

**《操作系统课程设计》报告**

**设计题目 进程管理演示**

**学院名称 计算机与网络安全学院（牛津布鲁克斯学院）**

**专业名称 软件工程**

**学生姓名 徐睿航**

**学生学号 202013160210**

**指导教师 王宇**

**设计成绩**

**教务处 制**

2022年09月 日

目 录

[第一章 设计思想说明 1](#_Toc114303698)

[1.1 设计目的 1](#_Toc114303699)

[1.2 设计环境 1](#_Toc114303700)

[1. 3 设计思想 1](#_Toc114303701)

[1.4 设计内容 2](#_Toc114303702)

[第二章 数据说明 2](#_Toc114303703)

[2.1 PCB的数据结构 2](#_Toc114303704)

[2.2 队列定义 3](#_Toc114303705)

[第三章 算法流程图 4](#_Toc114303706)

[3.1 FCFS算法流程图 4](#_Toc114303707)

[3.2 最短时间算法流程图 4](#_Toc114303708)

[第四章 程序源代码 5](#_Toc114303709)

[4.1函数声明 5](#_Toc114303710)

[4.2部分源代码 6](#_Toc114303711)

[4.3 测试数据集 7](#_Toc114303712)

[第五章 程序运行界面 8](#_Toc114303713)

[第六章 结论 9](#_Toc114303714)

# 第一章 设计思想说明

## 1.1 设计目的

加深对进程概念及进程管理各部分内容的理解；熟悉进程管理中主要数据结构的设计及进程调度算法、进程控制机构、同步机构及通讯机构的实施。

## 1.2 设计环境

（1）计算机及操作系统：MacBook Pro 2020，macOS Catalina

（2）程序设计语言：C++语言

（3）开发平台：Clion

（4）运行平台：MacBook Pro 2020

## 1. 3 设计思想

在采用多道程序设计的系统中，往往有若干个进程同时处于就绪状态。当就绪进程个数大于处理器数时，就必须依照某种策略来决定哪些进程优先占用处理器。根据不同的算法，模拟多个程序在计算机中的调配。

图形用户界面

描述已自动生成设计的算法有，先来先服务优先（FCFS）和用短作业优先算法（SJF）。处理器为两个处理器。先来先服务算法如图1-1所示。

图1-1 FCFS算法示意图

## 1.4 设计内容

设计一个允许 n 个进程并发运行的进程管理模拟系统。该系统包括有简单的进程控制、同步与通讯机构，其进程调度算法可任意选择(优先级调度，时间片 轮转，短进程优先中的一种)。每个进程用一个 PCB 表示，其内容根据具体情况设置。各进程之间有一定的同步关系(可选)。系统在运行过程中应能显示或打印各进程的状态及有关参数的变化情况，以便观察诸进程的运行过程及系统的管理过程。

# 第二章 数据说明

## 2.1 PCB的数据结构

在此设计中，将PCB所需的资源定义为需要的内存空间数、磁带数量。PCB具体定义如表2-1所示。PCB在代码中的表示如表2-2所示。

表2-1 PCB字段定义

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 注释 |
| name | string | 作业名 |
| account | string | 用户名 |
| Time | int | 需要的运行时间 |
| needSize | int | 需要的内存空间数目 |
| needTapes | int | 需要的磁带数 |
| staus | int | 目前作业状态,1表示收容状态，2表示运行，3表示结束 |

表2-2 PCB在代码中的定义

|  |
| --- |
| typedef struct HPCB {  string name;// 作业名  string account;//用户名  int time;//需要运行的时间  int needSize;//需要的内存空间数目  int needTapes;//需要的磁带数  int staus;//目前作业状态,1表示收容状态，2表示运行，3表示结束  }HPCB,\*qHPCB; |

