复用 multiplexing

隔离 isolation

交互 interaction

xv6 宏内核 monolithic kernel Unix.

整个操作系统都置于内核中. 整合性强，但不安全

一点bug导致整个内核崩

另一种 微内核 micro kernel

内核包含很少组件，内核其它部分作为用户程序运行.

特点：代码量小，安全性高；但开销大(消息传递)，性能不好；紧密性低

代表：minix

xv6 运行在多核的 RISC-V 微处理器上.

↓

多个 CPU 共享主存，并行执行，都拥有自己的寄存器副本

xv6 以 "LP64" 标准的 C 语言编写

- long, pointer 类型是 64 位.

  int 类型是 32 位.

  eg: sizeof(char*) = 8 (字节) 1字节 8bit

QEMU 模拟了硬件.

2.1 Abstracting Physical Resources

抽象化物理资源

隔离：禁止用户程序直接访问底层硬件资源，不能让进程影响到其他进程工作.

操作系统，将硬件资源从设备中抽象出来，提供简单的访问接口

同时，统一，便于使用，隐藏细节.

便利性，强隔离

eg: open, read, write, close 访问磁盘资源

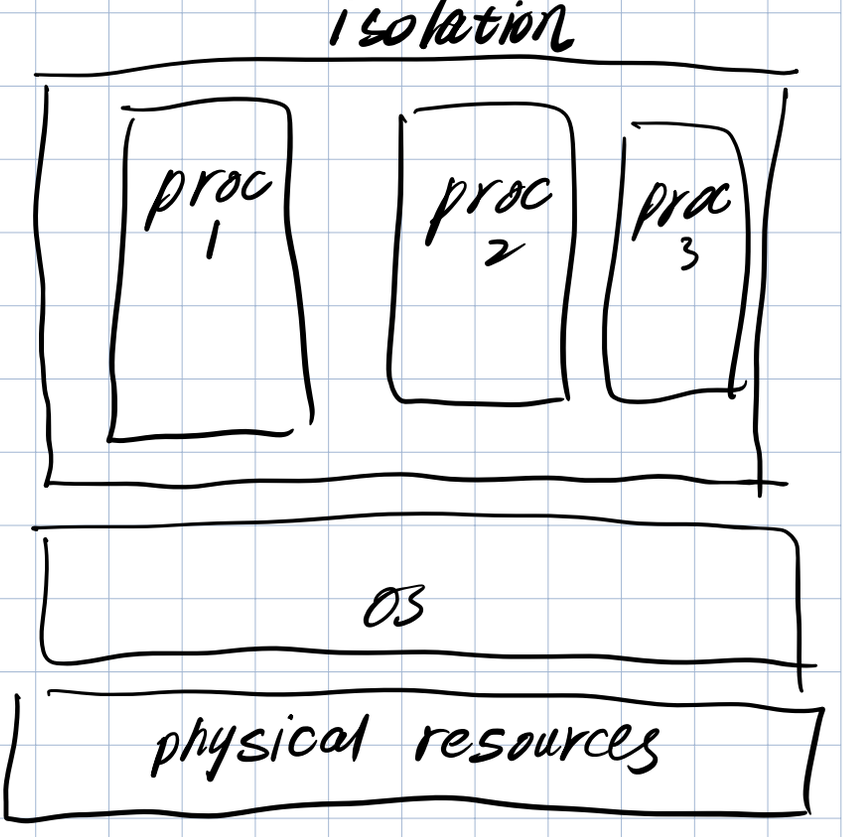
exec 使用内存，而不是自己访问物理内存

2.2 User Mode, Supervisor Mode, and System Calls

用户模式，监管者模式和系统调用

总结计算机隔离三点：

1. 用户进程　　硬件资源
2. 用户进程　　操作系统
3. 用户进程之间



隔离首先需要硬件支持，这是根本.

xv6 RISC-V CPU 有三种指令执行模式

1. machine mode 机器模式 (之后解释. 与启动有关)
2. supervisor mode 监管者模式 (可以执行特权指令)
3. user mode 用户模式 (不能...)

privileged instructions:

打开和关闭中断.

读写寄存器

用户进程只能执行用户模式下的指令，我们说

这个进程在用户空间下运行.

监管者模式下的代码可以执行一些特权指令.我们说

这是在内核空间下运行

在内核空间下运行的代码就是内核

(监管者模式.内核空间，内核这些概念是很相互纠缠的，
实际实现上也是如此，不必扣概念).

在第一章的时候我们已经介绍过，一个用户程序如果想要调用一个位于内核中的函数（例如我们现在只考虑系统调用），首先需要想办法移交CPU给内核运行，也就是说，我们需要切换并进入到内核。CPU应该提供一个特别的指令，可以令CPU从用户模式切换到监管者模式，同时通过一个由内核预先就设置好的进入点entry point，顺利地进入到内核当中。

RISC-V提供了这个特别的指令，它叫做**ecall**。一旦CPU成功切换到监管者模式下运行，内核就成功接管CPU了，接着内核就可以仔细检查传入的系统调用参数，从而决定该用户程序请求的操作是否能够被执行，然后为用户程序执行它或者拒绝其执行请求。

内核的进入点应该由内核自己来控制，否则，如果用户程序可以自己决定从哪里进入内核的话，那可能会有恶意的用户程序，在进入内核的同时，跳过一些重要参数的检查。

那很重要，

下附图示

2.3 Kernel Organization 内核组织架构

较深入说明宏内核设计和微内核设计

2.4 Code：Xv6 Organization in kernel/目录.

2.5 Process Overview 总览进程