1. **进程控制**

**1.1目的**

利用简单的结构和控制方法模拟进程结构、进程状态和进程控制

**1.2内容**

模拟进程就绪态、执行态、阻塞态三个状态的转换。

**1.3数据结构**

采用结构体来存储，动态开辟空间。通过是建立三个链表来分别模拟进程的三个状态，采用队列先进先出原则进行状态转换。

**1.4算法设计及流程图**

**用户输入c、r、s、g、t、f、h、m、v、e命令来分别实现创建进程、执行进程、阻塞进程、恢复进程、时间片到、结束进程、查看帮助、查看内存、查看进程状态、退出功能，并添加了错误指令的容错功能。**

**当用户输入c指令。检查是否能够分配内存，如果不能分配内存，输出提示。如果能分配内存，判断执行链表是否为空，若执行链表为空，那么创建新节点链接到执行态，并且执行态节点数+1；若执行态链表不为空，那么创建新节点连接到就绪态尾部，并且就绪态节点数+1。打印此时状态。**

**检查内存分配时扫描空闲表，如果空闲表里当前节点长度大于等于进程长度，那么继续判断碎片长度是否小于等于2k，如果碎片小于等于2k，那么更新进程长度，并将空闲表的当前节点删除，返回true。否则继续比较。如果扫描完后发现无法分配内存，返回false。**

**当用户输入r指令。首先判断执行链表是否不为空，若执行链表不为空，可判断当前命令错误；若执行链表为空，那么判断就绪链表是否为空，若就绪链表为空，可判断指令错误。若就绪链表不为空，我们根据先进先出原则，选取就绪链表的第一个结点，将其放入执行态，相应执行态链表节点数+1，就绪态节点数-1。打印此时状态。**

**当用户输入s指令。首先判断执行链表是否为空，若执行链表为空，可判断当前指令错误；若执行链表不为空，我们将执行链表的节点放入阻塞链表尾部，相应阻塞链表节点数+1，执行链表节点数-1。接下来判断就绪态链表是否为空，若就绪态为空，则不作考虑；若就绪态不为空，则根据先进先出原则选择就绪态第一个节点，将其转换为执行态，相应执行态节点数+1，就绪态节点数-1。打印此时状态。**

**当用户输入g指令。首先判断阻塞链表是否为空，若阻塞态为空，可判断命令错误；若阻塞态不为空，根据先进先出原则，将阻塞态第一个节点取出放入就绪态尾部。打印此时状态。**

**当用户输入t命令时，判断运行态为空，若为空输出错误提示。若不为空，判断就绪态是否为空。如果就绪态为空，将运行态节点加入就绪态；否则将就绪态首节点加入运行态，并将原运行态节点加入就绪态尾。**

**当用户输入f命令时，判断运行态是否为空，若为空，输出错误提示。若不为空，回收内存。接下来判断就绪态是否为空，若不为空，将就绪态首节点加入运行态。**

**回收内存时扫描空闲表，按照四种可能出现的情况进行判断。**

**当用户输入h命令时，显示菜单。**

**当用户输入m时，显示内存情况。**

**当用户输入e时，退出程序。**

**当前用户输入指令错误，给出错误提示。**

















