操作系统

Operating system

胡燕

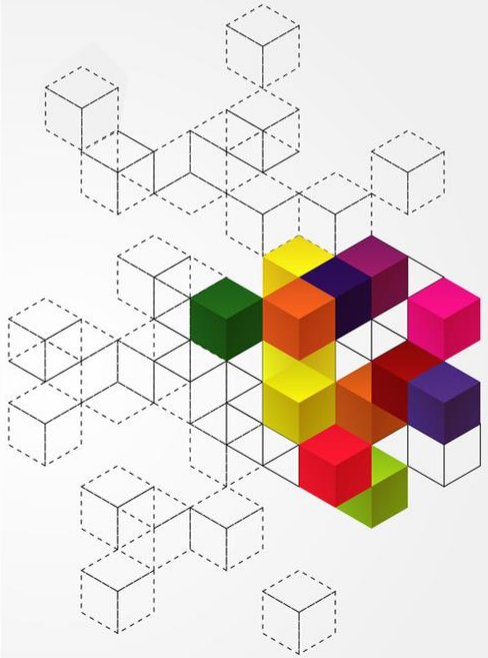
大连理工大学

主存管理（上）

8.1 主存基本概念

8.2 连续内存分配

8.3 分页机制



内容纲要

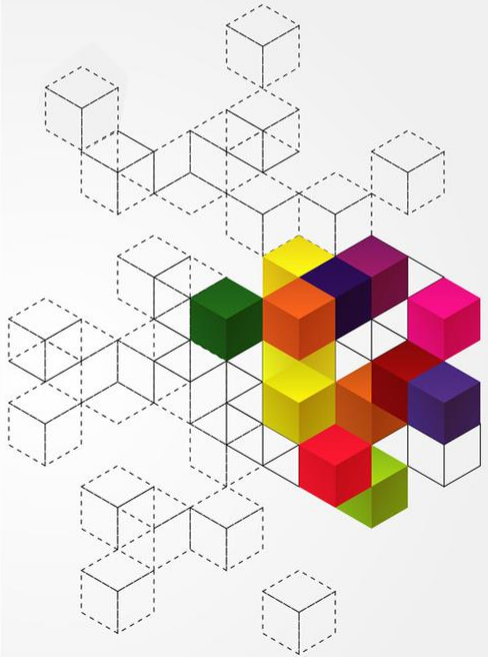
8.1 主存管理基本概念

一、内存管理的重要性

二、内存隔离保护

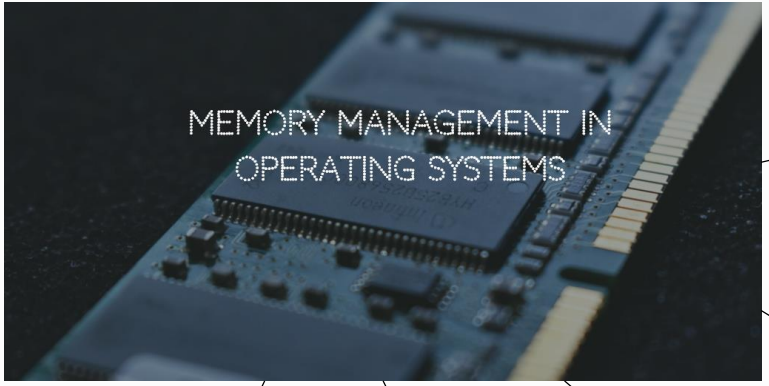
三、地址绑定

四、交换的概念

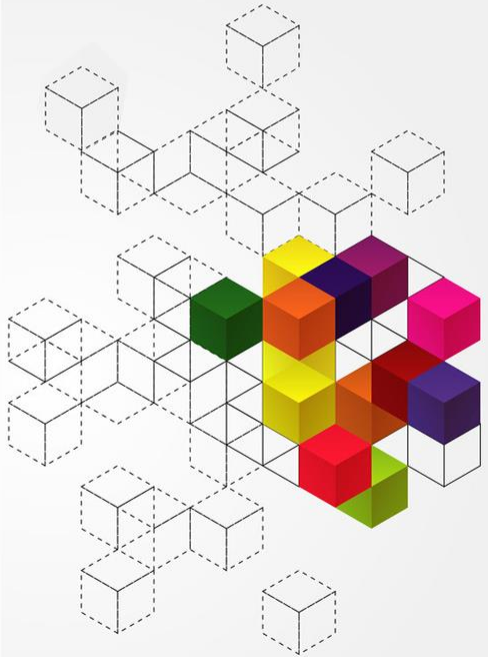


一、内存管理的重要性

内存是计算机系统中的核心资源，需要精心管理

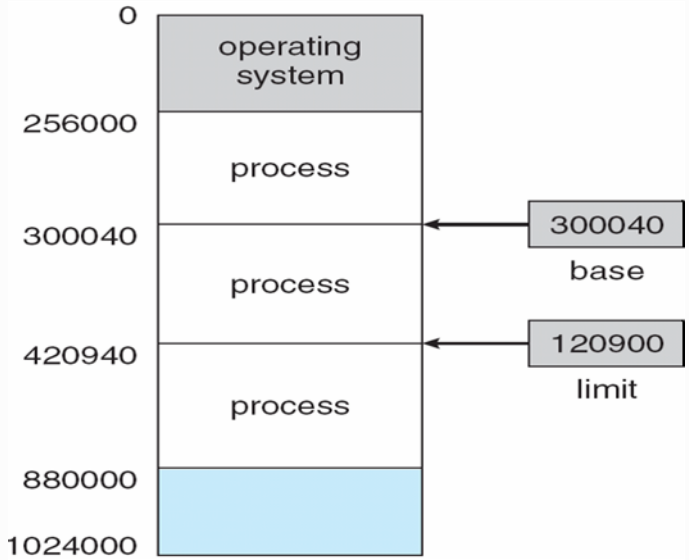


- 内存分配
- 内存保护
- 地址绑定
- 分页/分段
- 内存扩充



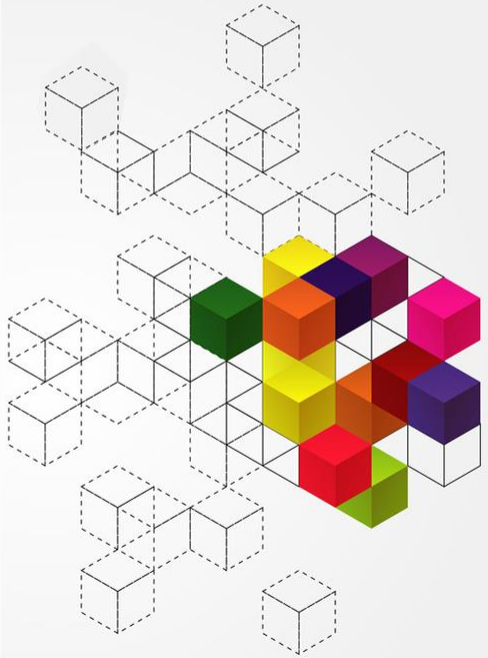
二、内存隔离保护

MM功能1：内存保护



为每个进程分配独立的一段内存空间

基本手段1：
为每个进程分配连续的内存，用两个寄存器分别存放地址上界和地址下界

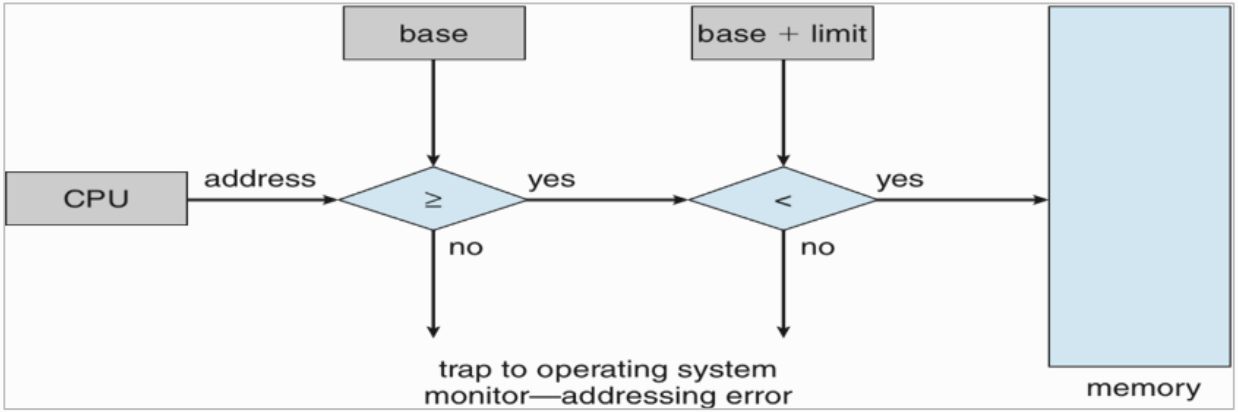


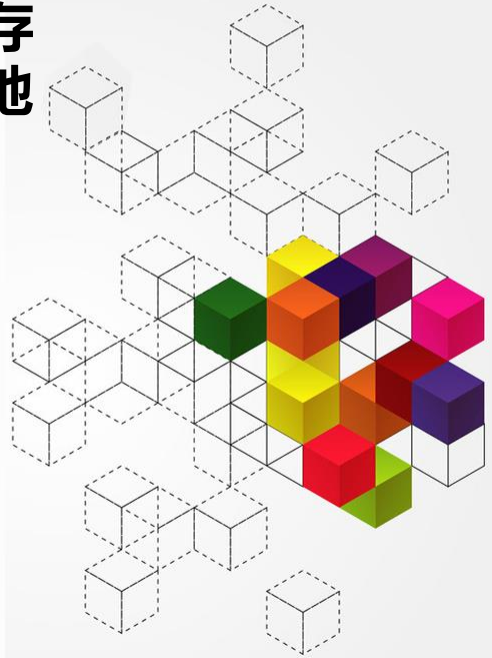
二、内存隔离保护

MM功能1：内存保护（硬件支持）

基本手段1：
为每个进程分配连续的内存，用两个寄存器分别存放地址上界和地址下界

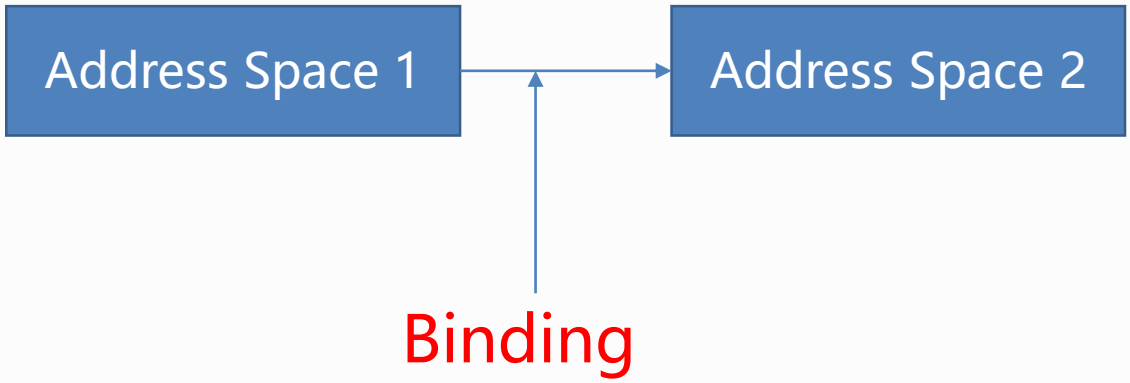
内存保护硬件逻辑





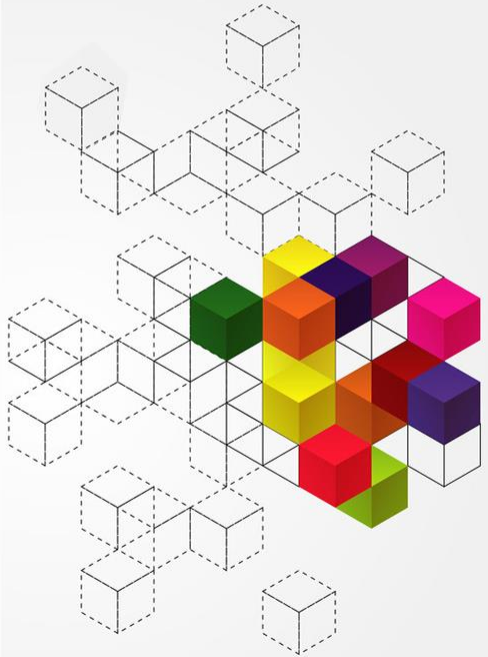
三、地址绑定

MM功能2：地址绑定



In computing, an **address space** defines a range of discrete addresses, each of which may correspond to a network host, peripheral device, disk sector, a memory cell or other logical or physical entity.

