北京邮电大学

《操作系统课程设计》实验报告

**指导教师 李文生**

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | **班级** | **学号** | **备注** |
| **王一平** | **2019211305** | **2019211301** | **组长** |
| **杨清林** | **2019211305** | **2019211302** |  |
| **黎芷淇** | **2019211305** | **2019211314** |  |
| **闭宗越** | **2019211305** | **2019211382** |  |
|  |  |  |  |
|  |  |  |  |

**计算机学院**

**2022 年 5 月**

目录

[1 课程设计题目 3](#_Toc104227526)

[2 课程设计目标和要求 3](#_Toc104227527)

[2.1 设计目标 3](#_Toc104227528)

[2.2 基本要求 4](#_Toc104227529)

[3 需求分析 4](#_Toc104227530)

[3.1 进程管理 4](#_Toc104227531)

[3.2 内存管理 4](#_Toc104227532)

[3.3 文件系统 4](#_Toc104227533)

[3.4 设备管理 4](#_Toc104227534)

[3.5 图形化界面 4](#_Toc104227535)

[4 开发环境 5](#_Toc104227536)

[4.1 运行环境 5](#_Toc104227537)

[4.2 编码环境 5](#_Toc104227538)

[5 总体设计 5](#_Toc104227539)

[5.1 程序运行流程图 5](#_Toc104227540)

[5.2 项目结构 5](#_Toc104227541)

[5.2.1 主模块 5](#_Toc104227542)

[5.2.2 虚拟硬件 6](#_Toc104227543)

[5.2.3 操作系统 6](#_Toc104227544)

[5.2.4 图形化界面 8](#_Toc104227545)

[6 详细设计 9](#_Toc104227546)

[6.1 全局变量定义 9](#_Toc104227547)

[6.2 各模块具体功能说明 10](#_Toc104227548)

[6.2.1 主模块 10](#_Toc104227549)

[6.2.2 虚拟硬件 10](#_Toc104227550)

[6.2.3 操作系统 11](#_Toc104227551)

[6.2.4 监视程序 17](#_Toc104227552)

[7 程序清单（摘录主要模块代码即可） 18](#_Toc104227553)

[8 测试报告 18](#_Toc104227554)

[8.1 测试指令规范 18](#_Toc104227555)

[8.2 测试环境 20](#_Toc104227556)

[8.3 测试的功能 20](#_Toc104227557)

[8.4 针对每个功能的测试情况 20](#_Toc104227558)

[8.4.1 进程创建和控制、内存的分配和回收 22](#_Toc104227559)

[8.4.2 进程的调度、进程的状态变化、进程在各种队列之间的迁移 24](#_Toc104227560)

[8.4.3 文件系统功能 26](#_Toc104227561)

[9 课程设计总结 27](#_Toc104227562)

[9.1 设计过程中遇到或存在的主要问题及解决方案 27](#_Toc104227563)

[9.2 改进建议 27](#_Toc104227564)

[9.3 体会/收获 28](#_Toc104227565)

# 课程设计题目

操作系统模拟程序的设计与实现

# 课程设计目标和要求

## 设计目标

设计并实现一个具有操作系统基本功能的软件：

操作系统的基本功能：

➢进程管理功能。 进程创建（new）、进程调度（scheduling）、进程阻塞（block）、

进程唤醒（wakeup）、进程同步（synchronous）等。

➢内存管理功能。 进程存储空间的分配和回收，空闲空间的管理等。

➢文件系统。

➢设备管理。

➢UI界面。

➢中断机制

## 基本要求

要求完成的最小功能集合：

1. 进程管理和调度
2. 内存管理（存储分配与回收，进程交换）
3. 时钟管理：timer
4. 中断处理：中断响应、中断处理
5. 用图形界面展示多道程序并发执行的过程

# **需求分析**

## 进程管理

* 1. 需要实现进程的创建、控制、调度策略的选择和实现（RR调度策略）。
  2. 需要实现进程的状态切换、状态查询。
  3. 需要设计PCB的存储内容与结构。
  4. 需要完成中断类型的设置，完成中断处理的过程（断点设置、保存环境等）。
  5. 提供外中断和用户调用。

