【模板：概要设计报告】

<Group04>小组 <公交车调度>概要设计
版本号：siji-Design-012（每次修订时改变编号）
编制时间：2022 年 7 月 4 日
编制人员：梁晨锐、闫晨浩、孙懿

1. 用户界面设计

1.1 文件方式（定义输入输出的格式）

A．[文件名称]dict.dic

B．[文件格式]“参数 = 值”的形式。其中参数有三个，即 TOTAL_STATION，代表站点总数，为大于 1 且小于等于 10 的整数；DISTANCE，代表每站之间的距离，为大于 0 且小于 6 的整数；STRATEGY，代表调度策略，只能是 FCFS（先来先服务），SSTF（最短寻找时间优先）和 SCAN（顺便服务）之一。

[文件使用/产生说明]程序要通过读配置文件获取参数

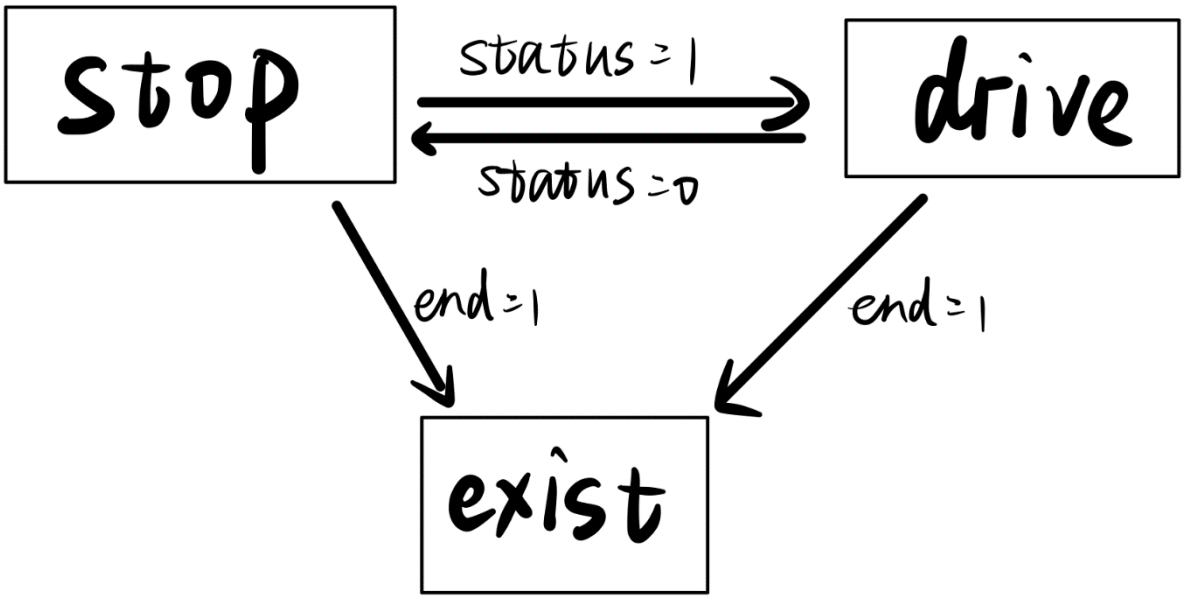
1.2 动画方式（画出图形界面）

[图片]后续补充

[界面操作说明]

