**华 南 农 业 大 学 数 学 与 信 息（软 件） 学 院**

《操作系统分析与设计实习》成绩单

开设时间：2021学年第一学期

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 小组成员、组内分工、工作量比例、各成员个人成绩 | | | | | | | | | | | |
| **学号** | 2019xx | **姓名** | xx | **分工** | 进程管理器、、桌面设计、界面美化、简易浏览器、报告攥写 | | | **工作量比例** | 33% | **成绩** | **100** |
| **学号** | 2019xx | **姓名** | xx | **分工** | 系统文件表、文件管理器、报告攥写 | | | **工作量比例** | 33% | **成绩** | **100** |
| **学号** | 2019xx | **姓名** | xx | **分工** | 占用管理器、报告攥写 | | | **工作量比例** | 33% | **成绩** | **100** |
| **学号** |  | **姓名** |  | **分工** |  | | | **工作量比例** |  | **成绩** |  |
| 实 验 题 目 | 题目六 模拟操作系统实现 | | | | | | | | | | |
| 各人分工与体会 | xx**：**本次实验，成员间沟通流畅，协作顺利。我负责的主要工作是进程管理器、桌面设计、窗口设计、界面美化、简易浏览器的开发并进行相关部分的报告攥写。同时，本次实验题目也是非常有趣的，通过将学到的知识以编程的方式进行实现，我对课本的知识也具有了更深刻的理解。无论从编程层面还是操作系统知识层面，都非常有收获。由于需要设计较好的界面效果，我根据自己想法浏览了大量关于JavaFx界面设计的技术博客，在设计过程中，还需要考虑窗口自适应等问题。在本次实验中，不仅让我对操作系统有了进一步的认识，同时也在一定程度上提高了我在Java界面开发方面的能力。  xx**：**在本次实验中，我负责的是系统文件表、文件管理器以及报告攥写。经过一个学期的对操作系统的学习后，将理论应用于实践，模拟写出一个操作系统。在这过程中，为了实现预想的功能，我不断学习新的知识，在对操作系统不断熟悉的同时提高了编程能力。团队协作是最重要的，一个人难以完成这样一个这样大的任务，在实验过程中，相互协助，优劣互补，最终较为完美的完成这一次实验。  xx**：**本次实验中，我们团队分工明确，我负责的是占用管理器以及报告攥写。基于一个学期的操作系统的学习，结合面向对象编程的思想，再搭配上新掌握的JavaFx界面开发技术，我们实现了这次操作系统的模拟。学习了新知识后，要进行实操才能够真正的掌握。更重要的是，让我清楚了团队合作不是说单纯分配任务就可完成的事，要经过不断地沟通交流。总的来说，这一次团队任务让我受益很多，让我的各方面能力都更进了一步。 | | | | | | | | | | |
| 教 师 评 语 |  | | | | | | | | | | |
|  |  | | | | | **教师签名** | 孙微微 | | | | |

目录

[1 需求分析 1](#_Toc87041089)

[1.1系统文件表 1](#_Toc87041090)

[1.2文件管理器 1](#_Toc87041091)

[1.3进程管理器 1](#_Toc87041092)

[1.4占用管理器 2](#_Toc87041093)

[1.5 界面设计 2](#_Toc87041094)

[2概要设计 2](#_Toc87041095)

[2.1人员分工 2](#_Toc87041096)

[2.1.1前端分工 2](#_Toc87041097)

[2.1.2后端分工 3](#_Toc87041098)

[2.2开发说明 3](#_Toc87041099)

[2.3系统架构图 3](#_Toc87041100)

[2.4模块层级图 3](#_Toc87041101)

[3详细设计 4](#_Toc87041102)

[3.1操作系统桌面 4](#_Toc87041103)

[3.1.1功能概述 4](#_Toc87041104)

[3.1.2主要模块 5](#_Toc87041105)

[3.1.3前端 5](#_Toc87041106)

[3.2 系统文件表 6](#_Toc87041107)

[3.2.1功能概述 6](#_Toc87041108)

[3.2.2主要模块 6](#_Toc87041109)

[3.2.3前端 6](#_Toc87041110)

[3.2.4后端 7](#_Toc87041111)

[3.3 文件管理器 8](#_Toc87041112)

[3.3.1功能概述 8](#_Toc87041113)

[3.3.2主要模块 9](#_Toc87041114)

[3.3.3前端 9](#_Toc87041115)

[3.3.4后端 10](#_Toc87041116)

[3.4进程管理器 21](#_Toc87041117)

[3.4.1功能概述 21](#_Toc87041118)

[3.4.2主要模块 22](#_Toc87041119)

[3.4.3前端 22](#_Toc87041120)

[3.4.4后端 22](#_Toc87041121)

[3.5占用管理器 27](#_Toc87041122)

[3.5.1功能概述 27](#_Toc87041123)

[3.5.2主要模块 27](#_Toc87041124)

[3.5.3前端 27](#_Toc87041125)

[3.5.4后端 28](#_Toc87041126)

[3.6简易浏览器 29](#_Toc87041127)

[3.6.1功能概述 29](#_Toc87041128)

[3.6.2 实现说明 30](#_Toc87041129)

[3.7界面实时更新线程 30](#_Toc87041130)

[3.7.1功能概述 30](#_Toc87041131)

[3.7.2 实现说明 30](#_Toc87041132)

[4调试分析 31](#_Toc87041133)

[4.1调试过程中遇到的问题是如何解决的以及对设计与实现的讨论与分析 31](#_Toc87041134)

[4.2.算法的时间复杂性和改进思想 31](#_Toc87041135)

[4.3. 设计过程的经验和体会 32](#_Toc87041136)

[4.3.1xx 32](#_Toc87041137)

[4.3.2xx 32](#_Toc87041138)

[4.3.3xx 32](#_Toc87041139)

[4.4实现过程中遇到的主要问题及解决方法 32](#_Toc87041140)

[4.4.1进程队列问题与解决方案 32](#_Toc87041141)

[4.4.2界面更新问题与解决方案 33](#_Toc87041142)

[5用户使用说明 33](#_Toc87041143)

[5.1简易浏览器 33](#_Toc87041144)

[5.2系统文件表 33](#_Toc87041145)

[5.3文件管理器 33](#_Toc87041146)

[5.4进程管理器 33](#_Toc87041147)

[5.5占用管理器 34](#_Toc87041148)

[5.6帮助 34](#_Toc87041149)

[6测试与运行结果 34](#_Toc87041150)

[6.1系统桌面 34](#_Toc87041151)

[6.1简易浏览器 35](#_Toc87041152)

[6.2系统文件表 36](#_Toc87041153)

[6.3文件管理器 36](#_Toc87041154)

[6.4进程管理器 39](#_Toc87041155)

[6.5占用管理器 41](#_Toc87041156)

[6.6帮助 42](#_Toc87041157)

1 需求分析

满足文件管理与用户接口、内存管理器、设备管理、进程管理、界面显示，并进行模拟操作系统设计。

1.1系统文件表

系统文件表，其主要作用是显示系统可执行文件和对应的可执行文件内容。左侧表格是可执行文件表，右侧表格是可执行文件的内容。点击左侧表格任意一个文件，相应的指令内容则会在右侧表格显示。

系统开启时，会随机生成10个可执行文件，用于后续的进程创建、内存分配、进程执行/调度和设备分配。

可执行文件包含五种指令：

1）x=？：给x赋值。

2）x++ ：x加1。

3）x - - ：x减1。

4）! ? ? ：！是指令前缀；第一个？为A、B、C中的某个设备；第一个？为设备使用的时间。

5）end ：表示程序结束。

其他说明：可执行文件中x的初值默认为0。

1.2文件管理器

文件管理器，其主要作用是对文件进行操作，以及显示磁盘使用情况。

1）当启动该软件时，会询问是否读取历史数据，若是，则进行读取；若否，则清空之前的数据。

2）第一个模块是文件管理页面。在该页面，通过鼠标及键盘操作，可进行对文件的新建、读写、移动、复制、删除、属性查看等操作，可进行对文件夹的新建、删除、移动、复制、属性查看等操作。

3）第二个模块是磁盘的使用情况页面。在该页面，显示了磁盘的占用情况：①占用分布图；②占用盘块、索引、类型、内容等信息；③当前打开的文件对应文件名、打开方式、起始盘块、文件长度、文件路径等信息。