## 内存管理

1. 在进程创建和运行时对其进行内存分配。
2. 在进程运行结束时对其占用的内存资源进行回收。
3. 当使用页式管理时，需要完成页表的设计和页面置换等内容。

## 文件系统

1. 实现目录、文件的创建和删除。
2. 实现文件的读写。
3. 实现目录和文件名称、大小等基本信息查询。
4. 实现文件空间分配和回收。

## 设备管理

对设备访问请求进行排队和调度。

## 图形化界面

1. 展示CPU运行状态。
2. 展示各进程状态、进程队列状态。
3. 展示内存分配状态。
4. 展示文件系统信息。
5. 展示设备使用情况信息。

# 开发环境

## 运行环境

Windows10 操作系统

## 编码环境

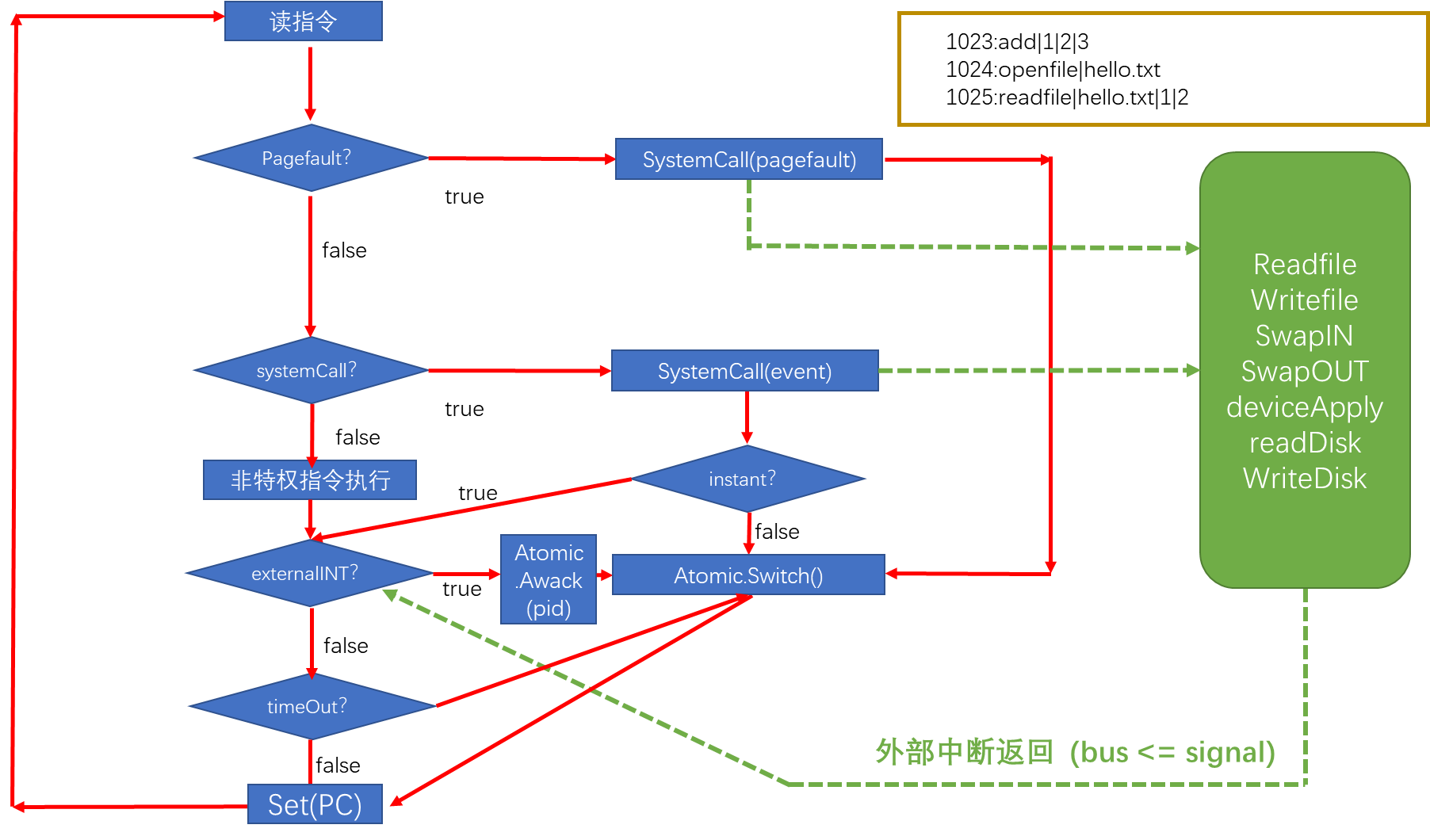
语言：Go

IDE：GoLang

前端：Vue开发环境

# 总体设计

## 程序运行流程图



## 项目结构

### 主模块

程序入口模块，调用其它功能组件，启动和维持整个程序运行

### 虚拟硬件

为所编写的操作系统提供虚拟的硬件环境，包括CPU、内存、硬盘和其它IO设备

1. **CPU**

模拟指令执行

1. **寄存器**

模拟指令执行所需寄存器

1. **内存**

模拟内存工作，可根据地址访问临时数据，系统关闭后数据丢失

1. **硬盘**

模拟硬盘工作，可根据地址访问数据，将数据写入文件

1. **其它设备**

虚拟其它IO设备，如打印机，仅可被一个进程占用

### 操作系统

1. **进程管理**
2. **进程状态切换**

提供各种进程状态切换相关原语如下

1. New 新建进程
2. Abort 判断进程状态
3. Switch 切换当前进程
4. Waiting 阻塞当前进程
5. Awake 唤醒指定进程
6. **中断处理**

内部中断：中断信号在CPU内部发出，包括trap（系统调用\*）、整数除0、越界中断和缺页中断。

外部中断：中断信号在外部发生：包括读写完成，设备完成，缺页返回等。

\*系统调用包括：文件的所有操作（新建、打开、关闭、读写）、内存的所有操作（申请、释放、分配）、设备的安装和申请操作、磁盘的读写和磁盘块的申请

1. **PCB**

管理进程PCB中的信息

1. **内存管理**

根据所需内存大小返回页表，表中包含所分配的指定数量页

1. **文件系统**

**a)文件目录**

采用树形结构组织文件，根据文件的路径及命令提供文件新建、读、写和删除等操作

**b)文件存储**

采用FAT的方式，以块（block）为单位存储文件

1. **设备管理**

**a)设备表**

进行设备状态维护

**b)设备调度**

对设备访问请求进行排队和调度

### 图形化界面

1. **操作系统桌面**

程序主窗口，展示各类应用信息

1. **CPU状态展示**

展示当前CPU运行状态，包括当前所执行的指令，后续要执行的指令，寄存器状态，中断相关信息等

1. **进程状态展示**

展示当前有哪些进程正在运行，它们各自的运行状态（new、ready等），以及进程排队情况

1. **内存状态展示**

展示当前内存使用状态，都有哪些进程占用了多少内存

1. **硬盘状态展示**

展示硬盘使用情况，占用量

1. **文件系统展示**

展示当前操作系统管理的文件及文件夹

1. **设备情况展示**

展示当前IO设备的占用情况，包括由哪些进程占用，占用时间等

1. **进程详情展示**

每个正在运行的进程都有一个窗口展示其PCB内容以及运行情况，包括当前正在运行的指令，IO设备占用情况等

# 详细设计

## 全局变量定义

var LOGS string *//日志生成*  
const ( *//wait类型和IO类型* MEM\_DISK = 0 *//wait磁盘* MEM\_PRINTER = 1*//wait打印机* IOREADDISK = 0 *//磁盘读命令* IOWRITEDISK = 1 *//磁盘写命令*)  
var BUS = make(chan Interrupt.Interrupt, 100000) *//模拟中断信号总线，往里面放中断信号*var PC int *//程序计数器，存储实际内存位置*var PCSave []int *//程序计数器暂存，中断处理程序时候立即触发，多级中断这里使用栈实现*var SuspendReadyQueue []int *//这个阻塞几乎不会发生，如果磁盘空间也不够，触发等待（不实现）*

## 各模块具体功能说明

### 主模块

调用时钟模块，发出时钟信号，启动程序

### 虚拟硬件

1. **CPU**
2. **指令执行**

模拟CPU每次执行用户程序一条指令（首先PC指向的位置是否发生缺页，如果缺页，缺页中断），如果该指令出现中断，那么写中断总线 ：BUS <- interrupt

1. **中断**

在时间片的末尾检测中断发生

1.中断未发生：继续执行下一条指令

2.中断发生：陷入操作系统**调用中断处理程序**

1. **内存**

使用数组模拟内存工作，单位为页

涉及操作：读/写内存

1. **硬盘**

使用数组模拟硬盘工作，单位为块

申请、读/写磁盘、释放、查看（返回）可用磁盘块位置（块号）

提供DMA

1. **其它设备**

虚拟其它IO设备，如打印机，仅可被一个进程占用

### 操作系统

**操作系统内核**

1. **进程管理**
2. **进程状态切换**

提供各种进程原语如下

1. **Process/Atomic/Waiting(stdin中断)** 将当前运行的进程设置为阻塞状态，并从ready队列中挑选一个新进程恢复状态，失败给出error
2. **Process/Atomic/Awake(pid, stdin中断)** 唤醒pid进程，加入到ready队列，并决定是否抢占，失败给出error
3. **Process/Atomic/Switch()** 进程发生切换，将当前进程放到ready对列，并选择一个在ready中的一个进程恢复状态，失败给出error
4. **pid = Process/Atomic/New()** 创建一个新进程并加到ready队列，并决定是否抢占，失败给出error
5. **PCB**

**存储内容**

1. 是否被占用
2. 父进程PID
3. 保存执行到哪的进程的PC值（**逻辑**地址） //移除running时候用
4. 保存执行到哪里的寄存器状态 //移除running时候用
5. 页表指针：存储的是 逻辑页号、物理页号（内存中的页号）、状态、访问字段、修改位、磁盘页号（被换出的时候存放的位置）
6. 进程内文件索引表指针：存储的是：文件名字、文件编号（句柄，用来在总索引表中找到该文件的物理位置信息）

具体实现

type PCB struct {Allocated bool *//进程ID是否被占据*  
 PID int *//进程ID*  
 Ppid int *//父进程ID*  
 PC int *//执行到代码的逻辑位置，当save的时候会更新* RegisterState Hardware.RegisterType *//寄存器值，当save的时候会更新* FileTable map[string]int *//文件系统* PageTable []PageAtom *//这里把页表存下来，真实的操作系统只会存一个指针,页表常驻内存，当swap的时候会更新，当前PC位置禁止被换出，防止影响load的原子操作* CodeSiz int *//规定PageTable,前CodeSiz块存储的是代码信息，后面的所有都是程序申请的内存* IsPageFault bool *//当前进程状态是否缺页* Disk int *//程序存放的外存地址，一经确认不在改变，虽然UNIX系统分了交换区和文件区* Flag bool *//换页标识*  
 Flag2 bool *//弃用，保留标识*  
 LogicAddr int  
}

**相关功能**

1. 创建一个PCB并返回PID，
2. 保存给定进程的PCB，
3. 加载某进程的PCB，
4. 删除某进程的PCB，
5. 换入换出内存时修改PCB的页表，
6. sort排出进程的优先级 **进程调度算法算法： RR**
7. **中断处理**
8. **中断类型**

const (  
 *//用户接口* USERINTERFACE = 0  
*//内中断 trap* STDIN = 1 *//标准输入* STDOUT = 2 *//标准输出* NEWPROCESS = 4 *//新建进程* ABORTPROCESS = 5 *//进程终止* PAGEFAULT = 6 *//缺页* READFILE = 8 *//读文件* WRITEFILE = 9 *//写文件* MEMORYAPPLY = 10 *//内存申请* DEVICEAPPLY = 12 *//设备请求* OPENFILE = 13 *//打开文件* DELETEFILE = 15 *//删除文件* CREATEFILE = 14 *//创建文件* CLOSEFILE = 16 *//关闭文件* UNDEFINE1 = 17  
 DEVICEINSTALL = 18  
  
*//外中断 interrupt* E\_STDIN = -1 *//标准输入完成* E\_STDOUT = -2 *//标准输出完成* E\_HARDERROR = -3 *//硬件错误* E\_READFILE = -8 *//读文件完成* E\_WRITEFILE = -9 *//写文件完成* E\_CREATEFILE = -10 *//创建文件完成* E\_PAGEFAULT = -6 *//调页完成* E\_DEVICEAPPLY = -12 *//设备处理完成* E\_CLOSEFILE = -13 *//关闭文件完成* E\_OPENFILE = -16 *//打开文件完成* E\_DISKREAD= -17 *//读磁盘完成* E\_DISKWRITE= -18 *//写磁盘完成*)