- wikipedia



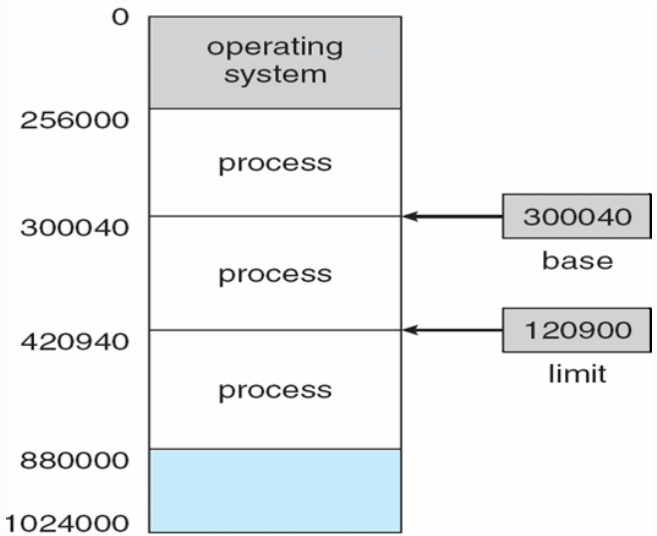
三、地址绑定

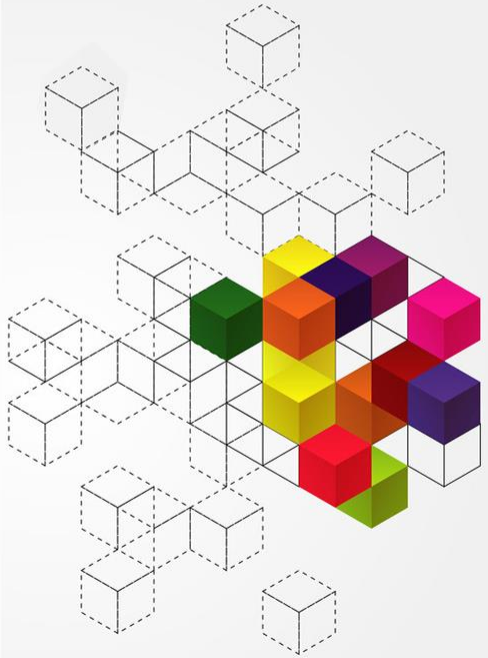
MM功能2：地址绑定

- 逻辑地址空间（Logical Address Space）
- 物理地址空间（Physical Address Space）

0-120900

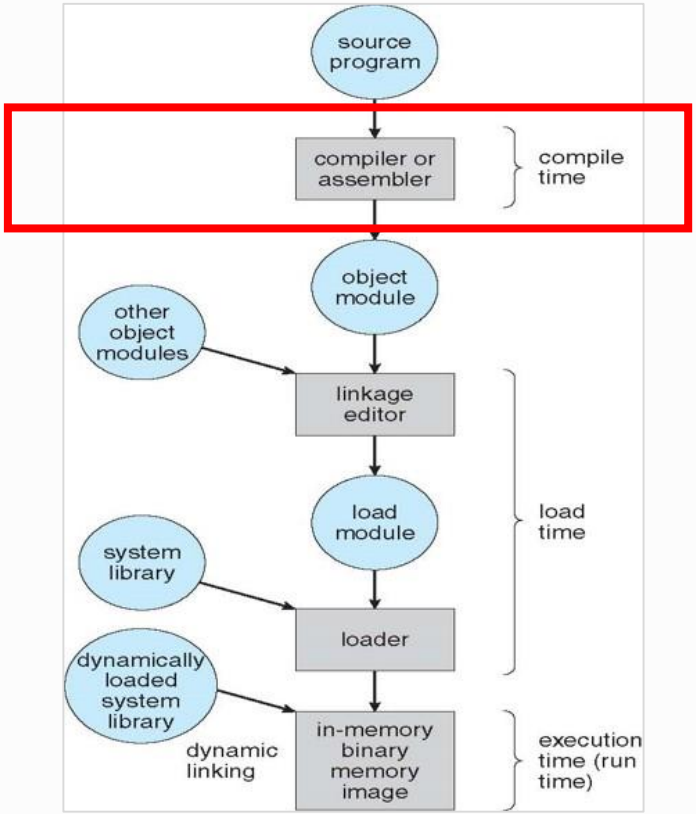
300040-420940



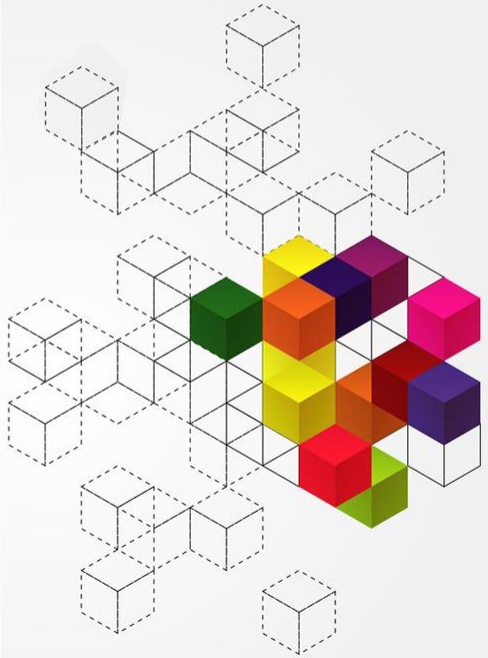


三、地址绑定

MM功能2：地址绑定

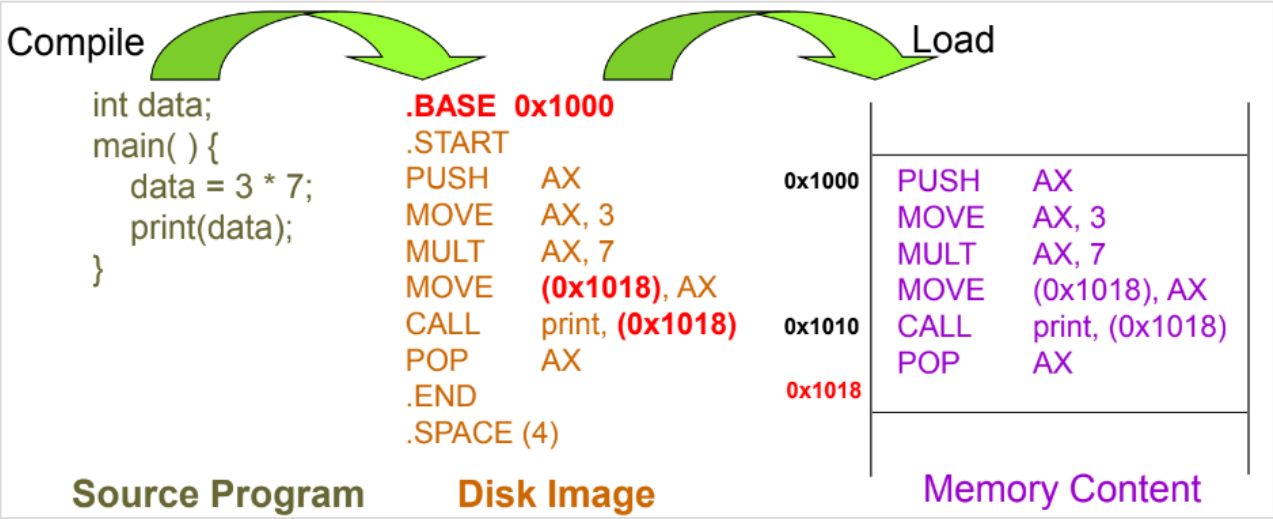


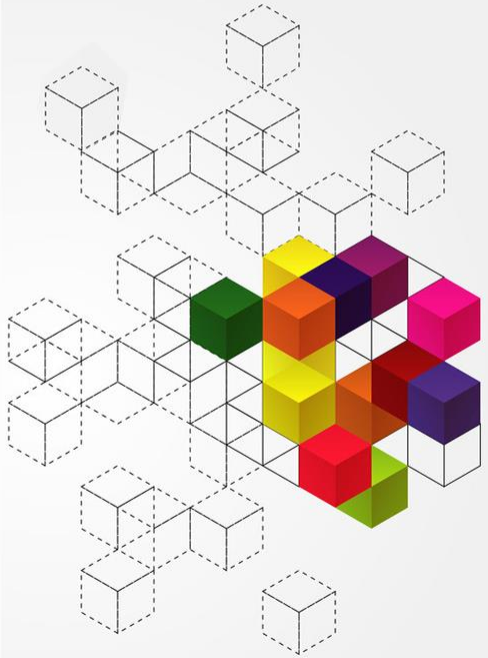
- 编译器直接将源代码中的符号地址转换为物理地址
- 如果程序装载位置发生变化，则需要重新编译



