**1**   **需求分析**

1.目的

（1）通过本实验可以加深理解有关进程控制块(PCB) 、进程队列的概念，体会并了解创建进程、切换进程、0阻塞进程、唤醒进程、结束进程、显示进程的具体实施过程

（2）完成设备管理功能的模拟，掌握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）通过设计一个磁盘调度模拟系统，从而使磁盘调度算法更加形象化，容易使人理解，使磁盘调度的特点更简单明了，能使使用者加深对先来先服务算法、最短寻道时间优先算法、扫描算法以及循环扫描算法等磁盘调度算法的理解

2. 内容

2.1进程调度

功能分析

(1)每个进程有一个进程控制块(PCB)表示。进程控制块包含如下信息:进程名、进程编号、进程的大小和进程的临接PCB的地址。

(2)每个进程的状态可以是就绪(ready) 、执行(running)和阻塞(block)三种状态之一。

(3)进程创建，由系统生成一个PCB结点，用进队函数放入就绪队列，如果没有正在执行的进程，则将等待队列中就绪进程调入执行。

(4)进程切换，通过函数实现将运行队列中的执行进程调入就绪队列，将等待队列中就绪进程调入执行。

(5) 进程阻塞，通过函数实现将运行队列中的执行进程调入阻塞队列，将等待队列中就绪进程调入执行。

(6) 进程唤醒，通过函数实现将阻塞队列中的阻塞进程调入就绪队列，将等待队列中就绪进程调入执行。

(7) 进程结束，通过函数实现将就绪队列中的就绪进程抢占运行队列。

(8)进程显示，根据队列进程的存储特性，顺序查找到每个进程并依次输出每个进程的名称和大小。

(9) 所创建进程将会在设备分配功能中为其提供所需必要设备。

2.2数据结构

进程控制块（PCB）

2.2.1设备分配

功能分析

（1）设备管理子系统涉及到通道控制表（CHCT）、控制器控制表（COCT）和设备控制表（DCT）来体现输入输出系统。

（2）实现上述设备、控制器以及通道的层次关系，同时能够添加或删除新的设备、控制器或通道。

（3）通过键盘命令模拟进程执行过程中提出的设备分配或释放请求，并为此请求分配或释放设备。分配设备成功后可将进程状态调整为阻塞，释放设备后变为就绪状态。

（4）分配设备时应如果该设备已被其它进程占用，则设备分配失败，请求进程进入阻塞状态，同时等待该设备的释放。如果设备空闲，进程占用设备的同时还应提出申请控制器请求，直到与设备相关的通道都已申请成功为止。

（5）设备、控制器或通道的释放应引起对应节点的等待队列中的第一个阻塞进程被唤醒。如果被唤醒的进程还未完成申请操作，应继续执行上级节点的申请操作。

2.3数据结构

设备控制表（DCT）

控制器控制表（COCT）

通道控制表（CHCT）

2.4磁盘调度

系统主界面可以灵活选择某种算法，算法包括：先来先服务算法（FCFS）、最短寻道时间优先算法（SSTF）、扫描算法（SCAN）、循环扫描算法（CSCAN）。

2.4.1先来先服务算法（FCFS）

根据进程请求访问磁盘的先后次序进行调度。此算法的优点是公平、简单，且每个进程的请求都能依次得到处理，不会出现某一进程的请求长期得不到满足的情况。此算法由于未对寻道进行优化，在对磁盘的访问请求比较多的情况下，此算法将降低设备服务的吞吐量，致使平均寻道时间可能较长，但各进程得到服务的响应时间的变化幅度较小。

**1**   **总体设计**



FCFS先进先出算法流程图

核心代码如下：

cout << "\t输入当前磁头所在磁道号: ";

    cin >> original\_track;

    cout << "\t输入磁道访问序列: ";

    while(scanf("%d", &track\_number) != EOF) {

        max\_distance += original\_track > track\_number ? original\_track - track\_number : track\_number - original\_track;

        original\_track = track\_number;

        q.push(track\_number);

    }

    que\_size = q.size();

    cout << "\t磁盘调度顺序为: " ;

    while(!q.empty())

    {

        cout << q.front() << " ";

        q.pop();

    }

    cout << endl;

    cout << "\t共 " << que\_size << " 个磁道被访问" << endl;

    cout << "\t总访问距离为 " << max\_distance << endl;

    cout << "\t平均访问距离为 " << " " <<  (double)max\_distance/que\_size << endl;