存储管理

- 基本概念
- 物理内存管理
- 伙伴系统
- 基本内存管理方案
- 交换技术（Swapping）

首先我们介绍一个在存储模型当中非常重要的概念 地址重定位。

地址转换、地址映射、地址翻译

基本概念：地址重定位

RELOCATION

首先我们介绍一个在存储模型当中非常重要的概念 地址重定位。

……已经了解的

- 程序装载到内存才可以运行

  通常，程序以可执行文件格式保存在磁盘上

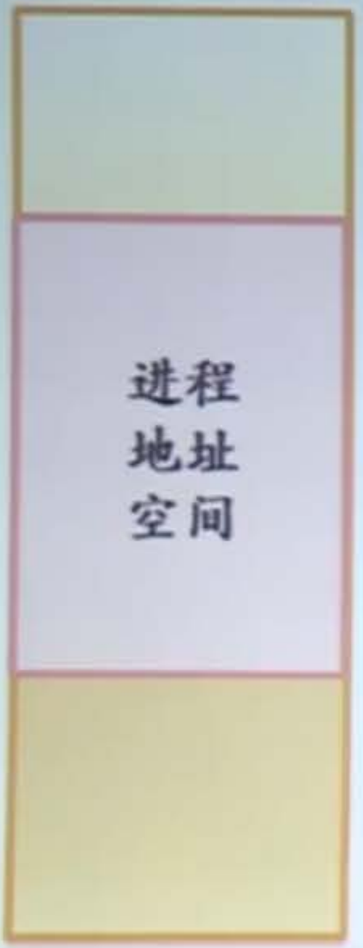
- 多道程序设计模型

  允许多个程序同时进入内存

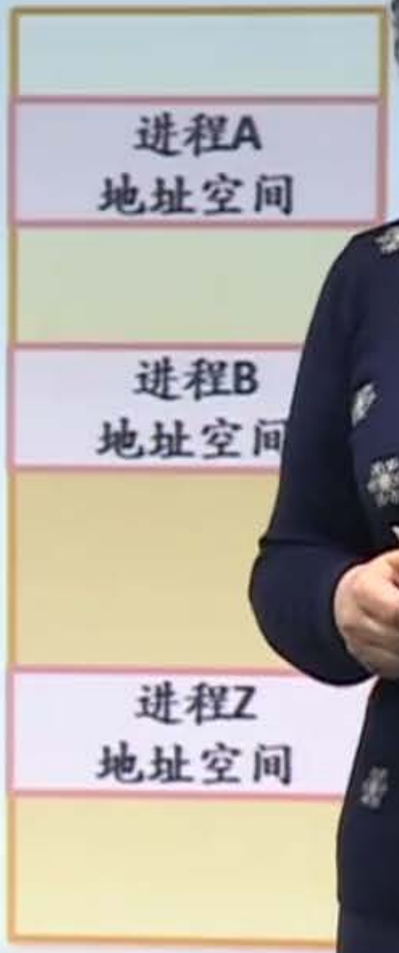
- 每个进程有自己的地址空间
  - 一个进程执行时不能访问另一个进程的地址空间
  - 进程不能执行不适合的操作