2 有限状态自动机状态转换图

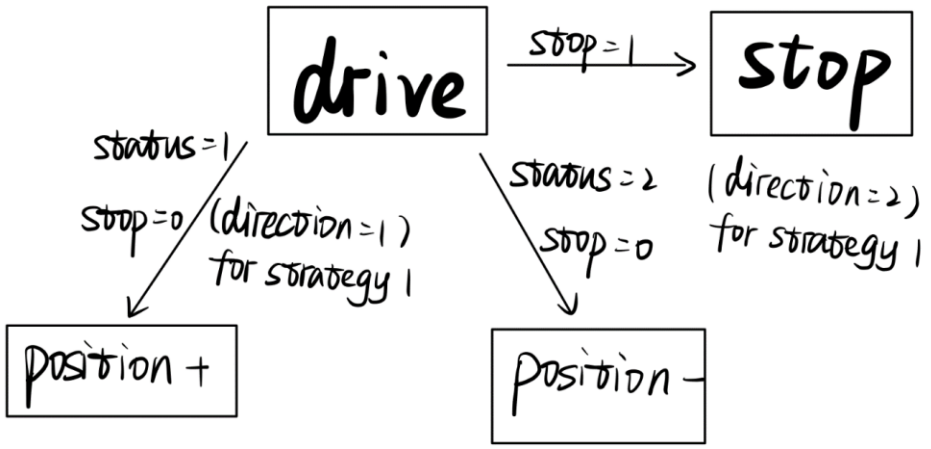
2.1 [一级状态图]



[状态转换说明]
stop:停止
drive:运行
exist:结束

status:表示车状态的变量
（status=0 时表示车处于等待状态，status 不等于 0 时表示车处于运行状态）
end:表示是否接收到停止指令

2.2 [二级状态图]



[二级状态说明]
stop:停止
drive:运行
status=1 且 stop=0 时，车顺时针运行，position=position->next
status=2 且 stop=0 时，车逆时针运行，position=position->prev

3 高层数据结构设计

(包括：全局常量定义、全局变量定义。注意命名，常量和变量含义要有说明)

3.1 常量定义

```
#define MAX_STATION 10    //最多有 10 个车站
#define MAX_REQUEST 10000 //最多有 10000 个请求（假设）
```

3.2 全局变量定义

```
//以下为公共变量，默认全部为 0
extern int time;
extern int goal;       //在 FCFS 中是当前目标在队列中的编号，在 SSTF 中是目标站的编号
extern int TOTAL_STATION = 5; //默认为 5
extern int STRATEGY = 1;
//默认为 1，STRAGETY 值含义：1 => FCFS，2 => SSTF，3 => SCAN
extern int DISTANCE = 2;      //默认为 2
extern int TOTAL_LENGTH = 10; //默认为 10
extern int stop;              //记录停站状态
typedef struct node
{ //采用了双向循环链表模拟轨道；定义以及相关变量与函数
```

```
    int pos;
    int sta;
    struct node *next;
    struct node *prev;
} track;
typedef struct
{                          //存储站点信息的 INFO 数组
    int target;            //公交车上对车站请求
    int clockwise;         //顺时针车站请求  0 表示无，1 表示有
    int counterclockwise; //逆时针车站请求
} sta;
typedef struct
{ // FCFS 和 SSTF 通用的请求队列，可存储一个请求的站点和类型
    int type;
    int sta;
} STATION1;
extern STATION1 req[MAX_REQUEST];
extern int reqNum;
extern int queue[MAX_REQUEST]; // SCAN 专用请求队列，仅存放请求目标站编号
```

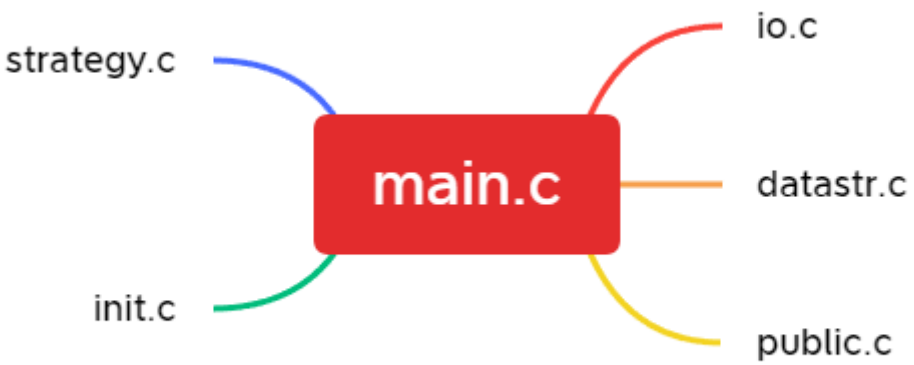
4 系统模块划分

4.1 系统模块划分(说明共分成哪些程序模块，各模块功能概述)

模块划分思路说明。

哪些模块需要独立成线程?

