## 内存管理项目报告之动态分区分配方式

2153401 赵一婷

##### 项目需求

利用代码模拟内存管理中的动态分区分配方式，基本任务要求实现：假设初始态可用内存空间为640K，人为输入一串请求序列，分别使用首次适应算法和最佳适应算法模拟进程内存块的分配和回收，同时通过可视化界面显示出每次分配和回收后的空闲分区链的情况。

##### 项目目的

1.强化对进程的数据结构、空闲分区链表的数据结构以及相关算法的理解

2.加深动态分区存储管理方式及其实现过程的理解

3.锻炼代码编写能力以及可视化设计能力

##### 使用平台及语言

平台：pycharm

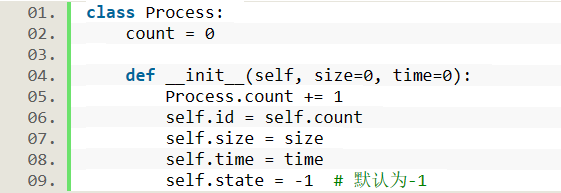
语言：python

可视化实现：pyqt5

##### 设计方案

###### 数据结构设计

①进程类设计



每个进程主要包含四个参数

·id：每个进程各拥有一个id用于做身份标识

·size：表示每个进程所需要占用的内存大小

·time：表示该进程执行完成还需要的时间，没执行一步time值都会减一，直到time值为0时表示该进程执行完成。

·state：表示该进程所处的状态，即是否被分配，初始值默认为-1表示未被分配内存空间，当被分配后将会被赋值为其所在内存块的首地址

此外有三个与进程相关的列表

·p\_wait[]：该列表内存放刚被创建成功，仍未被分配的进程

·p\_exe[]:该列表内存放已经被分配内存空间且正在执行的进程

·p\_finish[]:该列表内存放已经已经执行完成的进程

②空闲分区块链表设计

在本程序中空闲分区块链表的名称为freelist，数据类型为列表，列表中的每一项为一个二元组，分别存放空闲区域的首地址以及空闲区域的大小。freelist初始值为[[0,640]]。

