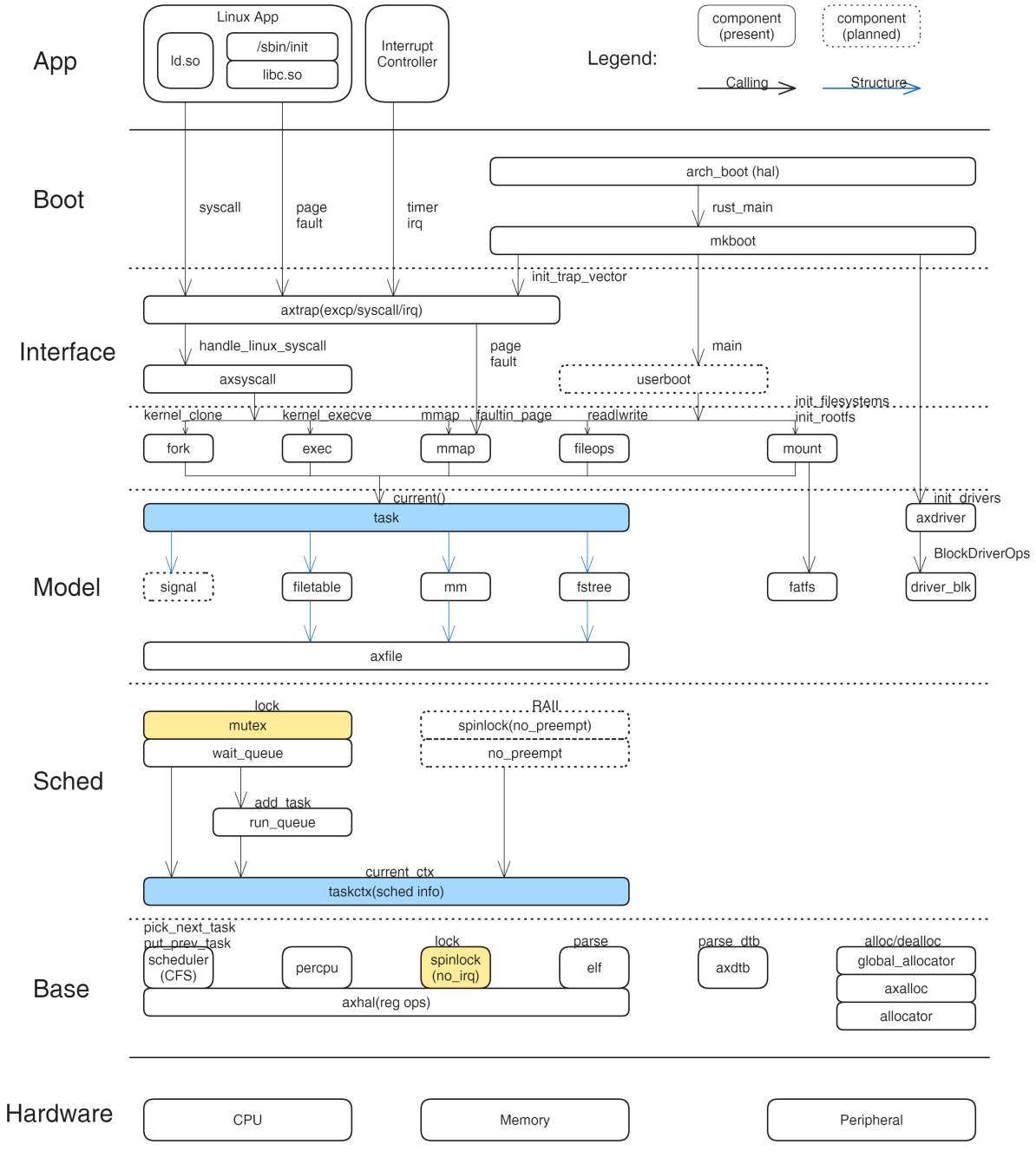
近两周的工作

1. 基于单向依赖的设想，与郑友捷同学对LK验证模型中的一些关键组件进行了拆分，重新调整了层次。
2. 又根据杨金博同学提出的Mutex层次的问题，结合郑友捷同学建议，把task拆分为资源管理的上层组件task和包含调度信息的下层组件taskctx(sched info)，让对file/dev之类的资源，可以利用Mutex睡眠锁。
3. 目前剩余还需要调整拆分的是与关闭抢占相关的那个kernel_guard，完成这个之后，应该基本达到单向依赖的要求。

   验证仓库和分支：git@github.com:shilei-massclouds/org_arceos.git 切换到 split_task分支
4. 对目前的思路、实践过程和下步想法，整理组织了报告材料，提交到kern_crates/doc下面。

