# NIS1336 大作业设计文档

2023.7.10

成员：夏季 周祎晨 闫家硕

## 一、小组完成情况及分工

| 组员 | 分工 |
| --- | --- |
| 周祎晨<br>521021911080 | 实现命令行的非循环版本和 Run 循环版本<br>实现基于 Qt 的 UI 界面<br>实现 UI 界面的多线程以及命令行的互斥访问 |
| 闫家硕<br>518021910532 | 实现任务管理，设计和实现任务的添加、删除和显示等功能<br>实现任务排序和筛选，按各种条件筛选任务。<br>实现命令行的多线程 |
| 夏季<br>521051910099 | 实现云端同步<br>实现用户管理，增删查改及用户权限设置 |

## 二、命令行版本说明

### 1.模块与类的设计：

#### 1.1 任务管理模块

任务管理模块负责处理任务的添加、删除、显示、保存、加载和提醒功能，包含 Task 结构体和 TaskManager 类。通过将 Task 结构体定义在类外，其他部分也可以使用 Task 来表示任务，可以提高代码的可重用性，同时更方便扩展任务的功能和属性，使得代码更具有适应变化的能力。

**1.1.1 Task 结构体：** Task 结构体用于表示任务的属性。它包含以下成员变量：taskId（任务的唯一标识符，用于区分不同的任务）、task_name、start_time、priority、category、attention_time、printedReminder（标识是否已经打印过提醒）。

**1.1.2 TaskManager 类：** TaskManager 类是任务管理模块的核心类，封装了任务管理的各种操作。以下是该类的主要函数和功能：

**addTask：**用于添加新任务。使用条件判断和错误处理机制，检查用户提供的开始时间、提醒时间和任务分类是否符合要求。通过将开始时间和提醒时间解析为 std::tm 结构体，使用时间函数和比较操作符来验证开始时间是否早于当前时间，以及提醒时间是否晚于开始时间。使用 std::vector 存储任务列表，并使用 std::unordered_set 存储任务名称+开始时间的组合，以检查任务的唯一性。

**deleteTask：**用于删除指定 ID 的任务。它使用了迭代器进行遍历和条件判断，利用迭代器的“erase”方法删除任务，并通过调用 updateTaskIds 更新任务的 ID，确保任务 ID 的连续性和准确性，同时更新任务标识符。

**showTask：**用于显示任务。通过使用条件分支结构，根据传入的参数筛选任务，并将符合条件的任务存储在 filteredTasks 向量中。通过使用字符串的查找方法和比较器函数，将符合日期、月份和分类要求的任务添加到筛选结果中。最后，使用 std::sort 对筛选结果进行按开始时间排序。

**checkAttentionTime：**用于检查任务的提醒时间并进行提醒。使用 std::chrono 库获取当前系统时间，并将任务的开始时间和注意时间转换为时间点，以便进行时间比较和计算时间差。使用迭代器遍历任务列表，通过循环逐个处理每个任务的提醒逻辑。通过多个条件判断

语句来判断任务的状态和提醒条件。根据当前时间和任务的开始时间、提醒时间，确定是否需要打印提醒信息。使用 exitflag 和 pthread_exit 控制多线程的退出操作，保证程序的并发执行和退出控制。

**readFromFile 和 write2File：**使用文件流 (std::fstream 和 std::ofstream) 实现文件的读取和写入操作。通过指定文件路径和打开模式，读取和写入任务信息。通过循环和索引对任务进行遍历和访问。

### 1.1.3 其他辅助函数：

**isStartTimeUnique：**遍历任务列表，对比任务的开始时间，检查开始时间是否唯一。

**updateTaskIds：**用于更新任务的 ID。该方法在删除任务后被调用，重新分配任务的 ID，保持 ID 的连续性。

**setDefaultAttentionTime：**设置默认的提醒时间。如果任务未提供提醒时间，将提醒时间设置为开始时间的前十分钟。

**updateAttentionTime：**更新提醒时间。比较任务的提醒时间与当前时间，如果提醒时间早于当前时间，则将提醒时间设为当前时间，从而立即提醒。

**parseTime：**使用 std::stringstream 类将时间字符串解析为 tm 结构体，以便进行时间的比较和处理。

## 1.2 用户管理模块

用户信息存储：创建 USER 类，存放用户名、密码哈希、是否管理员字段，使用 map 存储用户信息，键值对为<UID,USER>。创建 uid.txt 文件，存储所有用户的信息。