那么现在我们来看一下 在存储模型这个大的主题当中，我们要解决哪些问题

要解决的问题

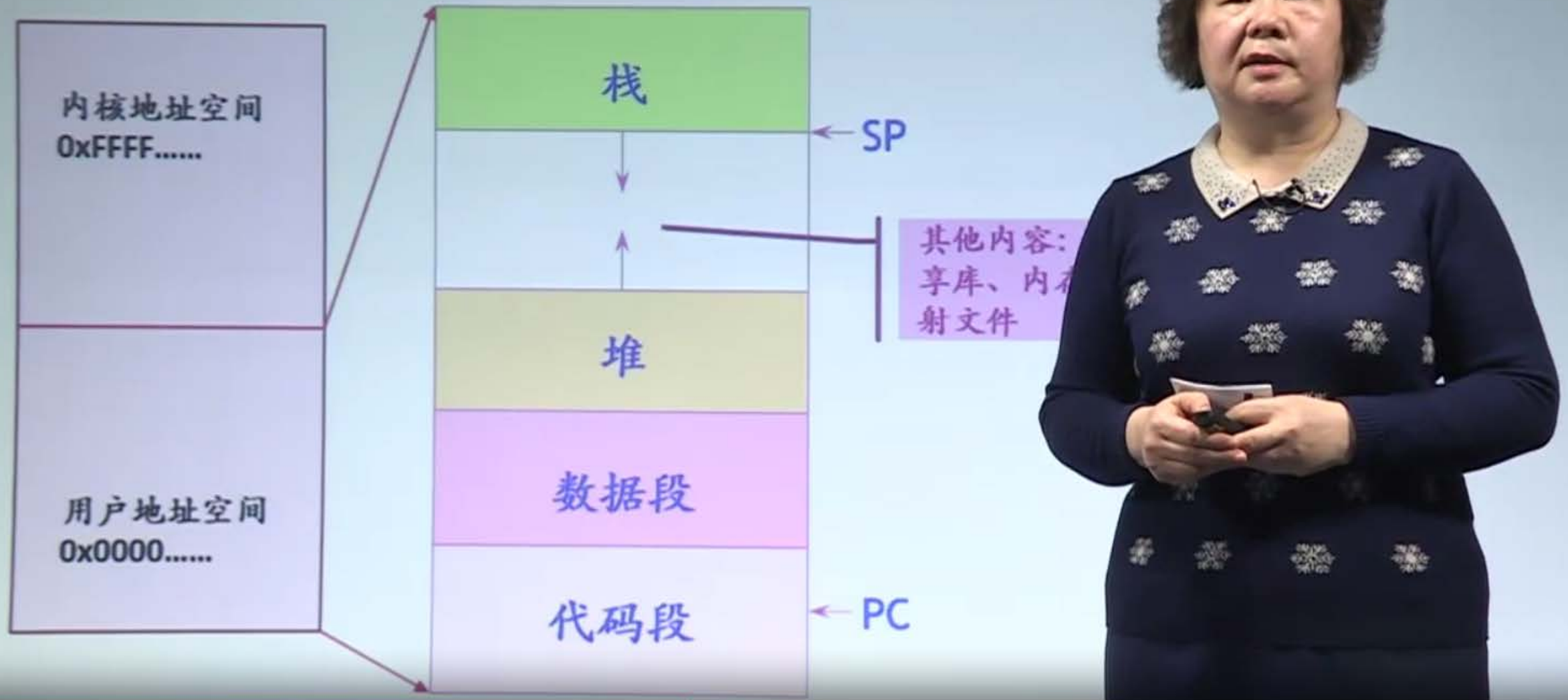


物理内存



物理内存

复习：进程地址空间



讨论

- 进程中的地址不是最终的物理地址
- 在进程运行前无法计算出物理地址

  因为：不能确定进程被加载到内存什么地方

→→ 需要 地址重定位 的支持

地址转换、地址变换、地址翻译、地址映射

Translation、Mapping

这个概念呢通常也有其他的一些名词 那么我们现在介绍什么是地址重定位

地址重定位

- 逻辑地址（相对地址，虚拟地址）

  用户程序经过编译、汇编后形成目标代码，目标代码通常采用相对地址的形式，其首地址为0，其余地址都相对于首地址而编址

  不能用逻辑地址在内存中读取信息

- 物理地址（绝对地址，实地址）

  内存中存储单元的地址　可直接寻址

为了保证CPU执行指令时可正确访问内存单元，需要将用户程序中的逻辑地址转换为运行时可由机器直接寻址的物理地址，这一过程称为地址重定位

静态重定位与动态重定位

静态重定位：

当用户程序加载到内存时，一次性实现逻辑地址到物理地址的转换

- 一般可以由软件完成

动态重定位：

在进程执行过程中进行地址变换

→→ 即逐条指令执行时完成地址转换

- 需要硬件部件支持

动态重定位实现

这就是一个示例

0:00 大家好，今天我给大家带来的是操作系统原理课的第 7 讲 存储模型。 在这一讲里头，我们主要介绍 一个非常重要的基本概念，地址重定位 另外我们要介绍物理内存管理和一种实例 伙伴系统。 接着我们会去介绍各种基本的内存管理方案 最后我们介绍交换技术这样一个概念 首先我们介绍一个在存储模型当中非常重要的概念 地址重定位。 那么这个概念呢，是贯穿整个存储模型 那么这个概念的同时又有几个不同的术语，比如说地址转换、 地址映射 还有地址翻译，其实都是同样的技术 首先，我们先看看作为 计算机专业的学生，他应该已经了解的一些内容 第一个 程序要加载到内存才可以运行 那么我们程序平常是以可执行文件的方式 保存在磁盘上。 