## 2.2 队列定义

队列为假设操作系统中的处理进程的队列，用于所有进程整体的调配。具体定义如表2-3所示。代码中的定义如表2-4所示。

表2-3 队列的字段定义

|  |  |  |
| --- | --- | --- |
| **属性** | **类型** | **注释** |
| **base** | HPCB | 初始化的动态分配存储空间 |
| **front** | int | 头指针，若队列不空，指向队头元素 |
| **rear** | int | 尾指针，若队列不空，指向队尾元素的下一个位置 |
| **count** | int | 主存中有的作业数 |

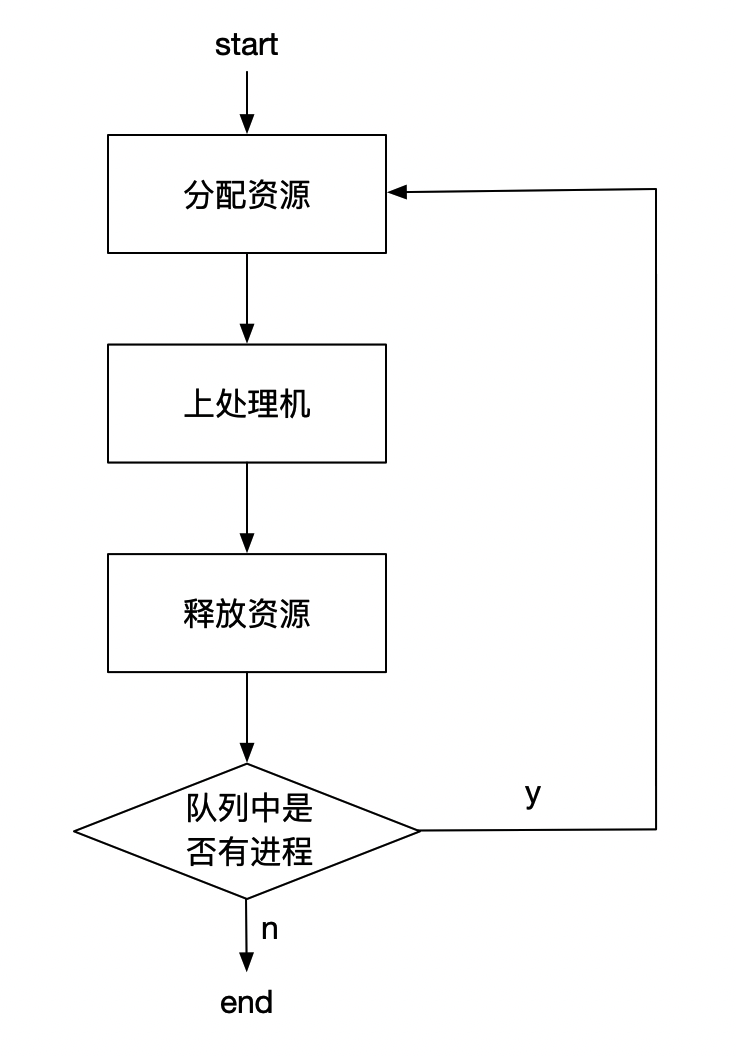
表2-4 队列在代码中的定义

|  |
| --- |
| typedef struct {  HPCB \*base; //初始化的动态分配存储空间  int front; //头指针，若队列不空，指向队头元素  int rear; //尾指针，若队列不空，指向队尾元素的下一个位置  int count; //主存中有的作业数  }sqQueue; |

