //1        标识为插入
#define sc 0  //0        标识为删除
volatile int readcount=0;  //          读者数目
const int lsarea=10000; //         链表大小随机数
const int earea=10000; //        元素范围随机数
const int sum=100000;  //       线程运行总次数
int th=0;  //       初始化当前线程总数
int th_cz=1;  //       初始化当前查找线程总数
int th_cr=1; //      初始化当前插入线程总数
int th_sc=1;  //       初始化当前删除线程总数
HANDLE h_Mutex;  //控制读者数量 readcount  的互斥访问量
HANDLE mutex;  //控制读写互斥，写写互斥的信号量
typedef int ElemType;  //          定义 ElemType为 int  类型的别名
typedef struct DuLNode *PNode;  //       结点指针
//定义结点结构体
typedef struct DuLNode{
    ElemType data;  //     定义数据域
    PNode prior;  //     定义前驱指针
    PNode next;  //     定义后继指针
}DuLNode,*DLN;
```

```
//定义双向链表结构体
typedef struct DuLinkList{
    DLN head;  //    定义头结点
    int Length;  //        定义链表长度
}DuLinkList,*DLL;
//定义读者传参结构体
struct Readarg{
    DLL List; //     定义链表
    ElemType e; //     定义查找元素
};
//定义写者传参结构体
struct Writearg{
    DLL List; //     定义链表
    int add; //      定义插入或删除的位置
    ElemType e; //     定义插入元素
    int Flag; //       定义传入标示符（cr 执行插入操作，sc 执行删除操作）
};
```

线程函数部分：

```
WaitForSingleObject(h_Mutex,-1);//         等待互斥量信号
WaitForSingleObject(mutex,INFINITE);//        等待信号量信号
ReleaseMutex(h_Mutex);//     释放互斥量信号
ReleaseSemaphore(mutex,1,NULL);//     释放信号量信号
```

主函数部分：

```
HANDLE hThread[sum];// 定义线程句柄
unsigned threadID[sum];//        定义 sum个线程
h_Mutex=CreateMutex(NULL,FALSE,NULL); // 创建互斥量 h_Mutex
mutex=CreateSemaphore(NULL,1,1,NULL); //     创建信号量 mutex
Readarg *RA=new Readarg[1];//     创建读者传参变量
RA[0].List=L;//     传参变量赋值
RA[0].e=random(100);//     传参变量赋值
```