   提交位置：docs/lk_model at main · kern-crates/docs (github.com)

当前模型中组件构成的状态



目前用于验证的应用还很简单，是动态编译的glibc应用，大约能够发出十几个系统调用。

原型在功能上基本覆盖到了内核的各个基本功能：

1. 支持syscall方面的fork/exec/mmap/fileops/mount等
2. 核心功能有内存管理、任务调度、驱动与文件系统以及底层的基础设施支持

目前原型基本打通了各个层及组件之间的关系，一定程度上说明了单向依赖的可行性。

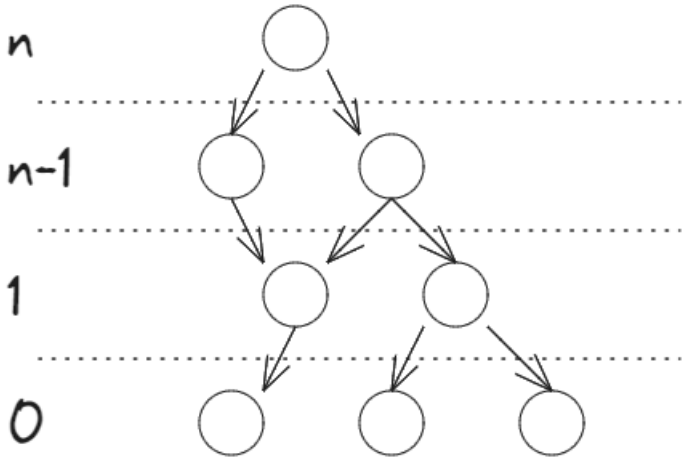
> 蓝色：把task拆分为资源管理的上层组件task和包含调度信息的下层组件taskctx(sched info)
>
> 黄色：Mutex睡眠锁和SpinLock自旋锁

建立单向依赖和控制组件细粒度的目的

在原型阶段，基于单向依赖和细粒度组件划分，给后面的开发提供一个**相对强制**的约束，达到目的：

1. 在将来的持续迭代开发过程中，一直能保持组件的相对独立性，控制他们之间的耦合关系
2. 为自底向上的测试和验证提供一个基础条件

在单向依赖关系下，一个系统的组件之间会形成一个单向的图结构。每个组件具有明确的层次位置。



0层不依赖任何其他组件。第n层直接依赖第n-1层组件。

我们可以基于此进行某种类似于数学归纳的方式来解决测试和验证问题。

每个组件都自成一个系统，它作为根代表了一个系统。

例如我们想证明或者表明一个系统符合某种要求或特性F，可以通过一下两步：

1. 如果一个第0层组件本身符合F，那么它代表的系统也符合F。
2. 如果一个第n层组件本身符合F，并且它直接依赖的第n-1层也都符合F，那么这个第n层组件代表的系统符合F。

证明一个复杂的系统符合某种特性或要求通常比较困难，但是证明一个单纯的组件符合特性或要求会相对简单。这样可以把复杂问题分解为一系列小问题，逐步解决。

目前对这个方法进行实践的方式是测试，起个名叫"塔顶测试"：自底向上逐级进行组件级测试，下层组件达成测试要求是上层组件测试的基础，最后证明整个系统符合了测试要求。

进行这项工作的**前提条件**：为每个组件增加一个默认的init方法，它会调用其**直接依赖**的组件的init方法，其作用是把组件初始化为待命状态。目前对所有组件的初始化工作，是在axruntime这个高层的组织型组件统一进行的，需要下移到每个组件的init方法中。类似于：

```
pub fn init() {
    dep_mod1::init();
    dep_mod2::init();
    ... ...
}
```

下步重点工作任务

1. 调整拆分与关闭抢占相关的那个kernel_guard，并确认目前符合了单向依赖的要求(axlog可以例外)。
2. 给每个组件至少加上一个默认的init方法，它会调用其**直接依赖**的组件的init方法，其作用是把组件初始化为待命状态。
3. 自底向上把组件放到单独仓库，就放在kern_crates组织下。
4. 在2和3的基础上，在每个组件单独仓库中加一个“塔顶测试”组件，通过实践确认“塔顶测试”方式的可行性。
5. 引入LTP(Linux Test Project)基于内核的syscall进行测试：目前已经做了两个准备

   (1) 基于riscv编译了ltp，它包含的syscall测试用例部分可以编译出来，都是基于glibc的简单的各个syscall测试用例

   (2) 扩展Makefile，支持下面的运行方式

   ```
   # 当前单独运行一个testcase
   make run INIT_CMD=/testcases/mmap01
   # 将来批量运行
   make run INIT_CMD=/testcases/runltp
   ```

   > 选择LTP的目的：syscall方面的各个测试用例比较单纯，比较有利于目前原型的验证工作