1.3进程管理器

进程管理器，主要作用是可视化进程的运行情况。

1）单独显示当前运行进程的编号、执行指令、数据寄存器的值、剩余时间片。[默认时间片为6]。

2）显示当前进程的执行进度，高亮当前执行指令。

3）显示进程详表，具体包括：进程编号、进程状态、执行文件、设备使用情况、进程控制块、当前执行结果、进程完成进度。

4）进程详表设置：包括显示设置、标记设置和进程控制。

①显示设置：用于选择想要显示的进程，包括当前进程、新建进程、就绪进程、阻塞进程、销毁进程、显示所有。

②标记设置：用于选择想要高亮的进程，包括新建进程、就绪进程、运行进程、阻塞进程、销毁进程。

③进程控制：用于控制继续新建进程或者暂停新建进程，以及控制进程运行的倍速（1倍速，2倍速，4倍速，8倍速）。

1.4占用管理器

占用管理器，主要作用是可视化用户区内存、磁盘、设备、PCB的占用情况。

1）用户区内存占用可视化：显示内存占用比、内存占用分布情况。

2）磁盘占用可视化：显示磁盘占用比、磁盘占用分布情况。

3）设备占用可视化：显示设备占用比，设备占用分布情况。

4）PCB占用可视化：显示PCB占用比，PCB占用分布情况，以及占用PCB的各进程状态（新建、就绪、运行、阻塞）。

1.5 界面设计

1）操作系统界面的设计与展示，包括系统桌面与系统软件两大部分。

2）系统桌面具体包括：软件图标、图标栏、时间显示、控制按钮。

3）系统软件具体包括：可执行文件表、文件管理器、进程管理器、占用管理器、帮助。除此之外，还包括简易浏览器。

2概要设计

2.1人员分工

2.1.1前端分工

1）蔡 棱：进程管理器、简易浏览器、桌面设计、窗口设计、界面美化

2）黄威雄：系统文件表、文件管理器

3）温翔昊：占用管理器

2.1.2后端分工

1）蔡 棱：进程调度

2）黄威雄：文件管理与用户接口

3）温翔昊：内存管理、设备管理

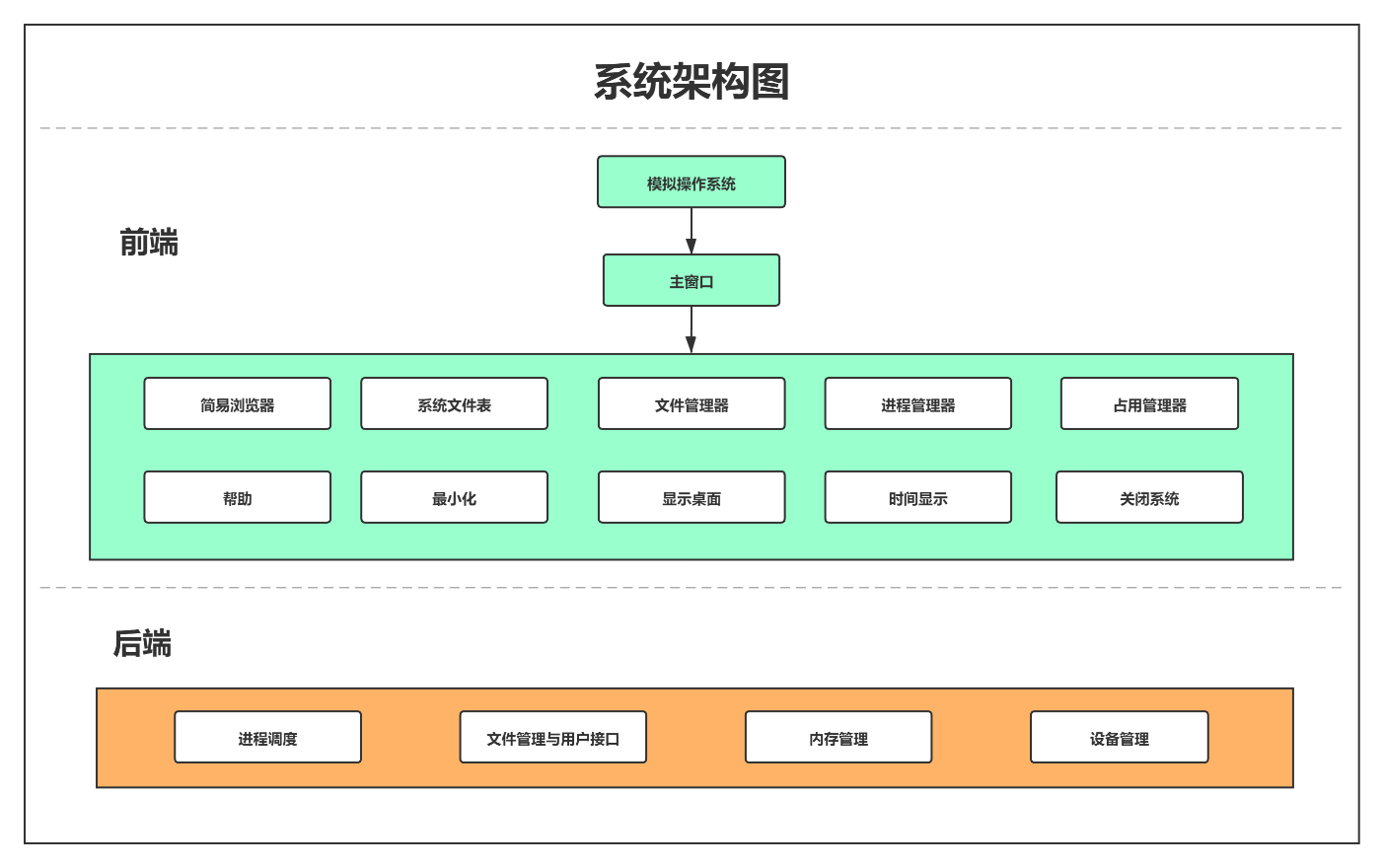
2.2开发说明

开发工具：IntelliJ IDEA、JavaFX Scene Builder

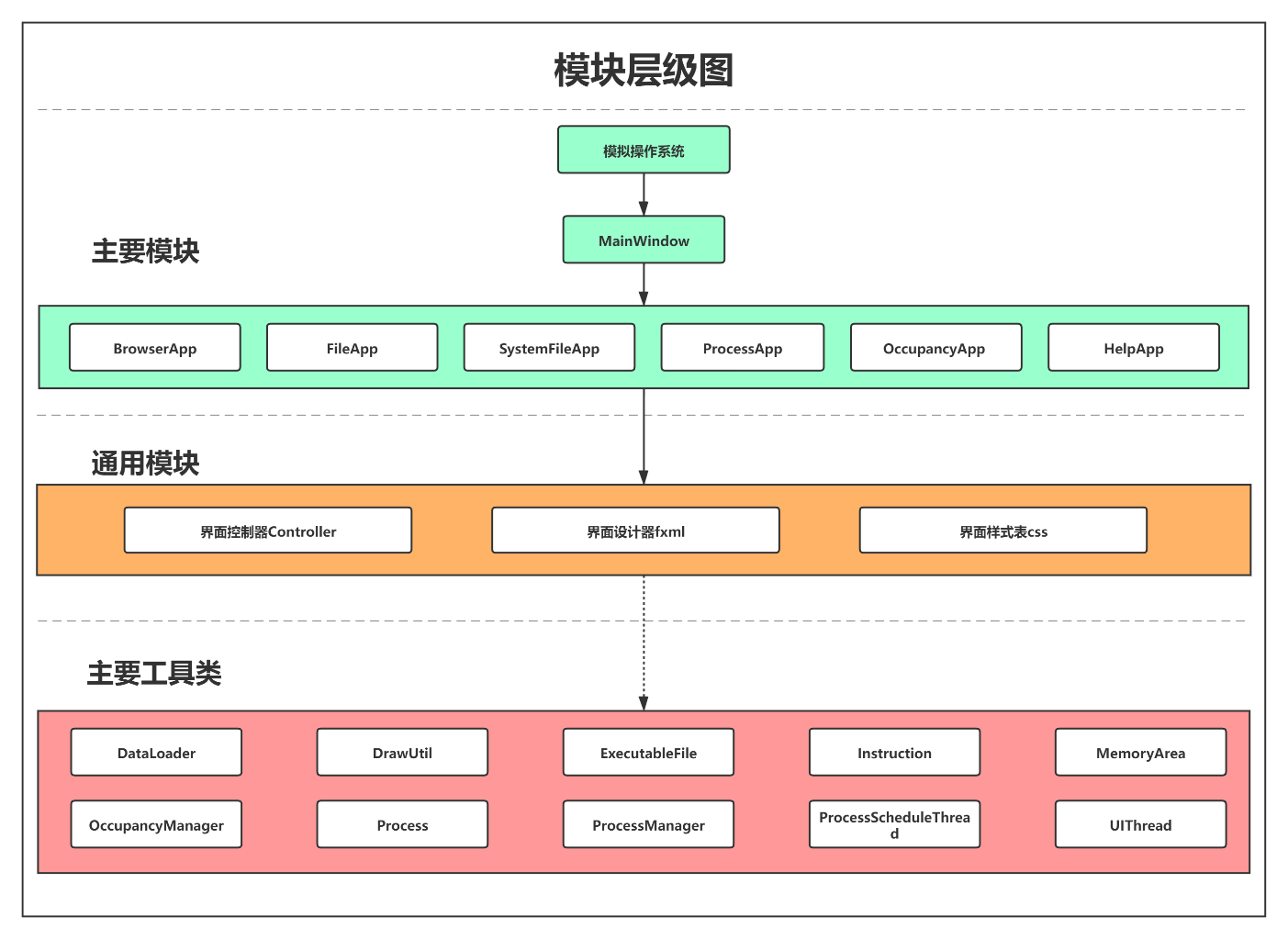
JDK版本：java/jdk1.8.0\_301

兼容系统：Windows

2.3系统架构图



2.4模块层级图



3详细设计

本程序主要模仿操作系统的桌面，将不同模块以“软件”的方式进行呈现。各软件在界面交互层面上，主要通过通过界面控制器Controller、界面设计器fxml，界面样式表css来实现，在数据操作层面上，主要通过预先设计好的类进行数据存储，并通过线程进行更新。

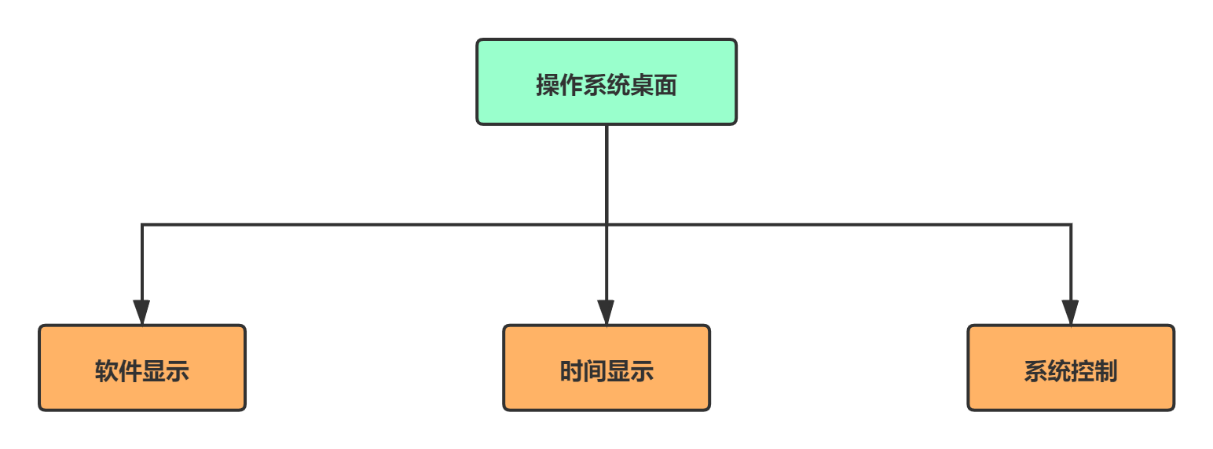
为避免篇幅过长影响阅读，仅描述程序设计的大概思路。若出现功能相似的模块、函数、数据类型，或实现逻辑较为简单，将避免详细介绍。

需要注意的是，以下介绍，均是基于java文件，主要描述后端的核心模块。其中，前端对组件进行布局的fxml文件、对组件进行美化的css文件，由于不涉及算法逻辑，不会过多提及。

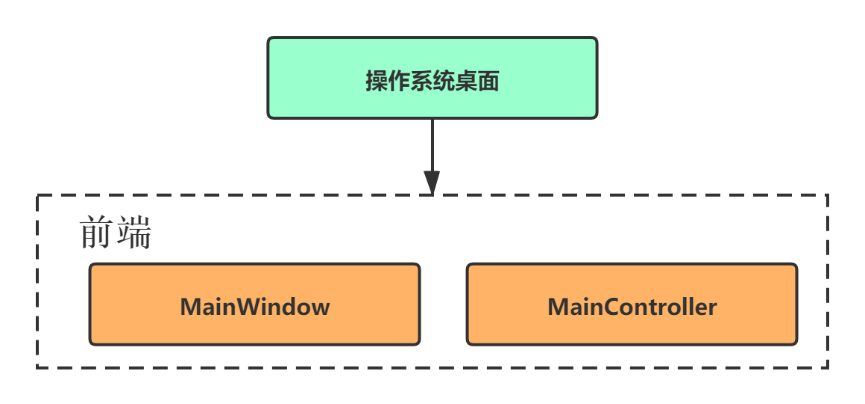
3.1操作系统桌面

主要包括桌面设计和软件的窗口样式设计。

3.1.1功能概述



3.1.2主要模块



3.1.3前端

（1）界面布局上，主要包括软件栏和任务栏。同时，在各软件窗口上，均作了风格统一的操作。第一，对所有的窗口标题栏进行自定义，并加入相关的监听事件，实现最小化、放大、关闭的操作，实现界面美化。第二，对组件的布局以及相关参数进行调整，实现窗口自适应。

（2）系统文件表窗口主程序MainWindow。主要作用是利用加载已设计的fxml文件进行窗口创建，并绑定进程管理窗口控制器MainController用以窗口创建和界面交互。

（3）系统文件表窗口控制器MainController。主要作用是MainWindow中的组件进行操作。以下进行相关部分的简要说明：

①初始化：依次调用图标初始化、时间初始化、进程初始化、UI线程初始化等。

②时间初始化：第一，新建Data对象，用来获取时间，并生成对应的文本字符串；第二，设置时间组件的文本显示。

③图标初始化：第一，根据所调参数，对相关组件进行设置；第二，对每个按钮增加监听事件，以读取鼠标操作，并进行相应处理；第三，将相关组件传给UI线程，方便其定时对界面组件进行更新。

④适应窗口：根据预先调好的参数，进行各组件的设置。

⑤软件启动：第一，检测该软件此前是否关闭但未清理，若是，先清理之前的缓存舞台；第二，若该软件为打开状态，则直接显示即可；否则，新建对应的软件程序，进行显示，并做好记录。

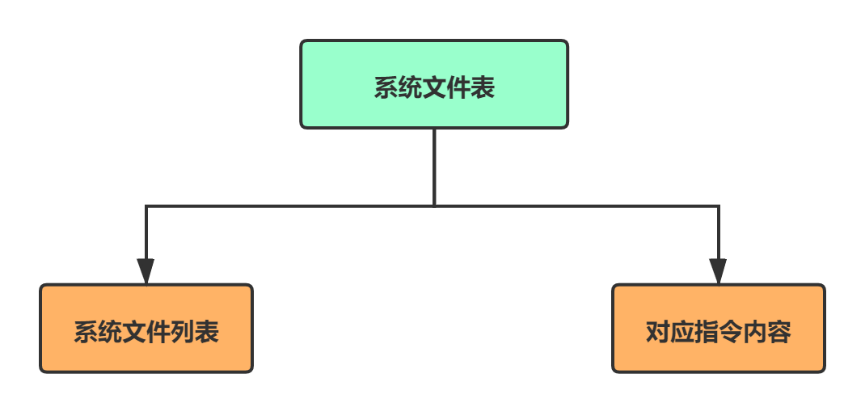
⑥最小化系统：系统中保存两个舞台队列：第一个舞台队列包含所有主要软件的舞台；第二个舞台队列包含文件管理器附属窗口的舞台队列。当点击“最小化”按钮时，遍历舞台队列中的所有舞台，依次进行最小化操作。最后把主舞台最小化。

⑦显示桌面：第一，设置标志位isTop，判断当前是否为处于桌面状态；第二，如果处于桌面状态，则遍历舞台队列中的所有舞台，依次进行最小化操作，更新isTop；第三，如果不处于桌面状态，则遍历舞台队列中的所有舞台，依次进行显示操作。

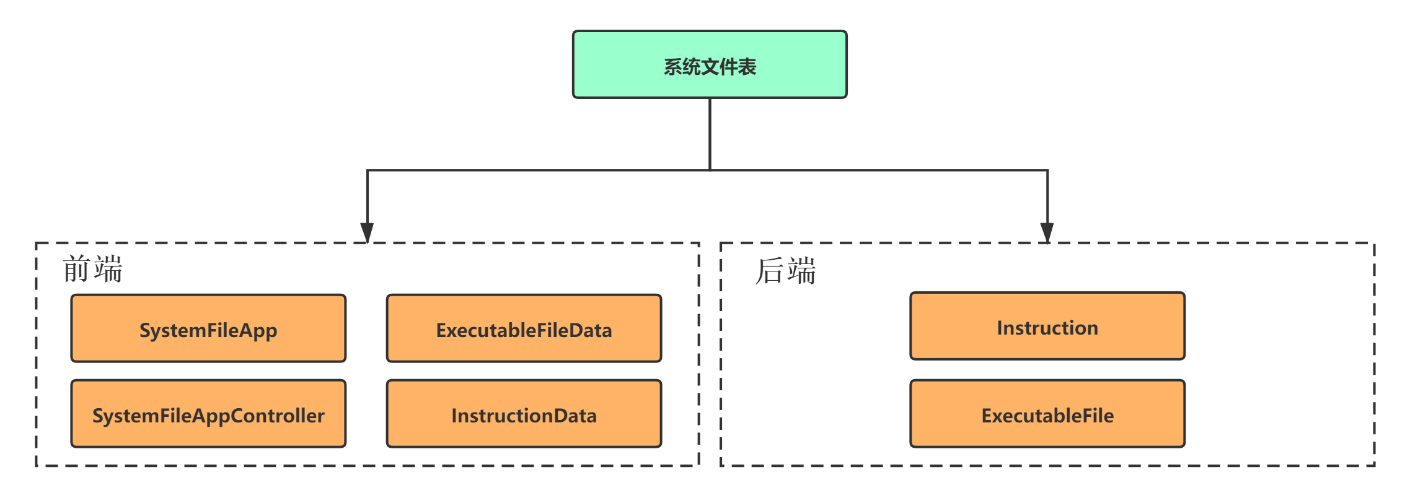
⑧关闭系统：具体包括：关闭时保存文件数据、系统终止、平台终止、舞台关闭等操作。

3.2 系统文件表

3.2.1功能概述



3.2.2主要模块



3.2.3前端

（1）界面布局上，主要包括系统文件列表、系统文件对应指令内容表。

（2）系统文件表窗口主程序SystemFileApp。主要作用是利用加载已设计的fxml文件进行窗口创建，并绑定进程管理窗口控制器SystemFileAppController用以窗口创建和界面交互。

