**48小组——机场营业厅排队系统概要设计**

版本号：V1.0

编制时间：2017年5月4日

编制人员：吴祥宏、张楷洪、林东杰

1. **用户界面设计**

**1.1文件输入格式**

文件结构体:

struct entry {

int no; //事件序号

int sec;//事件发生时间间隔

char type;//事件类型：C-乘客到达，X-安检口申请暂停，Q-下班；

int mans; //事件属性1-到达人数

int check; //事件属性2-申请暂停的安检口编号

};

文件名input.dat

文件输入(二进制文件)

取值范围

**1.2输出文件格式**

文件名 output.dat

(1)当安检口、安检口队列和排队缓冲区状态有变化时进行记录

T=<当前时间（3位）>

OFFDUTY=Y/N（是否下班）

WIN1:State=状态(11位)，WinList=队列中乘客编号列表，WinListCustCount=安检口队列人数

WIN2:同上

...

ListLines = 排队缓冲区队首乘客编号，队尾乘客编号。

ListCustCount = 排队缓冲区总乘客数

当有乘客进入排队缓冲区时，输出“\*\*\*乘客进入排队缓冲区”

当乘客想进入排队缓冲区但遭到拒绝时，输出“排队缓冲区已满”

当乘客进入安检队伍，输出“\*\*\*乘客进入\*\*安检口排队”

当乘客结束安检离开安检大厅，输出“\*\*\*乘客完成安检离开”

当下班指令到达，输出“接收到下班指令”<\n>

(2)每3秒周期性记录安检口、安检口队列和排队缓冲区的状态。

**1.3命令行输入界面**

进入程序时提示用户是否需要修改配置文件：

extern int NumOfWin;//安检口总数目

extern int MinRestSec;//安检口口最小休息时长

extern int MaxRestSec;//安检口最大休息时长

extern int MaxCustSingleLine：单队列最大等待乘客数

extern int MaxLines：蛇形缓冲区最多由MaxLines个直队组成

extern int MaxSeqLen：最大允许等待长度

extern int EasySeqLen: 短期等待长度

extern int MinCheck ：最少开放安检口数量

extern int MaxCheck ：最多开放安检口数量

extern int MaxCustCheck：安检口队伍最大长度

extern int MaxSec ：安检口最大安检时长单位是秒

extern int MinSec ：安检口最小安检时长

提示用户输入字符：

‘G’表示一个乘客申请进入“排队缓冲区”。

‘X’表示X号安检口请求暂停，X取值为1~MaxCheck。

‘Q’表示下班

**1.4令行输出界面**

屏幕输出：

【格式】【周期性输出】

|  |
| --- |
| 当前排队：乘客:XX  安检口：1-空闲/正在安检/休息中/关闭 2-…………  等待队列：普通客户：XX XX XX  上班状态：上班中/已下班 |