# 第三章 算法流程图

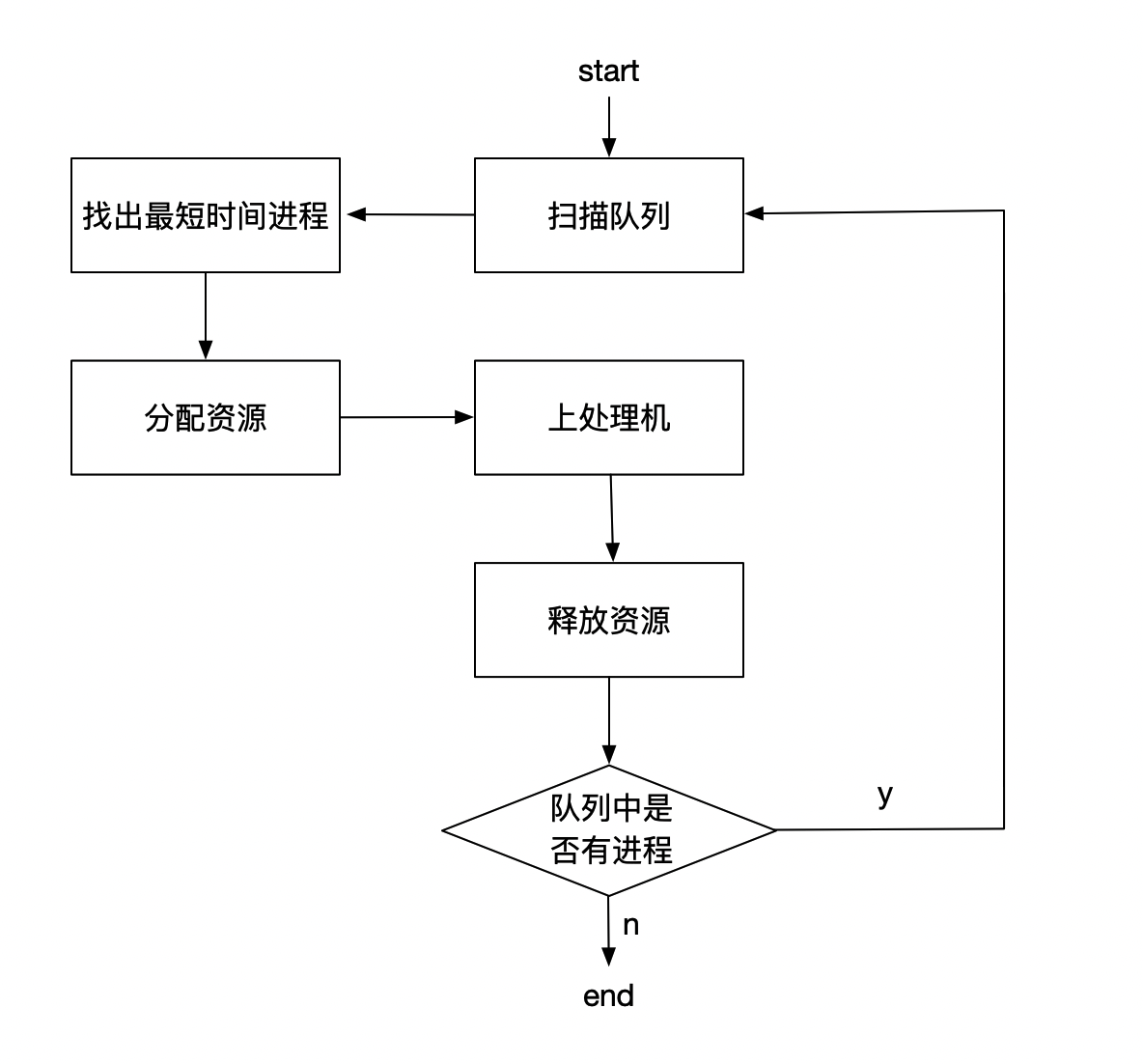
## 3.1 FCFS算法流程图

FCFS算法流程图如图3-1所示。

图3-1 FCFS算法流程图

## 3.2 最短时间算法流程图

最短时间算法流程图如图3-2所示。

图3-2 最短时间算法流程图

# 第四章 程序源代码

## 4.1函数声明

函数声明如表4-1所示。

表4-1 函数声明

|  |
| --- |
| void initQueue(sqQueue &Q); //对5个HPCB进行初始化  void display(sqQueue q); //展示所有作业的状态  void onControl(sqQueue &Q); //将队列中的一个进程调度到处理器中运行  void scan(sqQueue &Q); //扫描整个队列，将0-2个作业调入运行队列  bool checkIn(sqQueue Q); //检查队列里面是否还有有可以进入处理机的作业  void EnQueue(sqQueue &Q, HPCB e);//进队  void DeQueue(sqQueue &Q, HPCB &u);//弹出一个进程  bool QueueEmpty(sqQueue Q);//判断队列是否为空 |