（3）指令数据类InstructionData和可执行文件数据类ExecutableFileData。由于JavaFx中的表格不能直接读取常规数据类型，因此需要进行类型替换，以方便表格更新。此类只是对指令类Instruction和可执行文件类ExecutableFile的数据类型进行替换，并无其他细节。

（4）系统文件表窗口控制器SystemFileAppController。主要作用是将界面中的按钮、表格等组件传送给UI线程，方便后续UI线程统一更新界面。

3.2.4后端

（1）指令类Instruction。主要作用：对指令类别、指令中需要的值等一系列属性进行封装；构建指令不同情况的构造方法以及toString方法。以下进行详细说明：

Instruction类属性

|  |  |
| --- | --- |
| category | 指令类别 |
| additional | 指令中需要的值（第一位置） |
| additional1 | 指令中需要的值（第二位置） |

Instruction类方法

|  |  |
| --- | --- |
| Instruction | 构造方法 |
| toString | 指令实例转化成字符串 |

Instruction方法

|  |  |
| --- | --- |
| 步骤 | 对指令的类别category、指令中需要的值additional、additional1进行赋值。 |

toString方法

|  |  |
| --- | --- |
| 步骤1 | 首先声明String类型的解码串decode，对指令进行解码。 |
| 步骤2 | 如果指令类别为0（赋值操作），则decode等于“x”加上赋值位additional。 |
| 步骤3 | 如果指令类别为1（自增操作），则decode加上“x++”。如果指令类别为2（自减操作），则decode加上“x--”。 |
| 步骤4 | 如果指令类别为3（请求设备操作），则doced等于“！”加上additional对应字母（设备类别）+additional1（设备占用时间）。 |
| 步骤5 | 如果指令类别为4（终止操作），则doced等于“end”。 |
| 步骤6 | 如果指令类别为-1，则doced等于“闲逛”。 |
| 步骤7 | 返回decode，作为转化结果。 |

（2）可执行文件类ExecutableFile。主要作用：对文件编号、指令列表、文件名、文件后缀等一系列属性进行封装；构建ExecutableFile的构造方法和以及toString方法。以下进行详细说明：

ExecutableFile类属性

|  |  |
| --- | --- |
| id | 文件编号 |
| instructionArray | 指令列表 |
| name | 文件名 |
| addName | 文件后缀 |

ExecutableFile类方法

|  |  |
| --- | --- |
| ExecutableFile | 构造方法 |
| toString | ExecutableFile实例转化成字符串 |

ExecutableFile方法

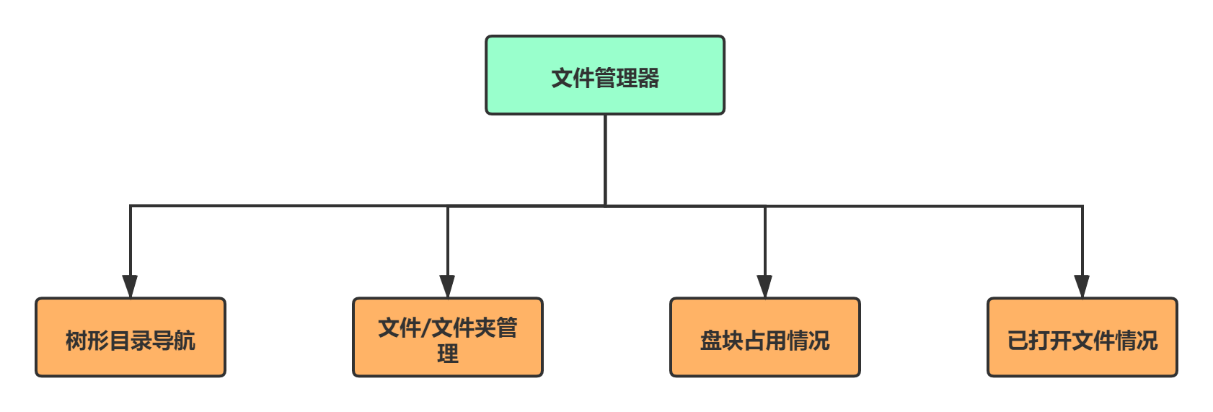
|  |  |
| --- | --- |
| 步骤1 | 随机生成想要生成的指令数量instructionNum，并依次对每条指令生成。 |
| 步骤2 | 当进行指令生成时，随机生成0-3的数字，代表指令的不同类别。 |
| 步骤3 | 依据指令的不同类别，对Instruction类进行实例化操作，并插入指令列表instructionArray中。 |
| 步骤4 | 最后实例化“end”指令，插入指令列表instructionArray中，以表示结束。 |

toString方法

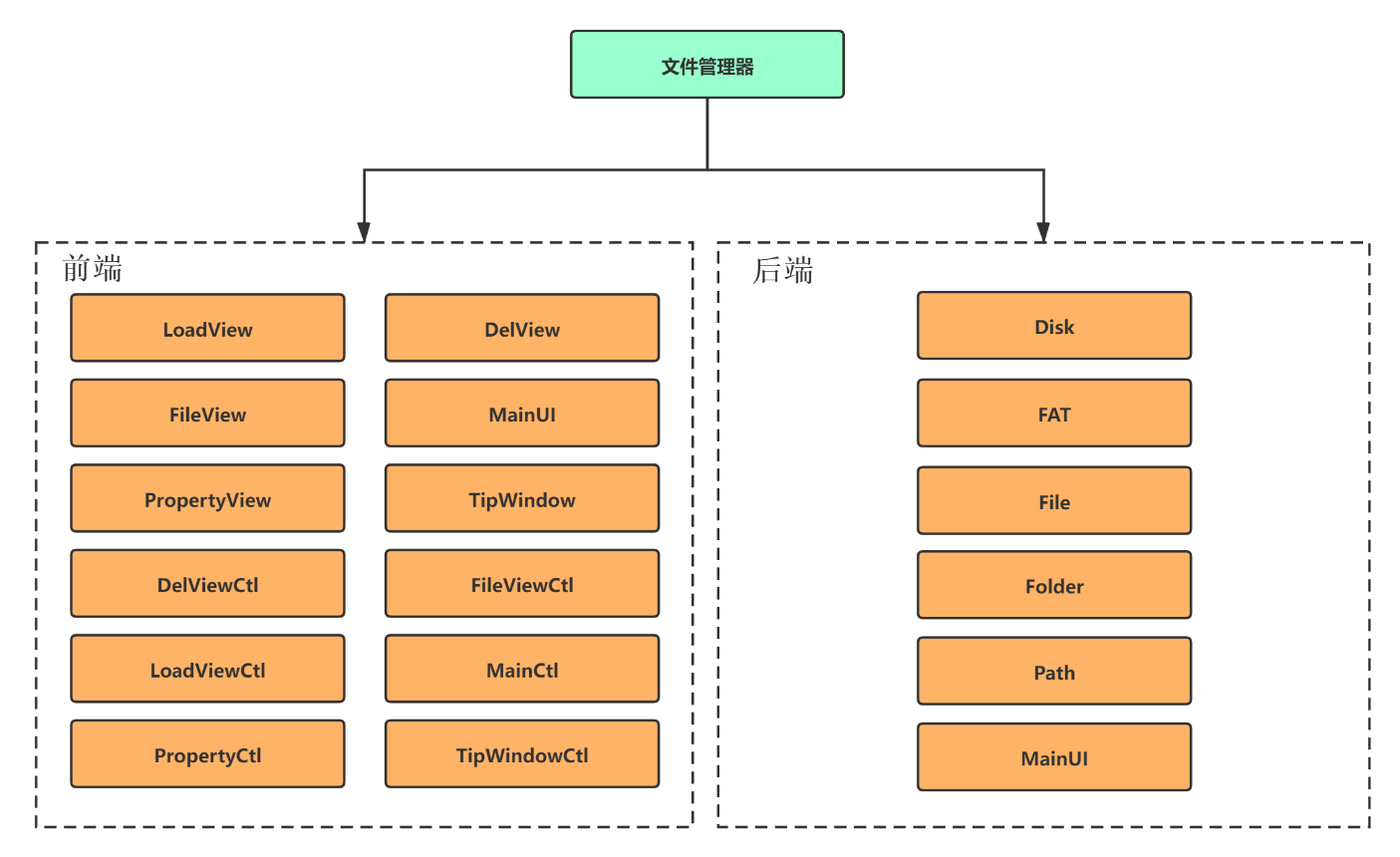
|  |  |
| --- | --- |
| 步骤1 | 新建String类型的字符串s。 |
| 步骤2 | 遍历指令列表instructionArray的每个元素，调用每个元素的toString方法，并依次加入到s中。 |
| 步骤3 | 返回s，作为转化结果。 |

3.3 文件管理器

3.3.1功能概述



3.3.2主要模块



3.3.3前端

（1）界面布局上，主要包括树形目录导航区域、文件/文件夹工作区、磁盘块占用情况、已打开文件列表展示。

（2）载入数据窗口程序LoadView，在主程序MainUI之前执行。主要作用是利用加载已设计的fxml文件进行窗口创建，供用户选择是否载入已有数据，并绑定文件管理器窗口控制器LoadViewCtl用以窗口创建和界面交互。

（3）文件管理器窗口主程序MainUI。主要作用是利用加载已设计的fxml文件进行窗口创建，并绑定文件管理器窗口控制器MainCtl用以窗口创建和界面交互，文件管理主要在该程序创建的窗口进行操作。

（4）在文件/文件夹工作区中，鼠标右击显示菜单，点击可以创建、删除、查看文件夹或文件，以及对相应的文件进行复制、粘贴或剪切。在打开文件时，利用加载已设计的fxml文件进行窗口创建，并绑定文件窗口控制器FileViewCtl用以窗口创建和界面交互；在查看文件或文件夹属性时，利用加载已设计的fxml文件进行窗口创建，并绑定属性窗口控制器PropertyCtl用以窗口创建和界面交互；在删除文件或文件夹时，利用加载已设计的fxml文件进行窗口创建，并绑定删除窗口控制器DelViewCtl用以窗口创建和界面交互。

（5）在磁盘使用情况展示区域中，使用饼状图和表格对磁盘块占用情况进行展示和使用表格对已打开文件信息进行展示。

（6）所有提示信息均通过使用TipWindow类进行窗口弹出。

3.3.4后端

（1）磁盘块类Disk。主要作用：对盘块编号、下一盘块的索引、盘块存储的类型、盘块存储的对象、盘块是否起始盘块等一系列属性进行封装；构建Disk的构造方法、清空盘块信息方法、重新分配盘块信息方法以及toString方法等。主要的属性和方法说明如下：

Disk类属性

|  |  |
| --- | --- |
| num | 编号 |
| index | 下一个盘块的编号 |
| type | 存储的类型 |
| object | 存储的对象 |

Disk类方法

|  |  |
| --- | --- |
| allocBlock | 重新分配盘块 |
| clearBlock | 清空该盘块 |
| toString | 返回该盘块对象的名称 |

（2）文件类File。主要作用：对文件名、类型、起始盘块号、读写标记、占用盘块数、内容、父文件夹、位置、大小、占用空间、创建时间、打开标志等一系列属性进行封装；构建File的构造方法以及toString方法等。主要的属性和方法说明如下：

File类属性

|  |  |
| --- | --- |
| num | 编号 |
| fileName | 文件名 |
| type | 文件类型 |
| diskNum | 起始盘块号 |
| flag | 读写标记 |
| length | 占用盘块数 |
| content | 内容 |
| parent | 父文件夹 |
| location | 位置 |
| size | 大小 |
| space | 占用空间 |
| creatTime | 创建时间 |
| isOpen | 打开标志 |

File类方法

|  |  |
| --- | --- |
| toString | 返回文件名 |

（3）文件夹类Folder。主要作用：对文件夹名、类型、起始盘块号、位置、大小、占用空间、创建时间、父文件夹、子文件夹、路径对象等一系列属性进行封装；构建Folder的构造方法以及toString方法等。主要的属性和方法说明如下：

Folder类属性

|  |  |
| --- | --- |
| folderName | 文件夹名 |
| type | 文件夹类型 |
| diskNum | 起始盘块号 |
| parent | 父文件夹 |
| location | 位置 |
| size | 大小 |
| space | 占用空间 |
| creatTime | 创建时间 |
| children | 子文件夹列表 |
| path | 路径对象 |
| catalogNum | 子目录项数 |

Folder类方法

|  |  |
| --- | --- |
| toString | 返回文件夹名 |

（4）路径类Path。主要作用：对路径名、父路径、子路径等一系列属性进行封装；构建Path的构造方法以及toString方法等。主要的属性和方法说明如下：

Path类属性

|  |  |
| --- | --- |
| pathName | 路径名 |
| children | 子路径列表 |
| parent | 父路径 |

Path类方法

|  |  |
| --- | --- |
| toString | 返回路径名 |

（5）文件分配表类FAT，主要作用：对盘块组、存储已打开文件的列表、根路径、存储所有路径的列表等一系列属性进行封装；构建FAT的构造方法、打开与关闭文件或文件夹方法、重新分配文件所需的盘块的方法等。主要的属性和方法说明如下：

FAT类属性

|  |  |
| --- | --- |
| disks | 磁盘块组 |
| c | 根文件夹 |
| rootPath | 根路径 |
| paths | 所有路径列表 |