【格式】【有事件发生时】

|  |
| --- |
| 窗口状态调整：  XX号窗口等待关闭/休息；XX号窗口进入休息；XX号窗口关闭；窗口不足，XX号窗口开启 |

**1．5配置文件格式**

配置文件 para.dat

启动后，进入配置信息维护界面，读取历史配置参数，

提示用户进行修改，然后运行。

extern int NumOfWin;//安检口总数目

extern int MinRestSec;//安检口口最小休息时长

extern int MaxRestSec;//安检口最大休息时长

extern int MaxCustSingleLine：单队列最大等待乘客数

extern int MaxLines：蛇形缓冲区最多由MaxLines个直队组成

extern int MaxSeqLen：最大允许等待长度

extern int EasySeqLen: 短期等待长度

extern int MinCheck ：最少开放安检口数量

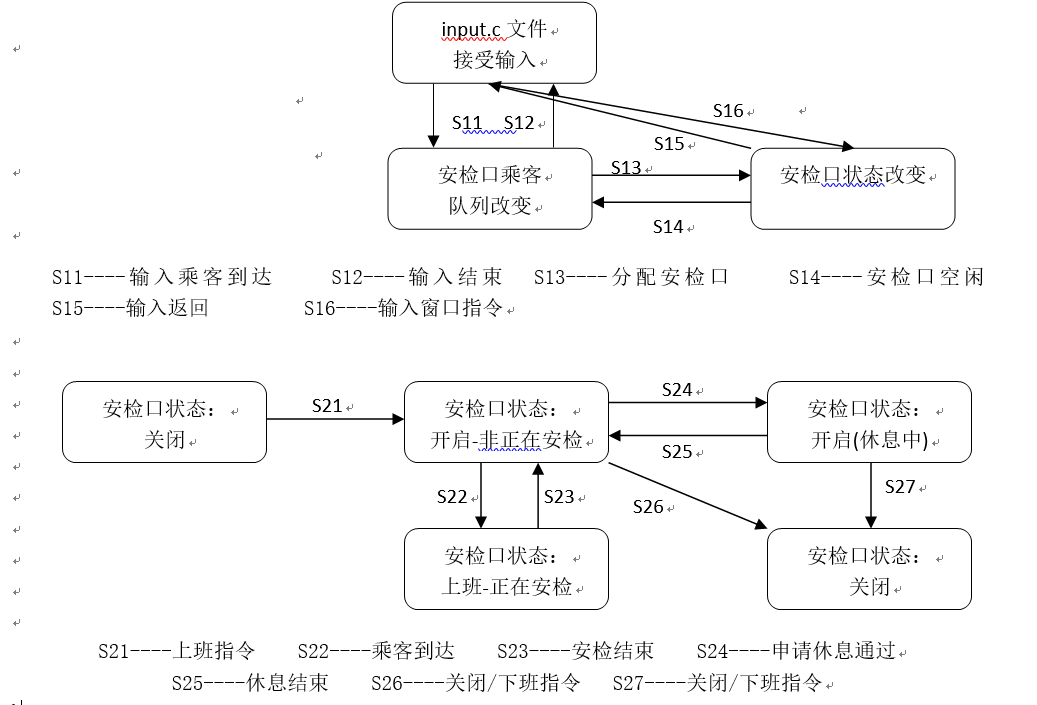
extern int MaxCheck ：最多开放安检口数量

extern int MaxCustCheck：安检口队伍最大长度

extern int MaxSec ：安检口最大安检时长单位是秒

extern int MinSec ：安检口最小安检时长

**2.自动机模型（状态转换图）**



**3.高层数据结构设计**

**3.1类型定义**

typedef struct Window{

int WinState;//工作状态：CloseWin窗口关闭 OpenWin空闲 OnSerWin正在服务 RestWin休息中 ReadyRestWin等待休息 ReadyClosWin等待关闭

int SerNum;//服务的号码

long SerTime;//本次服务结束时间

long RestTime;//本次休息结束时间

long TotalSer;//总共服务人数

long TotalTime;//总共服务时间

}Window;//窗口结构

typedef struct Passenger{

int State;//服务状态 0未分配，1已分配窗口，2已开始安检，3已离开

int SerNum;//窗口

long TaskTime;//安检时间

} Passenger;