###### 逻辑代码实现

（1）分配算法的实现

①首先适配算法

**首先适应算法介绍**：最先适配算法是一种尽可能利用低地址空闲区存储进程的一种分区分配方式，而在高地址空间尽可能多的保存较大的空闲区，以便满足内存占用较大的进程的需要。

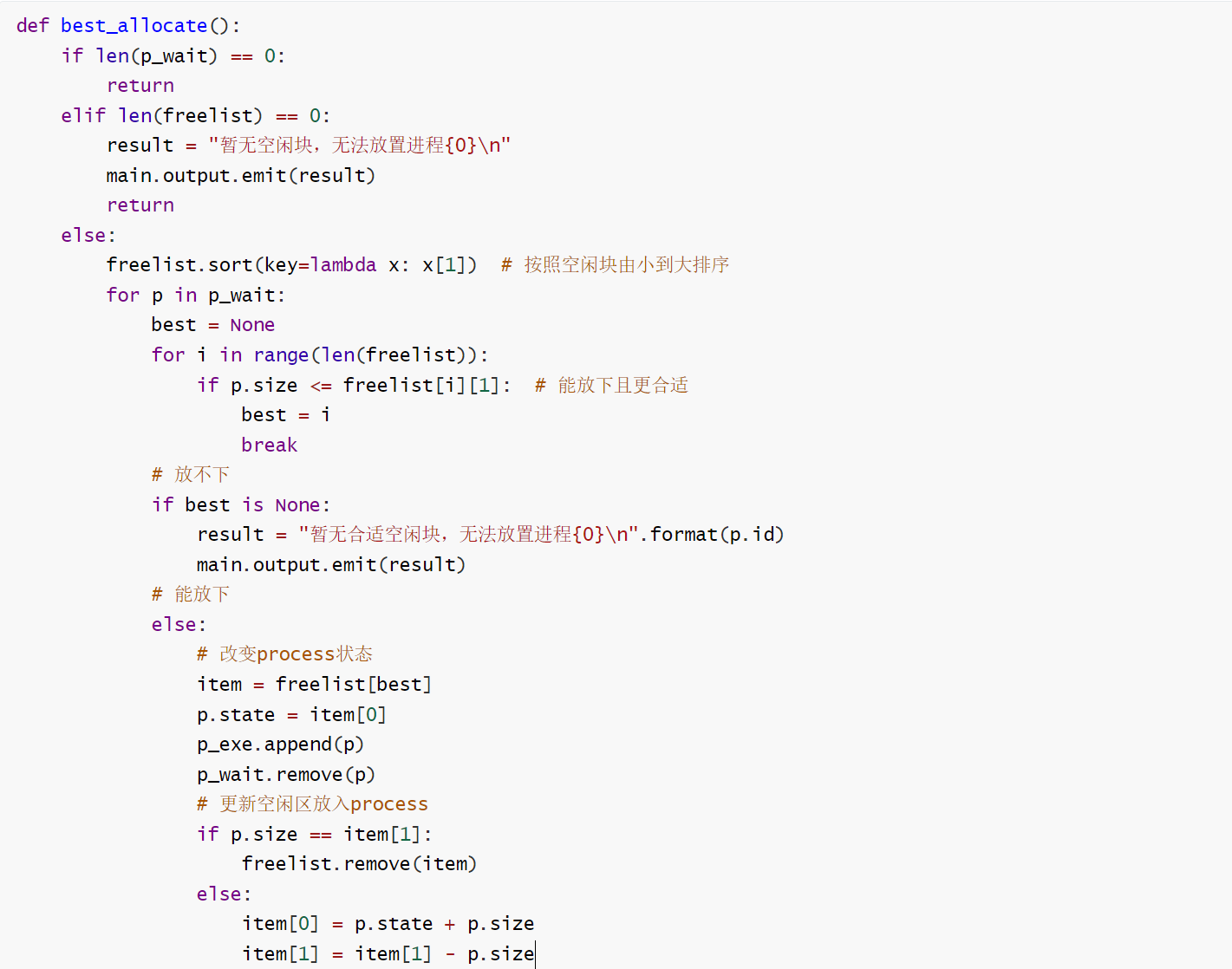
**代码实现逻辑**：通过遍历p\_wait列表，寻找到未被分配内存空间的进程，空闲块链表内的空闲块按照物理地址的顺序排列，因此只要从前往后找到第一个可以放入的空闲块就可以讲进程放入，放入后修改进程状态以及空闲块的情况。若从前到后没有找到合适的位置则提示“无合适空闲块，无法放置”



②最佳适应算法

**最佳适应算法介绍；**最佳适配算法的思想主要是在空闲块链表中寻找到一个最小的能放入进程的空闲块，即对于该分区所要求分配的大小来说时最合适的，分配后所剩余的块会最小。

**代码实现逻辑**：通过遍历p\_wait列表，寻找到未被分配内存空间的进程，再将空闲块列表按照空闲块有小到大的顺序排列，寻找第一个大小大于等于进程大小的内存块，将进程放置进去。若没有找到合适的位置则提示“无合适空闲块，无法放置”。放入后修改进程状态以及空闲块链表状态。



1. 释放空间算法

①free函数

**算法实现逻辑**：通过遍历整个p\_exe，即所有正在执行的进程列表，寻找到p.time值为0的进程，即代表这个进程已经执行结束，修改这个进程的状态，将其放入p\_finish列表中，同时寻找到进程占用内存空间的起始位置以及占用大小，将释放后的空闲块插入空闲块列表中。