## 4.2部分源代码

FCFS部分源代码如表4-2所示。

表4-2 FCFS算法部分源代码

|  |
| --- |
| #define MAXQSIZE 5  #define MAXINT 100;  int mainTapes = 5;//磁带数  int mainSize = 1024;//内存数  int forceTime = 0;//全局时间，从零开始  typedef struct HPCB {  string name;// 作业名  string account;//用户名  int time;//需要运行的时间  int needSize;//需要的内存空间数目  int needTapes;//需要的磁带数  int arvTime;//到达的时间  int staus;//目前作业状态,1表示收容状态，2表示运行，3表示结束  }HPCB,\*qHPCB;  //----队列的定义及操作--------  typedef struct {  HPCB \*base; //初始化的动态分配存储空间  int front; //头指针，若队列不空，指向队头元素  int rear; //尾指针，若队列不空，指向队尾元素的下一个位置  int count; //主存中有的作业数  }sqQueue;  /////////////////////////////////////////////////////////////////////////////////////////  //进队  void EnQueue(sqQueue &Q, HPCB e){  //插入元素e为Q的新的队尾元素  Q.base[Q.rear] = e;  Q.rear = (Q.rear + 1)%MAXQSIZE;  Q.count += 1;  mainSize -= e.needSize;  mainTapes -= e.needTapes;  }//EnQueue |

## 4.3 测试数据集

测试数据集包括每个进程所需要的磁带数、所需运行时间、占用内存的大小等资源。测试数据集如表4-3所示。

表4-3 测试数据集

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 名称 | 所需运行  时间 | 所需内存 | 所需磁带数 |
| 1 | JA | 20 | 150 | 2 |
| 2 | JB | 35 | 600 | 1 |
| 3 | JC | 6 | 500 | 3 |
| 4 | JD | 24 | 100 | 2 |
| 5 | JE | 12 | 300 | 3 |

