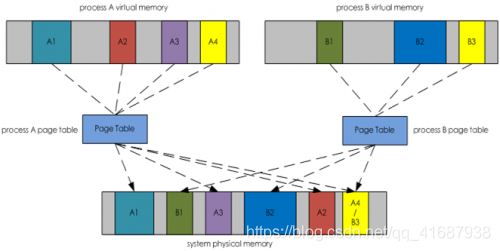
虚拟内存是操作系统内核为了对进程地址空间进行管理（ process address space management）而精心设计的一个逻辑意义上的内存空间概念。我们程序中的指针其实都是这个虚拟内存空间中的地址。比如我们在写完一段C++程序之后都需要采用g++进行编译，这时候编译器采用的地址其实就是虚拟内存空间的地址。因为这时候程序还没有运行，何谈物理内存空间地址？凡是程序运行过程中可能需要用到的指令或者数据都必须在虚拟内存空间中。

既然说虚拟内存是个逻辑意义上（假象的）的内存空间，为了能够让程序在物理机器上运行，那么必须有一套机制可以让这些假象的虚拟内存空间映射到物理内存空间（实实在在的RAM内存条上的空间）。这其实就是操作系统中页映射表（ page table）所做的事情了。内核会为系统中每个进程维护一份相互独立的页映射表。页映射表的基本原理是将程序运行过程中需要访问的段虚拟内存空间通过页映射表映射到一段物理内存空间上，这样CPU访问对应虚拟内存地址的时候就可以通过这种查找页映射表的机制访问物理内存上的某个对应的地址。“页（page）"是虚拟内存空间向物理内存空间映射的基本单元.



虚拟内存是一个假象的内存空间，在程序运行过程中虚拟内存空间中需要被访问的部分会被映射到物理内存空间中。虚拟内空间大只能表示程序运行过程中可访问的空间比较大，不代表物理内存空间占用也大。

一般我们所讲的进程占用了多少内存，其实就是说的占用了多少驻留内存而不是多少虚拟内存。

虚拟内存地址通常是连续的地址空间，由操作系统的内存管理模块控制，在触发缺页中断时利用分页技术将实际的物理内存分配给虚拟内存，而且64位机器虚拟内存的空间大小远超出实际物理内存的大小，使得进程可以使用比物理内存大小更多的内存空间。

**每个进程都有它自己的虚拟内存。包括堆区、栈区、代码区（程序的二进制代码存放）、全局存储区。所以我们平时所说的代码的运行，分配，操作等，都是指的虚拟内存。**

Linux内核在用户申请内存的时候，只是给它分配了一个线性区（也就是虚存），并没有分配实际物理内存；只有当用户使用这块内存的时候，内核才会分配具体的物理页面给用户，这时候才占用宝贵的物理内存。内核释放物理页面是通过释放线性区（也就是虚存），找到其所对应的物理页面，将其全部释放的过程。

**char** **\***p**=**malloc(2048) *//这里只是分配了虚拟内存2048，并不占用实际内存。*

strcpy(p,"123")*//分配了物理页面，虽只使用了3个字节，但内存还是为它分配了2048字节的物理内存。*

free(p) *//通过虚拟地址，找到其所对应的物理页面，释放物理页面，释放虚拟内存(线性区)。*

***上文可称为进程虚拟地址空间，下文是个虚拟内存。用作区分，下文仍然有用，时间紧迫时可不读。***

***~~物理内存是系统硬件提供的内存大小，是真正的内存。虚拟内存是为了满足物理内存不足而提出的策略，利用磁盘空间虚拟出一块逻辑内存，用作虚拟内存的磁盘空间称为交换空间。  
作为物理内存的扩展，Linux未在物理内存不足时，使用交换分区的虚拟内存（内核将暂时不用的内存块信息写到交换空间，物理内存得到释放并能用于其他地方，当需要用到原始内容时，这些信息会被重新从交换空间读入物理内存）。  
Linux的内存管理采取的是分页存取机制。为了保证物理内存能得到充分利用，内核在适当时候将物理内存中不经常使用的数据块自动交换到虚拟内存中，经常使用的信息保留到物理内存中。~~***

***~~深入了解Linux内存运行机制：  
首先，Linux系统会不时地进行页面交换操作，以保持尽可能多的空闲物理内存。即使并没什么事情需要内存，Linux也会交换出暂时不用的内存页面，可以避免等待交换所需的时间。  
Linux 进行页面交换时不是多有的页面在不用时都交换到虚拟内存中。Linux内核根据“最近经常使用”算法，仅仅将一些不经常使用的页面文件交换到虚拟内存中。有时我们会看到这么一个现象：Linux物理内存还有很多，但是交换空间也使用了很多。这是因为一个占用很大内存的进程运行时，需要耗费很多内存资源，因此就会有一些不常用的页面文件被交换到虚拟内存中。但后来这个占用很多内存资源的进程结束并释放了很多内存，刚才被交换出去的页面文件并不会自动交换进物理内存（除非有这个必要），那么此刻系统物理内存就会空闲很多，同时交换空间也在被使用，就出现了刚才的现象。  
最后，交换空间的页面在使用时会首先被交换到物理内存中。如果此时没有足够的物理内存来容纳这些页面，它们又会被马上交换出去，如此一来，虚拟内存中可能没有足够空间来存储这些交换页面，最终导致Linux出现假死机、服务异常等问题。~~***