实验二（操作系统大作业）

- 选题：《自制OS实验》

个人信息

- 姓名：苏俊杰
- 学号：2022211607
- 班级：2022211807

摘要

- “开发一个操作系统”，听起来是一个不简单的事情，事实上也不简单，可也并没有我们想的那么难。笔者本事只是有点兴趣，想要尝试一下，意外地发现其实起步并没有那么困难，只是需要一点时间去熟悉OS。不知不觉中，小小的尝试也产出了成果，而且仍有继续成长的趋势，说白了，容易上瘾。
- 本文将介绍笔者开发的一个微型系统，名为 `MindSync OS`，意为善解人意的系统，中文名还没想好。为什么要起这个名字？这是笔者起初的遐想，希望能开发一个便于用户管理文件和阅读的OS，现在看来确实是止于遐想了哈哈，毕竟当时还没着手写代码，只是跟着教程，听大牛们说在开发OS前一定要对OS有一些理想，或者说叫系统哲学？于是就起了这个名字。还像具体了解这个名字的含义的话，可以到 `MindSync OS`项目仓库的 `README.md`查看。
- `MindSync OS`虽然只是一个非常小的微型系统，但已经具有了比较完整的功能体系，能够覆盖操作系统的五个基本问题：处理机管理、存储器管理、设备管理、文件管理、OS与用户接口。虽然每个部分的实现都不是最优解，不如说优化比较差，但至少在功能上解决了问题，是可用的。而且系统大小不过几十k，还能够覆盖到这五个问题，笔者认为这就足够了。
- `MindSync OS`的整个所有项目文件已经上传到Github上，如果希望测试这个系统，请点击：MindSync OS。

第一章：系统概述

- `MindSync OS`（下称“本系统”）是一款微型的操作系统，具有绝大部分现代操作系统所需的基本功能，也支持许多让人兴奋的功能，比如支持多任务，具有图形化界面等等。本章将会简要介绍本系统的基本信息，列举本系统的功能，并说明本系统如何进行使用和测试。

1.1 系统信息概述

- 本系统在开发过程中使用的是Windows 11系统，并通过qemu将本系统作为虚拟机启动进行测试。此处引用主要参考文献的原话：
  - 本文的一切说明也将面向以IBM PC/AT兼容机（也就是所谓的Windows个人电脑）为对象进行说明。至于其他机型，比如Mac（苹果），虽然也参考了其中某些部分，但基本上无法开发出在这些机型上运行的操作系统，这一点还请见谅。严格地说，不是所有能称为AT兼容机的机型都可以开发我们这个操作系统，我们对机器的配置要求是CPU高于386（因为我们要开发32位操作系统）。换句话说，只要是能运行Windows 95以上操作系统的机器就没有问题，况且现在市面上（包括二手市场）恐怕都很难找到Windows 95以下的机器了，所以我们现在用的机型一般都没问题。
- 简单地说，就是本系统的在编写的时候面向的是特定的指令集，而且使用的编译软件也依赖特定的操作系统，而且Makefile也是依赖于命令行工具所支持的指令的，具体的说明请见1.3节。

1.2 系统功能概述

- 本节将简单列举系统具有的功能。

1. 处理机管理：
   1. 系统支持多任务并发运行，设置任务优先级，高优先级优先运行，同级任务按时间片轮转法调度。
   2. 系统支持绝大多数任务调度原语，能够将任务在创建、就绪、运行、阻塞、挂起、终止几个状态合理切换。
   3. 通过FIFO实现消息队列对任务进行控制，配合开关中断，支持任务的互斥与同步。
2. 存储器管理：
   1. 系统支持动态内存分配与释放，使用首次适应算法，且对内核与用户区域进行了设计与划分。
   2. 系统控制台支持使用 `mem`指令查看内存使用情况。
3. 设备管理：

    1. 系统支持键盘与鼠标控制，编写中断处理程序处理用户输入。
    2. 能够使用键盘输入文字，使用 `Tab`切换窗口，使用鼠标移动窗口。
4. 文件管理：
    1. 系统控制台支持使用 `dir`指令查看磁盘目录，使用 `type`指令查看文件内容。
    2. 支持检索并运行存储在系统硬盘的用户程序，系统内有一个示例程序 `hello.hrb`，可以输入 `hello`或 `hello.hrb`运行。
5. OS与用户接口：
    1. 系统具有可视界面，以窗口形式显示应用程序画面，支持多图层的叠加处理。
    2. 用户可以使用鼠标与系统交互。
    3. 用户可以使用键盘输入指令，并在屏幕现实的控制台查看回显。
    4. 用户可以使用系统提供的API进行编程，实现用户程序在控制台输出文字。