FAT类方法

|  |  |
| --- | --- |
| FAT | 初始化fat |
| removeOpenedFile | 从已打开文件列表中移除文件 |
| addOpenedFile | 从已打开文件列表中添加文件 |
| closeAll | 关闭所有文件 |
| createFolder | 创建文件夹 |
| createFile | 创建文件 |
| searchEmptyDiskBlock | 寻找第一个空盘块 |
| freeBlocksCount | 计数空盘块 |
| reallocBlocks | 为文件重新分配盘块 |
| reallocFolderBlocks | 为文件夹重新分配盘块 |
| getFolders | 返回指定路径下所有文件夹 |
| getBlockList | 返回所有文件夹和文件的起始盘块 |
| getFolder | 返回指定路径指向的文件夹 |
| delete | 删除文件或文件夹 |
| getFolderSize | 求文件夹长度 |

FAT方法

|  |  |
| --- | --- |
| 步骤1 | 为文件分配表分配盘块 |
| 步骤2 | 为根目录分配盘块 |
| 步骤3 | 为可执行文件分配盘块 |
| 步骤4 | 初始化剩下的盘块 |
| 步骤5 | 初始化已打开文件表和路径表 |

addOpenedFile方法

|  |  |
| --- | --- |
| 步骤1 | 根据给出的盘块取得对应的文件 |
| 步骤2 | 将文件加入已打开文件表并设置文件打开标记为true |

removeOpenedFile方法

|  |  |
| --- | --- |
| 步骤1 | 根据给出的盘块取得对应的文件 |
| 步骤2 | 遍历已打开文件表取得要关闭的文件 |
| 步骤3 | 将文件移出已打开文件表并且设置文件打开标记为false |

closeAll方法

|  |  |
| --- | --- |
| 步骤1 | 遍历已打开文件表，将里面所有的文件移出并设置文件打开标记为false |

createFolder方法

|  |  |
| --- | --- |
| 步骤1 | 取得该路径下不重名的文件夹名称。 |
| 步骤2 | 判断当前路径是否为根路径，如果是，则限制只能创建8个目录项，若超出，则不能创建新的文件夹，否则可以创建。如果当前路径不是根路径，则创建文件夹后，为父文件夹添加目录项，然后为父文件夹重新分配盘块。 |
| 步骤3 | 寻找空的盘块，取得其索引号。 |
| 步骤4 | 创建文件夹对象，并为其分配盘块。 |
| 步骤5 | 创建路径对象，为父文件夹添加该文件夹为子目录。 |
| 步骤6 | 将该路径加入到路径表，为该文件夹设置新路径对象。 |
| 步骤7 | 遍历该文件夹的路径上的父目录，更改所有父目录文件夹的大小。 |

createFile方法

|  |  |
| --- | --- |
| 步骤1 | 取得该路径下不重名的文件名称。 |
| 步骤2 | 判断当前路径是否为根路径，如果是，则限制只能创建8个目录项，若超出，则不能创建新的文件，否则可以创建。如果当前路径不是根路径，则创建文件后，为父文件夹添加目录项，然后为父文件夹重新分配盘块。 |
| 步骤3 | 寻找空的盘块。 |
| 步骤4 | 创建文件对象，并为其分配盘块。 |
| 步骤5 | 如果该文件为复制的文件，则将其设置为原来的文件的属性。 |
| 步骤6 | 在复制的前提下，如果该文件为剪切文件，在粘贴文件的同时需要将原来的文件删除。 |
| 步骤7 | 遍历该文件夹的路径上的父目录，更改所有父目录文件夹的大小。 |

searchEmptyDiskBlock方法

|  |  |
| --- | --- |
| 步骤1 | 遍历所有盘块。 |
| 步骤2 | 返回第一个空闲盘块的盘块号。 |

freeBlocksCount方法

|  |  |
| --- | --- |
| 步骤1 | 遍历所有盘块。 |
| 步骤2 | 计数所有空闲盘块数目并返回。 |

reallocBlocks方法

|  |  |
| --- | --- |
| 步骤1 | 根据所给的盘块取得对应的文件和之前文件所需的盘块数。 |
| 步骤2 | 如果新的文件占用盘块数更少，则将后面多余的盘块清空。 |
| 步骤3 | 如果新的文件占用盘块数更多，则需往后寻找多的空盘块并占用。 |

reallocFolderBlocks方法

|  |  |
| --- | --- |
| 步骤1 | 根据所给的盘块取得对应的文件夹和之前文件夹所需的盘块数。 |
| 步骤2 | 如果新的文件夹占用盘块数更少，则将后面多余的盘块清空。 |
| 步骤3 | 如果新的文件夹占用盘块数更多，则需往后寻找多的空盘块并占用。 |

getFolders方法

|  |  |
| --- | --- |
| 步骤1 | 遍历所有盘块，建立文件夹列表。 |
| 步骤2 | 如果该盘块对应的对象为文件夹且路径与给出的路径相同，则将该文件夹加入到刚建立的文件夹列表中。 |
| 步骤3 | 遍历完成后，返回该文件夹列表。 |

getBlockList方法

|  |  |
| --- | --- |
| 步骤1 | 新建盘块列表，遍历所有盘块。 |
| 步骤2 | 如果该盘块不为空并且为起始盘块，则将其加入到新建盘块列表中。 |
| 步骤3 | 遍历完毕后，返回该盘块列表。 |

getFolder方法

|  |  |
| --- | --- |
| 步骤1 | 取得给出路径的父路径。 |
| 步骤2 | 遍历父路径上所有的文件夹。 |
| 步骤3 | 若为该文件则返回，否则返回null。 |

delete方法

|  |  |
| --- | --- |
| 步骤1 | 判断该文件或文件夹是否为空，不为空则不能删除；若判断为文件且已打开也不能删除。 |
| 步骤2 | 开始删除文件或文件夹，修改路径上文件夹的大小。 |
| 步骤3 | 释放盘块。 |

getFolderSize方法

|  |  |
| --- | --- |
| 步骤1 | 遍历该文件夹的孩子。 |
| 步骤2 | 如果孩子为文件对象，则将其大小进行计数。 |
| 步骤3 | 如果孩子为文件夹对象，则将其继续调用该函数进行文件夹大小的计数。 |
| 步骤4 | 遍历完成后，返回计数总长度。 |

（6）删除窗口类DelView，主要作用：弹出删除确认窗口，窗口中显示所要删除的文件或者文件夹的信息。

DelView类

|  |  |
| --- | --- |
| block | 删除的文件或文件夹的起始盘块。 |
| tipString | 文件或文件夹的信息。 |
| mainView | 文件管理系统主程序类。 |

DelView类方法

|  |  |
| --- | --- |
| DeView | 初始化删除信息窗口，将所删除的文件或文件夹的信息填入窗口内。 |
| start | 创建确认删除文件或文件夹窗口。 |

（7）载入数据窗口类LoadView，主要作用：弹出是否载入数据的确认窗口，窗口中显示将要删除的文件或者文件夹的信息。

LoadView类方法

|  |  |
| --- | --- |
| LoadView | 创建是否载入已有的数据的窗口 |

LoadView方法

|  |  |
| --- | --- |
| 步骤1 | 创建是否载入已有的数据的窗口。 |
| 步骤2 | 若确认载入原有的数据，则不做操作。 |
| 步骤3 | 若取消，则读取原有的文件，清空文件数据内容。 |
| 步骤4 | 清空当前文件分配表。 |

（8）文件打开窗口类FileView。主要作用：对文件、文件分配表、文件所属盘块、文件旧内容、文件新内容等一系列属性进行封装；提供一个静态哈希表用于储存已经打开文件以及其对应的Stage的信息；构建FileView的构造方法以及保存文件内容方法等。以下进行详细说明：

FileView类属性

|  |  |
| --- | --- |
| file | 文件名 |
| block | 文件初始盘块 |
| newContent | 文件新内容 |
| oldContent | 文件旧内容 |
| maps | 储存已打开文件与面板对应的哈希表 |

FileView类方法

|  |  |
| --- | --- |
| saveContent | 保存文件新内容 |

（9）查看文件属性窗口类PropertyView。主要作用：对文件分配表、文件所属盘块、旧命名、新命名等一系列属性进行封装；提供一个哈希表用于储存路径对象以及其对应的树节点的信息；构建PropertyView的构造方法以及重命名方法等。以下进行详细说明：

PropertyView类属性

|  |  |
| --- | --- |
| block | 窗口所对应的盘块 |

PropertyView类方法

|  |  |
| --- | --- |
| applyBtn.setOnAction | 应用新属性 |
| cancelBtn.setOnAction | 取消修改属性 |
| okBtn.setOnAction | 确认新属性 |
| reLoc | 更改名称 |

applyBtn.setOnAction方法

|  |  |
| --- | --- |
| 步骤1 | 检查新的名称，如果不合法或与当前路径中的名称重名，则应用失败。 |
| 步骤2 | 如果是文件夹，则重新定位，修改路径上的名称。 |
| 步骤3 | 修改文件或文件夹的名称并修改对应图标的名称。 |
| 步骤4 | 如果是文件，根据复选框重新设置文件的读写标志。 |

cancelBtn.setOnAction方法

|  |  |
| --- | --- |
| 步骤1 | 关闭该属性窗口。 |

okBtn.setOnAction方法

|  |  |
| --- | --- |
| 步骤1 | 检查新的名称，如果不合法或与当前路径中的名称重名，则应用失败。 |
| 步骤2 | 如果是文件夹，则重新更改路径上的名称。 |
| 步骤3 | 修改文件或文件夹的名称并修改对应图标的名称。 |
| 步骤4 | 如果是文件，根据复选框重新设置文件的读写标志。 |

reLoc方法

|  |  |
| --- | --- |
| 步骤1 | 如果文件夹没有孩子，则直接修改该路径名称即可。 |
| 步骤2 | 如果文件夹还有孩子，则遍历每一个孩子，再次使用reLoc方法更改每一个孩子的路径名称。 |

（10）提示信息窗口类TipWindow。主要作用：创建窗口进行信息提示。

（11）文件管理器主窗口类MainUI。主要作用：对属树形目录根节点、树形目录当前节点、文件分配表、复制标记、剪切标记、路径对象及其对应的的树节点的哈希表、存放生成窗口的动态数组等一系列属性进行封装；构建MainUI的构造方法、各种窗口的创建方法以及面板各组件的初始化方法等。主要的属性和方法说明如下：

MainUI类属性

|  |  |
| --- | --- |
| rootNode | 根节点 |
| recentNode | 当前节点 |
| fat | 文件分配表 |
| paths | 目前所有路径列表 |
| copyBlock | 已复制的文件对应的盘块 |
| copyFile | 已复制的文件 |
| blockList | 盘块列表 |
| recentPath | 当前路径名 |
| copyFlag | 复制标志 |
| moveFlag | 剪切标志 |
| clearFlag | 清空标志 |
| pathMap | 存储路径对象与其对应的树节点的哈希表 |
| fileAppAdditionStageList | 已打开窗口面板的列表 |

MainUI类方法

|  |  |
| --- | --- |
| MainUI | 初始化文件管理器主页面 |
| loadData | 载入数据 |
| menuInit | 初始化菜单 |
| tableInit | 初始化表格 |
| pieInit | 初始化饼状图 |
| initTreeView | 初始化树形结构 |
| initTreeNode | 初始化每一个树节点 |
| removeNode | 移除树节点 |
| TextFieldTreeCellImpl | 监听树形目录的点击事件 |
| addIcon | 显示当前路径文件或文件夹图标 |
| onOpen | 打开文件或文件夹 |
| propertyOpen | 打开属性窗口 |
| tipOpen | 打开信息提示窗口 |
| fileViewOpen | 打开文件窗口 |
| delViewOpen | 打开删除窗口 |
| saveData | 保存数据 |

MainUI方法

|  |  |
| --- | --- |
| 步骤1 | 加载选择是否载入已有的数据的窗口。 |
| 步骤2 | 加载文件管理主窗口。 |
| 步骤3 | 加载数据，使用ObjectOutputStream对象的序列化流，初始化文件分配表对象fat，并将所有文件设为关闭状态。 |
| 步骤4 | 遍历完成后，返回计数总长度。 |
| 步骤5 | 初始化菜单，绑定每个菜单项的点击事件。 |
| 步骤6 | 初始化树形目录导航栏。 |
| 步骤7 | 初始化表格数据显示。 |

loadData方法

|  |  |
| --- | --- |
| 步骤1 | 使用ObjectOutputStream对象的序列化流读取文件，获得文件分配表fat。 |

menuInit方法

|  |  |
| --- | --- |
| 步骤1 | 初始化菜单项，初始化菜单。 |
| 步骤2 | 绑定不同的点击事件显示不同的菜单。 |
| 步骤3 | 绑定不同菜单项的点击事件执行不同的方法。 |

tableInit方法

|  |  |
| --- | --- |
| 步骤1 | 将已打开文件表的文件数据绑定到TableView中。 |

pieInit方法

|  |  |
| --- | --- |
| 步骤1 | 获取空闲盘块的数目和已被占用的盘块的数目。 |
| 步骤2 | 求出占比并使用PieChart进行显示。 |

initTreeView方法

|  |  |
| --- | --- |
| 步骤1 | 从根路径出发，新建根节点并加入到树形目录中。 |
| 步骤2 | 为该树形目录增添点击导航事件，即点击目录项，工作区即可跳转到该对应的文件夹内容。 |
| 步骤3 | 遍历已有的路径表的每一个路径，如果该路径为根路径的子路径，则将其节点初始化。 |
| 步骤4 | 在工作区添加文件/文件夹图标。 |

initTreeNode方法

|  |  |
| --- | --- |
| 步骤1 | 将所给的节点放入该树形目录中。 |
| 步骤2 | 遍历该节点的孩子，再次使用initTreeNode方法将其初始化。 |

removeNode方法

|  |  |
| --- | --- |
| 步骤1 | 从树形目录中删除所给路径对应的树节点。 |
| 步骤2 | 从path映射树节点的哈希表中删除该路径。 |