用户信息读取：userfile_read 用于从 txt 文件中读取用户的信息，并存储到 mao 容器中

用户信息写入：userfile_update 用于向 txt 文件中写入所有的信息。读取和写入都是一次性操作，将所有用户信息进行写入和读取。

用户注册：regist 函数，用于增加用户，函数会判断是否用户已经存在，若不存在，则创建新用户，分配新的 uid（设计为当前最大键值的后一位），写入到 txt 中。

用户登录：user_check 函数，用于检查用户名和密码是否正确，其中密码经过哈希后再与存储的密码哈希进行比较

用户删除：删除用户，首先确认权限，仅管理员具有删除权限，且管理员无法删除其他管理员，不能删除自己

用户密码修改：change_passwd 函数，首先确认权限，仅管理员或自己可以修改密码，且非 root 管理员不可修改其他管理员用户的密码，root 拥有最大权限

用户切换：su_user 用于用户切换，更新当前 UID，同时更新当前应该打开和读取的 task 文件

权限修改：admin_change 用于修改管理员权限，可以赋予也可以剥夺，仅 root 用户有此权限

## 1.3 云端同步模块

使用设备：云服务器（腾讯云），IP 地址：1.15.238.222

设计背景：由于对于信息的存储要求较为简单，使用纯文本格式文档存储即可，因此在数据传输上不需要采用过于复杂的方式，我们选择了 scp 文件传输的形式讲所有保存用户信息的文件和保存用户任务文件（*.txt）上传至服务器上。

主体函数设计：

connect2server()：连接至服务器，初始化进程并返回一个 ssh 进程类

scp_upload()：利用 scp 上传文件

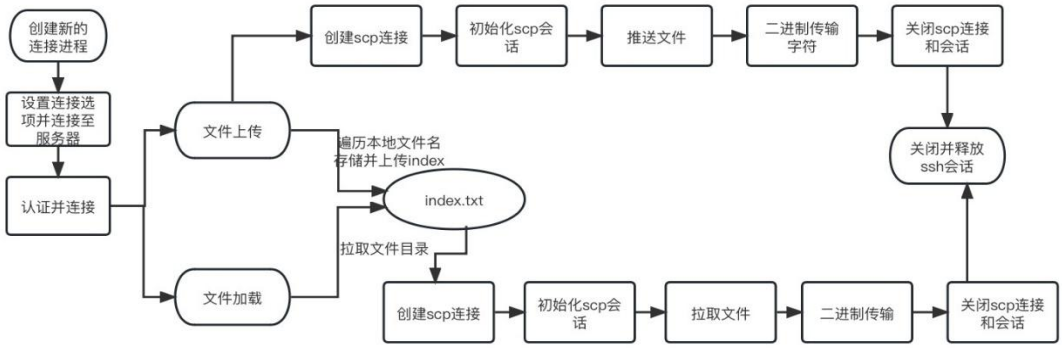
scp_download()：利用 scp 从远程拉取文件

free_ssh_session()：关闭和释放进程

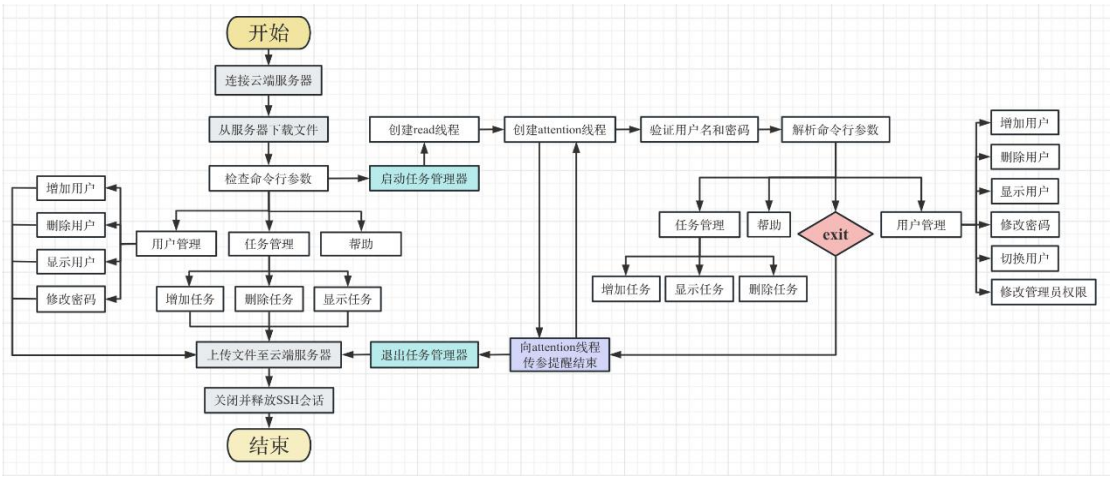
其他函数：

file_bytes()：读取文件字节数和内容，传递给函数参数(指针传递)

