任务调度器设计方案（简化版）

完整的 `任务调度器` 设计方案见 design，本方案删减了其中进程、线程切换、系统调用转发等部分功能，旨在 kernel bypass 或 unikernel 条件下，构建低时延服务。

本方案仅定义 `任务调度器` 暴露给软件的接口以及内部行为，将 `Executor` 集成到 `任务调度器` 中，不涉及具体的方案。（若不理解 `Executor` ，请参考资料）

软件接口

接口	描述
reset()	重置 任务调度器
lidt(idt: Idt)	设置中断向量表，其中保存对应的中断处理 TaskRef
spawn(task_ref: TaskRef)	创建 Task ，将 Task 添加至就绪队列
fetch() -> Option<TaskRef>	取出优先级最高的 Task

寄存器

1. `stask` ： 用于创建 `Task`
2. `ftask` ： 用于取出 `Task`
3. `idt` ： 用于 `lidt` 操作
4. `control` ： 用于控制

内部行为

内部集成 `Executor` ， `Executor` 内部为每个优先级设置一个 `FIFO` ，保存 `TaskRef` ，在软件调用 `spawn` 或 `fetch` 接口时，从对应的 `FIFO` 中添加或取出 `TaskRef` 。

当收到中断信号时，根据中断向量号，从中断向量表中获得对应的中断处理 `TaskRef` ，将 `TaskRef` 添加到 `Executor` 优先级最高的 `FIFO` 中。