系统软件课设 Pintos 项目上机题目与考核须知

2021 年 12 月 20 日

一、 Stride 调度介绍

经过了一学期 Pintos 的『顶峰体验』，你一定受益良多。在 Pintos 中，我们阅读了默认实现的 FCFS 算法，并亲手实现了优先级调度、MLFQ 调度。但是，以上两种算法均不能控制进程运行时间与其优先级的比例关系。下面，请你以本次上机测试给定的 Pintos 版本为基础，实现 stride 调度算法。

1.1 算法步骤

1. 为每一个进程设置一个当前 stride，表示该进程已经运行的『长度』。另外，设置其对应的 pass 值（只与进程的优先级有关系），表示进程在调度后，stride 需要进行的累加值。
2. *每次需要调度时*，从当前 ready 态的进程中选择 stride 最小的进程调度。对于获得调度的进程 P，将对应的 stride 加上其对应的步长 pass。
3. 一个时间片后，回到上一步骤，重新调度当前 stride 最小的进程。

可以证明，如果令 P.pass $= \frac{\text{BigStride}}{\text{P.priority}}$，其中 P.priority 表示进程的优先级（大于 1 的整数），而 BigStride 表示一个预先定义的大常数，则该调度方案为每个进程分配的时间将与其优先级成正比。证明过程我们在这里略去，有兴趣的同学可以于本次上机测试结束后在网上查找相关资料。

1.2 算法细节

- stride 调度要求*进程优先级* $priority \geq 2$，所以设定进程优先级 $p \leq 1$ 会导致错误。
- 进程初始 stride 设置为 0 即可。

1.3 注意事项

在工程实践中，我们会使用固定大小的数据类型（如 `int32_t`）来存储 stride，自然，我们会遇到溢出问题。你的算法应该能够在*上列细节的条件*下正确处理溢出后 stride 的比较，保证每次能够选出不溢出时 stride 最小的进程。

二、 考核要求

2.1 任务说明

你需要在*我们给定的 Pintos 版本*上实现 stride 调度。需要的常数和变量已经定义（`BIG_STRIDE` 和 `stride`）。你需要复用 Pintos 本身的、用于实现优先级调度的优先级变量（`struct thread` 中的 `priority` 域）并保证 `thread_set_priority` 和 `thread_get_priority` 工作正常。

你不应该修改 `threads.h` 中 `BIG_STRIDE` 的定义。

提示：使用 `git diff 8850fb~2..8850fb src` 命令可以查看我们对于 Pintos 基础版做出的修改。

2.2 测试点说明

考核共有 4 个测试，在 `threads` 文件夹中运行 `make check` 即可运行测试（*Pintos Project1 原有的测试已经被删除*）。其中，2 个测试点是开放的，同学可以在上机测试中查看结果；另外 2 个测试点是隐藏的，我们会在评测时补充隐藏测试点的内容，在上机考试中，隐藏测试点的结果固定为『测试不通过』。

表 1: 测试点内容说明

测试名称	测试内容	是否隐藏
stride-one	一个进程的情况是否能正确运行	否
stride-two	两个进程的情况下，能否正常运行，且运行时间是否与优先级成正比	否
stride-multiple	多个进程的情况下，能否正常运行，且运行时间是否与优先级成正比	是
stride-overflow	多个进程的情况下，且 stride 可能溢出时，能否正常运行，且运行时间是否与优先级成正比	是

A GIT 操作简要教程

克隆代码：`git clone --depth=1 < 你的项目地址 >`
提交代码：`git add .; git commit -a`
推送代码：`git push`