三、地址绑定

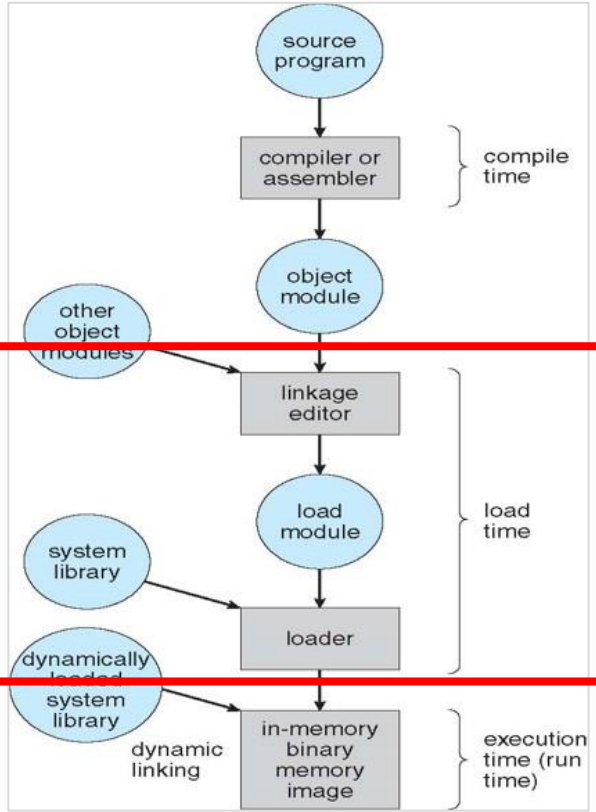
MM功能2：地址绑定（编译时绑定示例）



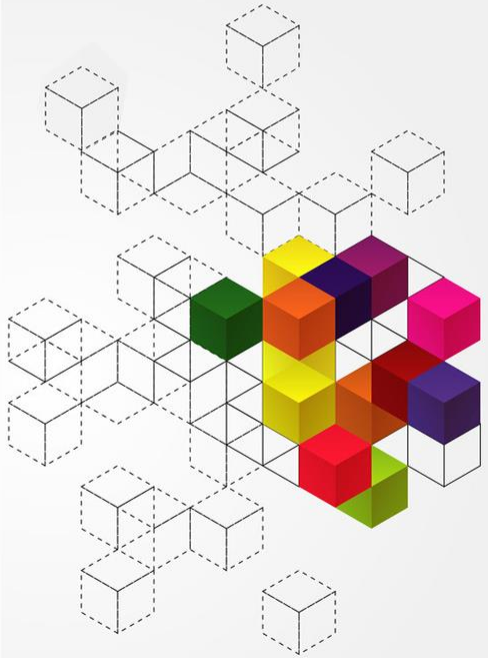


三、地址绑定

MM功能2：地址绑定

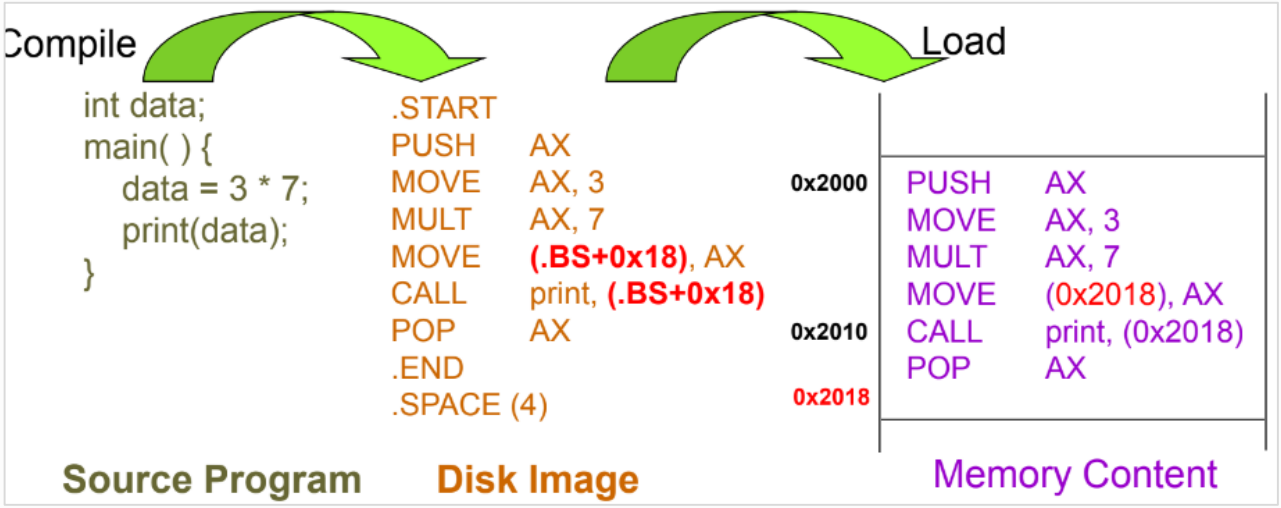


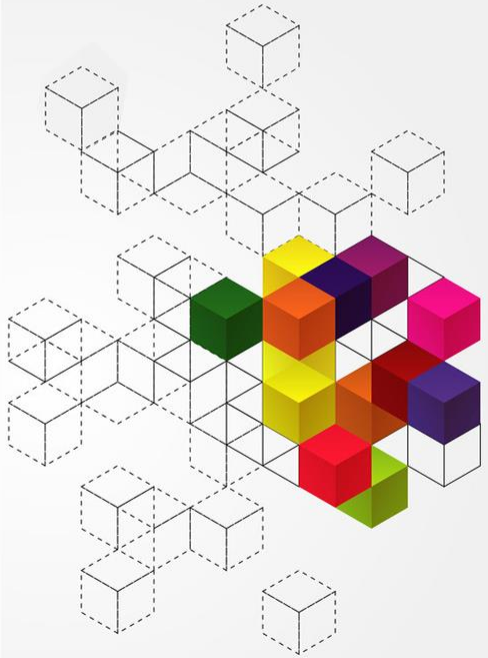
- 编译器将源代码中的符号地址翻译为可重定位地址
- 可执行程序中的可重定位地址在加载时被转换为物理内存地址



三、地址绑定

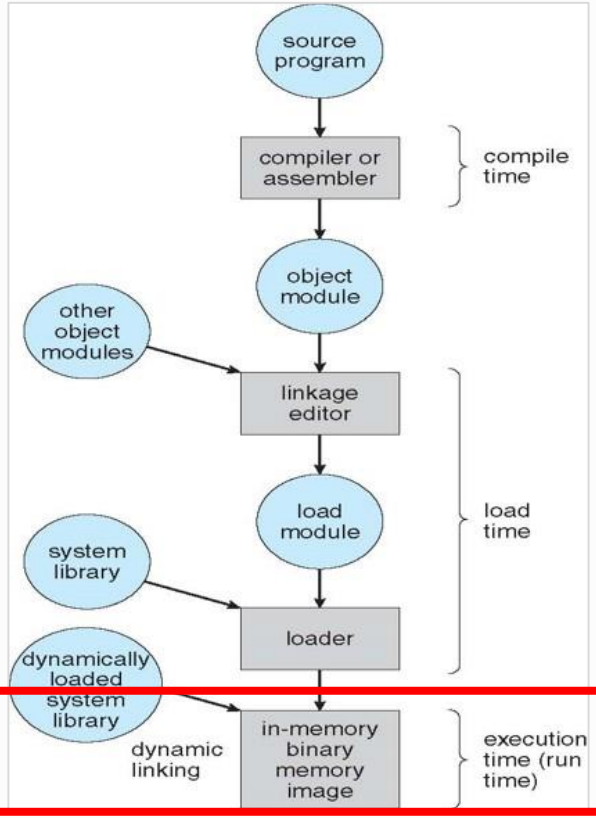
MM功能2：地址绑定（加载时绑定示例）



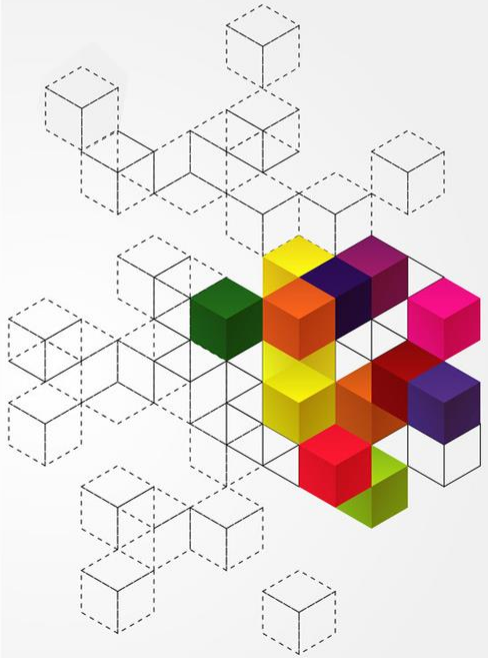


三、地址绑定

MM功能v2：地址绑定

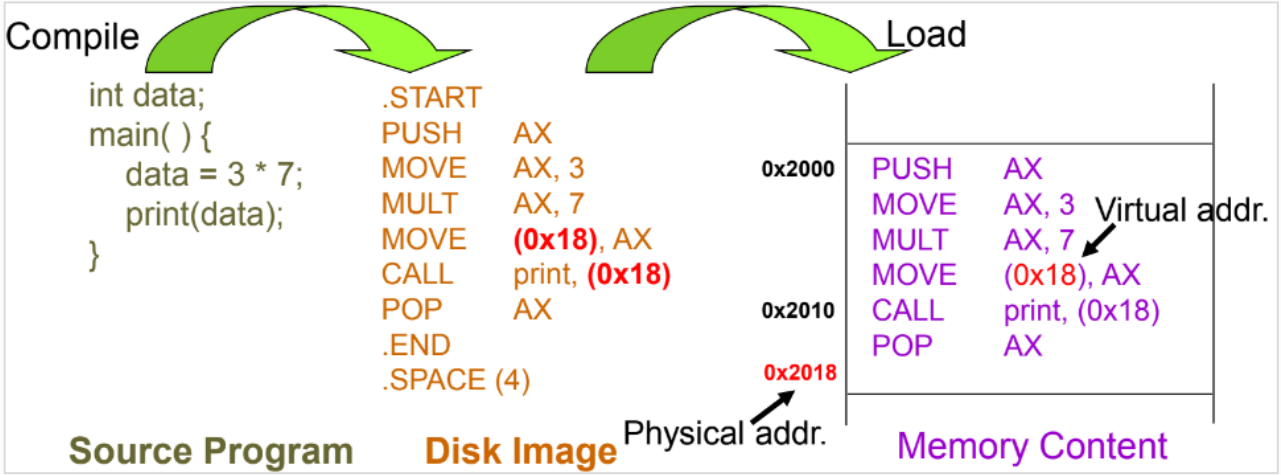


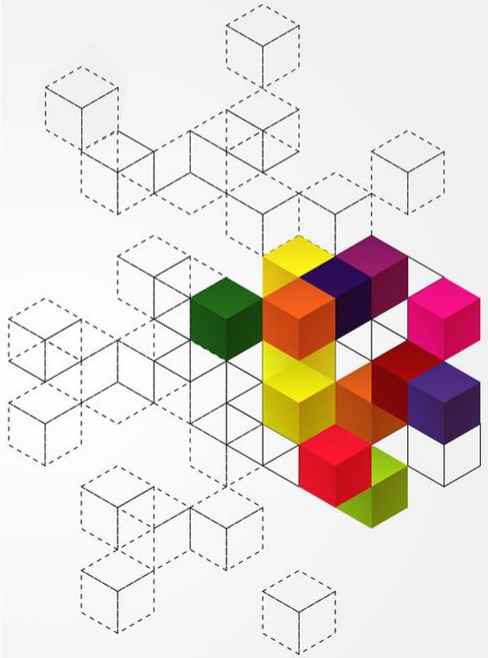
- 编译器将源代码中的符号地址翻译为逻辑地址
- 可执行程序中的可重定位地址在运行时被转换为物理内存地址