1. **中断信息存储数据结构**

**外中断设计：**

type Interrupt struct {  
 Type int *//中断类型，读/写 in out* SubType int *//子中断类型，例如如果是swap请求的磁盘读写，那么第一个Tyoe写磁盘，第二个Type写swap* Priority int *//优先级，默认0* Pid int *//请求/返回中断的进程id，就把我给你的pid原封不动返回* Device int *//什么设备返回的，磁盘操作还是打印机操作，原封不动返回deviceID* PhysicalPageNum int *//这次操作的内存页号* PhysicalBlockNum int *//这次操作的磁盘块号，这个和上面的原封不动返回 memoryADDR 和bufferADDR* Command string *//默认""*}  
**内中断设计：**  
type Trap struct {  
 Type int *//类型* Priority int *//优先级* Pid int *//请求/返回中断的进程id  
 //新建进程要给出的磁盘物理位置* PhysicalDiskPage []int  
 *//缺页* LogicPageNum int *//处理缺页中断的缺页的逻辑页号  
 //文件* FileLogicAddress int *//请求文件的逻辑位置* FilePath string *//请求文件的文件名* FileSiz int *//文件大小  
 //设备* Device int *//请求设备的设备名  
 //内存* MemSiz int  
 *//其他* Command string *//命令号（未实现）*}

1. **功能设计**

**提供函数Process/INTExecute(interrupt)：**

1. 立即保存当前环境，即保存各个寄存器的值
2. 查看发生什么类型的中断
3. 恢复环境，即加载寄存器的值

**提供外中断**

**在CPU时间片的最后监测系统中是否出现外部中断，有则处理**

1. **内存管理**

**功能设计**

1. 请求内存：Process/MemoryAllocate/MemoryAllocate(pid, in) 给定进程PID和申请的逻辑块号，需要申请一个内存，如果内存不足，可能会换出算法选定的内存快原有的东西，修改**PID进程的PCB**和**选定的内存块原有进程的PCB**，失败给出error；**这里需要调用IO处理（可能会有换出操作）**和 **磁盘操作（找一个合理的位置）**
2. 回收内存：Process/MemoryAllocate/MemoryRelease(块号) 给定块号，把它设为空闲块，失败给出error
3. swap：Process/MemoryAllocate/SwapInOut(pid, in) 将进程pid的逻辑块号为in的内存块读进内存，并换出选定的内存快原有的东西，修改**PID进程的PCB**和**选定的内存块原有进程的PCB**，失败给出error；**这里需要调用IO处理**和 **磁盘操作（找一个合理的位置）**

**页面置换算法采用：CLOCK**

1. **文件系统**

**功能设计**

1. 创建文件：传入参数创建文件的进程PID和文件信息，需要做这里需要修改文件表FileTable，不返回，失败给出error；**需要调用磁盘模块，申请**
2. 打开文件：传入打开文件的进程PID和文件路径，这里需要修改OpenFileTable、该进程PCB中的**进程内文件索引表**，不返回，失败给出error
3. 写文件：给出写文件进程PID和写文件的磁盘中的文件地址，以及内存中的位置和长度，不返回，失败给出error；**这里需要调用IO处理**
4. 读文件：同写
5. 关文件：给出PID，哪个文件，要求修改OpenFileTable、该进程PCB中的**进程内文件索引表**，不返回，失败给出error
6. 删除文件：给出PID，哪个文件，要求修改FileTable、OpenFileTable、该进程PCB中的**进程内文件索引表**，不返回，失败给出error；**需要调用磁盘模块，释放**

**算法: inode、树形目录结构、多级索引结构/混合索引**

1. **IO模块**
2. 磁盘IO处理，暂定使用DMA，**会产生磁盘IO中断**，会调用**磁盘**，需要指名读写操作，读写的位置（内存、磁盘的物理块号），读写长度、它会去**读/写磁盘**。
3. 设备IO处理，会产生**设备IO中断**，会调用**设备管理**

**DMA方法，与CPU分离，整体上表现为两个线程，通过 instructionChan异步交互**

### 监视程序

1. **图形化界面**

图形化界面以使用Vue框架的HTML页面实现，该页面通过HTTP接口与监视程序后端通信，以实现信息的展示与用户控制。

1. **操作系统桌面**

程序主窗口，展示各类应用信息

1. **进程状态展示**

展示当前有哪些进程正在运行，它们各自的运行状态（new、ready等），以及进程排队情况

1. **内存状态展示**

展示当前内存使用状态，都有哪些进程占用了多少内存

1. **设备情况展示**

展示当前IO设备的占用情况，包括由哪些进程占用，占用时间等

1. **监视程序后端**

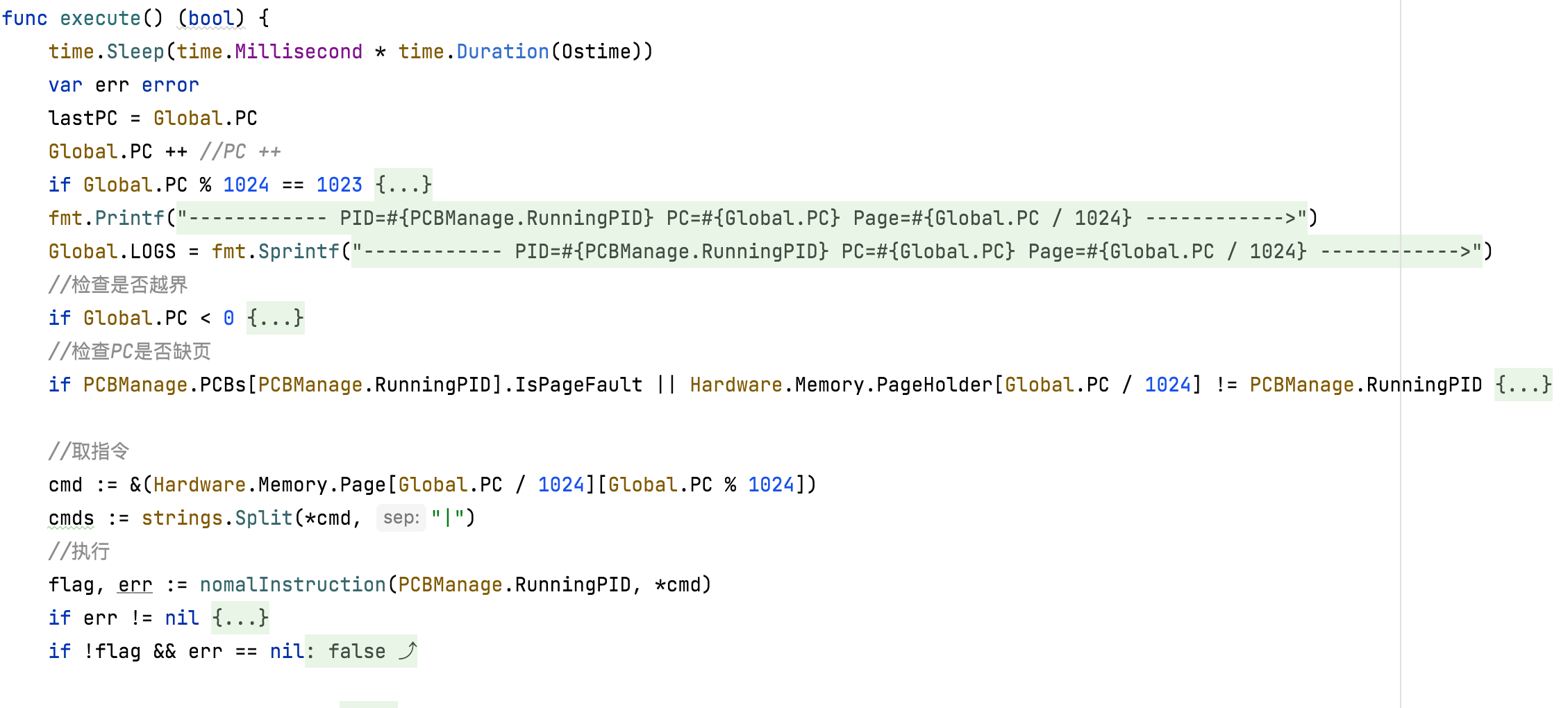
通过操作系统部分的各类接口获取当前操作系统的状态，并接收与处理前端的HTTP请求。

