CPU 虚拟化

为了实现 CPU 的虚拟化，操作系统需要实现一些低级机制和高级智能。低级机制称为*机制* `mechanism`，用以实现基本的所需功能，高级智能称为*策略* `policy`，也就是操作系统用于做出某种决定的算法

更本质地来说，机制主要回答「如何做」而策略主要回答「做哪个」

进程抽象

操作系统为正在运行的程序提供**抽象**，这也就是所谓的进程 `process`。进程的机器状态 `machine state` 由以下三类组成：

- 进程可以访问的内存（也就是进程的地址空间 `address space`）。这是因为指令和数据均存储在内存中，而一个进程必须要去访问这些内存
- 寄存器。进程在运行的过程中依赖于寄存器
- 文件列表。程序可能会访问 `I/O`，因此进程的组成需要包含相应的文件列表

在正式运行进程前，操作系统需要先将**代码**和**静态数据**（需要初始化变量）加载到内存当中（也就是进程的地址空间内）。这是因为程序都是以可执行程序的形式存储在**磁盘**中，因此操作系统需要预先将其加载到内存才可以继续后面的步骤

将代码和静态数据加载到内存后，操作系统需要为进程分配**运行时栈**和**堆**。运行时栈用于存放局部变量、函数参数、函数返回地址；堆用于存放程序内显示请求分配的数据

此外，操作系统还需要去初始化一些与 `I/O` 设置相关的工作。例如系统中每个进程都会默认打开三个文件描述符，分别为：标准输入、标志输出、标志错误

完成上述工作后，操作系统才会开始启动进程，将 CPU 的控制权移交新创建的进程中

数据结构

对于每个进程，操作系统需要去跟踪它并记录相关的信息，以下是 `vx6` 中相应的结构体：

```
//包含进程切换时需要保存和回复的寄存器
struct context {
  uint edi;
  uint esi;
  uint ebx;
  uint ebp;
  uint eip;
};


// 进程状态
```

```
enum procstate { UNUSED, EMBRYO, SLEEPING, RUNNABLE, RUNNING, ZOMBIE };

/*
    UNUSED: 未使用，表示该进程控制块当前未被使用。
    EMBRYO: 新建，表示该进程正在创建，但尚未准备好运行。
    SLEEPING: 休眠，表示该进程正在等待某个事件的发生，例如等待 I/O 完成、等待定时器超时
等。
    RUNNABLE: 可执行，表示该进程已经准备好运行，并且等待被调度。
    RUNNING: 正在运行，表示该进程当前正在 CPU 上执行。
    ZOMBIE: 僵尸状态，表示该进程已经终止，但其父进程尚未回收它的资源。
*/

struct proc {
  char *mem;                   // 内存位置，仅在使用kvmalloc时才有意义
  uint sz;                     // 进程占用内存大小
  char *kstack;                // 进程内核栈的起始地址
  enum procstate state;        // 进程状态：UNUSED, EMBRYO, SLEEPING, RUNNABLE,
RUNNING, ZOMBIE
  int pid;                     // 进程ID
  struct proc *parent;         // 父进程
  struct trapframe *tf;        // 指向当前进程的中断帧
  struct context *context;     // 进程的上下文信息，用于函数调用
  void *chan;                  // 进程等待的channel地址
  int killed;                  // 进程是否已经被杀死
  struct file *ofile[NOFILE];  // 打开文件的指针数组
  struct inode *cwd;           // 进程当前工作目录的inode
  char name[16];               // 进程名
};
```

进程 API