三、地址绑定

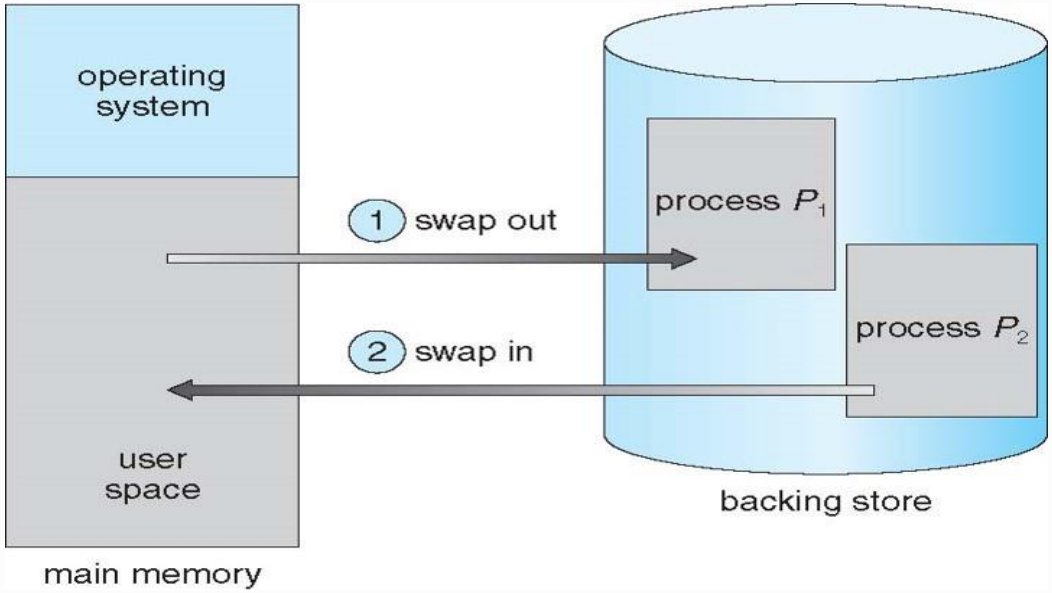
MM功能2：地址绑定（运行时绑定示例）

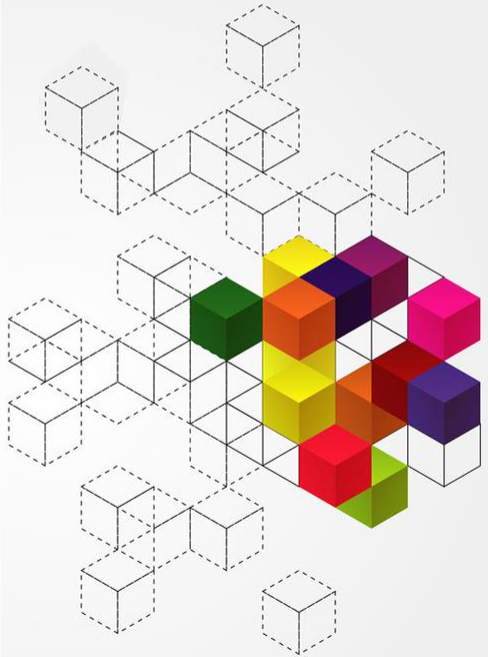




四、交换概念

MM功能3：交换

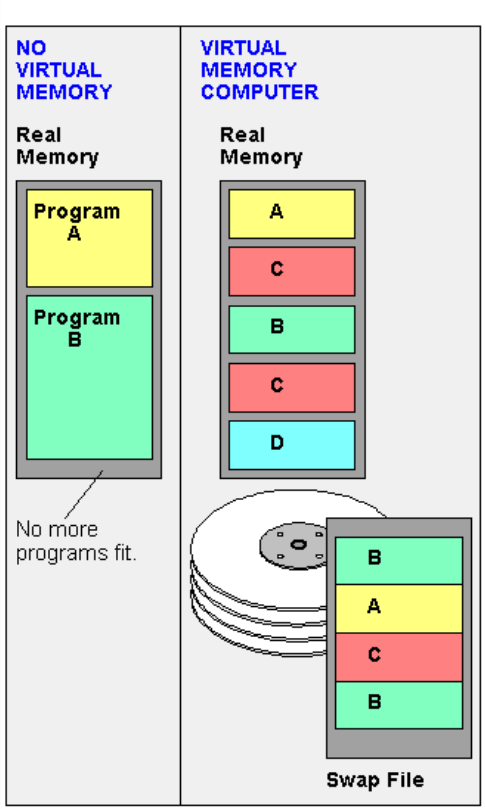




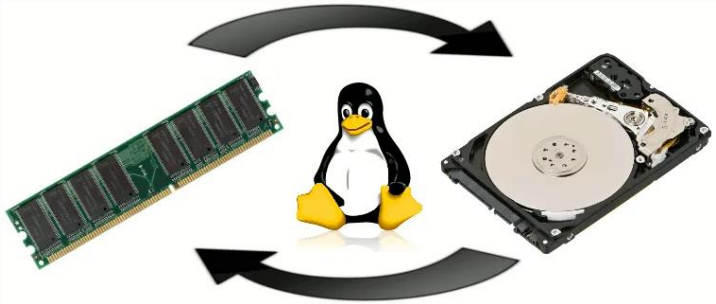
当内存资源紧张时，可以将进程内存映像转移到外存交换空间（扩充形式之一）

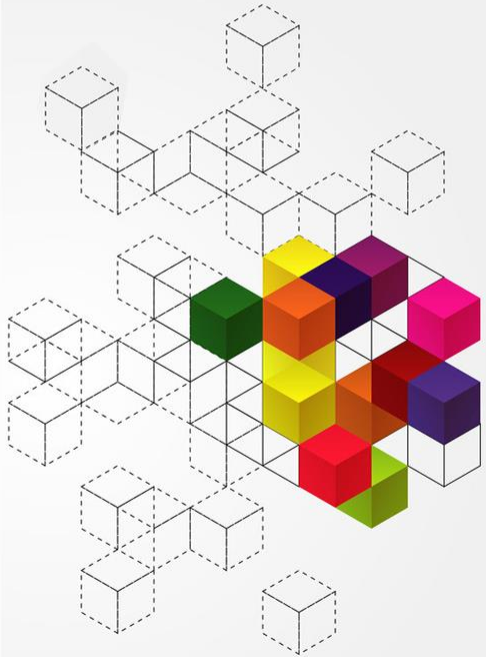
四、交换概念

MM功能3：交换



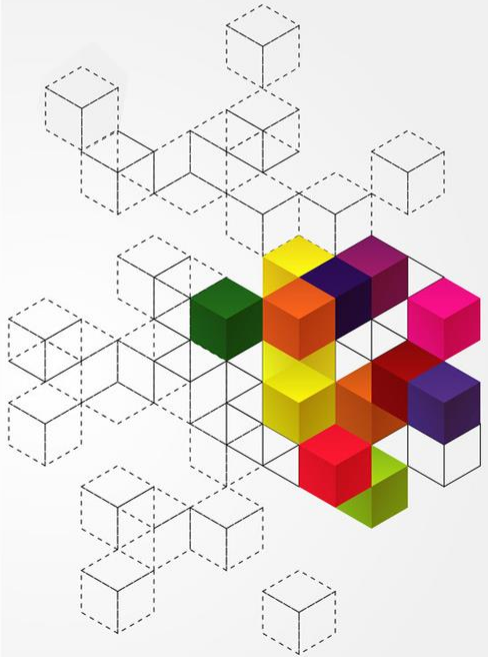
Memory is extended to storage





8.1 小结

- 内存管理重要性
- 内存隔离保护
- 地址绑定
- 交换概念



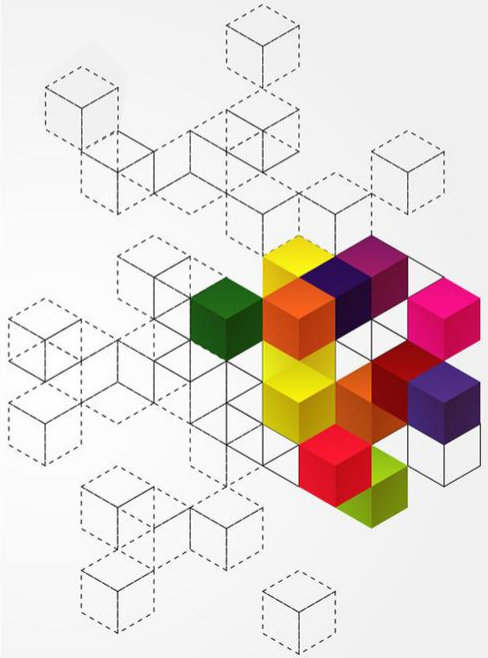
内容纲要

8.2 连续内存分配

一、连续内存分配方法概述

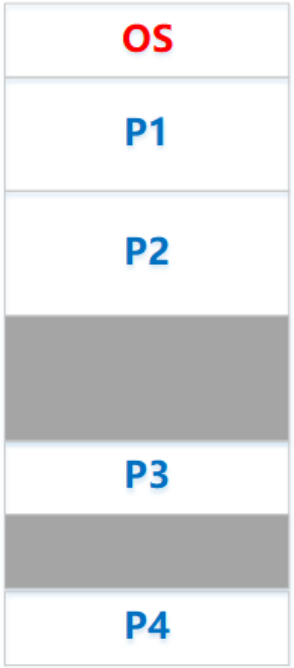
二、固定分区分配方法

三、可变分区分配方法

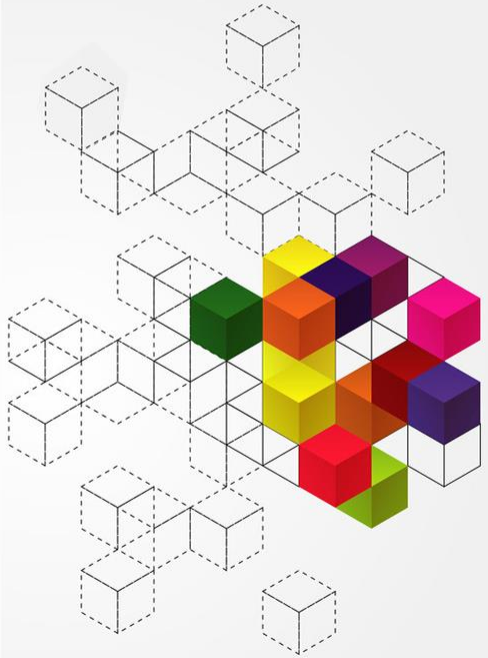


一、连续内存分配方法概述

- **连续内存分配：为每个进程分配连续的内存块**

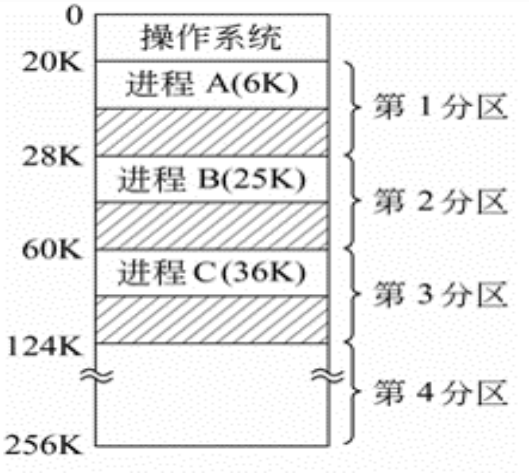


- 基于连续内存分配的操作系统常规做法
  - 将整个物理内存区分为两大部分：系统区与用户区
  - 系统内存区：为操作系统内核预留，用于保证操作系统正常运转
  - 用户内存区：为用户进程在用户态运行分配所需内存