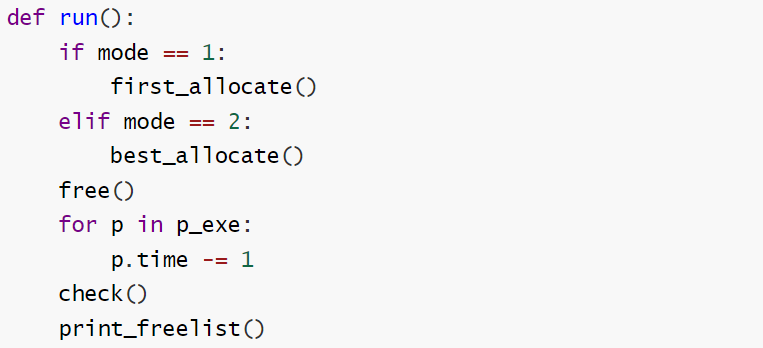
②check函数

**算法实现逻辑**：首先将空闲块链表按照起始地址由小到大进行排序，接下来依次判断前后的两个空闲块是否可以合并，若可以合并则将其合并为一个空闲块。



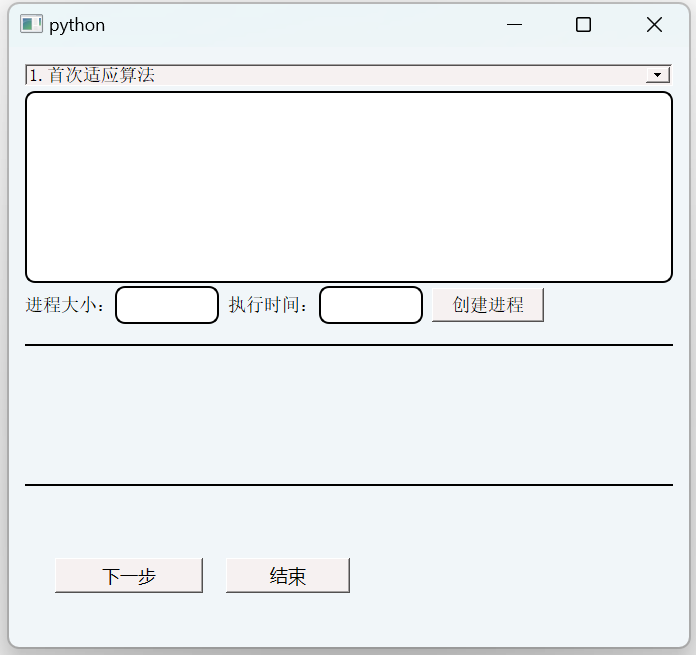
1. 主功能函数实现

算法实现逻辑：每当用户在界面上摁下“下一步”按钮后，就调用一次run函数，若当前模式为首先适应算法，则调用first\_allocate()函数，若若当前模式为最佳适应算法，则调用best\_allocate()函数，完成内存空间的分配

