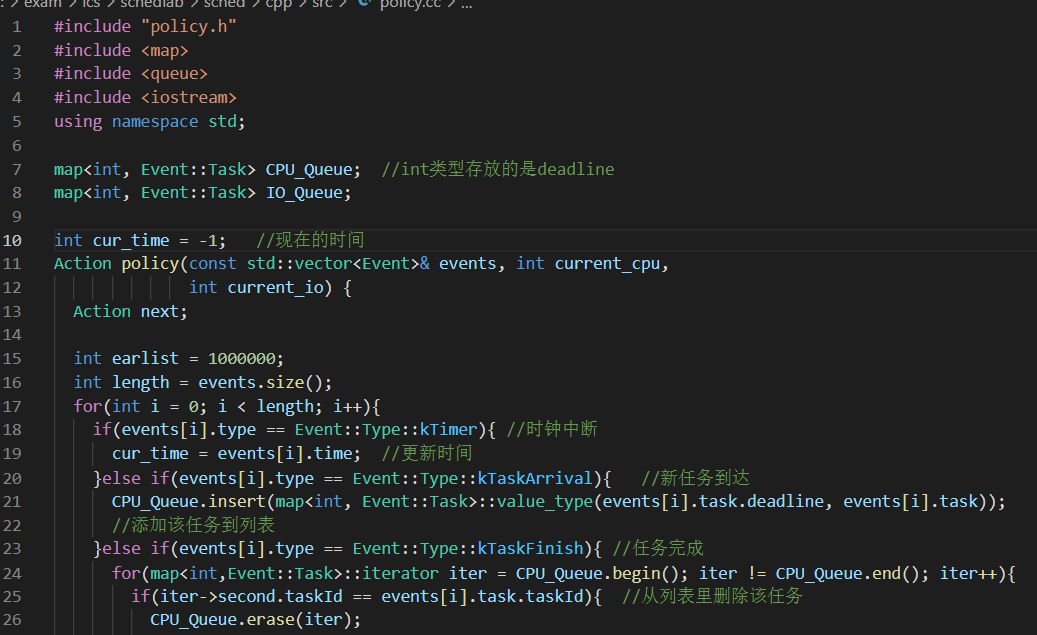
实验报告

一、调度算法思路

为需要使用CPU资源和IO资源的任务分别设置一个等待队列，队列中按照截止时间由小到大排序，即截止时间最近的最靠前。首先遍历事件列表中的事件，并根据事件类型执行不同操作：时钟中断，则更新时间；新任务到达，则将其添加至CPU等待队列；任务完成，则从CPU队列中删除；请求IO，则添加至IO队列并暂时从CPU队列中移除；结束IO，则从IO队列中删除并添加回CPU队列。然后，当需要选择使用CPU或IO资源的任务时，只需要取相应队列中截止时间最近且还没有超时的任务来执行即可。

二、代码







三、曾经尝试

1.第一版

（1）算法思路描述：

这一版先进行一个简单尝试，思路比较简单直接：只考虑任务的截止时间，不考虑其他因素如优先级等等。为需要使用CPU资源和IO资源的任务分别设置一个等待队列，队列以map结构存储，其中key是任务的截止时间，value是该任务，根据map的自动排序功能，等待队列即可按照截止时间从小到大排列，从而便于我们找到截止时间最近的任务。需要注意的是，由于一个任务不能同时占用CPU资源和IO资源，所以限制一个任务能在一个队列中，而不能同时等待CPU资源和IO资源。

首先遍历事件列表中的事件，并根据事件类型执行不同操作：时钟中断，则更新时间；新任务到达，则将其添加至CPU等待队列；任务完成，则从CPU队列中删除；请求IO，则添加至IO队列并暂时从CPU队列中移除；结束IO，则从IO队列中删除并添加回CPU队列。

然后，当需要选择使用CPU或IO资源的任务时，只需要取相应队列中截止时间最近且还没有超时的任务来执行即可。

（2）得分情况：

oj得分86，效果出乎意料地还不错

（3）存在问题：

只考虑了截止时间，而完全没有考虑任务的优先级，所以非常不全面。

2.第二版

（1）算法思路描述：

这一版希望在第一版的基础上，将任务的优先级也纳入考虑因素。仍然使用map结构存储两个队列，但是key值存放的是原始的截止时间乘以一个系数：对于低优先级任务，该系数大于1；对于高优先级任务，该系数等于一。因此，相同截止时间下，高优先级任务在队列中就会更优先。其余实现思路均与第一版相同。

（2）得分情况

低优先级任务权重系数为1.5：86

低优先级任务权重系数为2：80

低优先级任务权重系数为3：80

（3）存在问题：

虽然尝试考虑了任务优先级，但是这种加权考虑的方式似乎不太可靠，可能会让一些截止时间近的低优先级任务不能及时完成。

最终版本：

受时间和精力的限制，以及考虑到现有得分已经达到了86，就没有再进行其他调度思路的实现。而且在得分相同的情况下，希望算法能尽可能简单，所以选择了第一版作为最终版。