TextFieldTreeCellImpl方法

|  |  |
| --- | --- |
| 步骤1 | 如果点击树形目录的结点，找到该树结点的路径。 |
| 步骤2 | 查找该路径的所有子路径的所有文件和文件夹的起始盘块。 |
| 步骤3 | 清空工作区，重新将显示该节点的所有孩子对应的文件或文件夹图标。 |
| 步骤4 | 修改当前路径名。 |

addIcon方法

|  |  |
| --- | --- |
| 步骤1 | 根据给出的盘块列表，显示不同的图标，若为文件则显示文件图标，若为文件夹则显示文件夹图标。 |
| 步骤2 | 图标命名，如果名称长度大于4，则显示时在第四个字节后加省略号代替。 |
| 步骤3 | 设置鼠标悬停图标会显示该文件或文件夹完整名称。 |
| 步骤4 | 设置图标获得鼠标焦点或失去鼠标焦点会显示不同的背景颜色。 |
| 步骤5 | 设置图标点击显示菜单栏事件。 |

onOpen方法

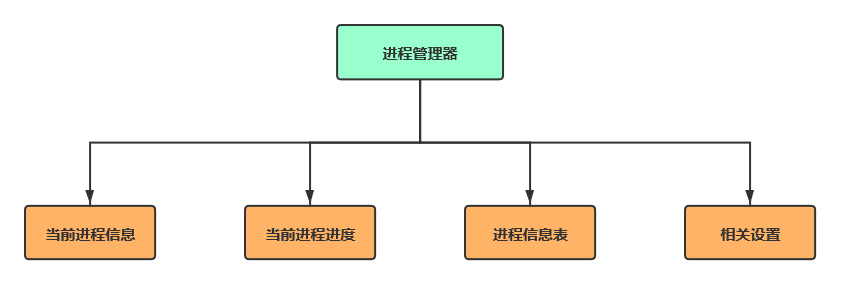
|  |  |
| --- | --- |
| 步骤1 | 如果打开的是文件，读取已打开文件表的长度，如果大于等于5，则打开失败。否则打开该文件的文件打开窗口。 |
| 步骤2 | 如果打开的是文件夹，则进入文件夹，更新工作区。 |

saveData方法

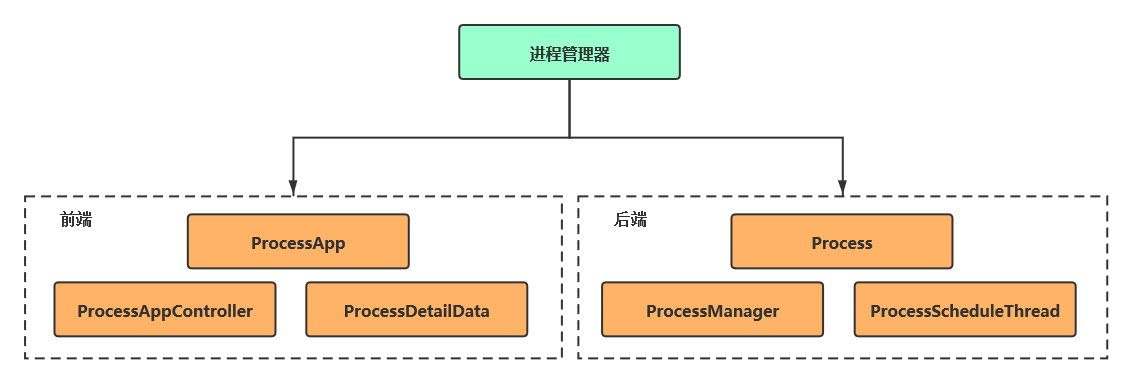
|  |  |
| --- | --- |
| 步骤1 | 使用ObjectOutputStream对象的序列化流，把文件分配表对象转成字节数据的输出到文件中保存。 |

3.4进程管理器

3.4.1功能概述



3.4.2主要模块



3.4.3前端

（1）界面布局上，主要包括所有进程表、当前进程对应文件内容表格、当前进程执行进度文本。同时提供了显示设置、标记设置、进程控制等按钮进行操控。显示设置细分为：当前所有、新建进程、就绪进程、阻塞进程、销毁进程、显示所有等选项；标记设置细分为： 新建进程、就绪进程、阻塞进程、销毁进程等选项；进程控制细分为：继续新建进程、暂停新建进程、倍速选择等选项。

（2）进程管理窗口主程序ProcessApp。主要作用是利用加载已设计的fxml文件进行窗口创建，并绑定进程管理窗口控制器ProcessAppController用以窗口创建和界面交互。

（3）进程表的信息数据类ProcessDetailData。由于JavaFx中的表格不能直接读取常规数据类型，因此需要进行类型替换，以方便表格更新。此类只是对进程类Process的数据类型进行替换，并无其他细节。

（4）进程管理窗口控制器ProcessAppController。主要作用是将界面中的按钮、表格等组件传送给UI线程，方便后续UI线程统一更新界面。

3.4.4后端

（1）进程类Process。主要作用：对进程的标识符、状态、占用PCB、占用内存、占用设备等一系列属性进行封装；构建进程的创建、撤销、阻塞、唤醒等一系列方法。以下进行详细说明：

Process类属性

|  |  |
| --- | --- |
| name | 进程标识符 |
| state | 进程状态 |
| counter | 程序计数器 |
| AX | 主要寄存器 |
| pcbID | 占用的进程控制块 |
| executableFile | 使用的可执行文件 |
| memory | 占用内存量 |
| memoryArea | 占有内存区域 |
| device | 占用设备 |
| deviceId | 占有设备的第几块 |
| deviceRemainTime | 使用设备剩余时间 |
| whichFile | 使用的可执行文件的编号 |

Process类方法

|  |  |
| --- | --- |
| create | 进程创建 |
| destroy | 进程撤销 |
| block | 进程阻塞 |
| awake | 进程唤醒 |
| cpu | 进程执行 |

create方法

|  |  |
| --- | --- |
| 步骤1 | 若当前不占用进程控制块，则调用Occupancy类的applyFreePcb方法进行申请，不成功则返回。 |
| 步骤2 | 若当前不占用内存块，则调用ccupancy类的applyMemory方法进行内存申请，不成功则返回。 |
| 步骤3 | 对进程状态state设为就绪，并移出ProcessManager类的新建队列creatingProcessList，就绪队列waitProcessList。 |

destroy方法

|  |  |
| --- | --- |
| 步骤1 | 更新当前进程状态为销毁。 |
| 步骤2 | 调用ccupancy类的retrieveMemory方法进行回收内存。调用ccupancy类的retrievePcb方法进行进程控制块的回收。 |
| 步骤3 | 移出ProcessManager类中的运行进程队列runProcessList。 |

block方法

|  |  |
| --- | --- |
| 步骤1 | 更新当前进程状态阻塞。 |
| 步骤2 | 移出ProcessManager类中的运行队列runProcessList，加入blockProcessList队列。 |

awake方法

|  |  |
| --- | --- |
| 步骤1 | 更新当前进程状态为就绪，更新设备为不占用。 |
| 步骤2 | 移出ProcessManager类中的阻塞队列blockProcessList，加入就绪队列waitProcessList。 |

cpu方法

|  |  |
| --- | --- |
| 步骤1 | 设置当前进程状态为运行，获取指令内容，程序计数器加1。 |
| 步骤2 | 分析指令内容，解析指令，并更新主要寄存器。若为申请设备指令，则调用OccupancyManager类的applyDevice方法进行申请设备操作。 |
| 步骤3 | 调用block方法进行进程阻塞，调用ProcessManager类的useDevice方法使用设备。 |

（2）进程管理器类ProcessManager。主要作用：封装可执行文件队列、所有进程队列、新建进程队列、运行进程队列、就绪进程队列、阻塞进程队列等属性；构建可执行文件的随机创建、新建进程队列的检测、阻塞进程队列的检测、使用设备等方法。以下进行详细说明：

ProcessManager类属性

|  |  |
| --- | --- |
| executableFileList | 可执行文件表 |
| allProcessList | 所有进程队列 |
| creatingProcessList | 新建进程队列 |
| runProcessList | 运行进程队列 |
| waitProcessList | 等待进程队列 |
| blockProcessList | 使用的可执行文件 |
| slice | 时间片长度 |
| speed | 倍率 |
| processNum | 已创建进程数 |

ProcessManager类方法

|  |  |
| --- | --- |
| createRandomExecuteFile | 可执行文件随机创建 |
| checkCreatingProcessList | 检测新建进程队列 |
| checkBlockProcessList | 检测阻塞进程队列 |
| useDevice | 使用设备 |

createRandomExecuteFile方法

|  |  |
| --- | --- |
| 步骤1 | 实例化ExecutableFile类，由其进行可执行文件的创建。 |
| 步骤2 | 将实例化的ExecutableFile，加入可执行文件队列。 |

checkCreatingProcessList方法

|  |  |
| --- | --- |
| 步骤1 | 遍历所有进程队列allProcessList |
| 步骤2 | 如果该进程为新建态，则对其进程调用create方法进行创建操作。 |

checkBlockProcessList方法

|  |  |
| --- | --- |
| 步骤1 | 遍历所有进程队列allProcessList |
| 步骤2 | 如果该进程为阻塞态：若因无设备为阻塞，则为其调用OccupancyManager类的applyDevice方法申请设备；若因为使用设备而阻塞，则为其调用useDevice方法使用设备。 |

useDevice方法

|  |  |
| --- | --- |
| 步骤1 | 如果该进程的设备剩余使用时间为0，则对其调用OccupancyManager类的retrieveDevice进行设备回收，并调用awake方法将进程唤醒。 |
| 步骤2 | 否则，该设备剩余使用时间减1。 |

（3）进程调度线程类ProcessScheduleThread。主要作用：对可执行文件、进程队列、内存、设备等属性进行封装。构建进程的创建、时间片轮转算法等一系列方法。以下进行详细说明：

ProcessScheduleThread类属性

|  |  |
| --- | --- |
| executableFileList | 可执行文件表 |
| allProcessList | 所有进程队列 |
| creatingProcessList | 新建进程队列 |
| runProcessList | 运行进程队列 |
| waitProcessList | 等待进程队列 |
| blockProcessList | 使用的可执行文件 |
| slice | 时间片长度 |
| speed | 倍率 |
| processNum | 已创建进程数 |
| sliceLength | 时间片长度 |
| allMemory | 用户区内存 |
| aDevice | 设备A |
| bDevice | 设备B |
| cDevice | 设备C |
| residueSlice | 剩余时间片 |
| controlButton | 进程控制按钮 |

ProcessScheduleThread类方法

|  |  |
| --- | --- |
| createProcess | 创建进程 |
| updateByRR | 以时间片轮转算法更新进程 |
| run | 主函数 |

createProcess方法

|  |  |
| --- | --- |
| 步骤1 | 判断新建进程是否超过3，如果超过3，就返回，避免新建进程过多。 |
| 步骤2 | 生成随机数，随机选择可执行文件。 |
| 步骤3 | 新建进程，加入新建进程队列creatingProcessList；加入所有进程队列allProcessList，以做好记录。 |
| 步骤4 | 将新建进程，调用create方法，进行新建操作。 |

updateByRR方法

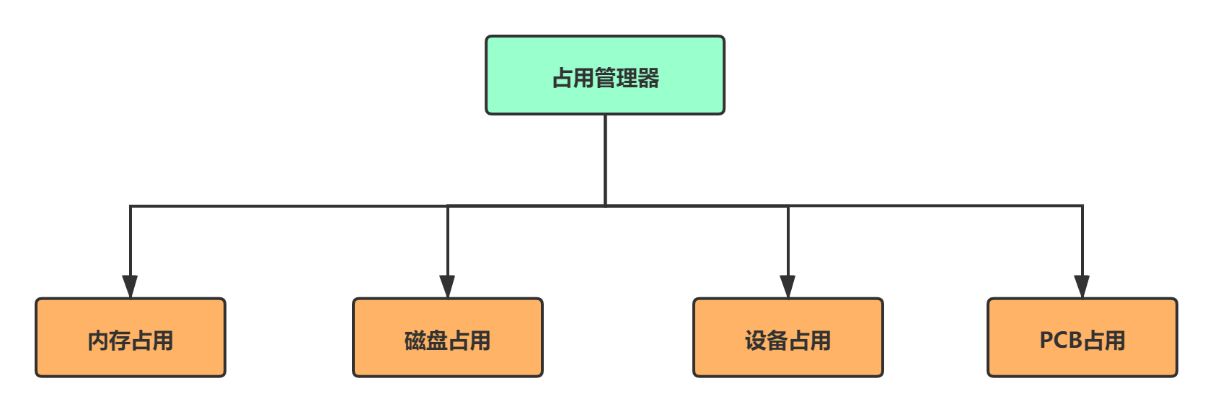
|  |  |
| --- | --- |
| 步骤1 | 调用createProcess方法，进行新建进程，避免cpu空闲。 |
| 步骤2 | 从就绪进程waitProcessList取出队首，移入运行进程队列runProcessList，得到当前进程nowProcess。 |
| 步骤3 | 对当前进程进行最多6次的cpu运行循环：若该进程指令以执行完，则进行销毁；若该进程执行完指令已转为阻塞状态，则退出循环。 |
| 步骤4 | 对当前进程移出运行进程队列，如果不为阻塞或销毁状态，则加入就绪进程队列。 |

run方法

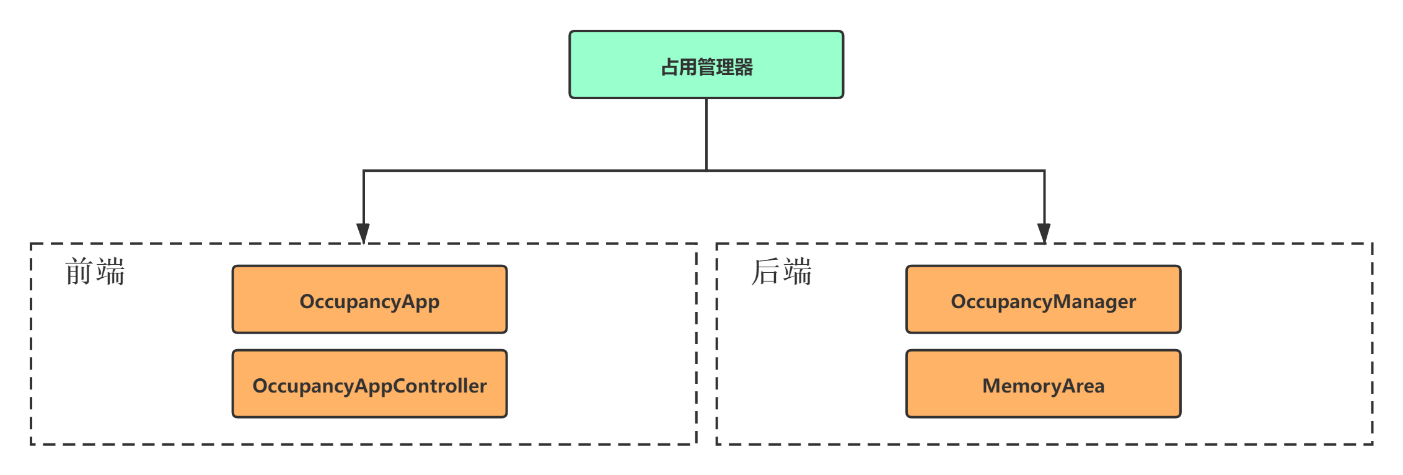
|  |  |
| --- | --- |
| 步骤1 | 新建一个定时器Timer，定时执行checkCreatingProcessList方法检测新建进程队列、checkBlockProcessList方法检测阻塞进程队列方法。 |
| 步骤2 | 调用updateByRR方法。 |