# 程序清单（摘录主要模块代码即可）

## CPU部分

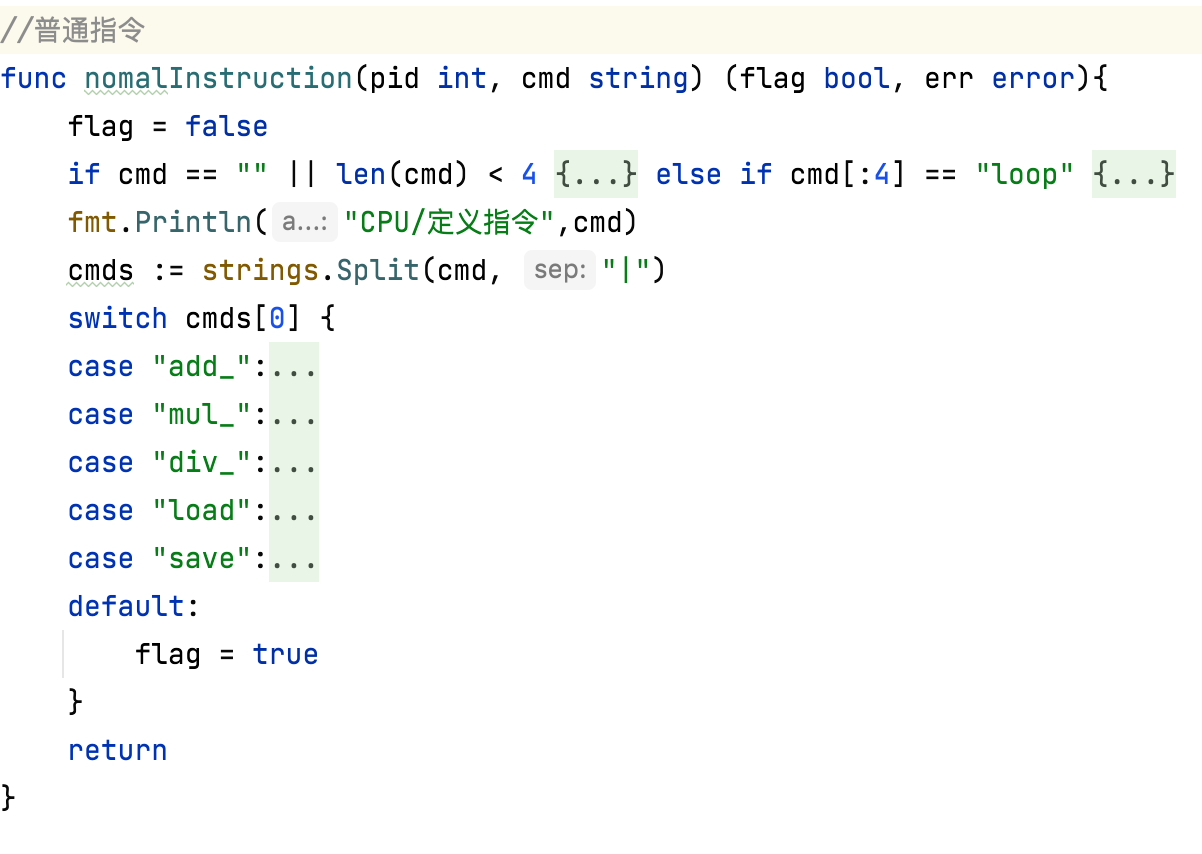
### 执行总体

判断缺页等一般错误、判断指令类型、对接特权指令陷入操作系统。

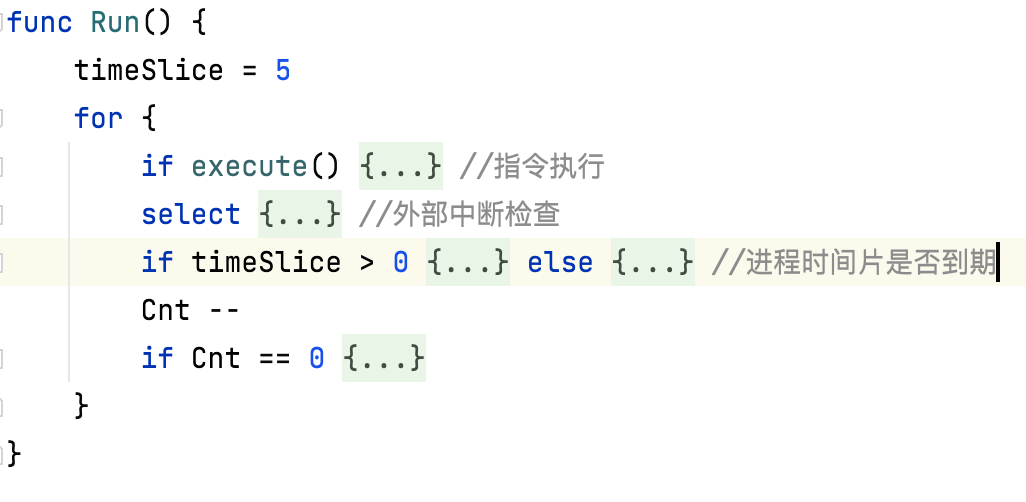




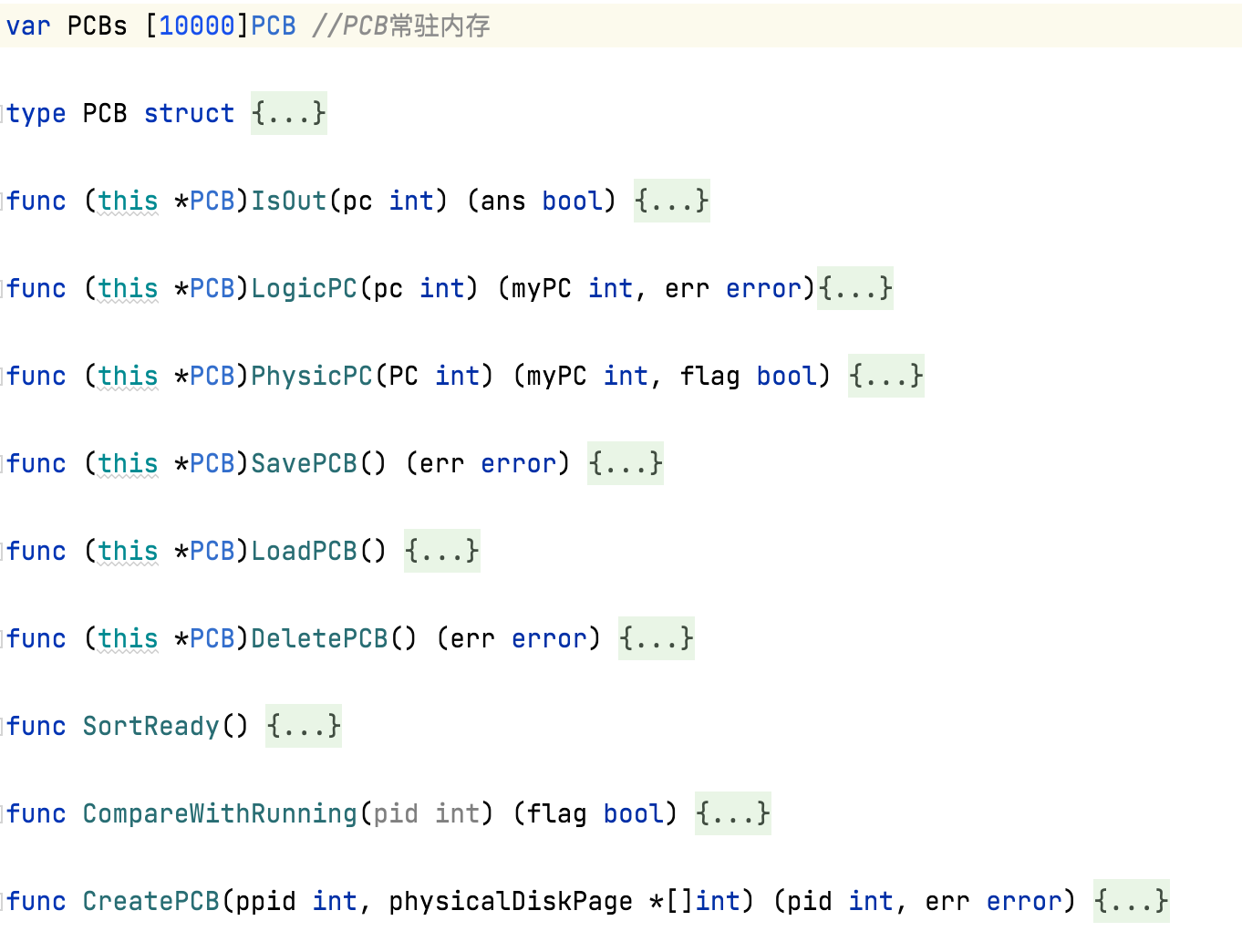
### 非特权指令



### 程序入口

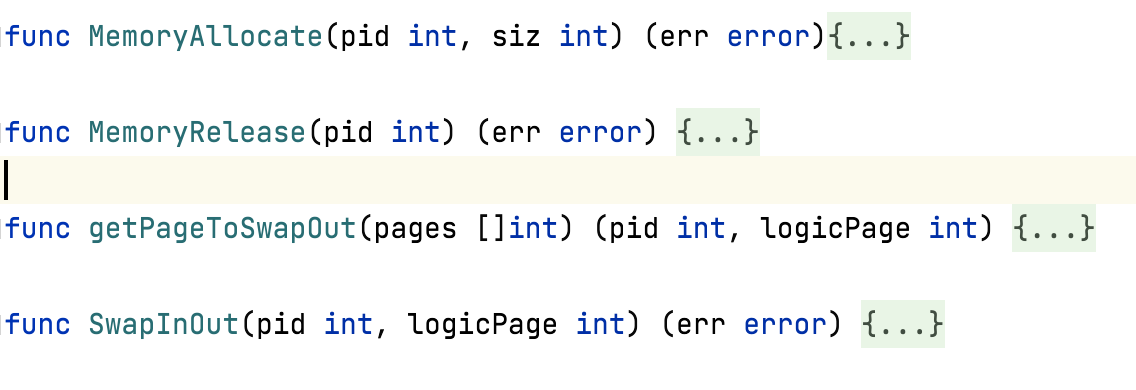


## PCB部分及其进程原语部分



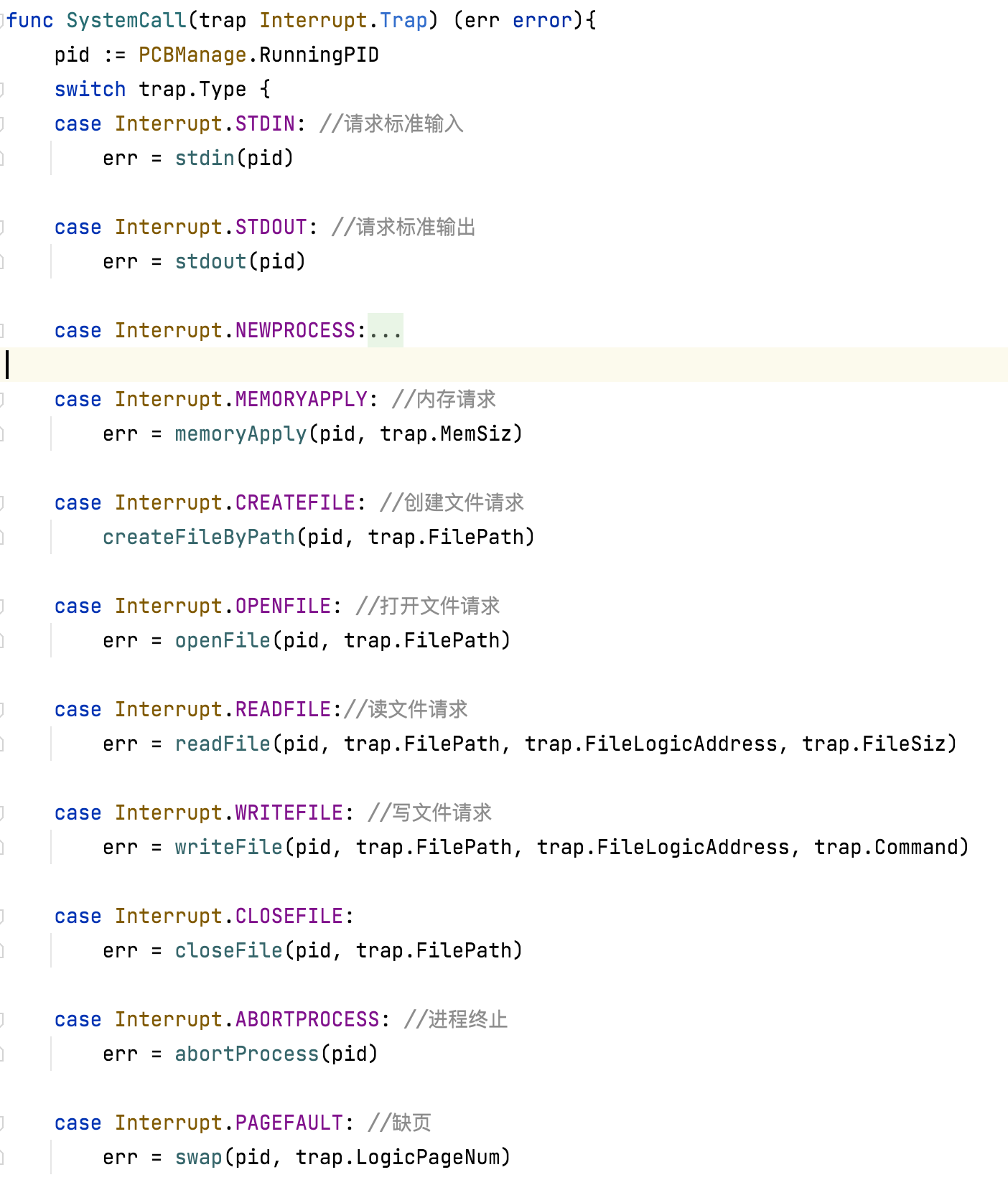


## 内存部分



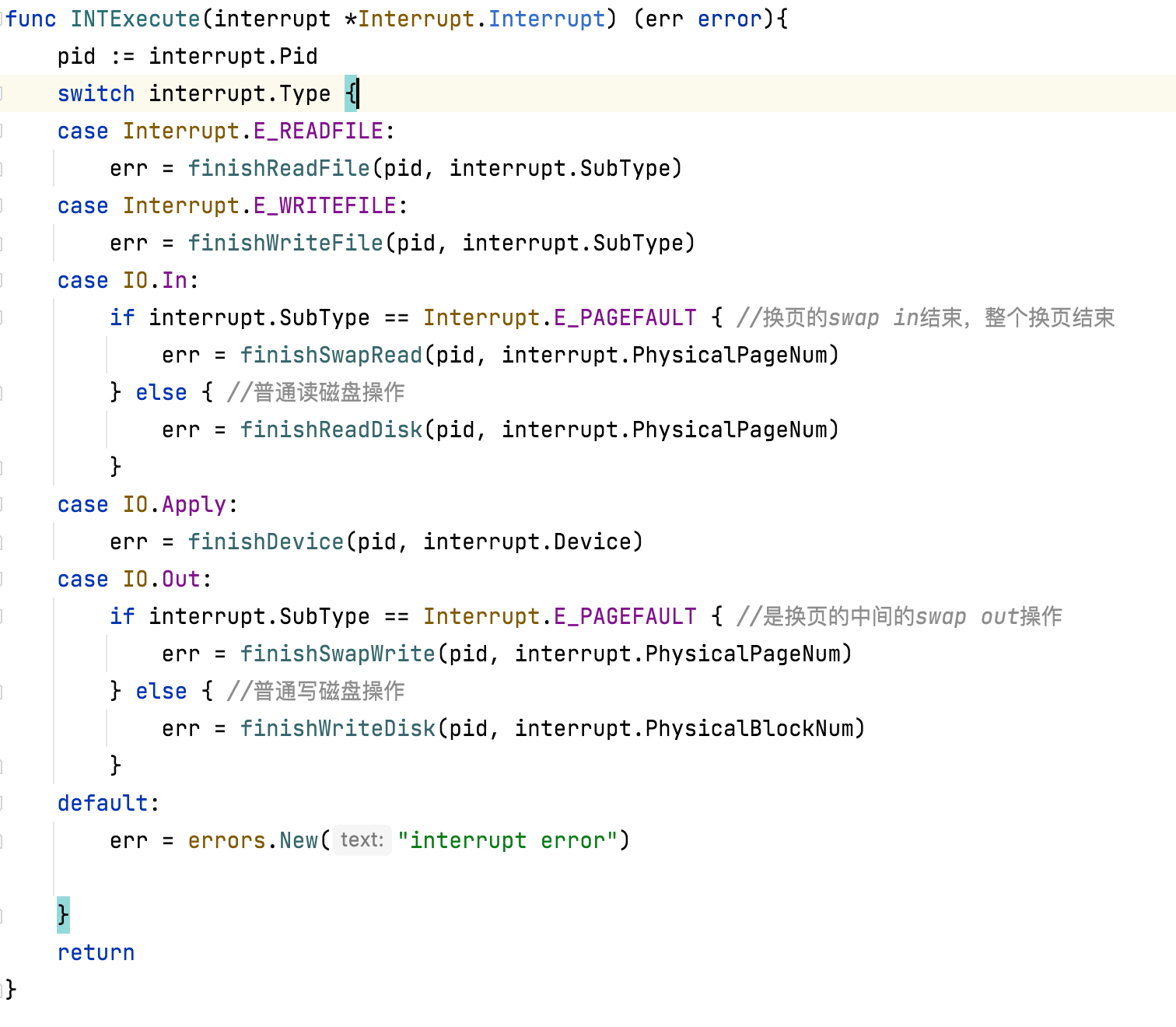
## 中断部分

### 内中断





### 外中断部分



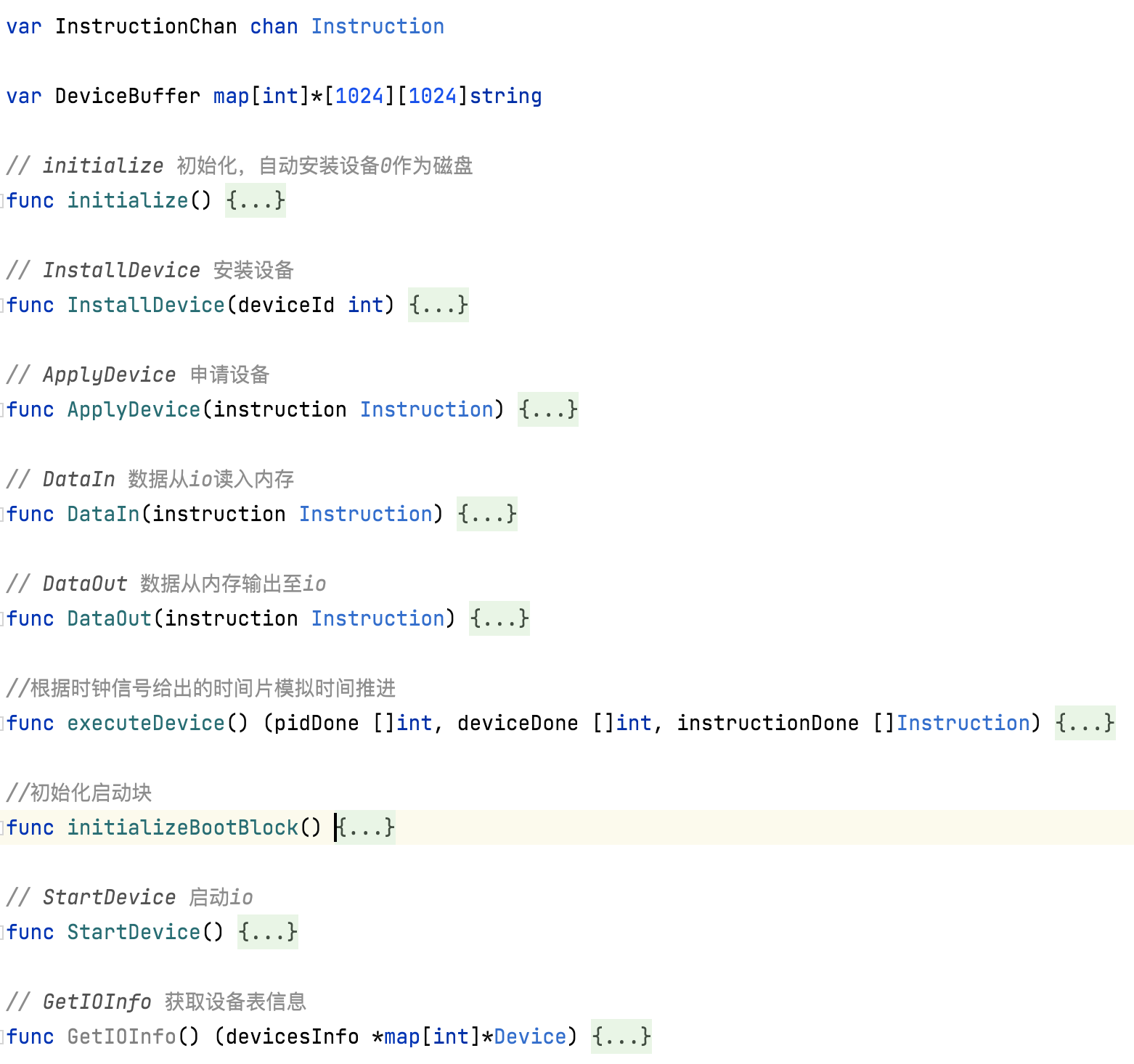
## 文件部分







## IO设计



## 监视模块

文本

描述已自动生成

## 服务器模块

文本

描述已自动生成

## 前端

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

# **测试报告**

## 测试指令规范

测试文件在in.in里面

指令规范：

1.输入有多少个作业  
2.输入第一个作业占多少个磁盘块  
3.输入这个作业对应的前端映射，如果不映射前端，填>2的数字；对应作业0，填0； 作业1，填1  
4.循环输入作业放在那几个磁盘块  
5.输入这个作业有多少个有效指令  
6.循环输入有效指令：   
 一行是指令在逻辑第几行，  
 一行是具体指令  
。  
。  
。  
8.默认填0（是否由1号进程启动）  
9.输入第二个作业 。。。  