load_index()：从远程拉取 index.txt 文件，读取远程所有文件的名称，以便后续逐一拉取该模块的运行流程图：




## 2.流程图




## 3.关键技术问题说明

### 3.1 任务管理模块

**3.1.1 时间计算和时间格式转换：**涉及到时间计算的时候，将时间解析为 std::tm 结构体，使用 std::mktime 函数将其转换为 std::time_t 类型的时间戳。使用时间戳进行时间计算和比较提供了一种简洁、统一和方便的方式来处理时间相关的操作。它具有精确性、直观性和跨平台兼容性等优势，使得时间的处理更加高效和可靠。

**3.1.2 多线程实现任务管理和提醒：**通过多线程，可以同时执行读取用户输入和检查任务提醒的功能，提高了程序的响应性和效率。用户可以在任务提醒的同时继续输入指令，不会阻塞程序的运行，并且可以实时地提醒用户关注即将到来或已经过期的任务。同时，通过使用互斥锁，可以保护共享资源，避免竞态条件，实现线程间的同步，提高程序的稳定性和可靠性，有效解决了多线程访问共享资源的问题。在功能实现的时候遇到在 read 线程输入 exit 无法及时有效退出的问题，经查阅资料，在调用 pthread_join 函数时，主线程将会阻塞，直到指定的线程结束执行，所以在输入 exit 后 run 函数需等待 attention 线程结束，导致无法及

时退出。因此设置了全局变量 exitflag，当 read 线程接收到 exit 后，将全局变量赋值 1，此时 attention 线程检测到全局变量的变化，立即结束线程，从而使 run 函数可以及时结束。

### 3.2 用户管理模块

#### 3.2.1 权限管理

进行权限划分，避免误操作，遵循最小权限原则。和 linux 的权限管理相似。

Root 用户拥有最大权限，可以进行管理员权限的赋予和剥夺。管理员可以删除普通用户，root 用户可以删除除自己以外的所有用户。同时用户可以修改自己的密码，管理员可以修改普通用户的密码，root 用户可以修改所有用户的密码。

### 3.3 云端同步模块

#### 3.3.1 官方文档的阅读

由于资料较少，需要阅读官方 api 文档找寻相关信息 https://api.libssh.org/stable/libssh_tutorial.html

确认主要的函数（ssh_session、ssh_options_set、ssh_userauth_password、ssh_scp_new、ssh_scp_init、ssh_scp_push_file、ssh_scp_write、ssh_scp_pull_request、ssh_scp_accept_request）等，确定每个函数所需要的参数和返回值。

#### 3.3.2 遍历当前目录所有 txt 文件

使用 glob 库遍历当前文件夹下所有 txt 文件。

#### 3.3.3 index.txt 的使用

从远程拉取文件时，无法实现遍历操作，需要精准匹配文件名才能实现拉取，这与网页的访问是类似的，只有完整匹配才能获取到页面信息。因此，需要提前记录我们上传到文件的文件名，并存储到 index.txt 中，这样，在从远程拉取文件时，我们先获取 index 文件，根据文件获取所有 txt 文件名，再从 txt 中获取所有文件名，并逐个拉取所有文件。

#### 3.3.4 二进制流式传输

scp 传输的文件创建和传输是两个单独的操作，且传输是以二进制的形式来进行的。因此要以二进制的形式打开文件，获取二进制字符串，存储至字符数组中，再将字符数组写入到文件，进而以文本的形式打开读取。