# 第五章 程序运行界面

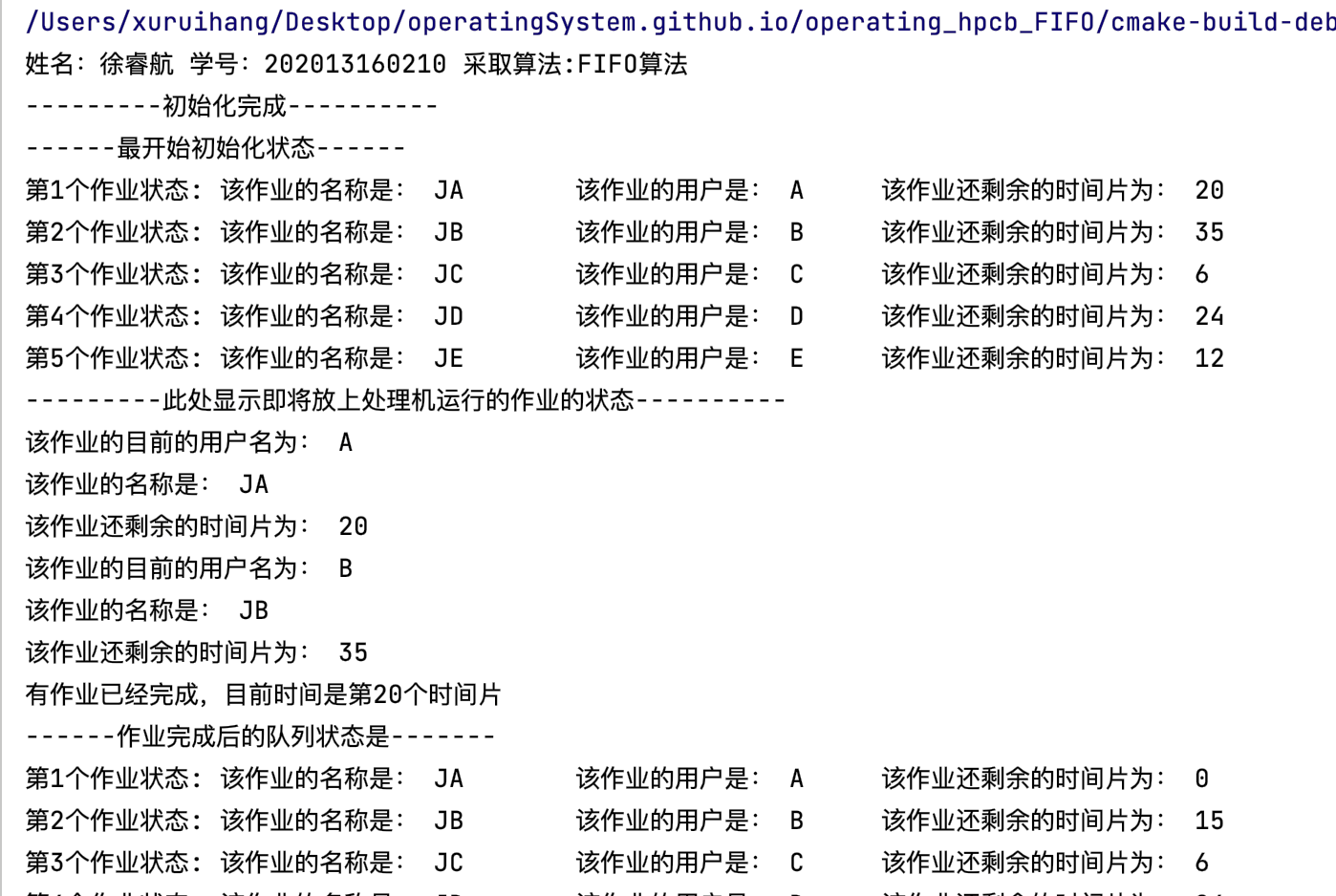
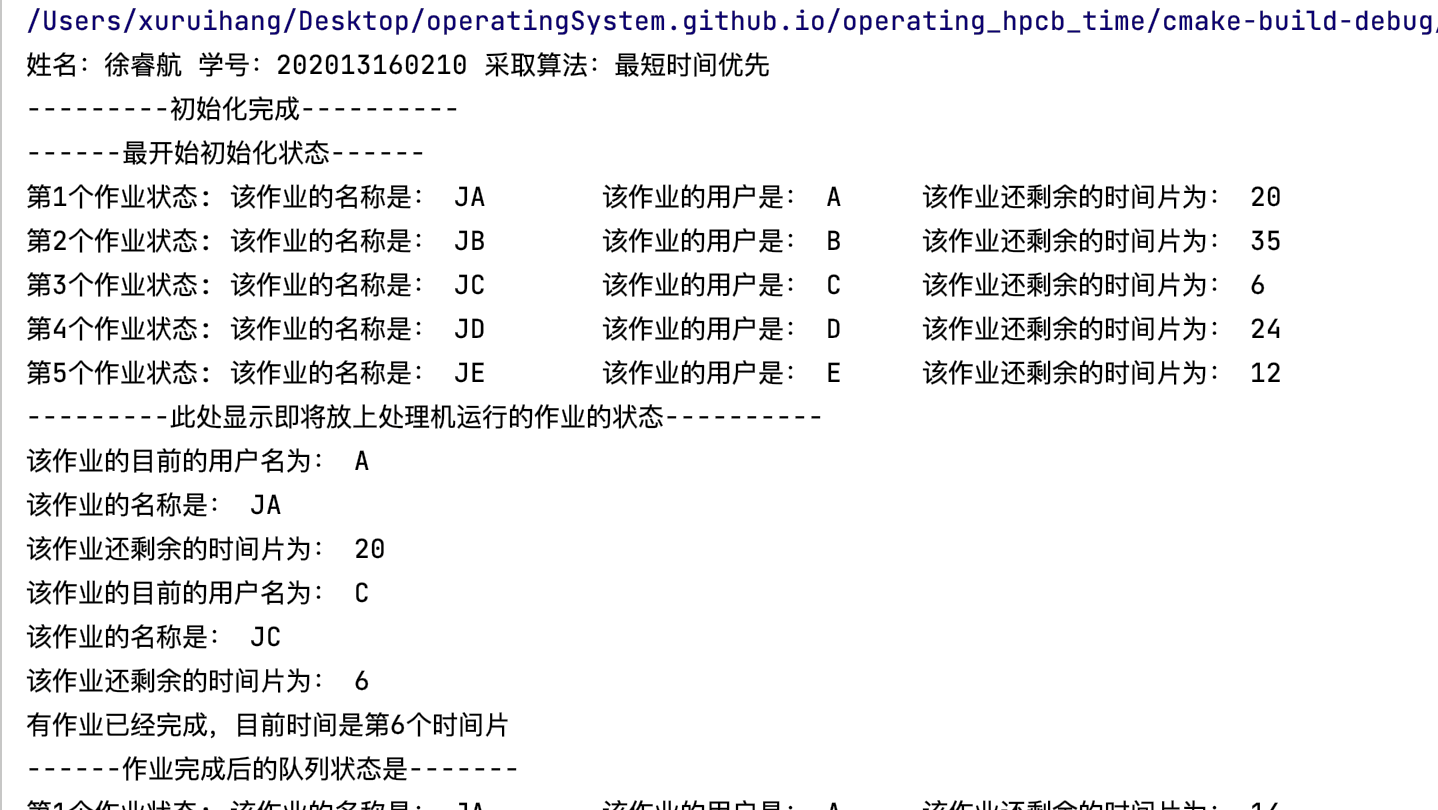
FIFO算法运行如图5-1所示。最短时间算法运行如图5-2所示。