。  
。   
。  
m.输入第二个作业。。。  
。

。

。  
n.输入CPU时间，单位ms  
n+1.输入CPU执行次数，-1表示一直执行

之后go build

./GOOS会在终端有提示

提供的指令：

//打开文件  
openfile|filename  
​  
//读文件,从逻辑位置logicStart读siz个字节  
readfile|filename|logicStart|siz  
​  
//写文件，从逻辑位置logicStart写siz个字节，内容是strings  
writefile|filename|logicStart|siz|strings  
​  
//创建文件  
createfile|filename  
​  
//关闭文件  
closefile|filename  
​  
//新建进程，后面是进程程序的物理块号  
newprocess|physicalPag1|physicalPag2|physicalPag3|....  
​  
//终止进程,终止当前进程  
abortprocess  
​  
//内存申请,num为申请的页数  
memoryapply|num  
​  
//设备申请（部分实现）  
deviceapply|deviceID  
​  
//loop循环指令,下面表示PC -= -2   
loop|-2  
​  
//寄存器加法，取寄存器a的数和寄存器b的数相加放在寄存器c  
add\_|a|b|c  
​  
//寄存器乘法，同上  
mul\_|a|b|c  
​  
//寄存器除法，同上  
div\_|a|b|c  
​  
//取数（同页面取数）逻辑地址m处取数，放到寄存器a  
load|m|a  
​  
//存数，从a寄存器取放到逻辑地址为m处  
save|a|m  
​

## 测试环境

硬件：PC机

软件：Windows操作系统、谷歌浏览器、本地2333端口

