**三、设备管理**

**3.1目的**

在前面的实验基础上实现设备管理功能的模拟，主要包括通道和控制器的添加和删除，设备的添加、删除，设备的分配和回收。

**3.2内容**

实现添加设备、删除设备、申请设备、回收设备、查看设备这五个功能。

**3.3数据结构**

typedef struct PCB

{

string name;

struct PCB \*next;

}PCB;

typedef struct CHCT{

string name; //通道名

bool stau;

PCB \*process; //占用该通道的进程

PCB \*waitinglist; //等待该通道的进程队列

struct CHCT \*next; //下一个通道

}CHCT;

typedef struct COCT{

string name;

bool stau;

CHCT \*chct; //控制器的上级通道

PCB \*process; //占用该控制器的进程

PCB \*waitinglist; //等待该控制器的进程队列

struct COCT \*next; //下一个控制器

}COCT;

typedef struct DCT{

string name;

string type;

bool stau;

COCT \*coct;

PCB \*process; //占用该设备的进程

PCB \*waitinglist;

struct DCT \*next;

}DCT;

typedef struct SDT

{

string name;

string type;

DCT \*dct;

struct SDT \* next;

}SDT;

CHCT \*channels; //通道队列头节点

COCT \*controllers; //控制器队列头节点

DCT \*devices; //设备队列头节点

SDT \*system;

**3.4算法设计及流程图**

添加设备时先判断该设备是否已经存在，若存在则输出错误提示；若不存在则进行初始化，询问是否创建新的控制器，最后询问选用通道1还是通道2。

删除设备时先判断该设备是否存在，若不存在则输出错误提示；若存在则从后往前进行删除，更新DCT、COCT的等待队列信息。

申请设备时直接用按类型申请，逐层按照SDT、DCT、COCT、CHCT去查找，若在某曾发现无法继续，则阻塞在当前层的阻塞队列里。

回收设备时先判断该设备是否存在，若不存在给出错误提示；若存在则逐层进行回收，更新每层的等待队列，若等待队列不为空，则将第一个等待的唤醒，依次向下检查。





**3.5小结**

本次实验用到指针的地方非常多，整个过程几乎都是在改变指针的指向、创建或销毁指针。加深了对指针使用的熟练程度。