3.5占用管理器

3.5.1功能概述



3.5.2主要模块



3.5.3前端

（1）界面布局上，主要包括内存占用显示块、磁盘占用显示块、设备占用显示块、PCB占用显示块，以及相关得文字信息显示。其中，各部分占用主要由三部分组成：占用比例动画显示、占用比例文字显示、占用分别情况动画显示。

（2）占用管理窗口主程序OccupancyApp。主要作用是利用加载已设计的fxml文件进行窗口创建，并绑定占用管理窗口控制器OccupancyAppController用以窗口创建和界面交互。

（3）占用管理窗口控制器OccupancyAppController。主要作用是将界面中各组件传送给UI线程，方便后续UI线程统一更新界面。

3.5.4后端

（1）内存区类MemoryArea。主要作用：对内存区的起始、终止位置进行封装，无其他细节。

（2）占用管理器类OccupancyManager。主要作用：内存块、占用设备、pcb等一系列属性进行封装；构建初始化、申请空闲pcb、申请内存、回收内存、回收pcb、请求设备、回收设备等一系列方法。以下进行详细说明：

OccupancyManager类属性

|  |  |
| --- | --- |
| allMemory | 用户区内存条 |
| aDevice | A设备表 |
| bDevice | B设备表 |
| cDevice | C设备表 |
| freePcbList | 空闲的pcb队列 |

OccupancyManager类方法

|  |  |
| --- | --- |
| init | 初始化 |
| applyFreePcb | 申请空闲pcb |
| applyMemory | 申请内存 |
| applyDevice | 申请设备 |
| retrievePcb | 回收pcb |
| retrieveMemory | 回收内存 |
| retrieveDevice | 回收设备 |

init方法

|  |  |
| --- | --- |
| 步骤1 | 将allMemory数组所有元素初始化为0，标记为未占用。 |
| 步骤2 | 将aDevice数组、bDevice数组、cDevice数组所有元素初始化为0，标记为未占用。 |
| 步骤3 | 将freePcbList添加0到9共10个元素。 |

applyFreePcb方法

|  |  |
| --- | --- |
| 步骤1 | 若空闲pcb队列freePcbList不为空，且有元素，则取出首部元素并返回。 |
| 步骤2 | 否则返回-1，表示没有空闲pcb。 |

applyMemory方法

|  |  |
| --- | --- |
| 步骤1 | 根据需要取得的内存区容量num，遍历allMemory数组。 |
| 步骤2 | 若找到第一个具有num容量的内存区，则返回对应的memoryArea实例。 |
| 步骤3 | 否则返回null，表示没有num容量的内存区。 |

applyDevice方法

|  |  |
| --- | --- |
| 步骤1 | 根据需要取得的设备类别device，访问对应的设备数组。 |
| 步骤2 | 遍历设备数组，找到第一个0元素，返回对应索引。 |
| 步骤3 | 若找不到，则返回-1，表示没有num容量的内存区。 |

retrievePcb方法

|  |  |
| --- | --- |
| 步骤 | 根据返回的的num，将其添加进空闲pcb队列freePcbList中 |

retrieveMemory方法

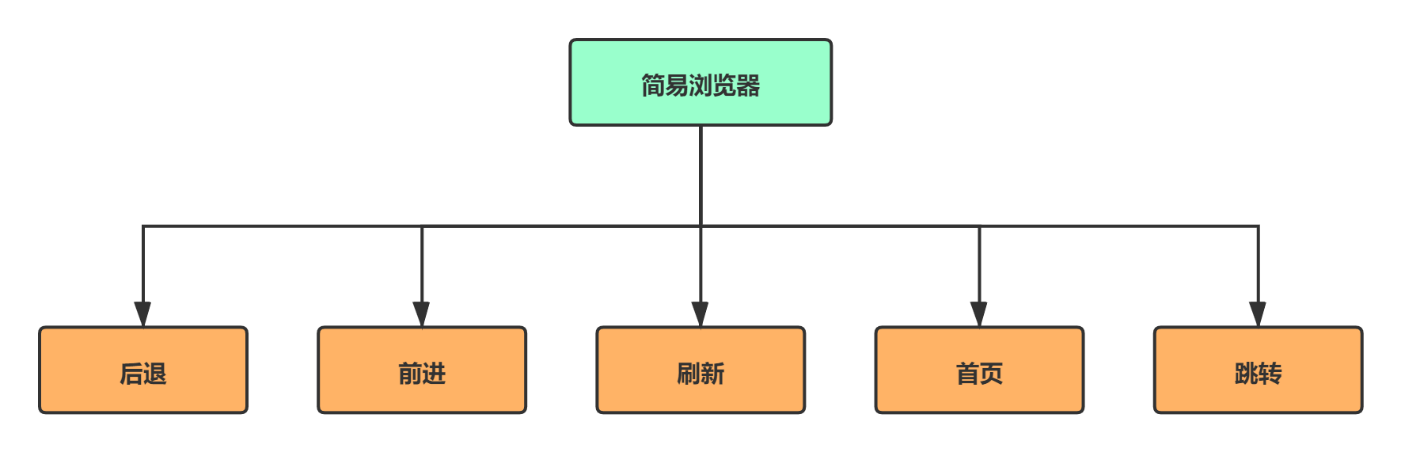
|  |  |
| --- | --- |
| 步骤1 | 根据传入的内存区memoryArea，判断是否为空，若为空，则直接返回不处理。 |
| 步骤2 | 得到内存区memoryArea的首尾索引，从而得知占用的范围。 |
| 步骤3 | 遍历内存块allMemory，对该占用的范围进行重置为0，表示未被占用。 |

retrieveDevice方法

|  |  |
| --- | --- |
| 步骤 | 根据传入的设备类别device和设备号diveId，对相应的设备数组和对应设备重置为0，表示未被占用。 |

3.6简易浏览器

3.6.1功能概述



3.6.2 实现说明

该功能基于javafx中的webview组件以及对应的API进行开发，开发思路较为简单。

（1）后退：判断当前是否为第一页，若不是，则指向上一页。

（2）前进：判断当前页是否为最后一页，若不是，则指向下一页。

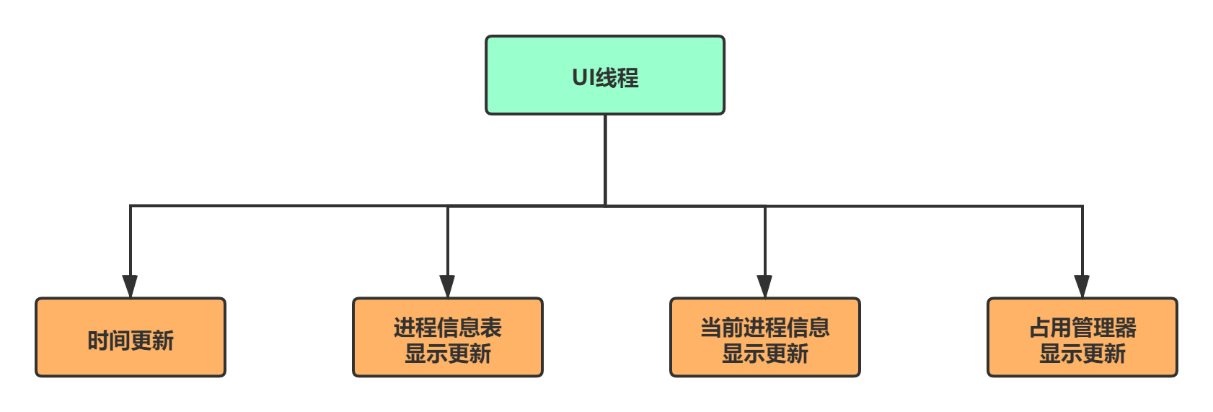
（3）刷新：重新加载当前引擎。

（4）首页：跳转至默认网址。

（5）跳转：加载输入网址框的文本。

3.7界面实时更新线程

3.7.1功能概述



3.7.2 实现说明

由于该部分大多是对各窗口的相关组件进行更新，大多属于前端范畴，又因为涉及的组件过于繁杂，因此只说明大概思路。

（1）时间更新

首先，新建Data对象，以得到最新的时间信息；其次，对时间组件的文本信息进行更新。

（2）进程信息表更新

第一，根据进程信息表的显示选项，得到想要显示对应的进程队列；第二，将进程队列中的信息转化为javafx中的Table组件能够读取的信息；第三，将进程信息表中的内容设置为转化后的信息。

（3）当前进程信息更新

第一，获取当前运行进程；第二，将当前运行进程的指令信息转化为javafx中的Table组件能够读取的信息；第三，对相关组件的信息进行更新。

（4）占用管理器更新

①文本显示更新：得到内存、磁盘、设备、pcb的占用比例，然后将比例信息更新即可。

②用管理器第一个矩形框更新（比例显示）：得到内存、磁盘、设备、pcb的占用比例，然后更新矩形框中的填充比例即可。

③用管理器第二个矩形框更新（分布显示）：

内存区：遍历整个内存区数组，如果该位置被占用，则涂色。

磁盘区：遍历整个磁盘区数组，如果该位置被占用，则涂色。

设备区：分别设定好每个设备对应的颜色。遍历所有设备区数组，如果该位置被占用，则涂色。

Pcb区：遍历遍历所有pcb，如果该位置被占用，则进行涂色。需要注意的是，占用pcb的颜色，与占用进程所对应的进程状态有关。

（5）表格的高亮实现

主要是通过设定setRowFactory函数实现。预先在css文件中设定好高亮的颜色，之后筛选指定的行作为高亮区即可。

4调试分析

4.1调试过程中遇到的问题是如何解决的以及对设计与实现的讨论与分析

在调试中遇到问题，首先会判断是自己模块的设计不当还是其他组员模块的设计不当。若是自己模块的设计不当，则通过后台查看具体出错位置，在逻辑设计、算法设计等方面进行分析，通过网上资料查找、自行修改来解决解决。若是其他组员模块的设计不当，则通过组员间的交流与修改来解决问题。

4.2.算法的时间复杂性和改进思想

时间复杂性分析：本程序中，在进程调度、内存分配与回收等方面，基本都是线性查找，时间复杂度都是。其余的模块，都是调用上述的功能模块，具体时间复杂度需根据具体情况具体分析。

改进思想：设置一些标志位，合理地利用已知信息来避免无效查找。

4.3. 设计过程的经验和体会

4.3.1 xx

本次实验，成员间沟通流畅，协作顺利。我负责的主要工作是进程管理器、桌面设计、窗口设计、界面美化、简易浏览器的开发并进行相关部分的报告攥写。同时，本次实验题目也是非常有趣的，通过将学到的知识以编程的方式进行实现，我对课本的知识也具有了更深刻的理解。无论从编程层面还是操作系统知识层面，都非常有收获。由于需要设计较好的界面效果，我根据自己想法浏览了大量关于JavaFx界面设计的技术博客，在设计过程中，还需要考虑窗口自适应等问题。在本次实验中，不仅让我对操作系统有了进一步的认识，同时也在一定程度上提高了我在Java界面开发方面的能力。

4.3.2 xx

在本次实验中，我负责的是系统文件表、文件管理器以及报告攥写。经过一个学期的对操作系统的学习后，将理论应用于实践，模拟写出一个操作系统。在这过程中，为了实现预想的功能，我不断学习新的知识，在对操作系统不断熟悉的同时提高了编程能力。团队协作是最重要的，一个人难以完成这样一个这样大的任务，在实验过程中，相互协助，优劣互补，最终较为完美的完成这一次实验。

4.3.3 xx

本次实验中，我们团队分工明确，我负责的是占用管理器以及报告攥写。基于一个学期的操作系统的学习，结合面向对象编程的思想，再搭配上新掌握的JavaFx界面开发技术，我们实现了这次操作系统的模拟。学习了新知识后，要进行实操才能够真正的掌握。更重要的是，让我清楚了团队合作不是说单纯分配任务就可完成的事，要经过不断地沟通交流。总的来说，这一次团队任务让我受益很多，让我的各方面能力都更进了一步。