大家可能要回忆一下，有哪些典型的可执行文件的格式 那么代码和数据都保存在这个文件里头 第二呢，我们现在计算机系统采用的是多道程序设计模型 所谓多道程序设计模型，就指的是同时允许多个程序 同时进入内存，也就是在内存当中 不同的进程都会占据一部分空间 第三个 我们前面已经介绍过，在创建进程的时候 操作系统为每个进程分配了一个地址空间 也就是说每个进程有自己独立的地址空间 因此一个进程在执行的过程中，它是不能够访问 另外一个进程的地址空间的。 同时 对于进程来讲，那么它不能，对内存 执行一些不合适的操作，那么这样会导致 出现一些问题。 所以这是由于每个进程具有一个自己独立的地址空间 那这是我们应该已经学到的、 了解到的知识 那么现在我们来看一下 在存储模型这个大的主题当中，我们要解决哪些问题 我们用一个图来示意一下 我们有两个空间。 进程呢有一个进程的地址空间 那么进程要执行必须进入内存 所以呢，我们有一个物理内存。 那么这个进程地址空间要 把它装到了物理内存，加载到物理内存，那么这个进程才可以去执行 这是一个一般意义的一个示意 那么如果我们是多道程序设计模型 那么也就是说有多个进程，那么它们都要进入物理内存 也就是可能进程 A 占据一部分空间 进程 B 占据一部分空间，每个进程都占据一份空间 所以在我们这个存储模型的主题之下，我们就是要解决 如何把一个进程的地址空间的内容把它装载到内存 然后呢合理地来分配使用内存，使得每一个进程能够非常 正确地执行，这就是我们这个主题要解决的问题 对于进程地址空间，我们简单地复习一下前面所讲的这个概念 那么这是进程地址空间。 当然这个空间呢一般通常一分为二 那么有一部分是操作系统内核占据的空间 然后一部分是用户的代码、 数据、 堆栈等所用的地址空间 那么我们当然比较关注的是用户的这部分地址空间 那么它又应该有一个布局，那么这个布局呢我们举的是 Linux 的一个例子 那么主要的内容都在这里头，比如说进程的代码部分、 数据部分 还有在运