```
Writearg *WA=new Writearg[2];//      创建写者传参变量
WA[0].List=L; //    传参变量赋值
WA[0].add=random(lsarea); //       传参变量赋值
WA[0].e=random(earea); //      传参变量赋值
WA[0].Flag=cr; //    传参变量赋值
WA[1].List=L; //    传参变量赋值
WA[1].add=random(lsarea); //       传参变量赋值
WA[1].e=0; //   传参变量赋值
WA[1].Flag=sc; //    传参变量赋值
hThread[i]=(HANDLE)_beginthreadex(NULL,0,ReaderThread,(void*)&RA[0],
0,&threadID[i]);//       创建线程函数
WaitForSingleObject(hThread[i],INFINITE);//          等待线程执行完毕
CloseHandle(hThread[i]);//       关闭线程句柄
CloseHandle(h_Mutex);//     关闭互斥量句柄
CloseHandle(mutex);//      关闭信号量句柄
```

3.课程设计报告内容

3.1 模块功能

此程序包含 6 个模块，分别是初始化兼创建模块，插入模块，删除模块，清空模块，查找模块和打印模块。先是有进程自动初始化并随机产生一个链表，然后创建多个线程，线程同时进行插入结点，删除指定位置结点，查找指定元素及打印链表操作，为清楚区分读和写的操作这里分出了两个线程一个为读者线程一个为写者线程，前者具有查找指定元素功能，后者具有插入和删除兼打印链表功能。对于所有的查找，插入和删除数都是随机产生的，更具有代表性。

如下图程序工作原理：

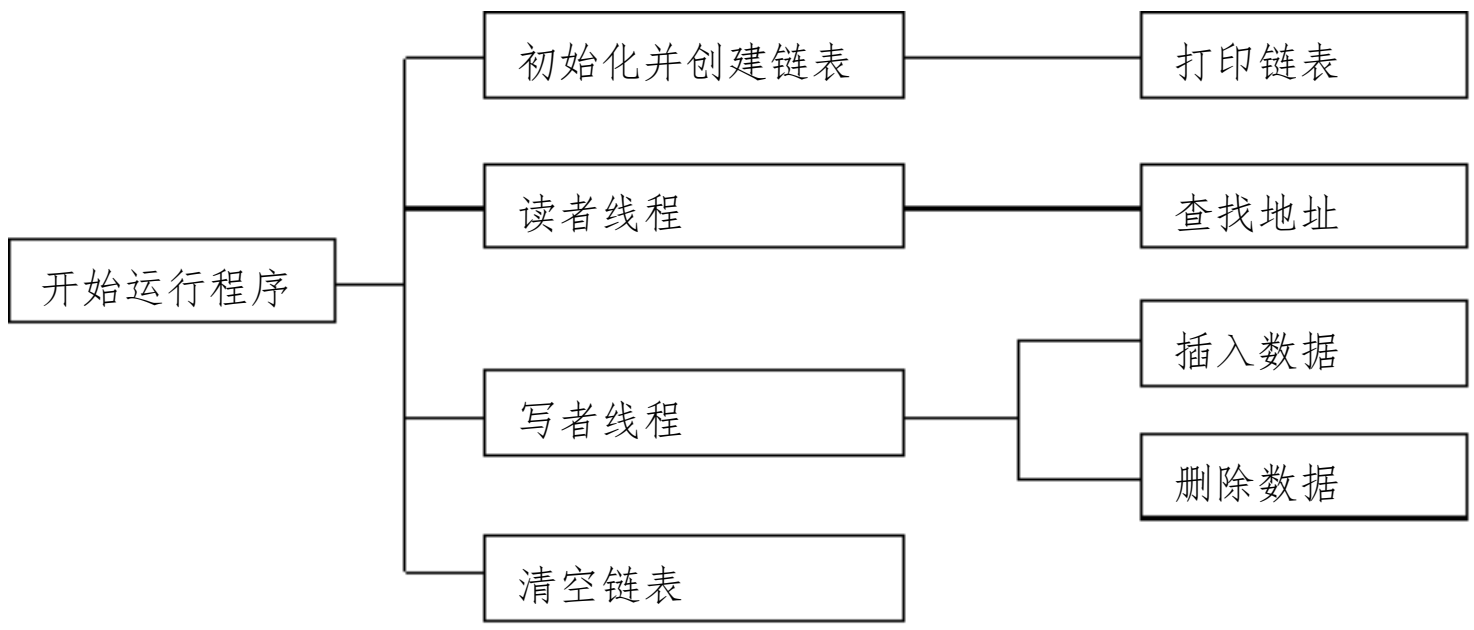

图 3.1 程序工作原理图

3.2 详细流程图

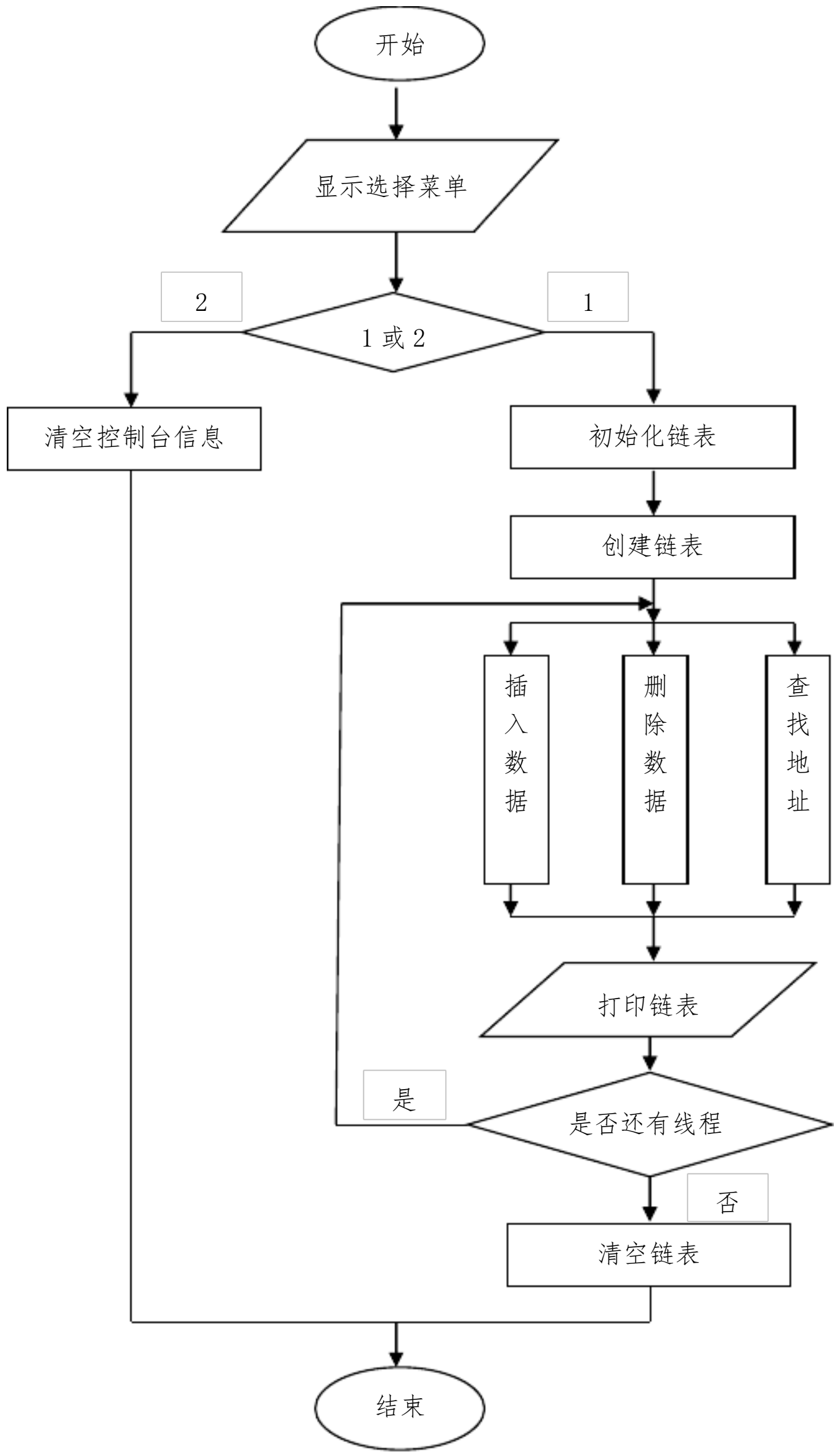

图 3.2 系统流程图

3.3 实现思路说明

第一步，构建双向链表的基本属性。包括结点结构体，链表结构体，初始化及创建链表函数，插入函数，删除函数，清空函数，查找函数，打印函数。

第二步，创建三个线程分别进行插入、删除和查找操作，主函数内创建完链表后通过传参结构体向线程传递链表参数，在线程内使用互斥量对链表的修改操作进行保护。运行过程中所有变量给定固定初始值，测试多线程同步的正确性。

第三步，构建两个线程分别为读者线程（执行查找操作），写者线程（执行插入和删除操作），所有变量使用随机数定义，利用互斥量对读者数的修改操作进行保护，利用信号量对链表的修改操作进行保护，从而实现读读共享，读写互斥，写写互斥的读者写者问题。并测试读者优先状态下链表多线程操作的正确性。

3.4 程序清单

