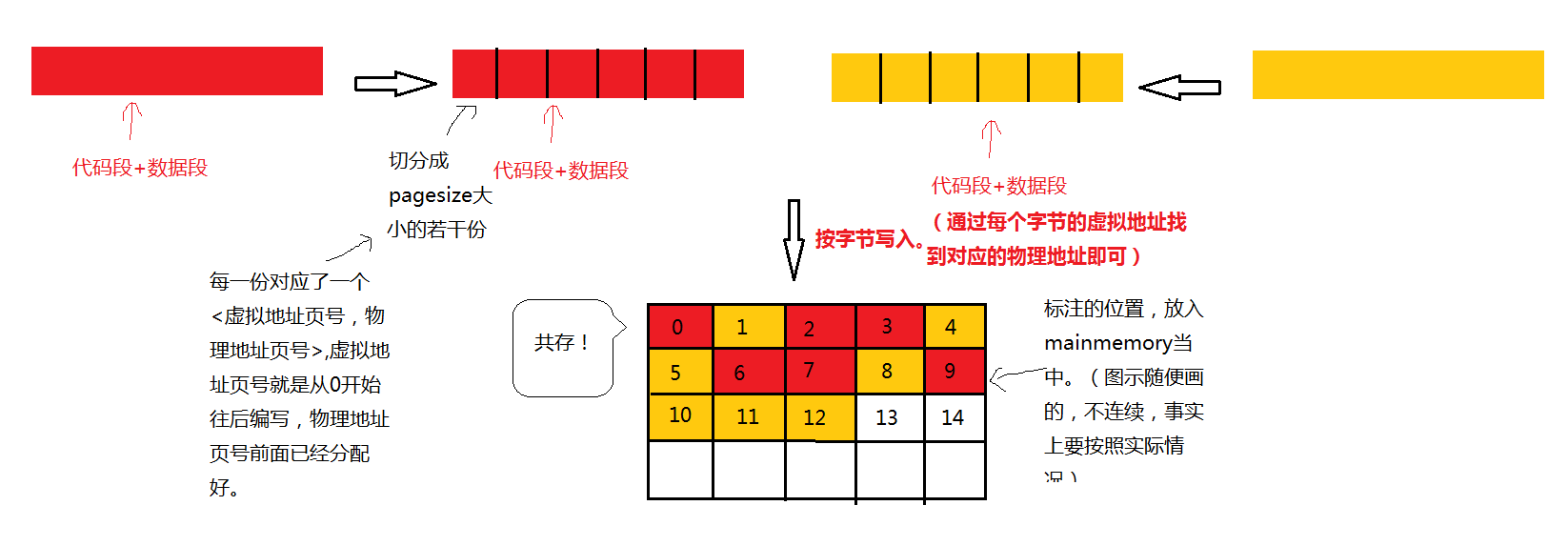
# 虚拟内存实践

## 多线程支持与全局物理内存管理

目前，因为在线程运行之前，系统就将所有的物理页加载，而且它们并不理会系统中别的线程，只是暴力地将从0开始的覆盖，直到物理页被加载完为止。总的来说，pro2的系统只能运行一个程序，并且在运行之前一次性地将程序所有的内容加载到内存中。

因此，我们希望实现的目标是，将主存分页（物理页），也将用户程序分页（虚拟页）这样不同的用户程序的某些虚拟页就可以映射到某些物理页上，也就可以“共存”了，原理如下图：



实现完成后的效果是，在程序运行之前，系统确定线程的虚拟页数和各页对应的执行文件内存，但是真正将页面加载到物理内存中，是由缺页中断实现的。

多线程支持的实现仅仅有物理内存与虚拟内存的分页是不足够的，系统中必须要有对物理页的管理，对物理页作出正确的分配、回收及信息的必要记录才行。下面来介绍具体的数据结构：

1. 物理页的空闲队列和已用队列——太好理解了。
2. Page数据结构，可以将这个数据结构称为页面信息，它保存了用户进程、虚拟页、物理页的对应关系，是最为关键的。

## 内存分配与回收

Lazy-loading的内存分配算法，使得当且仅当程序运行过程中缺页中断发生时，才会将所需的页面从磁盘调入内存。而在物理页（内存）的选择上，是在空闲队列中选择，但在内存耗尽的时候，则会选择一个未被使用过的物理页换出（注：这种置换方法是极其简陋的）。关于内存的回收，回收的操作是简单的，关键在于何时回收？系统选择在exit系统调用中回收。

## 跳表的使用

首先，这个跳表的使用时单线程的，因为在另一个线程将初始化和运行时，都会将跳表给清空。而且线程切换的时候也会将跳表给清空。

对于跳表的原理，学过操作系统的都是很清楚的，不多做解析。