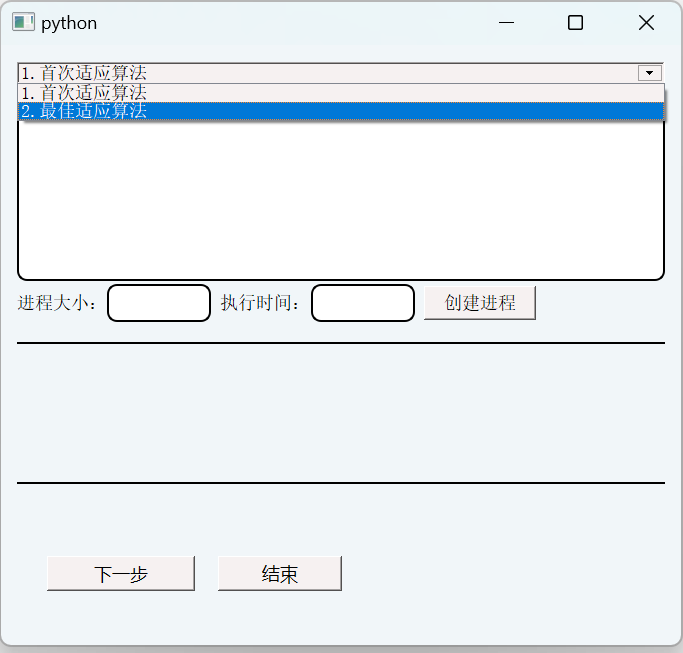


##### 界面展示以及使用说明

①运行代码后我们会看到如下界面：



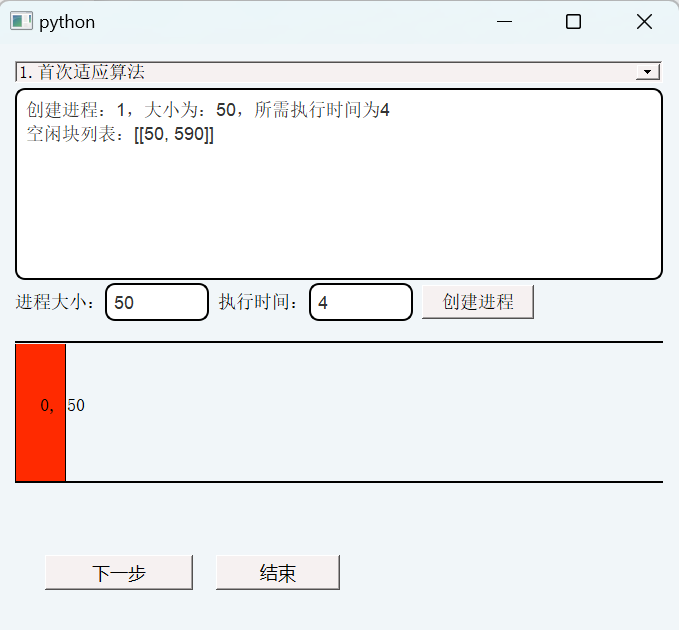
②通过下拉列表我们可以选择分配算法：



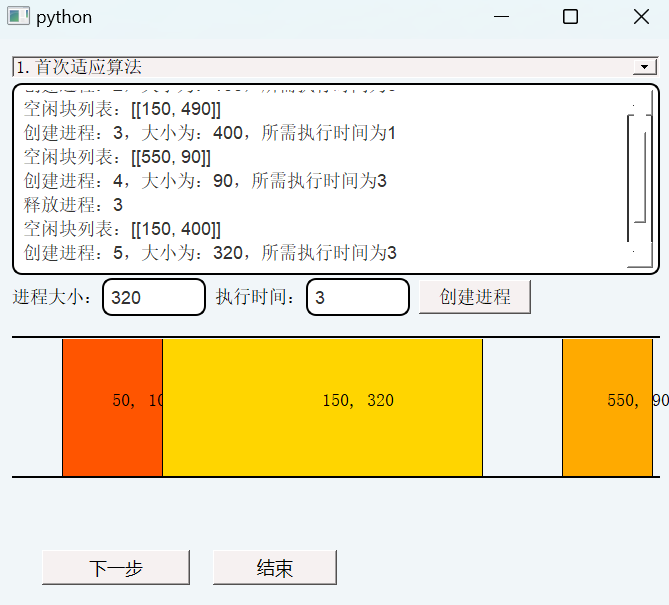
③通过输入进程的大小和执行时间，点击创建进程按钮可以创建一个进程，同时会在控制屏上显示相应进程数据



④点击“下一步”进行进程的分配与释放过程，此时会在下方绘制进程块，模拟内存的分配情况，同时打印出当前空闲块链表



⑤添加多个进程效果如图：



⑥点击结束按钮即可终止模拟

##### 实验改进

①关于内存分配的算法还可以实现最坏适配算法以及循环适配算法，以提高模拟的全面性，同时也可加深对不同算法的理解

②本代码中默认内存大小为640k，可以将其设置为根据用户输入数值来动态设定内存的大小以及界面的展示，这样可以更具有交互性和灵活性

③可以增加一些输入错误的判断，当前的进程只支持输入进程的大小和执行时间为大于0的正数，可以增加一些错误判断来提高代码的鲁棒性

##### 心得体会

通过本次实验使我加深了对动态内存分区分配算法的理解，各类算法的优缺点如下表：

|  |  |  |
| --- | --- | --- |
| 算法名称 | 算法优点 | 算法缺点 |
| 首次适应算法 | 高址部分的大的空闲分区得到保留，为大作业的内存分配创造了条件 | 每次都是优先利用低址部分的空闲分区，造成低址部分产生大量的外碎片 |
| 最佳适应算法 | 第一次找到的空闲分区是大小最接近待分配内存作业大小的 | 会产生大量难以利用的细小的外部碎片 |
| 最坏适应算法 | 效率高，分区查找方便 | 最后剩余的分区会越来越小，无法运行大程序 |
| 循环最先适应算法 | 可以使空闲分区分布更加均匀 | 较大的空闲区不易被保留 |