模块关系图，独立线程的模块要用红色标记出来！参考如下图例。



1. 模块名称 io.c

模块功能简要描述：完成对用户输入的线程的实现，根据输入更新用户请求数据结构；读取配置文件，根据指令分析类型，调用相应策略；输出当前车辆的状态和各站点的请求信息。

2. **模块名称 public.c**

模块功能简要描述：三个策略的共同部分，如公交车状态的更新等。

3. **模块名称 datastr.c**

模块功能简要描述：存放三个策略以及输出状态时用到的数据结构。

4. **模块名称 strategy.c**

模块功能简要描述：先来先服务策略、最短寻道优先算法策略、扫描策略的具体实现。

5.**模块名称 init.c**

模块功能简要描述：负责初始化整个系统。

4.2 各模块接口说明

1. **模块接口 init.h**

本模块声明 initSystem()函数，进行程序的初始化。

2. **模块接口 public.h**

本模块声明了三个策略通用的几个函数，如控制运行方向的 drive()，判断是否到目的地的 isArrive()等，还有一系列共用的全局变量，如时间变量 time,策略类型 STRATEGY 等。

3. **模块接口 default.h**

本模块使用 define 定义了两个常量，MAX_STATION=10 和 MAX_REQUEST=10000.

4. **模块接口 datstr.h**

本模块定义了程序的数据结构（data structure），采用了双向循环链表模拟轨道；定义以及相关变量与函数。

5. **模块接口 datastr.h**

本模块声明了一系列策略共用的函数，如删除链表的 deleteList(),增加站点信息的 addINFO(),删除站点信息的 delINFO()以及判断是否结束退出程序的 isExist()，以及表示公交车信息的结构体数组 INFO 等。

6. **模块接口 io.h**

本模块声明了用于输入输出的函数，如读取配置文件的函数 readCommand(),获取指令的函数 getRequest(),以及输出打印公交车目前状态的函数 printStatus()等。

7. **模块接口 strategy.h**

本模块声明了策略一：先来先服务策略、策略二：最短时间寻道策略、策略三：扫描算法的函数。

4.3 各模块函数说明

模块文件	模块说明	模块包含的函数名	函数功能
io.c	完成对用户输入的线程的实现,根据输入更新用户请求数据结构；输出当前车辆的状态和各站点的请求信息。	void readCommand()	实现连续读入公交调度的指令/请求，并将指令/请求的类型及相关值（如果有）传递给调度算法
		int isEnd(char command[])	判断是否读到 end 指令
		int getRequest(char command[])	判断指令/请求的类型，并返回相应的代表指令的正整数
		void getConfig()	读取配置文件，记录配置信息和选取策略
		void printStatus()	输出当前车辆的状态和各站点的请求信息。

public.c	三个策略的共同部分	`void drive()`	控制车辆运行
		`int cacu(int now, int sta, int mn)`	计算有请求的站点到公交车的距离，返回最小值所表示的站点位置
		`int isArrive()`	判断是否到达目标站点
		`int isRequest()`	当前位置是请求站点时被调用，返回相关信息
datastr.c	定义了程序的数据结构相关变量与函数。	`void initSystem()`	程序系统的初始化
		`track *createList()`	创建双向循环链表模拟轨道
		`void deleteList()`	删除双向循环链表模拟轨道并释放空间
		`void addINFO(int commandType, int value)`	将读取到的输入信息根据不同的类型添加到 INFO 数组中
		`void delINFO(int type, int sta)`	从 INFO 数组中删除已经处理过的站点信息
		`void push_back(int commandType, int value)`	实现 req 数组的出队操作
		`int isExist(int type, int value)`	判断是否读取到结束指令，控制程序退出
		`int isExist_SSTF(int value)`	根据指令类型，判断是否退出
		`void push_SCAN(int queue[], int *size, int value)`	控制 SCAN 函数的入队操作
		`void pop_FCFS()`	FCFS 策略的出队操作实现
		`void pop_SCAN(int queue[], int *size, int value)`	控制 SCAN 函数的出队操作
		`int isExist_SCAN(int queue[], int *size, int value)`	判断 SCAN 函数是否已经读取到退出指令，并控制其退出
		`void pop_SSTF(int type, int value)`	将相关站点的请求出队
strategy.c	实现三种调度策略	`void busStop()`	汽车到站时被调用，回调入队及出队函数
		`void setDirection()`	FCFS 策略确定运行方向
		`void arrStation()`	到站函数，到站之后的接客送客操作
		`void FCFS(int commandType, int value)`	调用 FCFS 策略
		`void SSTF(int commandType, int value)`	调用 Search Shortest Time First 策略，实现其相关功能
		`void SCAN(int commandType, int value)`	调用 SCAN 策略，实现其相关功能
		`void initBus()`	初始化结构体 bus 的相关变量
		`int getDistance(int tget)`	获取公交车当前位置与某一车站之间，沿当前方向的距离