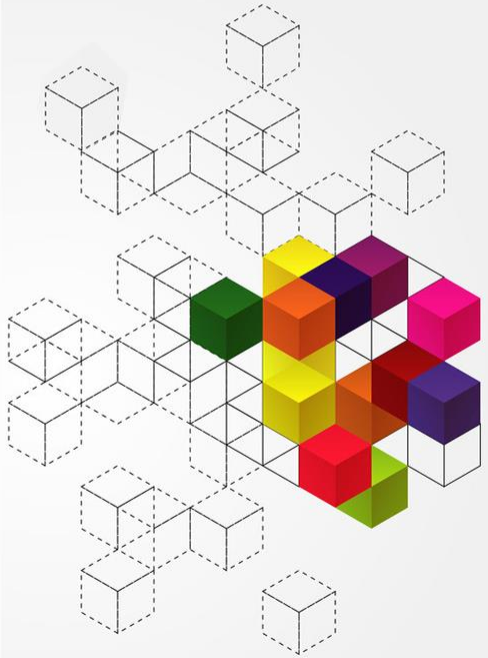


二、固定分区分配

- **Fixed-Partition Allocation**

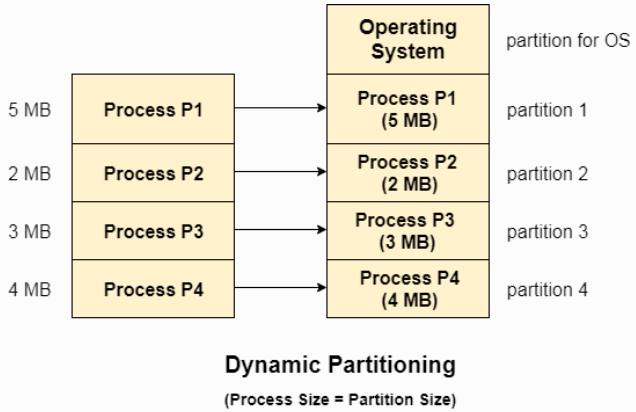


- 示例：
  - 系统区(0-20K)为操作系统所用
  - 其他四个分区供用户进程使用

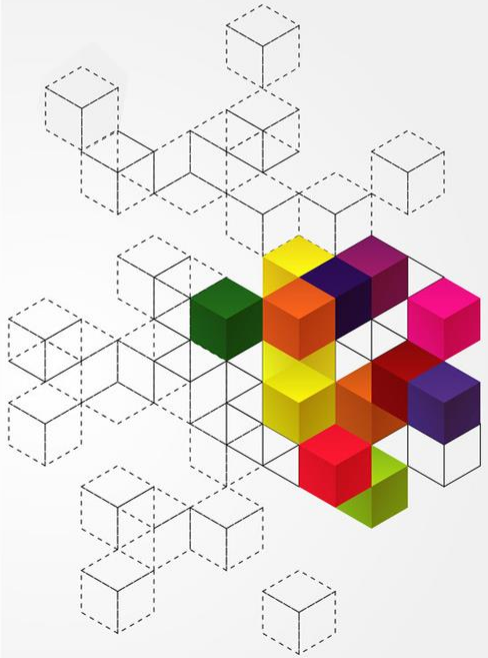


三、可变分区分配

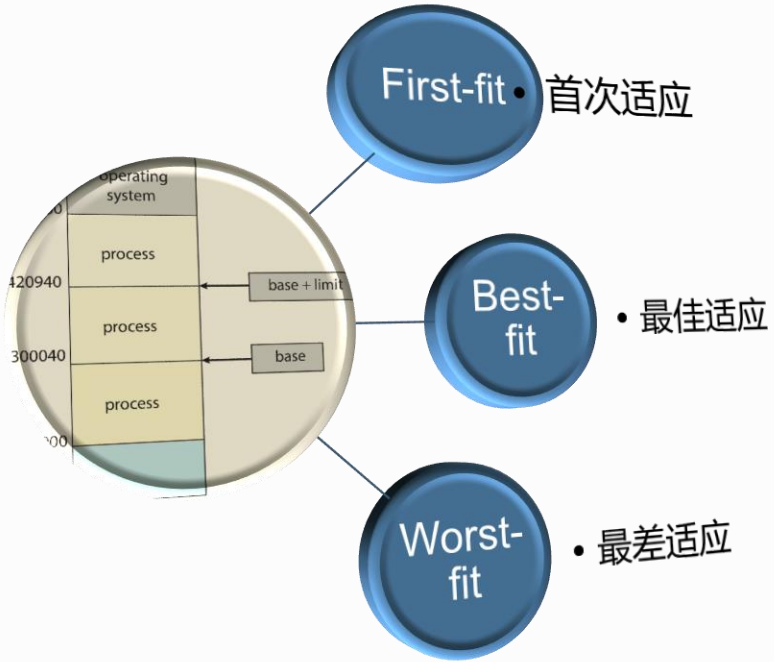
- **Variable-Partitioning**
  - 分区大小可变

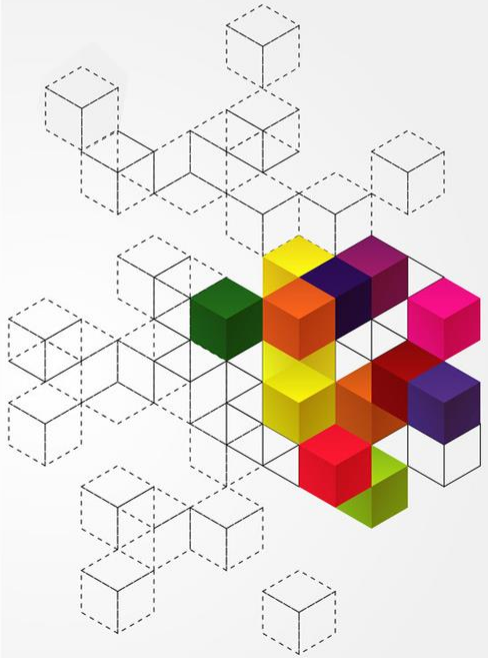


- 为每个进程分配的内存块大小=进程大小



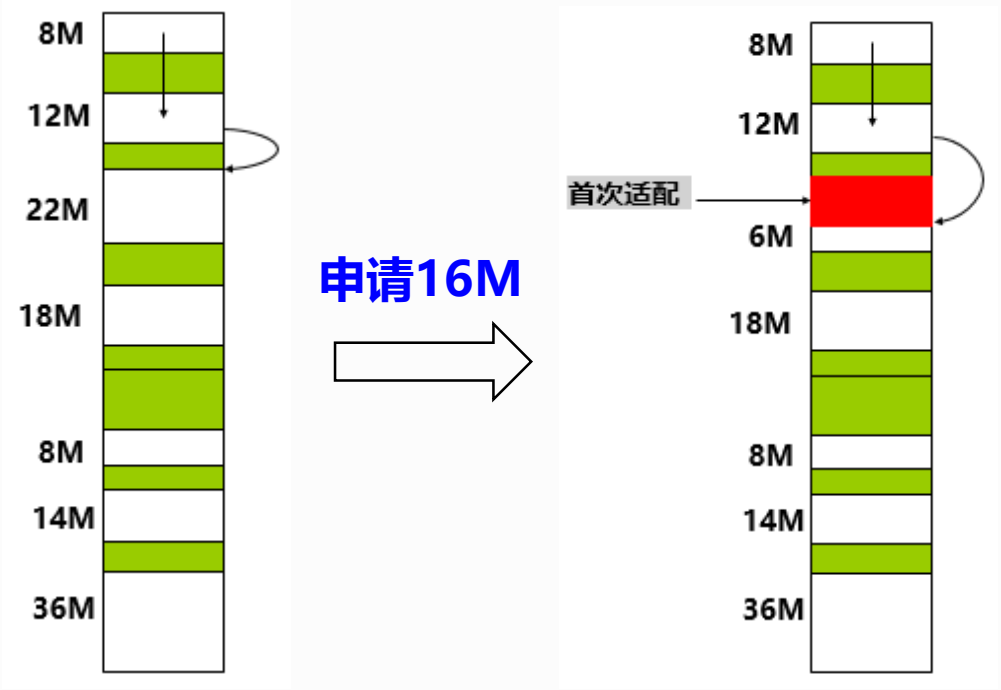
三、可变分区分配

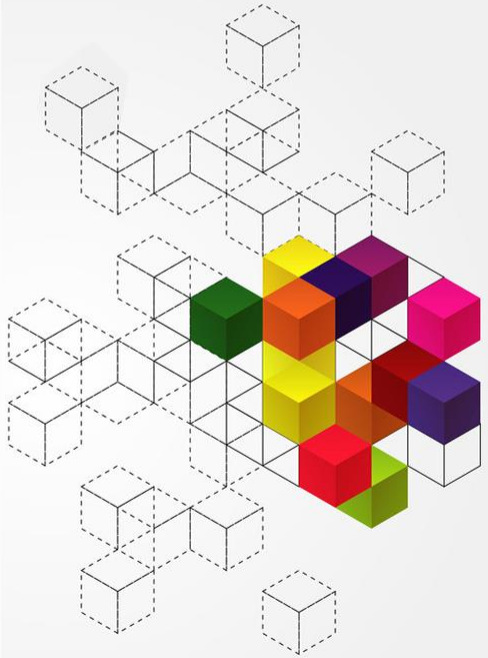

First-fit
首次适应
operating system
process
420940
base + limit
process
300040
base
process
Best-fit
最佳适应
Worst-fit
最差适应