4.4实现过程中遇到的主要问题及解决方法

4.4.1进程队列问题与解决方案

问题：一开始进程队列等相关信息都使用ArrayList进行存储。当多个线程对同一进程队列进行操作时，经常出现异常现象。后通过网上查询，才发现ArrayList是线程不安全的。

解决方案：第一种方案，就是对ArrayList进行加锁，但是代码量大，改起来费时费力，弃用；第二种方案，不使用ArrayList进行信息存储，而使用线程安全的Vector，代码改起来也简单，因此，最后使用第二种方案，问题解决。

4.4.2界面更新问题与解决方案

问题：在非Fx线程要执行Fx线程相关的任务时，会导致界面停滞。

解决方案：使用platform.runlater（）可以防止阻塞Fx线程。

5用户使用说明

5.1简易浏览器

简易浏览器，主要作用是浏览网页。

（1）其包含后退、前进、首页、跳转等功能，在地址栏输入地址，点击跳转，即可到达指定网页。

（2）若需要正常运行，建议在jdk1.8.0\_301环境上运行。

5.2系统文件表

系统文件表，其主要作用是显示系统可执行文件和对应的可执行文件内容。

左侧表格是可执行文件表，右侧表格是可执行文件的内容。点击左侧表格任意一个文件，相应的指令内容则会在右侧表格显示。

5.3文件管理器

文件管理器，其主要作用是对文件进行操作，以及显示磁盘使用情况。

（1）当启动该软件时，会询问是否读取历史数据，若是，则进行读取；若否，则清空之前的数据。

（2）第一个模块是文件管理页面。在该页面，通过鼠标及键盘操作，可进行对文件的新建、读写、移动、复制、删除、属性查看等操作，可进行对文件夹的新建、删除、移动、复制、属性查看等操作。

（3）第二个模块是磁盘的使用情况页面。在该页面，显示了磁盘的占用情况：①占用分布图；②占用盘块、索引、类型、内容等信息；③当前打开的文件对应文件名、打开方式、起始盘块、文件长度、文件路径等信息。