		int isStation()	判断是否路过车站
		void setStatus2();	SSTF 用更新行车状态
		void setStatus3();	SCAN 用更新行车状态
init.h	实现初始化	void initSystem()	进行程序的初始化

5 核心算法设计

系统核心算法的概要设计，例如电梯系统的控制策略算法。要求对任务书中的策略进一步细化分解，结合上面的数据结构和模块设计，描述出算法的实现思路。

1.FCFS:First Come First Service 先来先服务策略

(1)接受一个请求，利用循环体和 if 判断，确定进程是否被调度

(2)若进程未被调度，则改变其调度状态并存入队列里

(3)按照队列中的顺序依次完成调度

2.SSTF: Shortest Seek Time First 最短寻找时间优先策略

（1）首先判断指令类型。若指令为 clock，再判断此时公交车是否处于未启动状态（ status==0 ）。若未启动，则初始化。

（2）初始化后，根据距离距离公交车最近的请求站点，调用 drive() 函数确定行驶方向，驾驶公交车。

（3）每一秒判断一次公交车是否位于某个站点，如果是，则判断是否位于目标站点且是否有可以调用的顺便服务策略。

（4）若指令为 clock，执行完所有操作后还需调用 printStatus() 输出一遍当前状态。

（5）若指令不为 clock，则需将指令表示的站点请求入队，并添加到表示请求站点信息的数组中去。

3.SCAN 顺便服务策略

（1）首先判断指令类型。若指令为 clock，再判断此时公交车是否处于未启动状态（ status==0 ）。若未启动，则初始化。

（2）初始化后，根据距离距离公交车最近的请求站点，调用 drive() 函数确定行驶方向，驾驶公交车。

（3）每到站一次，就判断一次当前请求队列中最近的目的地到目前地点的距离是否大于总长度的一半，如果是，则切换行驶方向，否则继续按照当前方向行驶。

写在最后：

计导课设贯穿了这一整个特殊的时期，在临近告一段落的此刻，我们有着如释重负的轻松，也感到怅然若失的空落。

在这段特殊的时期，我们三位组员跨越着大半个中国大地，隔着小小的屏幕，一字一符艰难地敲出了最终的程序。我们也曾为读不懂文档而苦闷，也曾因想不出思路而惆怅，也曾面对满屏的 bug 和 wa 抓耳挠腮，也曾想过躺平摆烂放弃挣扎。

但是我们挺过来了，哪怕最后的结果并不完美，至少是我们三人用尽心血花费半年熬出来的成品。看着即将提交的验收材料，我们有种看着自己的孩子踏上高考考场的责任感和幸福感。也许，这就是程序设计的魅力所在。

最后的最后，我们想感谢本学期一直坚守在课程群为我们答疑解惑的老师和助教们、同学们，是你们

的帮助让我们共同进步。感谢我们自己，不抛弃，不放弃，踏踏实实提升编程技术和项目水平。

此次课设是我们迈入编程实战的重要里程碑，愿我们都能成为优秀的北邮计算机人。