三、可变分区分配

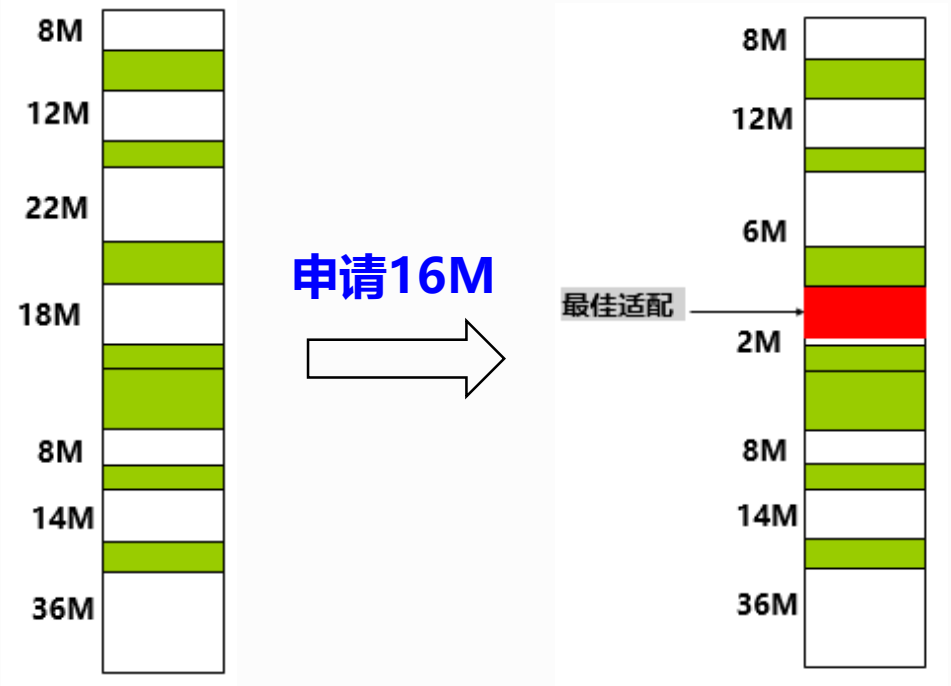
- **First-fit**
  - 示例：空白部分表示空闲空间

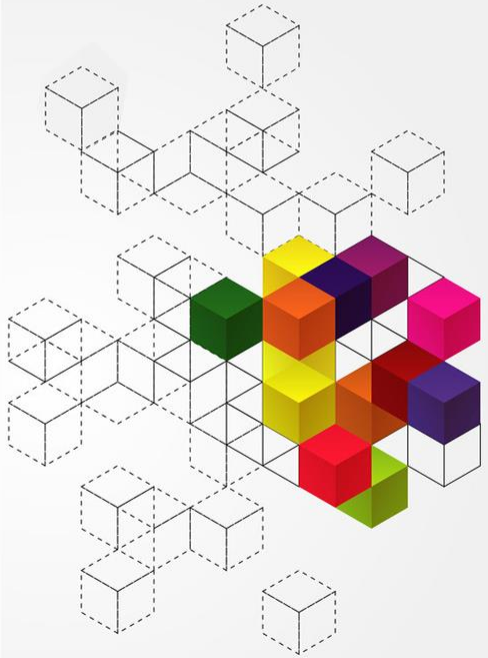




三、可变分区分配

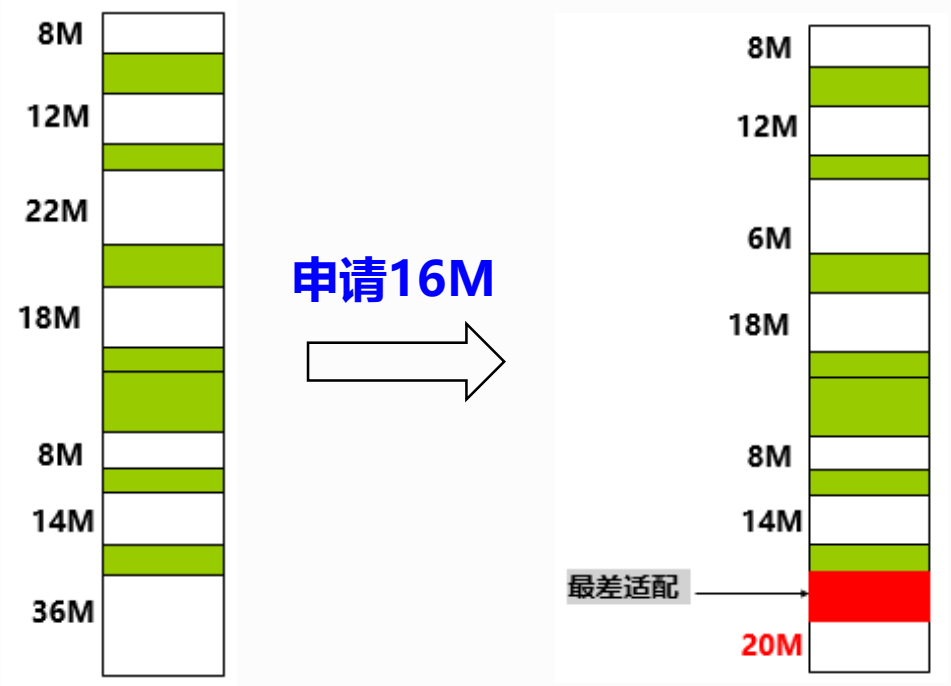
- **Best-fit**
  - 示例：空白部分表示空闲空间

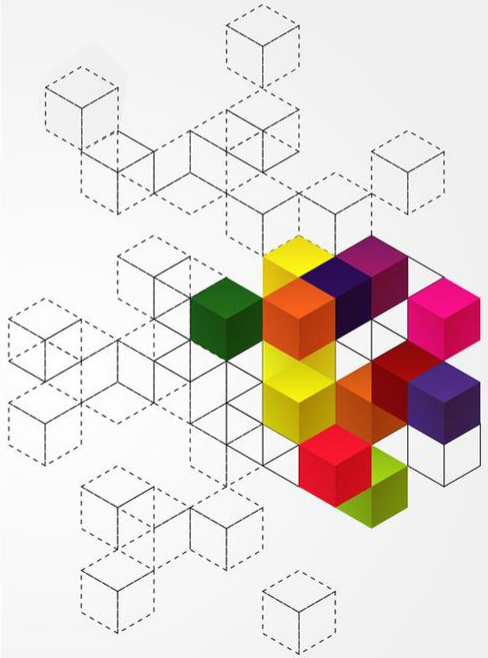




三、可变分区分配

- **Worst-fit**
  - 示例：空白部分表示空闲空间

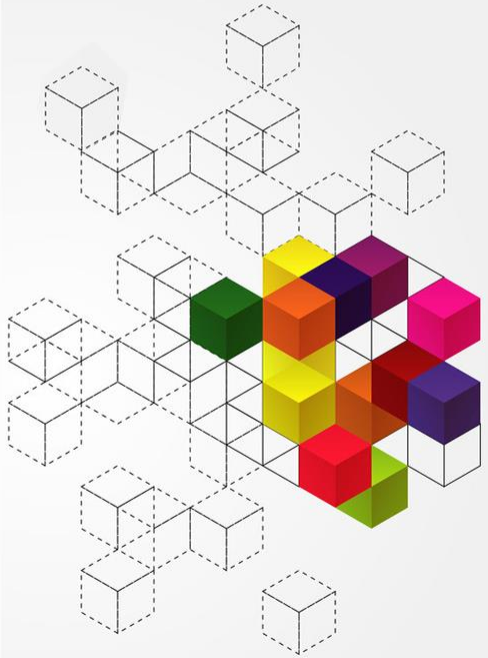




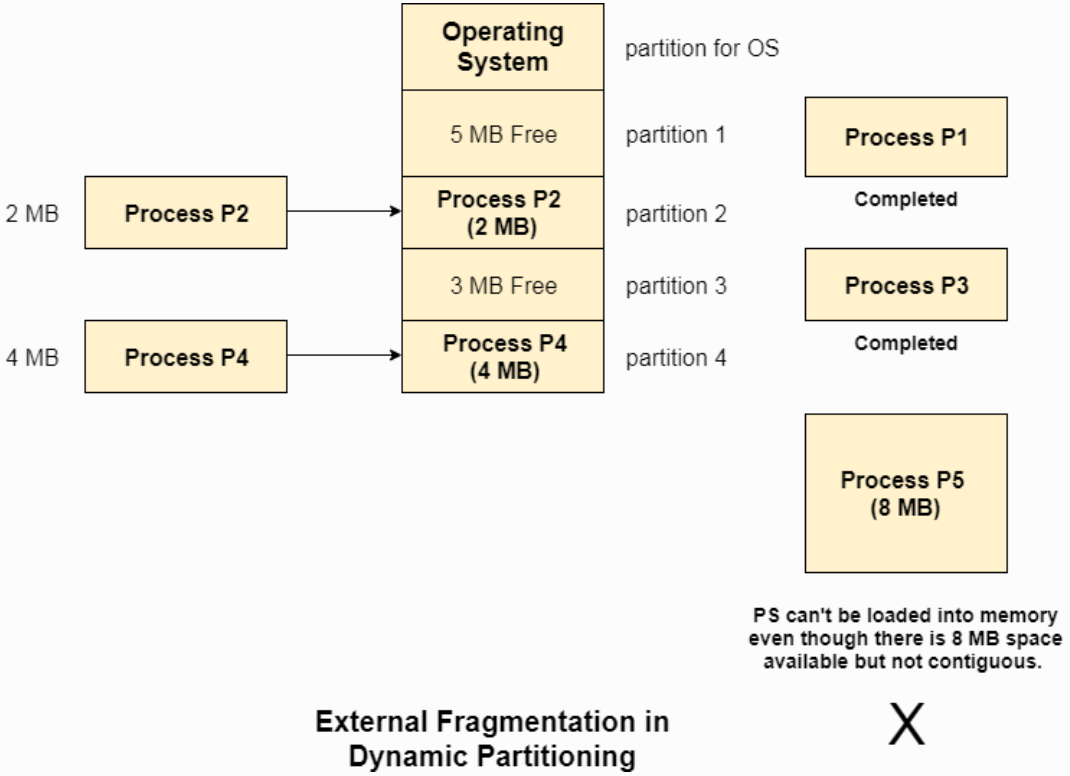
三、可变分区分配

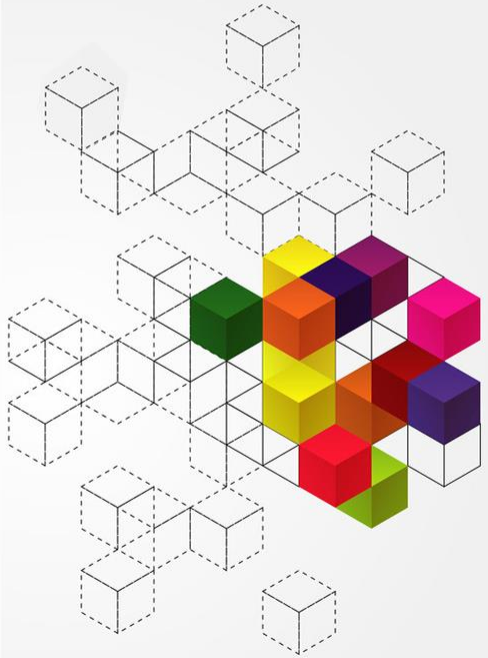
- 慕课堂-讨论1

比较固定分区内存分配和可变分区内存分配算法



三、可变分区分配

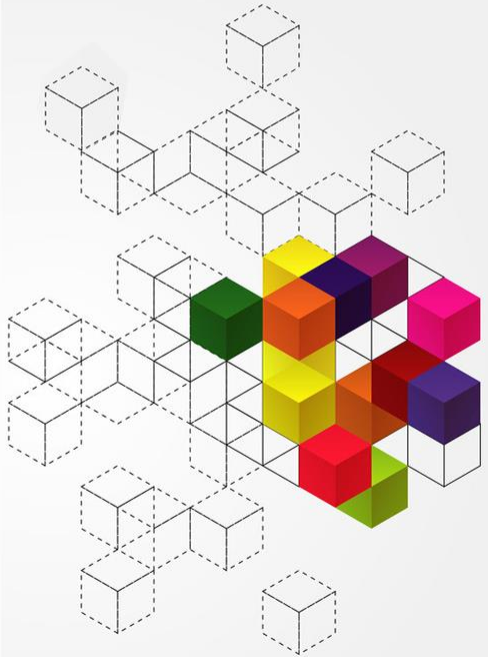

Operating System
partition for OS
5 MB Free
partition 1
Process P1
Completed
2 MB
Process P2
Process P2 (2 MB)
partition 2
3 MB Free
partition 3
Process P3
Completed
4 MB
Process P4
Process P4 (4 MB)
partition 4
Process P5 (8 MB)
PS can't be loaded into memory even though there is 8 MB space available but not contiguous.
External Fragmentation in Dynamic Partitioning
X