行过程中呢可能扩展的像堆或者是栈 存放一些中间结果或者是一些函数调用的一些参数啊等等 那么这些是在进程运行过程中会不断地扩展 那么当然了还有一部分空间呐用于来存放一些 其他的内容，比如说可能是共享库，可能是一些啊 内存映射文件，所以这是一个复习啊，我们看一下地址空间这个概念。 我们现在 就是要把这些内容把它读进内存，把它装载在内存里头去，让它能够 正确执行。 那么我们再讨论一下 看到刚才这样一个地址空间，我们得出这样一个结论啊 就是进程当中的地址不是最终的物理地址 这句话的含义就是说每个进程都有自己独立的一处空间，那么它的这个编址 指令或者是一些数据所在的这个地址不是内存的物理地址 因此呢，在进程运行之前，由于我们不知道 这个进程装载到内存的什么部分 那么因此在进程运行之前我们不能计算出这个物理地址到底是什么，所以我们用的地址呢 不是物理内存地址，这就是因为我们不能 确定进程是被加载到内存的什么地方 这是因为这个原因，因此呢我们就需要一个机制 来解决我们这个问题。 那么这个机制呢 就是我们这一小节里头最重要的一个概念叫地址重定位 需要这个地址重定位这个机制来解决从 进程空间的地址一直到物理地址的这么一个转换问题，因此呢 这个概念呢通常也有其他的一些名词 那么我们现在介绍什么是地址重定位 刚才我们说的进程地址空间的地址不是最终的物理地址 那么它，我们把它称之为叫做逻辑地址 逻辑地址呢有的时候也叫相对地址或者叫虚拟地址，在不同的场合呢用不同的词 如果我们的计算机支持虚存机制，那我们通常得到的这个地址我们可以称之为虚拟地址 那么一般情况下我们叫逻辑地址。 所谓逻辑地址呢是 用户程序经过了编译、 汇编之后形成了目标代码 那么目标代码当中呢通常是采用这种相对地址的形式 也就是说首地址我假设是 0 其余的地址都是相对于首地址而编址 那么这就是逻辑地址它的特征 CPU 取到了逻辑地址 它是不能用这个逻辑地址到物理内存去读取信息的 因为这不是最终的物理地址。 那我们再看看 另外一个概念，就是物理地址，也称为 绝对地址和实地址，相对于上面的相对地址就是绝对地址，相对于虚拟地址就是实地址 那么物理地址呢实际上就是内存存储单元的地址 那么这个地址呢是可以直接寻址的，也就是可以直接 到内存相应的单元去取到相应的指令或者数据的 那么由于这是两个不同的空间，所以呢我们在 计算机执行这个进程的时候 为了保证 CPU 执行的指令可以正确访问到 对应的内存单元就需要将用户程序中的逻辑地址 转换为在运行时可以直接寻址的物理地址 那么这个过程实际上呢就是称为地址重定位 所以地址重定

位是指的是一个转换的一个过程 在 谈到地址重定位的时候呢，我们有两种不同的方式 一种叫做静态重定位，一种叫做动态重定位 所谓静态重定位，指的是当这个把用户程序 加载到内存中的时候，这个时候要一次性的把所有的逻辑地址把它 转换到物理地址，这样的话呢我们可以通过一个软件，就加载 这个软件就可以完成这样一个过程。 那么它的 好处呢就是在程序的执行过程中，那么这个地址就直接可以拿来去到内存中取指令或者取数据了 但是呢，这个程序在内存它的位置就不能改变 因为一旦改变就要重新计算这个转换过程 另外呢，我们来看 常用的呢是动态重定位。 那么动态重定位呢实际上就是在 程序加载到内存的这个时候呢，不改变任何的地址，原来你是逻辑地址，那么还是逻辑地址- ，不改变 在程序执行过程当中再进行地址的变换 那么每执行一条指令就要去做相应的地址转换工作，这就是动态重定位 而动态重定位呢，往往呢我们为了加快这个速度，通常是需要一个硬件部件的支持的 那我们来看看举一个例子 啊，动态重定位，我们来一个，举一个例子。 好，那我们看看 CPU 执行指令，这是物理内存。 那么这有一个硬件的一个小部件，这里头呢 有一些机制，那么这里头其中有一个叫做 重定位寄存器，有的时候也可以叫做基地址寄存器 当一个进程把它加载到内存 之后，那我们就是知道了这个进程在内存的起始的位置、 起始的地址。 那么假设 这个起始的地址是在这里头，把它送到了这个寄存器 然后 CPU 在执行的时候呢，它取到的地址 因为我们讲取到的地址是一个逻辑地址，比如说取到了一个逻辑地址 然后它就把这个地址送到了这个部件 由这个部件来完成这个地址转换的工作，得到了 一个真正的物理地址。 所以逻辑地址呢经过了这样一个 部件的转换就得到了物理地址，然后用这个物理地址到内存中去 存取相关的指令或者数据。 那么这个部件呢 有不同的名称，我们这里头统一给它一个名称，就叫做内存管理单元 也简称为就是 MMU ， MMU 一般的这个计算机的系统当中总有这样一个部件 完成的是把逻辑地址到物理地址转换的这么一个功能 这就是一个示例