# 三、UI 界面版本说明

使用 Qt Creator 对于带 UI 界面的版本进行开发。

## 1.模块与类的设计

### 1.1 主线程：

登录模块：获取用户的登录信息，并检验用户名和密码是否正确；

login_widget 类：获取用户的登录信息；

任务管理模块：

mainwindow 类：在对话框中展示用户添加的任务，提供增删任务的按钮；

addtask_widget 类：获取用户要添加的任务信息；

task_manager 类：维护用户的任务队列，支持增删操作、从文件中读取任务、将任务信息写入文件等功能；

任务排序模块：支持根据不同的关键字队任务信息进行排序；

用户管理模块：

Users 类：维护用户信息，支持增加用户、删除用户、修改密码、展示用户、赋予管理员权限等功能；

adduser_dialog 类：获取要增加的用户信息；

deluser_dialog 类：获取要删除的用户信息；

changepassword_dialog 类：获取要修改密码的用户信息；

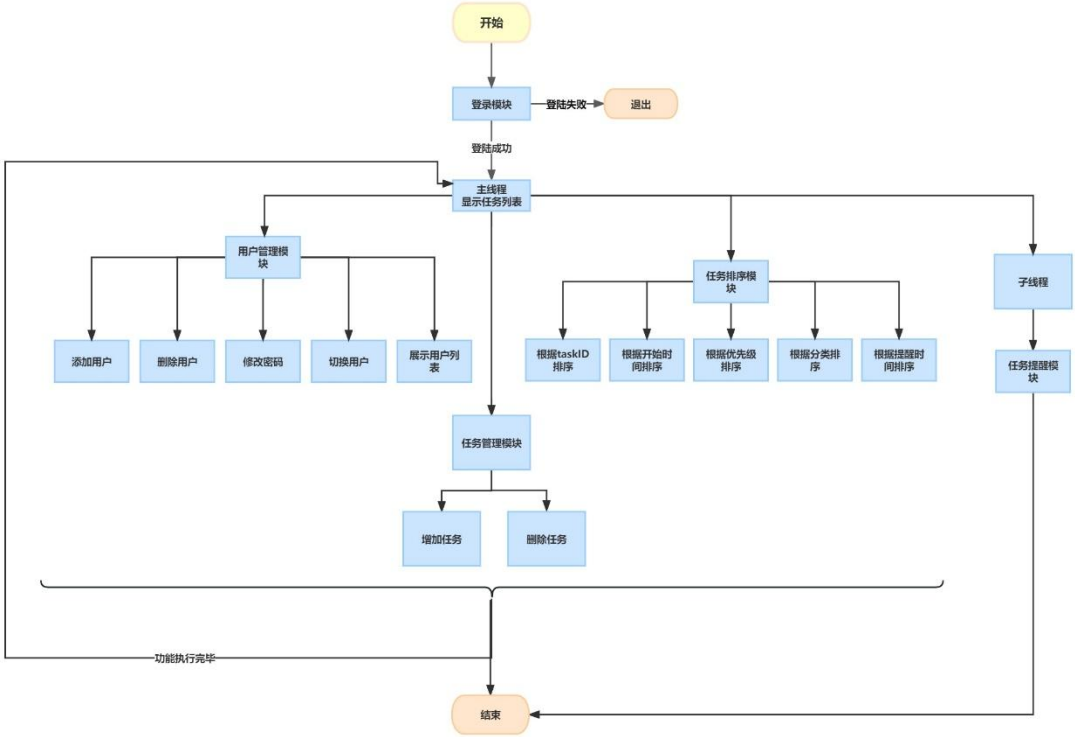
showuser_dialog 类：展示所有用户的信息；

give_admin_dialog 类：通过 root 用户赋予其他用户管理员权限；

### 1.2 子线程（任务提醒模块）：

MyThread 类：在后台对于 task_manager 中的任务进行扫描，对到达提醒时间的任务进行提醒；

## 2.流程图




## 3.关键技术说明

### 3.1 Qt 中的多线程实现

使用 Qt 中的 QThread 类，定义 MyThread 类继承自 QThread 类，重写其中的 run 函数，实现提醒功能。在主线程中的构造函数中，创建 MyThread 的一个实体并调用 mythread->start()函数使其运行。并且在主线程的析构函数中将子线程结束，以避免出现内存泄漏等问题。

### 3.2 Qt 中多线程间的通信

通过 Qt 中的信号与槽功能实现子线程与主线程之间的通信。在子线程中定义信号函数 attend(int)，每次发现到达提醒时间的任务（taskID 为 i）时激活该信号 attend(i)。在主线程中定义槽函数 get_attention(int)，将子线程中的 attend(int)与主线程中的 get_attention(int)进行关联。主线程每次接收到信号 attend(i)时，对于 taskID 为 i 的任务进行提醒。代码实现如下：

```
//创建并运行子线程
mythread = new MyThread(this);
//子线程与主线程通过信号与槽进行通信
void (MyThread::*signal)(int) = &MyThread::attend;
connect(mythread,signal,this,&MainWindow::get_attention);
mythread->start();
```