三、可变分区分配

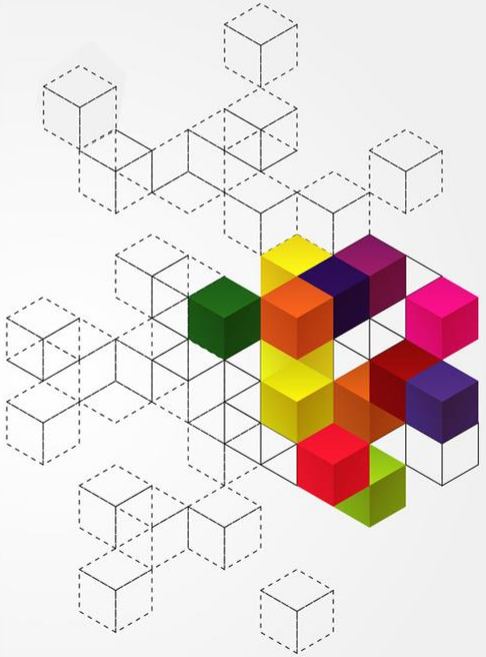
- 慕课堂-讨论2

如何理解分区分配中的碎片问题



8.2 小结

- 连续内存分配方法概述
- 固定分区分配
- 可变分区分配



内容纲要

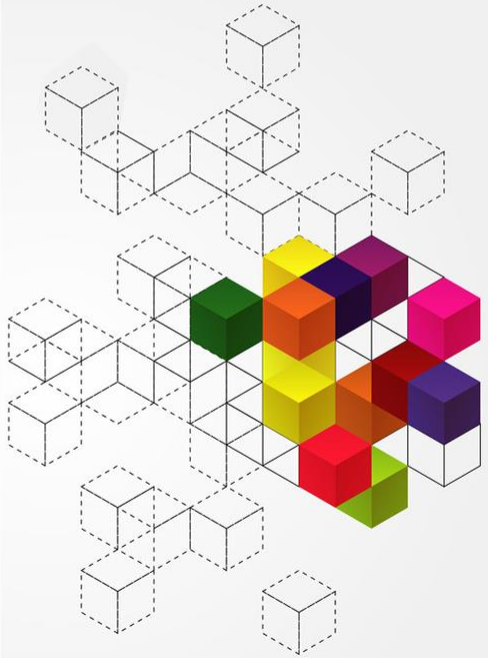
8.3 分页机制

一、引入分页的意义

二、分页原理

三、分页机制下的内存保护

四、分页机制下的内存共享



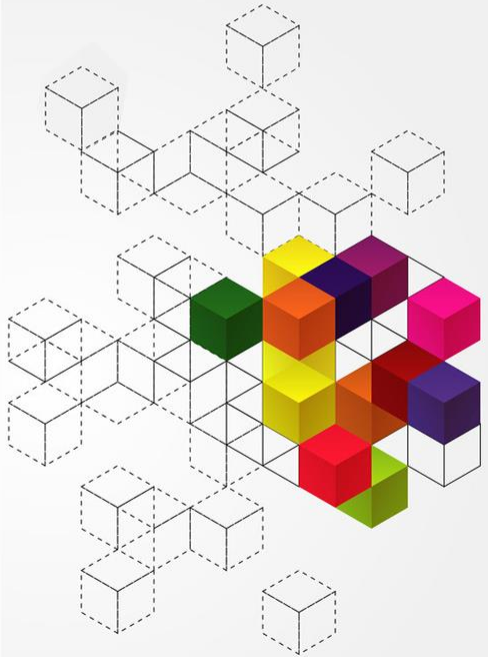
一、引入分页的意义

- **为什么要引入分页?**
  - 这是在计算机硬件性能提升，内存容量增大，多任务概念得以实现的情况下，分析连续内存分配的弊端，而进行的技术创新

连续分区分配的问题

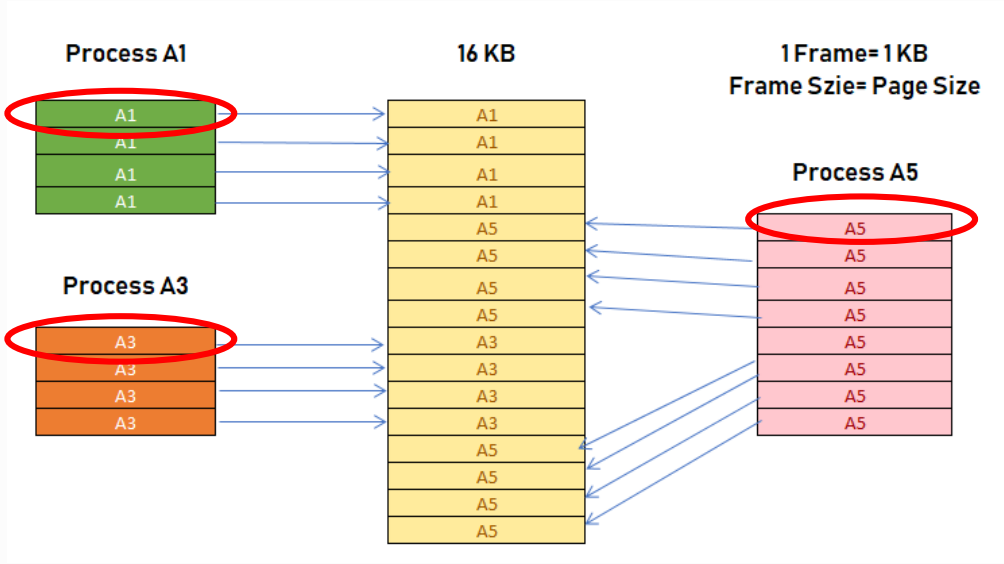
- **容易形成数量较多的较大内存碎片，导致内存使用效率降低**
- **碎片大小不可预估，不好控制**