## 测试的功能

（1）进程创建和控制、内存的分配和回收

（2）进程的调度、进程的状态变化、进程在各种队列之间的迁移

（3）文件系统功能

## **针对每个功能的测试情况**

**总测试文本in.in：**

4  
1  
0  
1  
7  
2  
createfile|/hello  
4  
openfile|/hello  
6  
writefile|/hello|0|ABCDEF  
44  
memoryapply|20  
60  
readfile|/hello|1|3  
70  
closefile|/hello  
250  
abortprocess  
0  
1  
1  
30  
8  
2  
4  
6  
0  
7  
load|2|5  
8  
load|6|4  
10  
add\_|5|4|4  
34  
save|4|100  
46  
memoryapply|5  
90  
abortprocess  
0  
2  
2  
2  
3  
9  
2  
memoryapply|10  
4  
openfile|/hello  
5  
writefile|/hello|0|abcdef  
7  
readfile|/hello|2|3  
8  
closefile|/hello  
12  
newprocess|4  
13  
loop|-1024  
200  
abortprocess  
1100  
loop|950  
0  
1  
4  
4  
5  
4  
openfile|/hello  
5  
writefile|/hello|0|123456  
7  
readfile|/hello|2|3  
10  
closefile|/hello  
120  
abortprocess  
0  
200  
-1

