PPC 项目

十字路口

1. 目标

本编程项目的目标是使用 Python 设计和实现一个多进程仿真系统。

本文件定义了你必须至少实现的功能，其中一些问题是故意开放的，以便你可以提出自己的解决方案。任何额外的扩展和优化都将对你的评估有利。请务必严谨且富有创造力！

2. 介绍

2.1 用户视角

我们考虑一个十字路口，由两条垂直交叉的道路组成，一条沿南北方向，另一条沿东西方向。该十字路口由四个双色交通灯管理，每个角落各一个。

在正常情况下，同一条道路上的信号灯始终保持相同颜色，而垂直方向上的信号灯颜色相反。例如，当一条道路的信号灯为红色时，垂直方向的信号灯为绿色，反之亦然（不考虑橙色）。

车辆遵守交通规则：

- 只有在绿灯时才能前行。
- 车辆在右转时优先通行。
- 如果有高优先级车辆（如消防车、救护车等）在任意道路上到达，它必须尽快通过。

为实现这一目标，当高优先级车辆接近十字路口时，系统会检测到它，并更改信号灯的颜色以使其能够按期望方向通过。在这种情况下，仅有一个信号灯会被设置为绿灯。我们假设不会有车辆堵塞在十字路口中央，即不会出现交通堵塞。

2.2 技术规格

你的实现至少需要涉及 5 个进程：

- **normal_traffic_gen**（普通交通生成器）：模拟普通交通的生成。对于每辆生成的普通车辆，随机或根据某些预设规则选择其出发道路和目的地道路。
- **priority_traffic_gen**（高优先级交通生成器）：模拟高优先级车辆的生成。对于每辆高优先级车辆，随机或根据某些预设规则选择其出发道路和目的地道路。
- **coordinator**（协调器）：根据交通信号灯状态和交通规则，允许所有车辆（包括普通车辆和高优先级车辆）通行。
- **lights**（交通信号灯管理）：在正常模式下按固定时间间隔更改信号灯颜色，并在收到 **priority_traffic_gen** 通知时切换到适当的颜色。
- **display**（显示模块）：让操作员可以实时观察仿真过程。

进程间通信

- **消息队列**：十字路口的四个道路部分由四个消息队列表示，每个部分一个。车辆通过消息进行表示，这些消息编码了车辆的属性。
- **信号通知**：高优先级车辆的接近情况通过信号通知 **lights** 进程。
- **共享内存**：交通信号灯的状态存储在共享内存中，至少应对 **coordinator** 进程可访问。
- **套接字通信**：**display** 进程通过套接字与其他进程通信，以获取实时仿真数据。

3. 实现

3.1 设计

请先绘制交互流程图，以便更好地可视化进程/线程之间的交互。状态机图在此阶段非常有帮助。你需要考虑并解释以下要点：

- **进程之间的关系**（父子进程或无关联）。仅在绝对必要时定义父子进程关系。
- **消息队列**：进程间的通信方式，包括消息类型、内容及交换顺序。
- **套接字通信**：进程间的通信方式，包括数据类型、内容及交换顺序。

- **共享内存结构**：存储的数据类型及访问方式。
- **进程信号**：进程间交换的信号类型及其处理程序。
- **同步机制**：用于保护共享资源访问或资源计数的同步原语。
- **管道通信**（如果使用）：涉及的进程对及其类型。

编写类 Python 语法的伪代码，涵盖每个进程/线程的主要算法和数据结构，以帮助实现。

3.2 实施计划

请规划你的实现和测试步骤。我们强烈建议：

1. **单独实现和测试每个进程/线程**，在初始阶段使用硬编码数据进行测试。
2. **逐步实现和测试进程间通信**，先单独测试每对通信进程/线程（必要时仍然使用硬编码数据）。
3. **集成所有进程并测试仿真系统**，确保进程间通信正确运行。
4. 关注 **仿真启动和正确关闭**，释放所有资源。
5. 识别可能的 **故障情况**，并考虑恢复或终止策略。
6. 确保能够在 **短时间演示** 你的实现成果。ss