1.3 系统使用说明

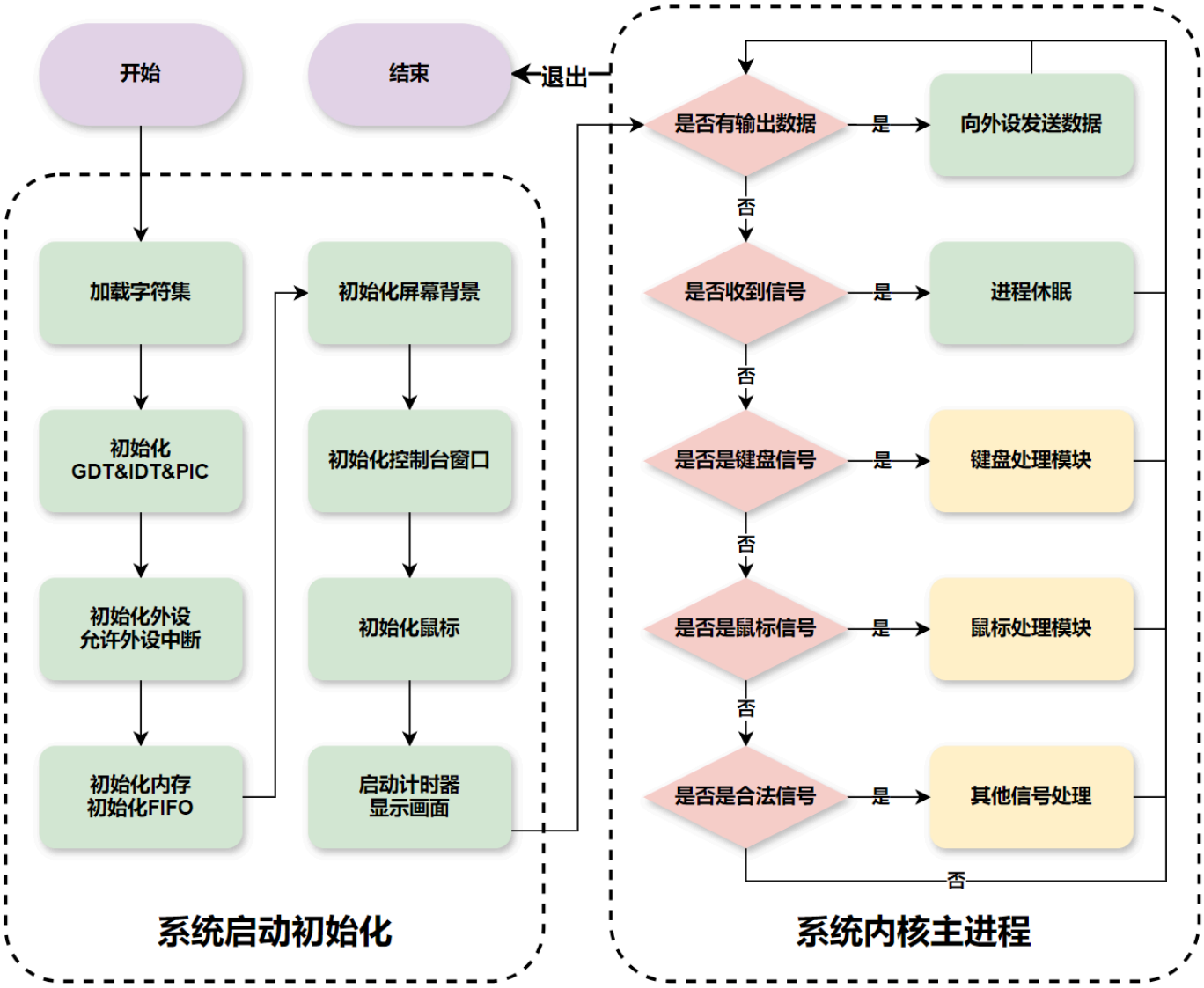
1. 本系统的在开发的时候面向的是特定的指令集（x86），所有使用汇编语言实现的部分都是按照x86的规格进行编写，因此需要在x86架构的机器上使用nasm等工具编译成2进制才可以运行。
2. 本系统使用的编译软件也依赖特定的操作系统，仓库中所有的依赖软件都是.exe的二进制文件，即Windows系统的可执行文件，所以想要直接使用只能在Windows上运行。
3. Makefile使用了Bash shell指令，因此理论上只能使用git bash运行，其他命令行无法保证能够正常运行。事实上，笔者自己就尝试在Windows命令行（cmd）和Powershell使用对应的指令编写Makefile，但出现了各种bug，只有git bash得以正常运行，因此，强烈建议直接使用git bash运行笔者写好的Makefile，如果非要自己改用其他命令行，请自行研究如何运行。