### 进程创建和控制、内存的分配和回收

1. **测试用例**

作业1：

1.将立即数4存入逻辑地址2处

2.将立即数0存入逻辑地址6处

3.将逻辑地址2处的数放到寄存器5中

4.将逻辑地址6处的数放到寄存器4中

5.寄存器4和寄存器5的数相加，结果存入寄存器4

6.将寄存器4的数存入逻辑地址100处

7.申请5页的堆栈空间

1. **预期的结果**

系统会新建一个进程执行该作业，立即数可以顺利存入规定的逻辑地址，寄存器可以顺利从内存取数进行加法计算，并将结果存入指定逻辑位置。

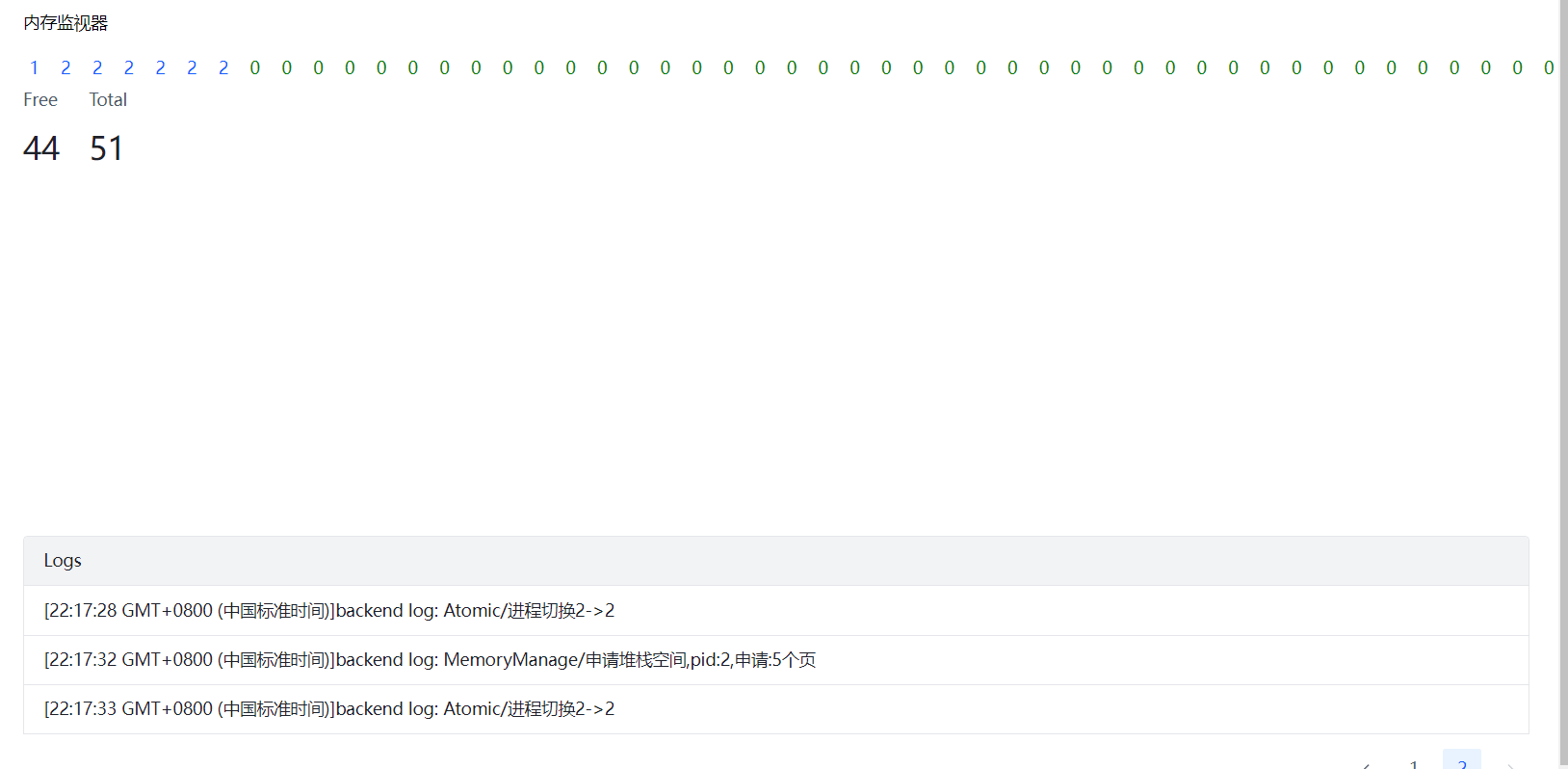
进程开始running时会产生缺页中断，进程可以顺利申请到5页的内存空间，进程运行结束后申请的资源会被释放。

1. **测试结果及其分析**

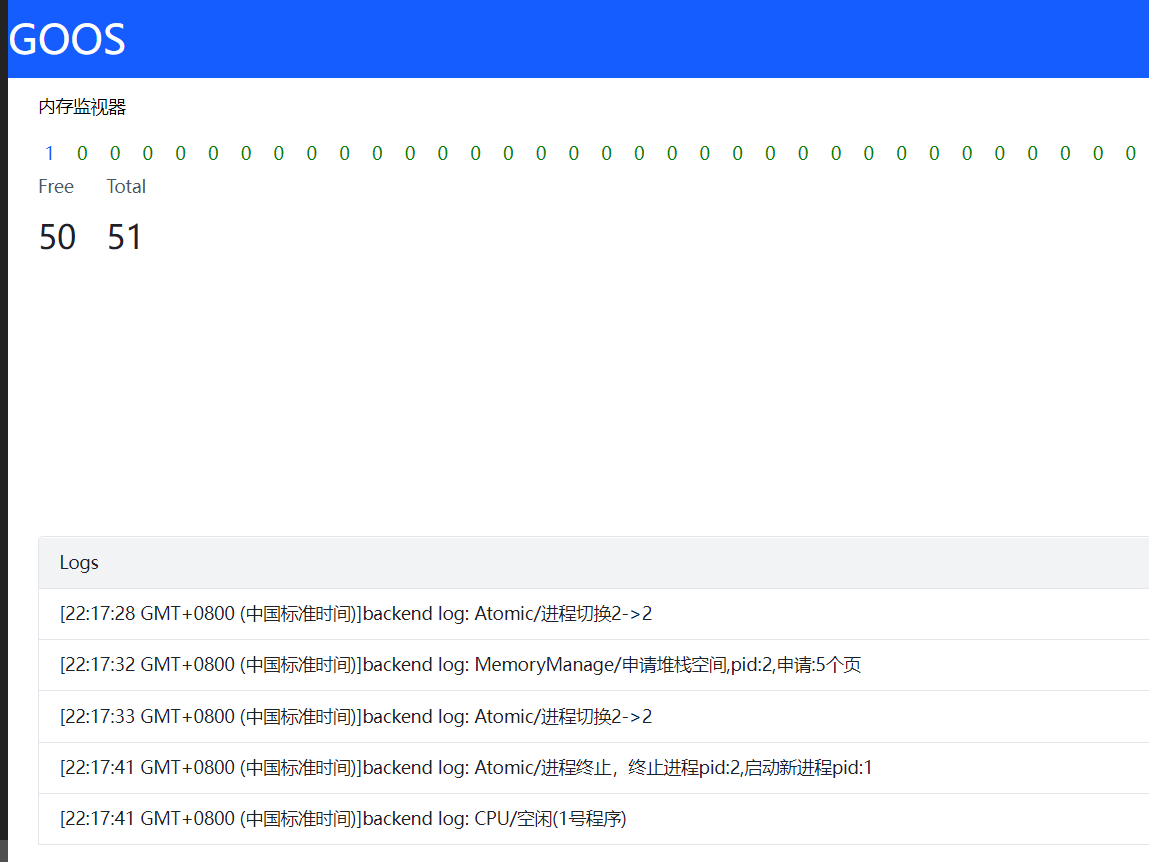
可以看到系统会新建一个2号进程执行作业1，同时发生了缺页中断，需要进行换页处理。