图5-1 FIFO算法运行图

图5-2 最短时间优先算法运行图

# 第六章 结论

经过接近3天的设计与实现，“进程演示” 已经基本完成。在这设计过程中，我更加理解了操作系统中进程的各种状态。另外，通过对clion的使用，对c++语言的运用更加熟悉。基于该演示的一些设计，下面对演示做一个总结。

在整个设计过程中，主要的工作有：

进程的初始化、进程的状态检查、进程的调度算法的设计

在设计时，由于时间和能力有限还有更高级的调度算法的功能没有实现，希望系统能在以后升级成更高版本时得以实现。

设计的两个算法，已经实现了多处理器情况以及进程就绪、进程处理、进程阻塞三种状态。较为清晰的展示了进程在操作系统下的状态。

本课设中只设计了两个算法，还有很多其它的算法没有涉及。此外，算法的展示不够图形化，没有进行前端的搭建。希望未来能够进行补充完整。

|  |  |
| --- | --- |
| **学生实验 心得** | 此次操作系统课程设计，采用 c++ 语言设计进程调度模拟系统，在给定题目的基础上，在此模拟系统中实现了两种基本进程调度。在编程实践中，我对 c++ 语言有了进一步的深入，对进程调度有了更深入的认识。感谢在算法设计过程中，室友的帮助。  这是一次难忘的经历,因为它同去年操作系统课程一样，被封闭在寝室里面完成。  学生（签名）：  2022年 09 月17日 |
| **诚信承诺** | 本人郑重声明所呈交的课程报告是本人在指导教师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注的地方外，论文中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同学对本文研究所做的贡献均已在报告中作了明确的说明并表示谢意。  学生（签名）：  2022年 09 月 17 日 |
| **指导**  **教师**  **评语** | 成绩评定：  指导教师（签名）：  年 月 日 |