- 综上所述，如果想要开箱即用，就要使用符合此处说明的环境进行测试：
    - 请在裸机使用软盘启动，或在 `Windows`系统使用 `qemu`运行此项目，暂不支持其他启动方式。
    - 理论上，映像文件可以直接在裸机运行，但仍建议在 `qemu`进行运行。裸机运行出现任何问题，后果自负。
    - 请在拉取仓库后，打开命令行，进入 `src`目录，输入 `make run`即可运行操作系统。
    - 请使用 `git bash`命令行工具，否则可能会出现错误。
    - 如果修改代码后 `make run`出现错误，可以尝试 `make clean`清除中间文件后再次 `make run`。

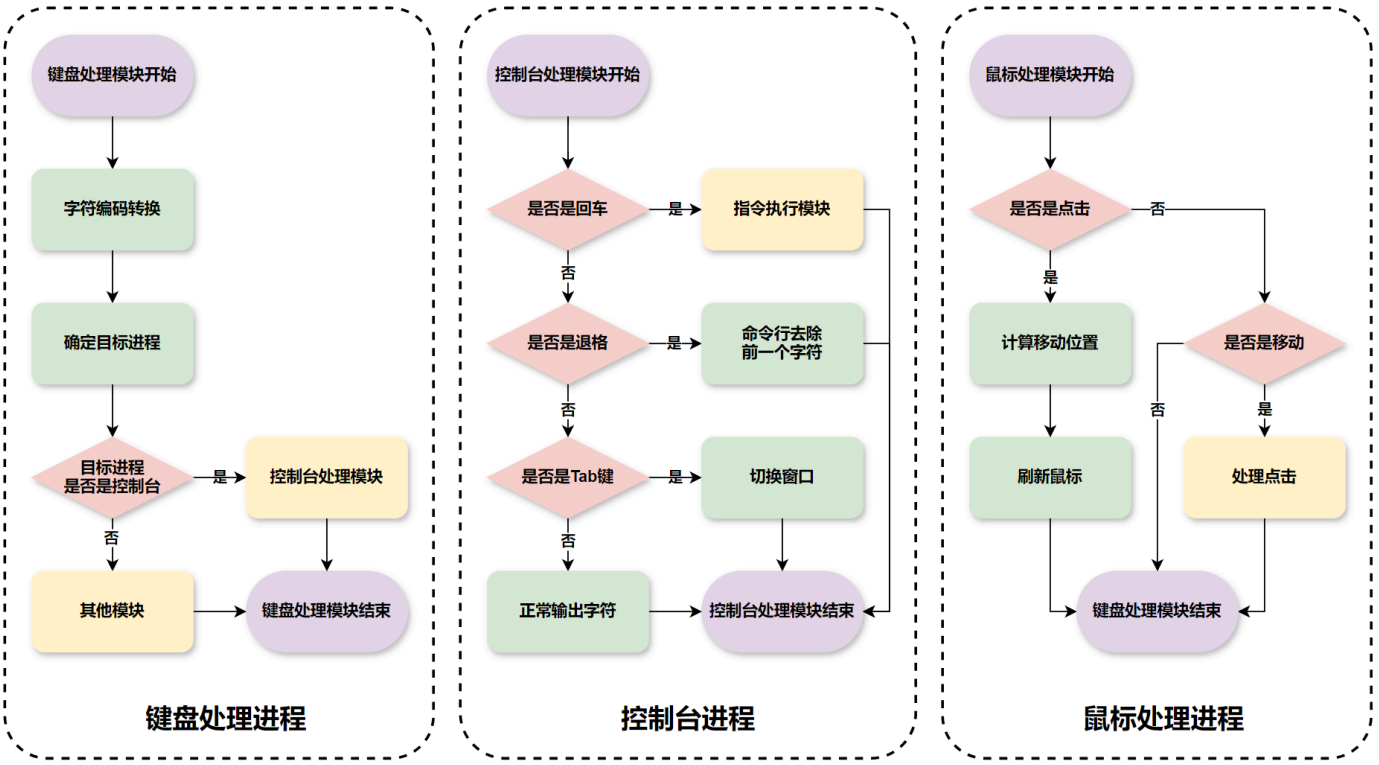
第二章：系统设计

2.1 系统流程图

2.1.1 系统主进程



2.1.2 其他模块



2.2 系统功能模块

2.3 系统内存分布图

第三章：系统代码简析

3.1 处理机管理

1. 本系统实现了基本的任务调度，并实现了任务调度原语操作，包括：
    - `task_alloc()`：实现任务创建
    - `task_add()`：将任务加入就绪队列
    - `task_switch()`：将运行的任务终止，激活队列中的下一个任务
    - `task_sleep()`：将任务挂起
    - `task_run()`：激活挂起任务
    - `task_remove()`：任务释放

```
struct TASK *task_alloc(void);                  // 任务创建
void task_add(struct TASK *task);               // 任务就绪
void task_switch(void);                         // 时间片完&任务调度
void task_sleep(struct TASK *task);             // 任务挂起
```

```
void task_run(struct TASK *task, int level, int priority); // 任务激活
void task_remove(struct TASK *task);                       // 任务释放
```

- 其中，任务阻塞将通过其他中断程序实现，此处不一一列举。
- 通过这些原语，可以实现任务在创建、就绪、执行、阻塞、挂起、阻塞、挂起之间灵活变换。

2. 本系统使用多级队列调度算法，实现了多任务并发。

3.2 存储器管理

3.3 设备管理

3.4 文件管理

3.5 OS与用户接口