可以看到进程成功申请到了5页的堆栈空间。



可以看到进程运行结束之后，申请的内存资源会被回收。



### 进程的调度、进程的状态变化、进程在各种队列之间的迁移

1. **测试用例**

作业0：

1.创建/hello文件

2.打开/hello文件

3.向/hello文件写入内容"ABCDEF"

4.申请20页的堆栈空间

5.读/hello文件第一个逻辑位置起的三个字符

6.关闭文件/hello

作业1：

1.将立即数4存入逻辑地址2处

2.将立即数0存入逻辑地址6处

3.将逻辑地址2处的数放到寄存器5中

4.将逻辑地址6处的数放到寄存器4中

5.寄存器4和寄存器5的数相加，结果存入寄存器4

6.将寄存器4的数存入逻辑地址100处

7.申请5页的堆栈空间

1. **预期的结果**

1.启动作业后新建的进程会进行状态的变化。

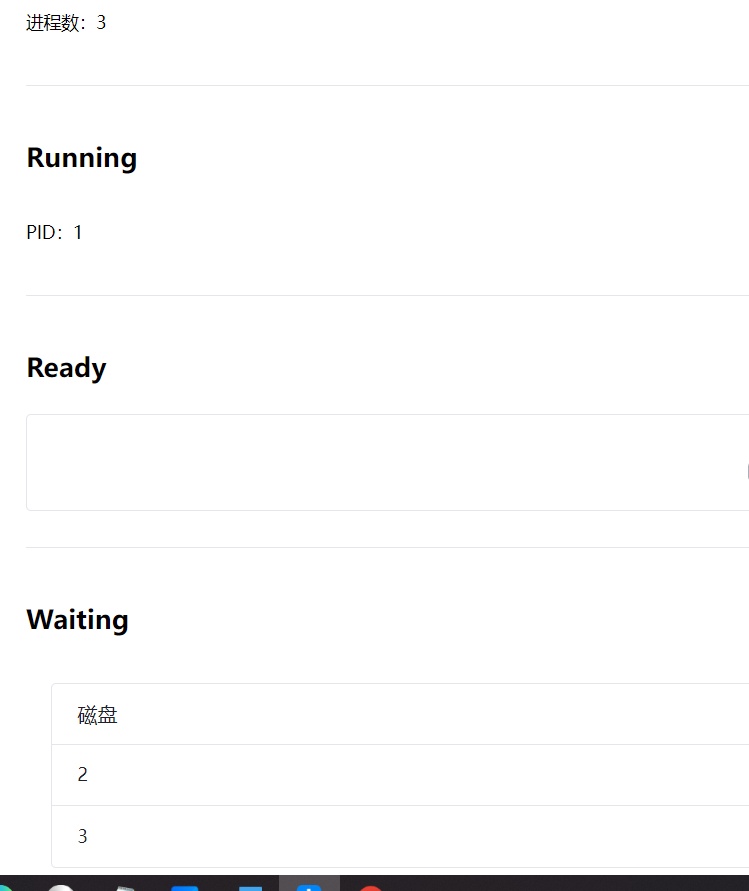
2.这里磁盘相当于0号设备。换页处理时进程会变成waiting状态，之后根据RR调度每隔一个时间片会进行进程的切换。

3.进程运行结束后申请的资源会被释放。

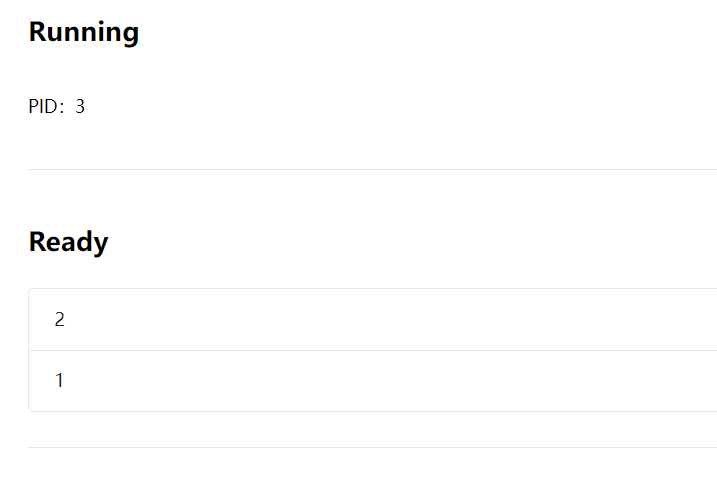
4.进程成功创建文件并进行文件读写，

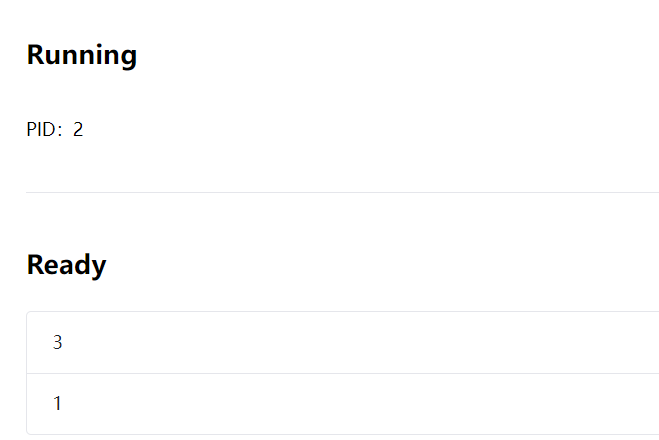
1. **测试结果及其分析**

缺页中断后分配物理页号。磁盘相当于0号设备，当进程发生缺页中断时会访问磁盘，调入相应页面，在waiting队列这里可以看到。



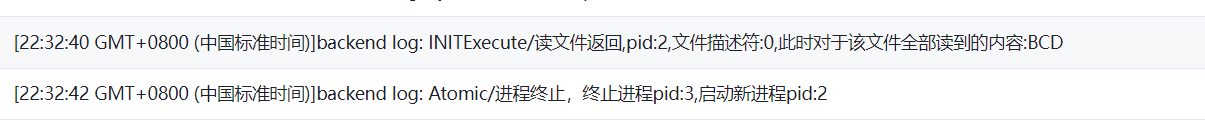
系统使用的是RR轮询调度，为了方便进行展示，我们模拟的时间片大概是5s，也就是说，如果进程的cpu执行时间超过5s，那么该进程会被抢占，并被放回到ready队列，这里看到进程2和进程3对cpu的交替抢占。





可以看到hello文件的创建和读写都成功了。





### 文件系统功能

1. **测试用例**

作业2：

1.申请10页的堆栈空间

2.打开/hello文件

3.向/hello文件写入内容"abcdef"

4.读/hello文件第二个逻辑位置起的三个字符

6.关闭文件/hello

7.新建进程，进程在第四块物理块

8.跳转到另一页再跳转回来

作业3（存在第四块物理块中）：

1.打开/hello文件

2.向/hello文件写入内容"123456"

3.读/hello文件第二个逻辑位置起的三个字符

4.关闭文件/hello

1. **预期的结果**

作业2可以成功对已创建的hello文件进行内容的改写，并能够顺利读取。

之后新建的进程可以再次对hello文件进行改写，同样可以顺利读取。

1. **测试结果及其分析**

可以看到作业2新建的进程成功把hello文件的内容改写为“abcdef”。

并且之后成功读取到逻辑起始为2的3个字符“cde”。



可以看到进程2新建的进程3成功把hello文件的内容改写为“123456”。

并且之后成功读取到逻辑起始为2的3个字符“345”。



# 课程设计总结

## 设计过程中遇到或存在的主要问题及解决方案

问题：处理不同页面的指令会出现问题，无法正常运行。

解决方案：调整页面置换算法，这样处理不同页面的指令不会出现问题。

## 改进建议

（1）调度算法不够丰富，目前只支持时间片轮转算法。

（2）系统的功能比较单薄，可以在文件操作等功能上继续丰富。如在文件关闭时，可选择保留原有内容。

（3）界面交互较为简单，可以继续加强用户的交互体验。

## 体会/收获

在此次操作系统模拟程序设计的实验中：

（1）我们加深了对操作系统进程管理、内存管理、文件管理和中断处理等基本内容的理解，更理解其内在的原理和工作机制。

（2）掌握了一定的操作系统设计与实现技术。

（3）同时，通过小组合作，提高了每个成员的合作、沟通和组织协调能力，深刻体会到集体的力量。

（4）该次实验还提高了我们的编程能力，使我们享受到编程的乐趣。