```
#include<stdio.h>//c      语言标准输入输出头文件
#include<malloc.h>//    动态存储分配函数头文件
#include<stdlib.h>//      标准库头文件（malloc(),rand(),srand()      等等）
#include<process.h>//      包含用于和宏指令的作用声明与螺纹和过程一起使用的 C标头文件（线程的创建和终结等等）
#include<windows.h>//win32     头文件
#include<time.h>//     日期和时间头文件

void InitList(DLL L)//       初始化一个空的双向链表并创建链表
void Insert(DLL L,int i,ElemType e) //          在链表指定位置插入一个结点
void Erase(DLL L,int i) //         删除指定位置的结点
```

void Clear(DLL L) //　　删除链表中所有结点

DLN Find(DLL L,ElemType e) //　查找链表中是否有指定的元素，若有，返回能够访问该结点的指针；若无，返回 NULL

void Print(DLL L) //　　打印当前链表中的所有元素

unsigned __stdcall ReaderThread(void *arg) //　　读者线程(查找)

unsigned　__stdcall　WriterThread(void　*arg)　//写者线程（包括插入和删除）

3.5 注释

宏定义和主函数内部分代码的详细注释已经在前面章节阐述，下面主要为函数内容注释。

```
//初始化一个空的双向链表并创建链表
void InitList(DLL L){
    int c,i,e;//      定义三个整型变量 c，i，e
    DLN p;// 定义结点 p
    //初始化操作
    L->head=0;//  链表头结点置零
    L->Length=0;//  链表长度置零
    //创建操作
            双向链表初始化完毕
    srand((int)time(0));//      随机数时间种子设置
    c=random(lsarea);//     变量 c 取范围 0~Lsarea  内的随机整数
if(!c){
            链表创建失败!
    exit(0);//      异常处理，如果用户未输入结点个数则跳出该段代码。
}else{
    p=(DuLNode*)malloc(sizeof(DuLNode));    //p  动态分配存储空间
    if(!p){
            结点 p 动态分配内存失败!
        exit(0);    //异常处理，如果节点 p 动态分配内存失败则跳出该段代
```

(网络 121 朱正杰)

码。

```
}
e=random(earea);//     变量 e 取范围 0~earea  内的随机整数
p->data=e;//    将变量 e 的值送入结点 p 的数据域
p->next=p->prior=p;//      将结点 p 的前驱和后继指针指向它自己
L->head=p;//  将 p 结点作为链表头结点
L->Length++;//  链表长度加 1
//下面循环插入后续结点操作
for(i=1;i<c;i++){
      p=(DuLNode*)malloc(sizeof(DuLNode));      //p  动态分配存储空间
            if(!p){
                        结点 p 动态分配内存失败!
                  exit(0);    // 异常处理，如果用户未输入结点个数则跳出该段代
码。
            }
            e=random(earea);//     变量 e 取范围 0~earea  内的随机整数
            p->data=e; //     将变量 e 的值送入结点 p 的数据域
            p->next=L->head;//    将结点 p 的后继指针指向当前链表的头结点
            p->prior=L->head->prior;//       结点 p 的前驱指针指向当前链表的尾
结点
            L->head->prior->next=p;//       将当前链表尾结点的后继指针指向结
点 p
                  L->head->prior=p;//     将当前链表头节点的前驱指针指向结点 p
                  L->Length++;//  链表长度加 1
            }
      }
}

//在链表指定位置插入一个结点
```