

Unikernel地址空间与分页、多任务支持（协作式）

实验 U.3.0 ReadPFlash

在组件化中的实现 Paging

实验U.4.0 ChildTask

Task的数据结构

实验U.5.0 MsgQueue

实验 U.3.0 ReadPFlash

恒等映射 —— 恒等映射+ 偏移映射 —— 后期重建映射

在组件化中的实现 Paging

```
[dependencies]  
axstd = { features = ["alloc", "paging"],  
  }
```

实验U.4.0 ChildTask

Task的数据结构

```

pub struct TaskInner {
    id: TaskId,
    name: String,
    is_idle: bool,
    is_init: bool,

    entry: Option<*mut dyn FnOnce()>,
    state: AtomicU8,

    in_wait_queue: AtomicBool,
    #[cfg(feature = "irq")]
    in_timer_list: AtomicBool,

    #[cfg(feature = "preempt")]
    need_resched: AtomicBool,
    #[cfg(feature = "preempt")]
    preempt_disable_count: AtomicUsize,

    exit_code: AtomicI32,
    wait_for_exit: WaitQueue,

    kstack: Option<TaskStack>,
    ctx: UnsafeCell<TaskContext>,
    task_ext: AxTaskExt,

    #[cfg(feature = "tls")]
    tls: TlsArea,
}

```

is_idle: 本身是系统任务idle
is_init: 本身是任务main(其实就是主线程)

entry: 实现任务逻辑函数的入口
state: 任务状态, 即上页所示的四个状态

kstack: ArceOS任务相当于线程, 所以具有自己的栈空间

ctx: 上下文类型TaskContext,
调度的核心数据结构
保存恢复任务的关键

任务的扩展属性, 是面向宏内核和Hypervisor扩展的关键;
但对于Unikernel, 它是空, 本节略过。

注意默认内置任务: Main, gc, idle

Main: 执行应用逻辑的主线程, 它完成退出会导致系统退出。

gc: 除main之外的任务(线程)退出后, 由gc负责回收清理。

IDLE: 当其它所有任务都阻塞时, 执行它。对某些arch, wait_for_irqs对应非忙等指令

实验U.5.0 MsgQueue

保存任务上下文——保存任务状态的最小的寄存器状态集合。

协作式调度定义: 任务之间通过“友好”协作方式分享CPU资源。当前任务是否让出和何时让出
CPU控制权完全由当前任务自己决定。