Passenger Pas[1000];//乘客数组

**3.2全局常量定义**

//机场

#define OffWork 0

#define OnWork 1

#define ShutDown 2

//窗口状态

#define CloseWin 0

#define OpenWin 1

#define OnSerWin 2

#define RestWin 3

#define ReadyRestWin 4

#define ReadyClosWin 5

**3.3关键变量常量定义**

extern int AirportState;//机场

extern int TotalOdinCus;//当前总的乘客人数

extern int OdinWatNum;//当前乘客等待人数

extern int WinNum;//安检口工作数目

extern int PassengerArriveTask;//给到达的乘客安排窗口代码

extern clock\_t TimeStart;//开始时间

extern clock\_t TimeFinish;//结束时间

extern clock\_t TimeNow;//现在的时间

extern int NumOfWin;//安检口总数目

extern int MinRestSec;//安检口口最小休息时长

extern int MaxRestSec;//安检口最大休息时长

extern int MaxCustSingleLine：单队列最大等待乘客数

extern int MaxLines：蛇形缓冲区最多由MaxLines个直队组成

extern int MaxSeqLen：最大允许等待长度

extern int EasySeqLen: 短期等待长度

extern int MinCheck ：最少开放安检口数量

extern int MaxCheck ：最多开放安检口数量

extern int MaxCustCheck：安检口队伍最大长度

extern int MaxSec ：安检口最大安检时长单位是秒

extern int MinSec ：安检口最小安检时长

**4.系统模块划分**

**4.1系统模块划分**

state.c文件

input.c文件

main.c文件

output.c文件

maintain.c文件

1. 模块名称 main.c

模块功能简要描述：初始化界面，主线程。

1. 模块名称 input.c

模块功能简要描述：完成对用户输入的实现。

1. 模块名称 maintain.c

模块功能简要描述：完成对配置文件修改的实现。

1. 模块名称 state.c

模块功能简要描述：完成更新机场状态。

1. 模块名称 output.c

模块功能简要描述：完成对机场状态的输出

**4.2各模块接口说明**

1. 模块名称 global.h

模块功能简要描述：提供全局变量。

1. 模块名称 passenger.h

模块功能简要描述：提供乘客的结构体和乘客数组。

1. 模块名称 window.h

模块功能简要描述：提供窗口的结构体和窗口数组。

**4.3各模块函数说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 模块文件 | 模块说明 | 模块包含函数名 | 函数功能 |
| main.c | 数据初始化，程序运行开始 | void SetAndBegin() | 初始化 |
| int main() | 主进程 |
| input.c | 接受命令输入 | void MainInput() | 主菜单，营业前接收输入命令 |
| void BankOnServe() | 机场营业时接收输入命令 |
| void DistriNum(int) | 对到达的乘客分配号码和业务 |
| void WinRun() | 窗口处理客户 |
| long SerTaskTime() | 对不同的业务分配不同的服务时间 |
| maintain.c | 对配置文件修改 | void MainPara() | 修改配置文件 |
| void CusOutput() | 当有事件发生时，输出信息 |
| void NowTime() | 读取当前系统时间 |
| state.c | 储存当前机场信息 | void RegularOutput() | 周期性在命令行窗口输出状态 |
| Output.c | 输出机场信息 | void ResultOutput() | 输出机场统计信息 |

**4.4函数说明**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 函数原型 | 功能 | 参数 | 返回值 |
| 1 | void SetAndBegin() | 初始化，在main.c里，程序开始执行时先调用此函数 | void | void |
| 2 | int main() | 主线程 | void | 0 |
| 3 | void MainInput() | 营业前，接收输入命令 | void | void |
| 4 | void BankOnServe() | 营业时，接收输入命令，机场开始运营后调用 | void | void |
| 6 | void DistriNum(int) | 对客户分配号码，当input.c接收到客户，分配客户号码 | int 客户类型 | void |
| 8 | int CheckWin() | 当发出下班指令后，检查机场是否能关门 | int | int 0/1 |
| 9 | void WinRun() | 窗口处理乘客，不断地检查窗口状态并给客户分配窗口，安检乘客。 | void | void |
| 10 | long SerTaskTime() | 对不同的业务分配不同的服务时间，由DistriNum调用 | void | long 业务时长 |
| 11 | void NowTime() | 周期性读取当前系统时间 | void | void |
| 12 | void CusOutput() | 当有事件发生时，输出信息，由DistriNum调用 | void | void |
| 13 | void RegularOutput() | 周期性在命令行窗口输出当前机场状态，并储存到文件中 | void | void |
| 14 | void ResultOutput() | 在安检结束后，输出安检统计信息到屏幕及文件 | void | void |
| 15 | void MainPara() | 维护配置文件 | Void | void |

1. **核心算法设计**

**5.1 先来先服务策略：**

通过DistriNum()函数通过遍历客户数组实现将当前等待的最小号码放到等待对列的最前方，即先到的乘客先分配给窗口的功能。

**5.2 输入、输出和窗口服务：**

通过多线程实现三者“同时”运行

1）输入：MainInput函数实现“G”“X”“Q”等的输入和识别，以及客户办理业务的输入、办理时长的确定，同时在时间发生时，调用CusOutput函数进行输出。

2）输出：StoreInfo每隔5s读取当前营业厅状态一次，并调用RegularOutput函数进行输出

3）窗口服务：WinRun实现窗口服务的函数通过while循环不断检测、调整窗口状态以删除、服务乘客。