第四章：系统效果展示

第五章：总结

- 通过本次实验，笔者收获颇丰。首先，着手开发让我对操作系统的底层实现有了更深的理解；其次，实践过程中，也加强了笔者的汇编与C语言开发能力，为以后进行如操作系统等底层开发积累了许多有价值的经验；最后，本次实验让笔者体会到了开发OS的乐趣，为笔者未来进路的选择产生一定影响，当然目前来看是好的影响。笔者也希望以后的学弟学妹们都能尝试一下这个实验，很有价值（不如改成必选吧，哈哈！）。
- 本次开发主要参考的是川合秀实写的《30天自制操作系统》，对笔者的开发起了非常重要的引导作用。起初只是跟着敲代码，但这本书实在是太面向新手了，仿佛把读者都当成了小学生，所以不自觉地会在代码上做一些改进呢。但是改进的后果就是，后边作者自己又改进了，如果与自己的思路不太相符，后边的文章就没法直接应用在自己改动后的代码上了，这真是太让人遗憾了！不过后来逐渐习惯了川合的套路，也就逐渐能够自己对代码进行一些小的优化了，虽然函数定义大概率是改不了。还有一件事，川合的函数与变量命名习惯实在是与笔者八字不合，可是为了能

够在自己代码出现bug时快速使用他的代码替换进行测试debug，只好屈从了他的风格，或许这就是我和底层大佬的差距所在？哈哈！可惜的是，这个学期过的很快，在快到学期末时，笔者已经意识到自己不可能按书的要求完成完整的30个day的开发，于是便在当时进度的基础上自己进行了一些封装与开发，让系统从一个用于展示的demo变为一个更像真的OS的程序（这是川合的说法，实际上他开发的Haribote在完成到最后30day之前，都只是一个长得像OS的程序），体现出系统的一些特性。毕竟时间有限，还要把这篇报告写出来呢，后续的开发等暑假再说吧~。

- 在文末，笔者要特别感谢**王申老师**对笔者的支持。王老师是笔者大二春季学期操作系统课程的主讲老师，让笔者对操作系统的理论基础有了较好的理解。起初笔者只是带着试一试的心情，向老师了解了一下关于开发OS的问题，结果收到了许多有用的资料，这对我来说是很有价值的帮助（《30天自制操作系统》也是老师推荐的）。之后的两天草草地看了一下资料，发现好像也就是这一回事吧，应该问题不大吧？于是就走上了MindSync的开发之路，直到现在。真的非常感谢！

参考文献

[1] （日）川合秀实著；周自恒，李黎明，曾祥江，张文旭译. 30天自制操作系统. 北京：人民邮电出版社, 2012.08. [2] （日）凑川爱著；苗琳娟译. 跟Wakaba酱一起学 兼容Windosw和Mac Git使用. 合肥：中国科学技术大学出版社, 2020. [3] 于渊编著. 自己动手写操作系统. 北京：电子工业出版社, 2023.04. [4] 闪客著. Linux源码趣读. 北京：电子工业出版社, 2023.09.