5.4进程管理器

进程管理器，主要作用是可视化进程的运行情况。

（1）单独显示当前运行进程的编号、执行指令、数据寄存器的值、剩余时间片。[默认时间片为6]。

（2）显示当前进程的执行进度，高亮当前执行指令。

（3）显示进程详表，具体包括：进程编号、进程状态、执行文件、设备使用情况、进程控制块、当前执行结果、进程完成进度。

（4）进程详表设置：包括显示设置、标记设置和进程控制。

①显示设置：用于选择想要显示的进程，包括当前进程、新建进程、就绪进程、阻塞进程、销毁进程、显示所有。

②标记设置：用于选择想要高亮的进程，包括新建进程、就绪进程、运行进程、阻塞进程、销毁进程。

③进程控制：用于控制继续新建进程或者暂停新建进程，以及控制进程运行的倍速（1倍速，2倍速，4倍速，8倍速）。

5.5占用管理器

占用管理器，主要作用是可视化用户区内存、磁盘、设备、PCB的占用情况。

（1）用户区内存占用可视化：显示内存占用比、内存占用分布情况。

（2）磁盘占用可视化：显示磁盘占用比、磁盘占用分布情况。

（3）设备占用可视化：显示设备占用比，设备占用分布情况。

（4）PCB占用可视化：显示PCB占用比，PCB占用分布情况，以及占用PCB的各进程状态（新建、就绪、运行、阻塞）。

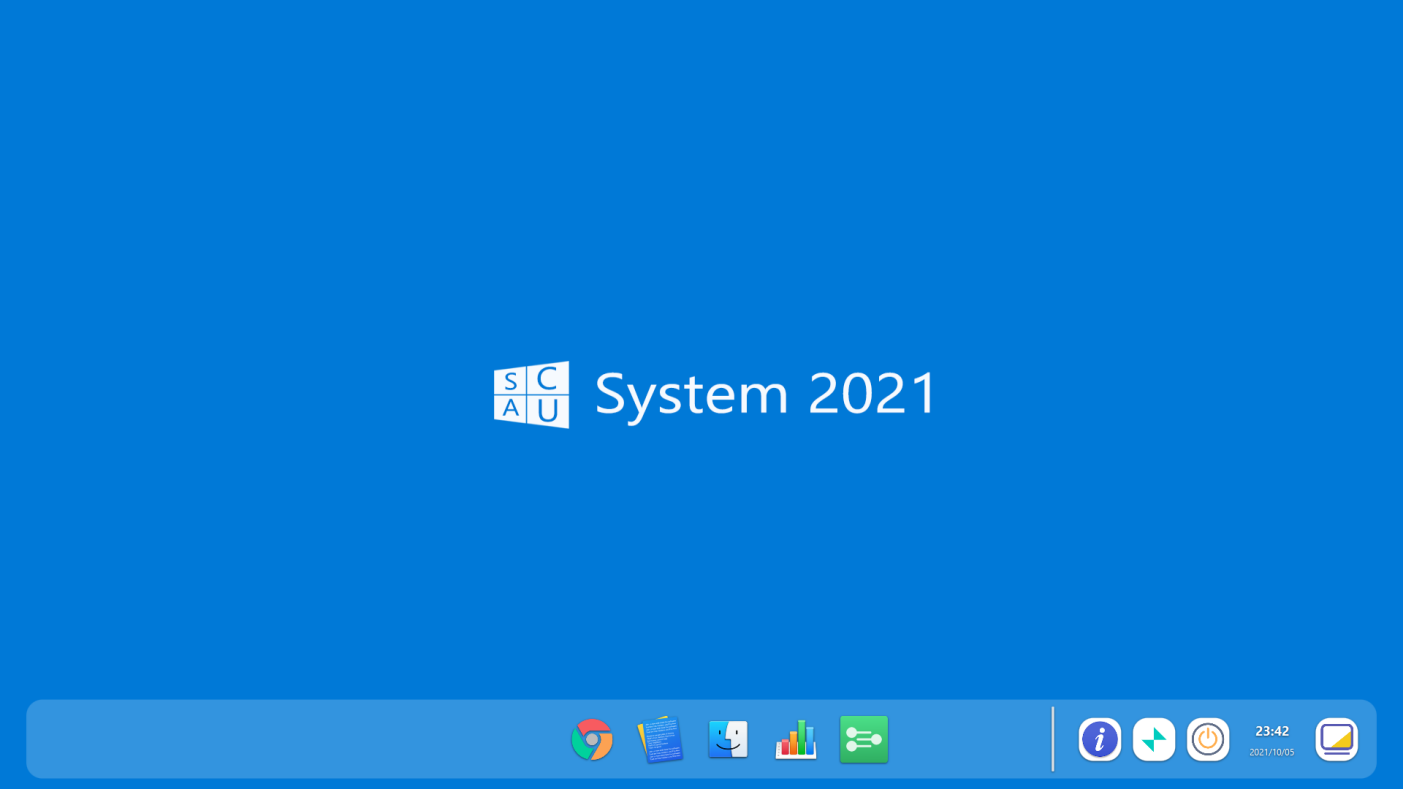
5.6帮助

帮助，主要作用是对整个系统的操作进行说明。

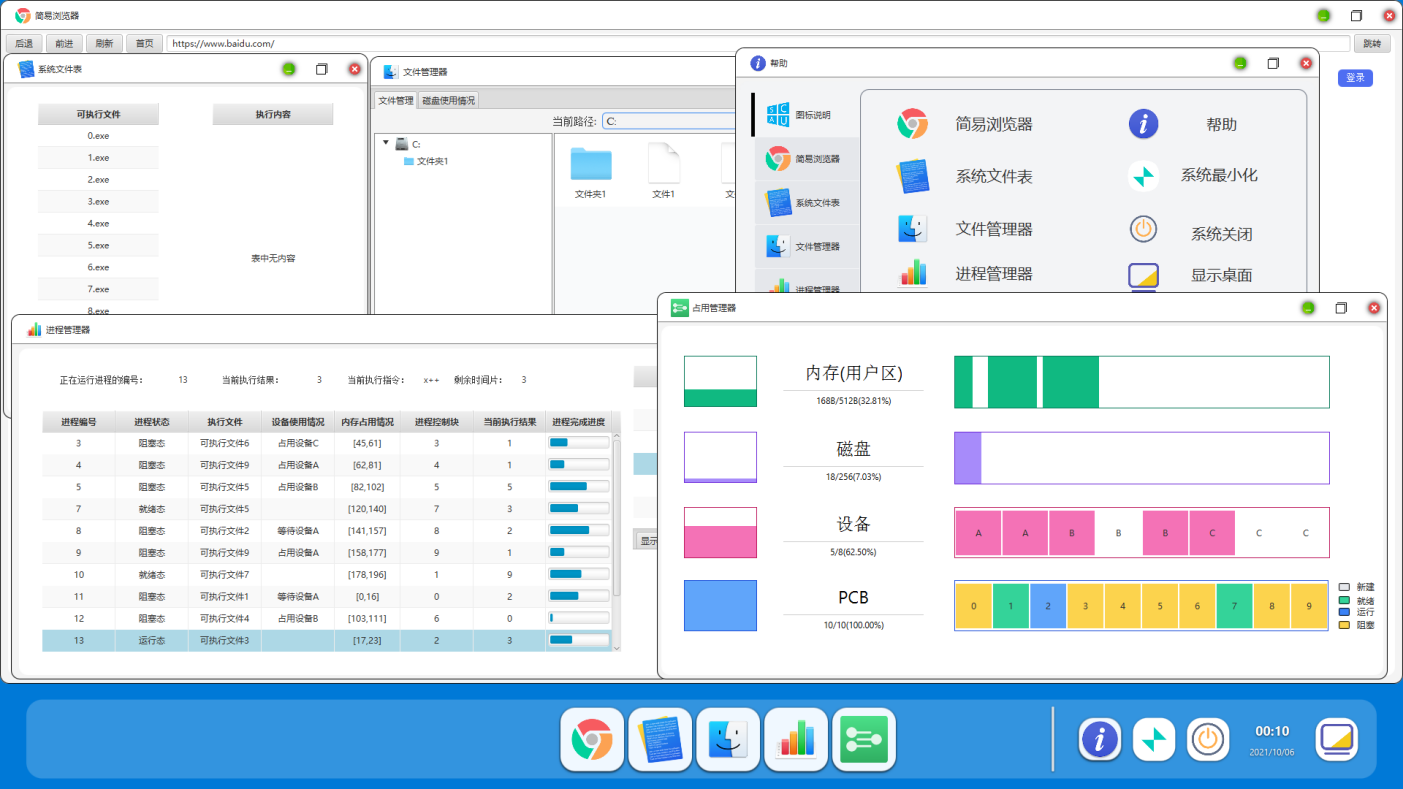
6测试与运行结果

6.1系统桌面

（1）程序启动初始界面

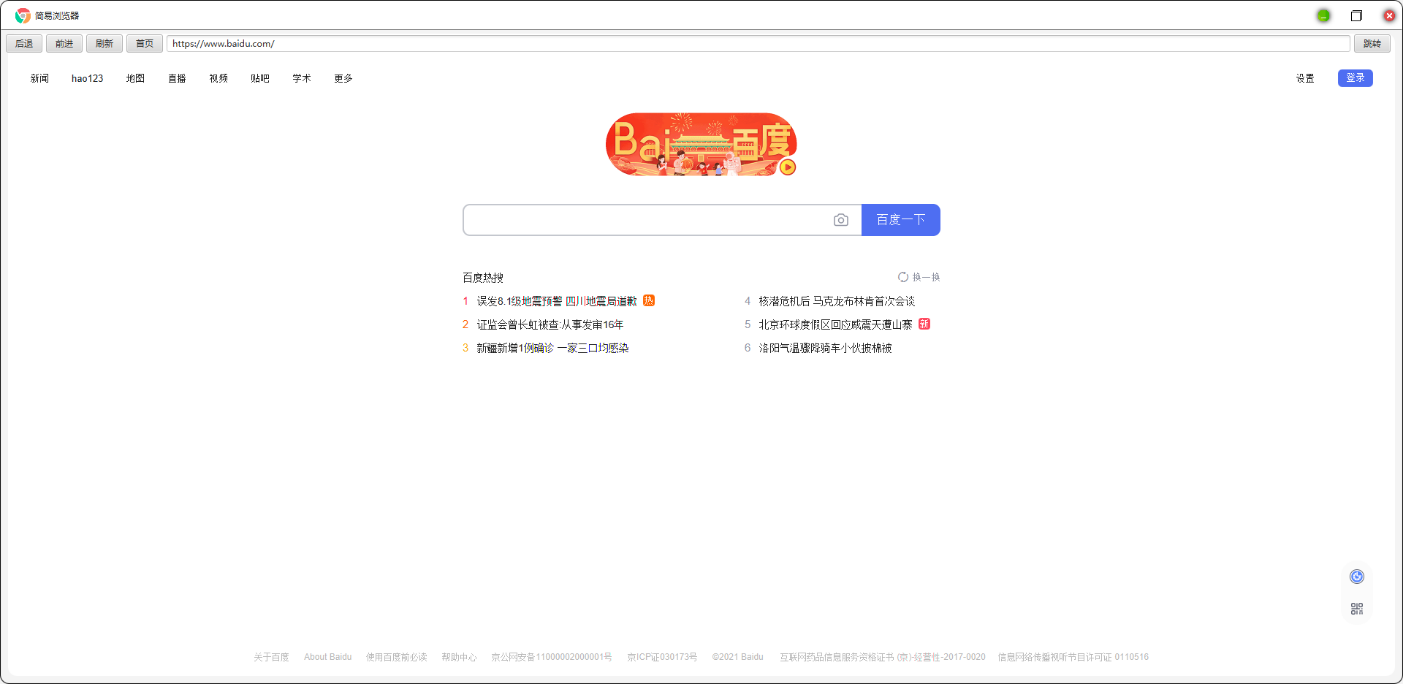


（2）所有软件打开后的界面效果



6.1简易浏览器

简易浏览器运行时的网页效果。



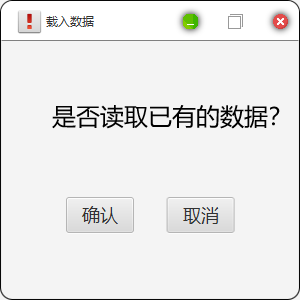
6.2系统文件表

左侧显示可执行文件列表，右侧显示对应的指令内容。

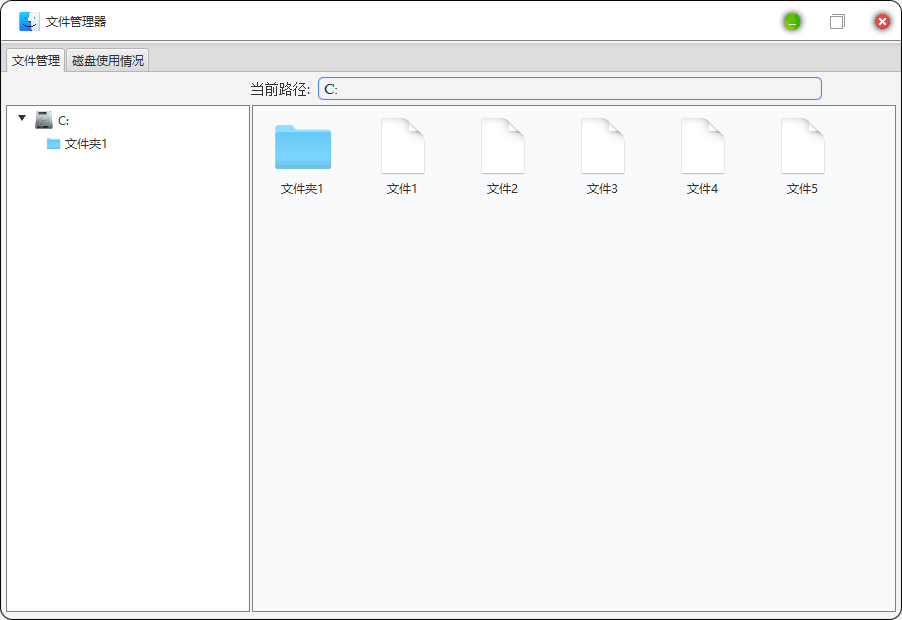


6.3文件管理器

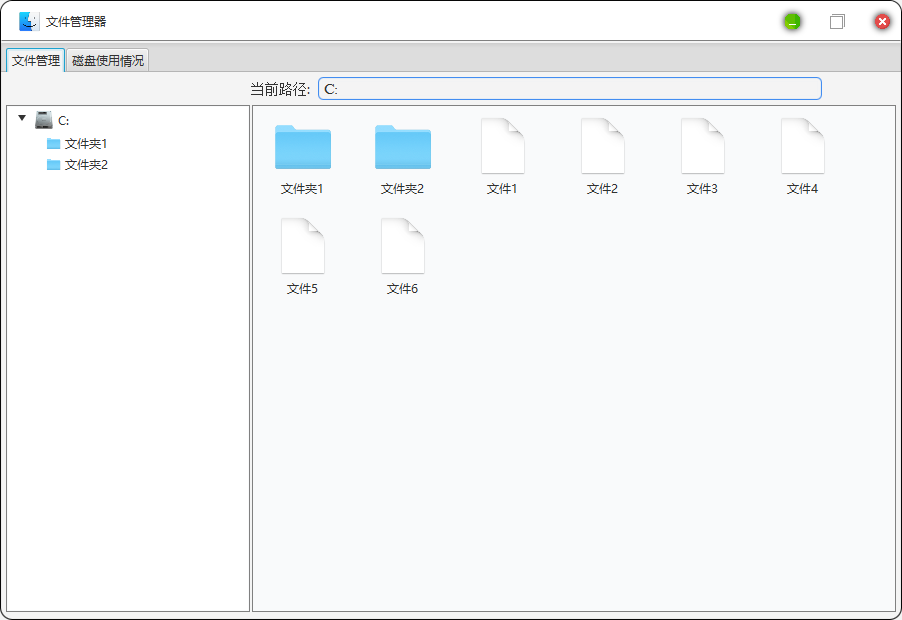
（1）首次打开软件时，弹出是否加载数据。



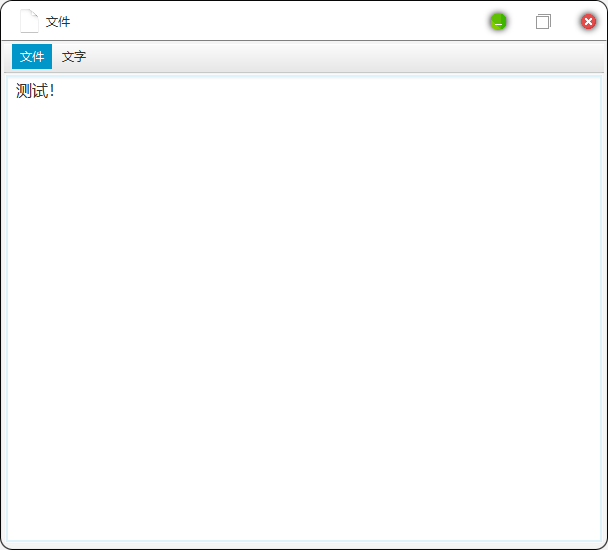
（2）加载历史数据后，显示如下。



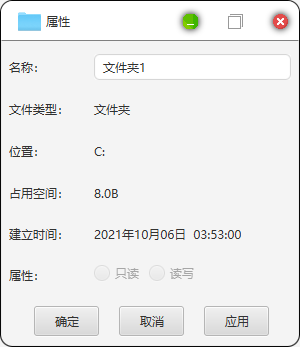
（3）新建文件夹和文件，如图所示，多出了一个文件夹和一个文件。



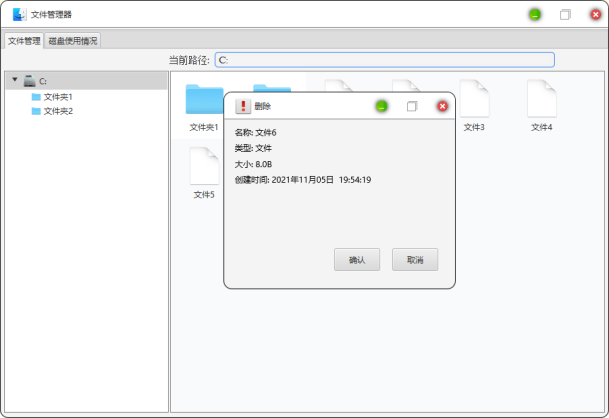
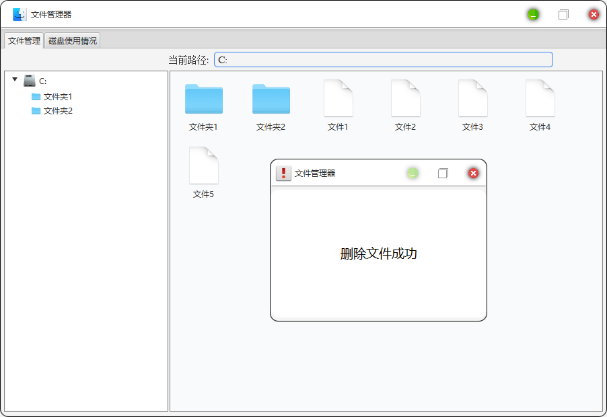
（4）对文件进行读写，显示如下。



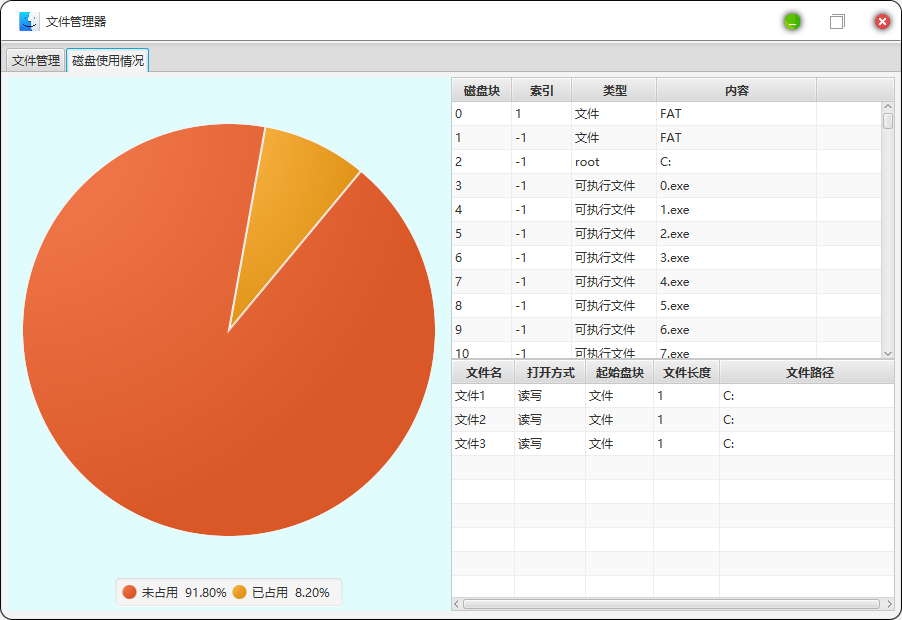
（5）查看文件夹属性和文件属性。

（6）删除文件提示以及删除后效果。

（7）磁盘占用情况显示。



6.4进程管理器

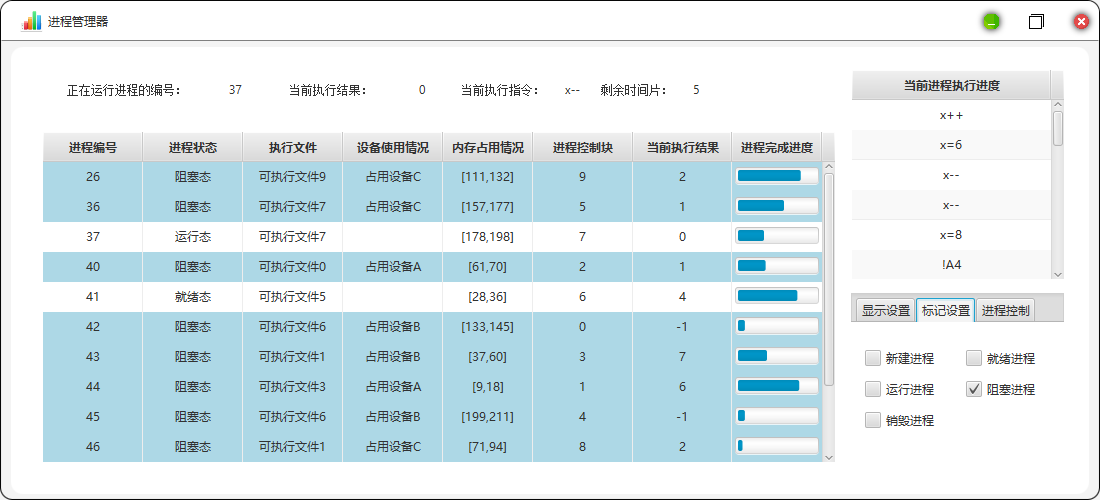
（1）软件整个界面效果，包括正在运行进程的信息、进程信息表以及相关设置面板。



（2）显示设置的操作演示。以下以选中显示设置的新建进程为例。（注意：完整设置为：显示设置——新建进程、标记设置——运行进程、进程控制——继续新建进程）



（3）标记设置的操作演示。以下以选中标记设置的阻塞进程为例。（注意：完整设置为：显示设置——当前进程、标记设置——阻塞进程、进程控制——继续新建进程）

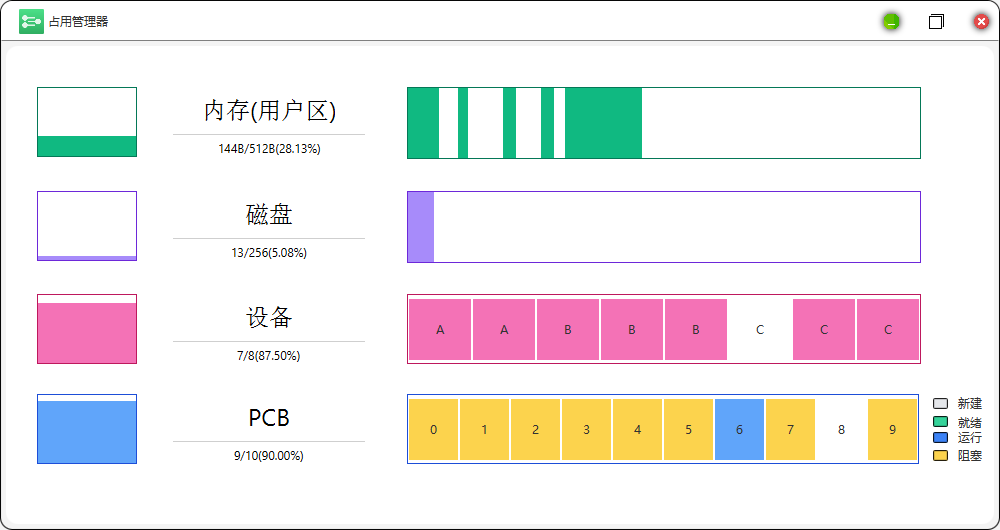


（4）进程控制的操作显示。以下以暂停新建进程为例，一段时间后当前进程表无内容。（注意：完整设置为：显示设置——当前进程、标记设置——运行进程、进程控制——暂停新建进程）



6.5占用管理器

内存、磁盘、设备、PCB的占用比例和占用情况如下。



6.6帮助

帮助的各个界面显示如下。