分页机制可以有效解决连续内存分配的问题，提升内存使用效率

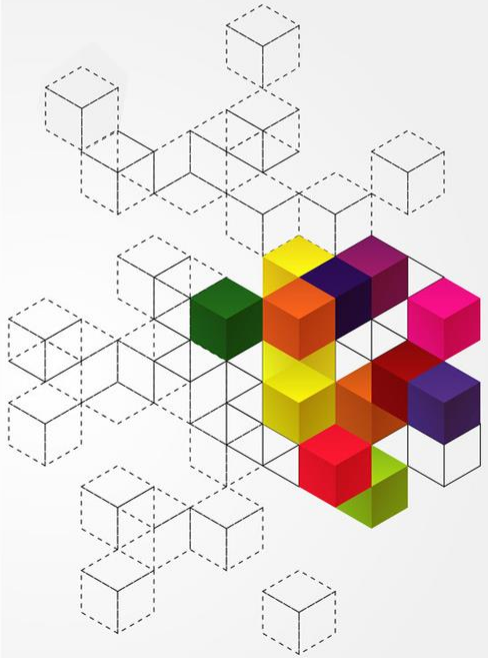


二、分页原理

• 分页基本思想

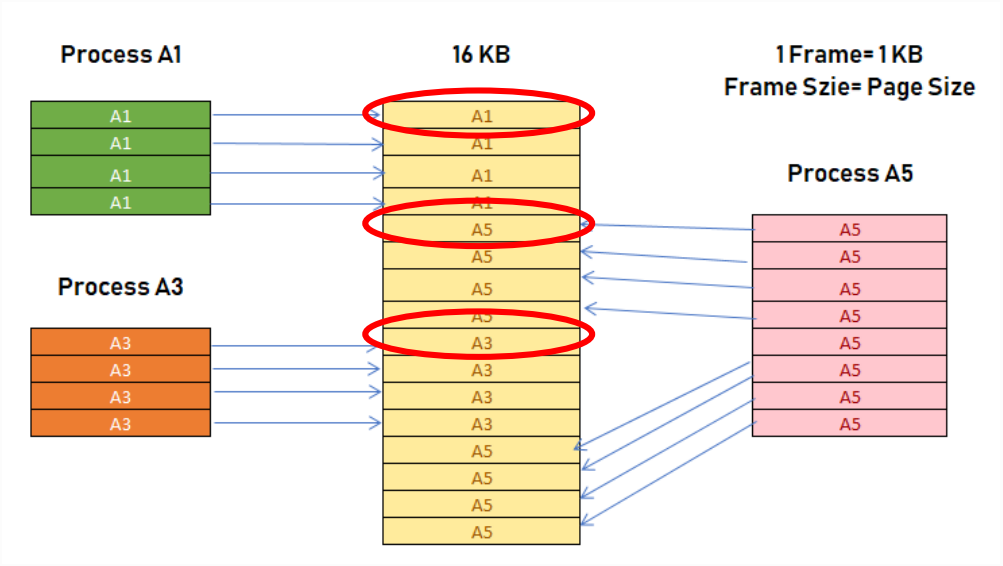


进程的逻辑地址空间等分为同样的大小的块
这些块称为逻辑页（Page）

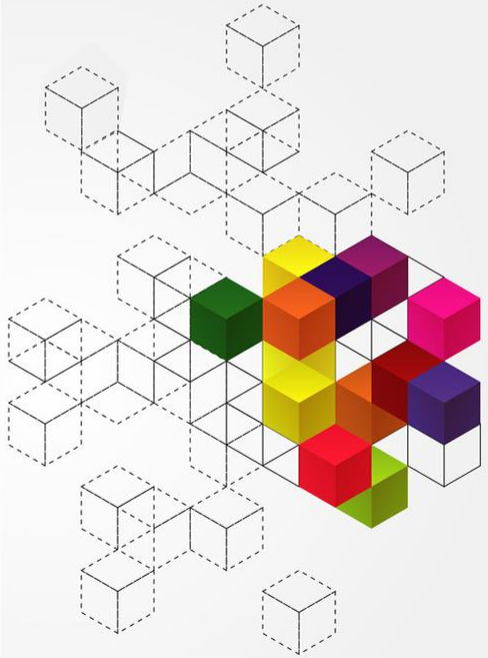


二、分页原理

• 分页基本思想

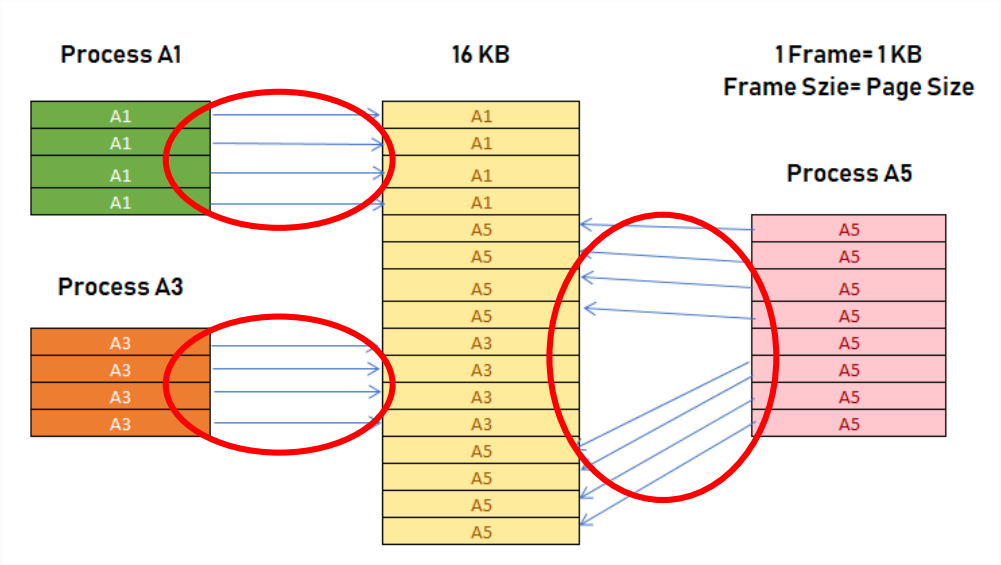


将物理内存等分为同样大小的块
每个这样的块称为物理页（frame）

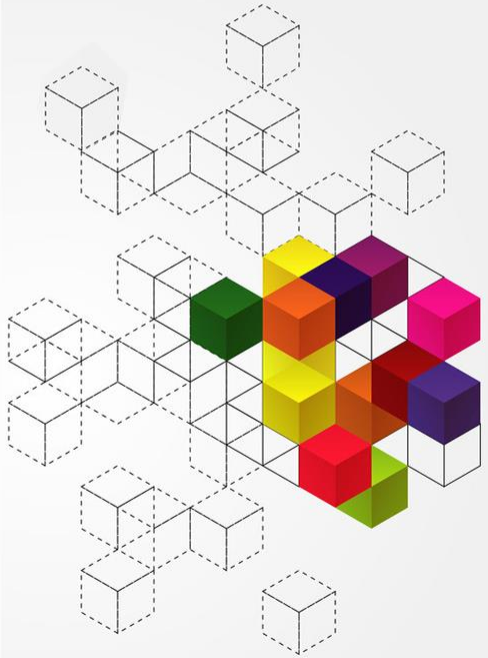


二、分页原理

• 分页基本思想



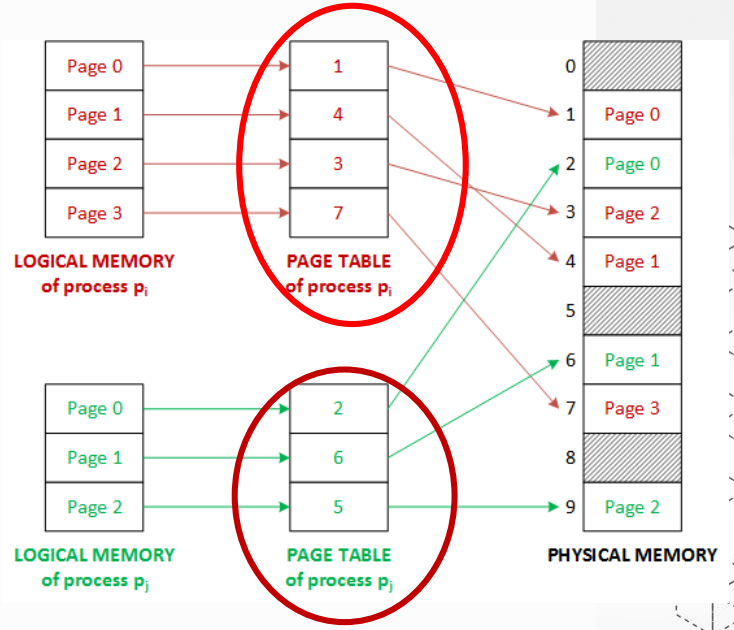
逻辑页与物理页形成1：1映射关系



二、分页原理

- 分页机制下，内存管理主要任务

1. 跟踪进程的页面使用情况

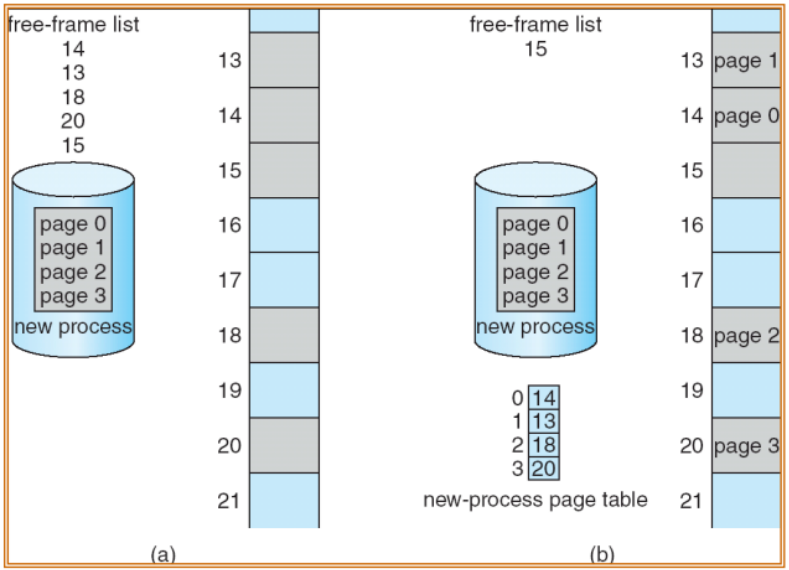


为每个进程维护一个页面与物理页框之间的映射表格（页表 Page Table）

二、分页原理

- **分页机制下，内存管理主要任务**

2. 跟踪进程的页面使用情况



在进程需要新的物理页时，从现有空闲页中进行分配；
在进程退出或进程释放内存时，回收物理页资源

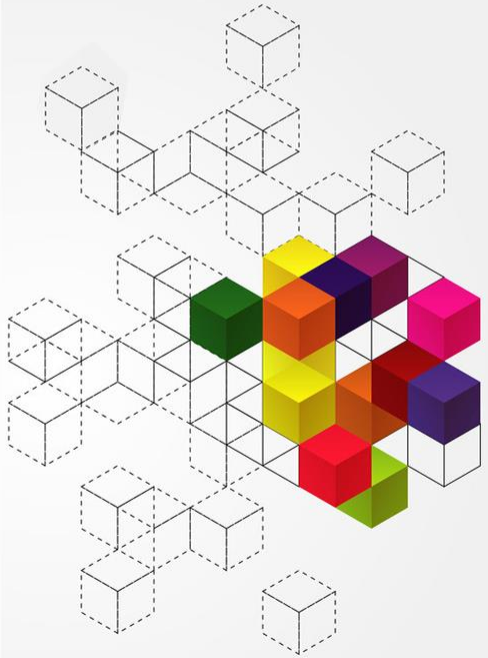
二、分页原理

• 分页的三个重要环节

地址划分 → 建立地址映射（页表） → 地址翻译

page number	page offset
p	*d*
m - n	*n*

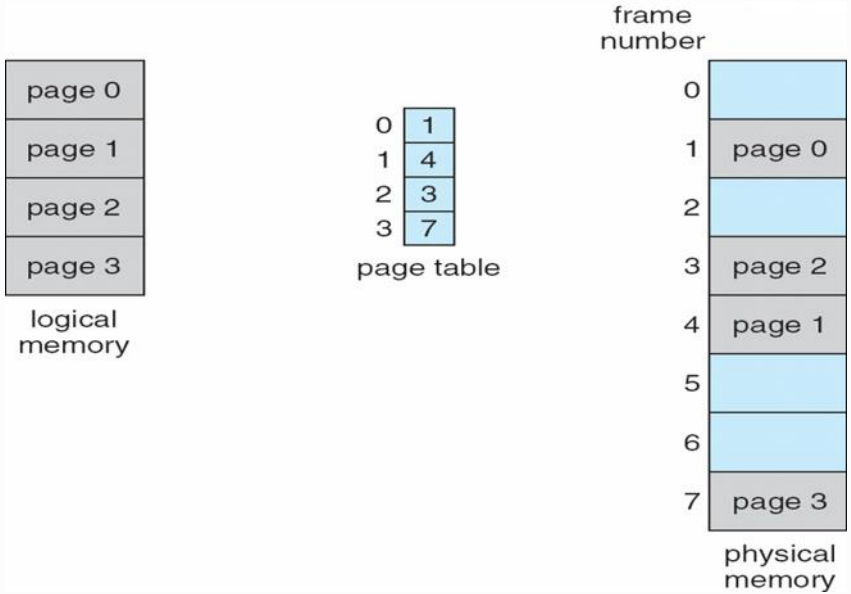
地址被按位拆分成页号和页内偏移这2部分

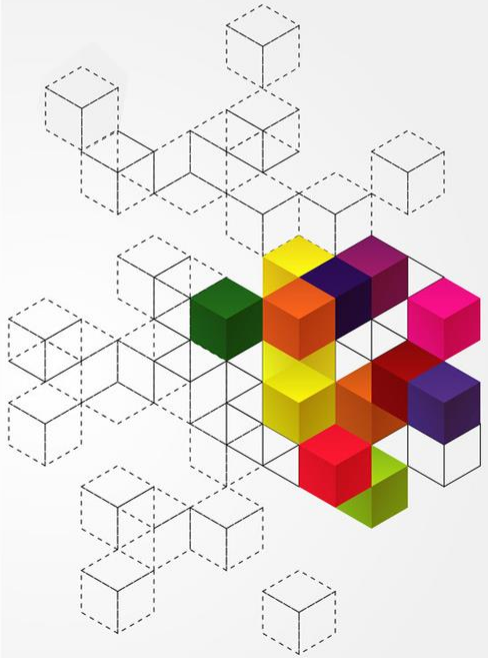


二、分页原理

• 分页的三个重要环节

地址划分 → 建立地址映射（页表） → 地址翻译

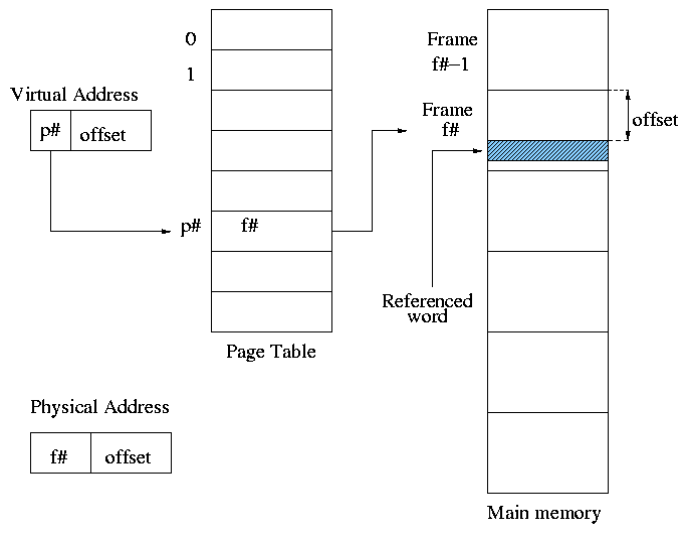


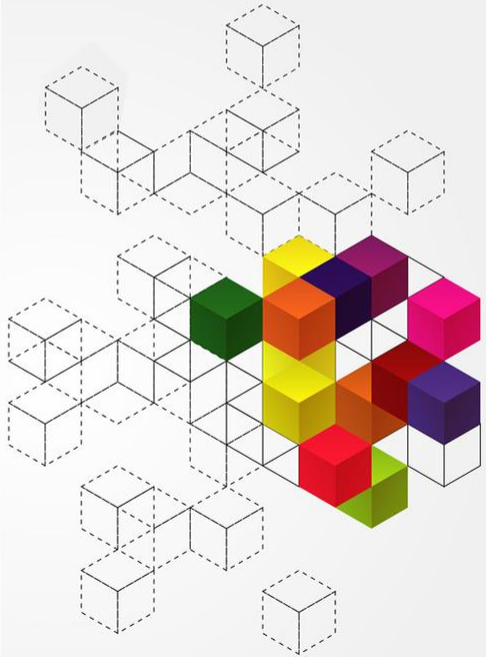


二、分页原理

- **分页的三个重要环节**

地址划分 → 建立地址映射（页表） → 地址翻译

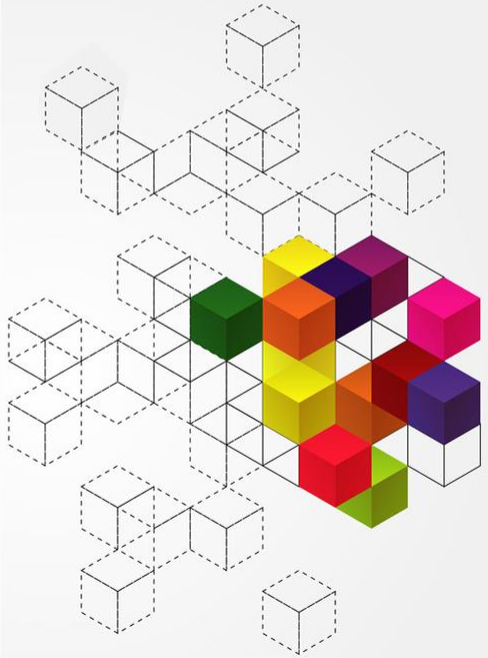




三、分页机制下的内存保护

- **通过页表项中的低位，存储页保护信息**
  - 有效位
  - 读写位
  - 禁止执行位

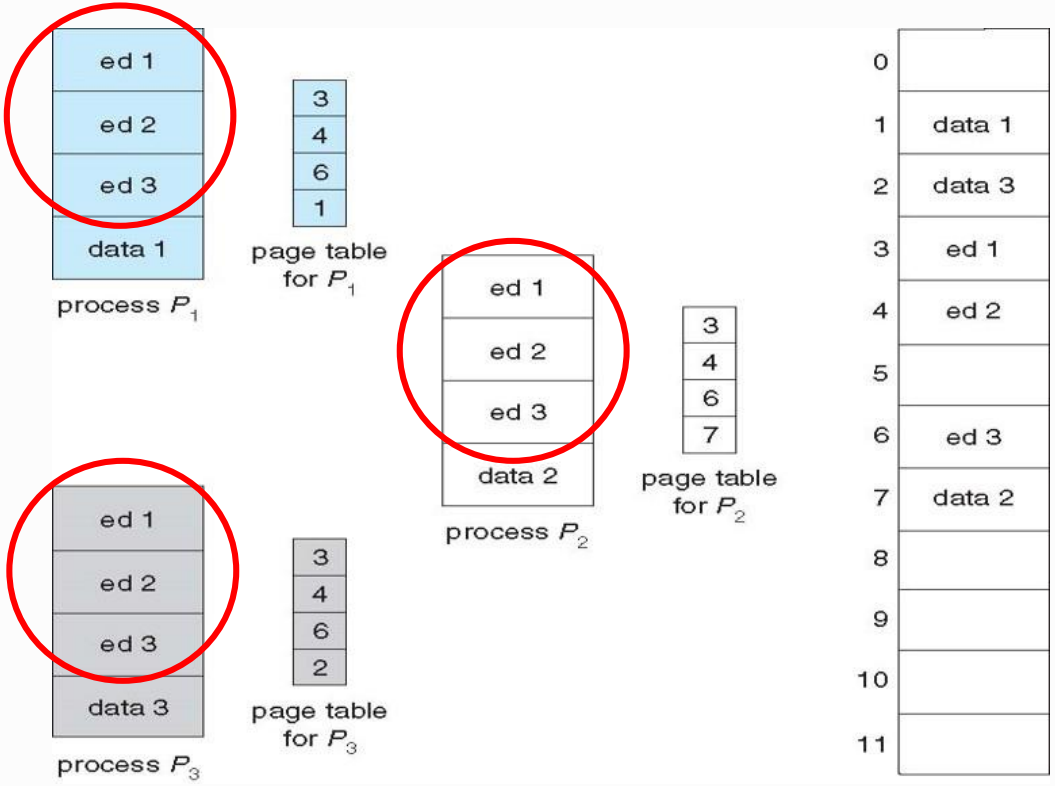
Page Number	Frame Number	Valid	Read Only	NX
0x40a1	0x0100	1	0	1
0x40a2	0x0200	0	0	1
0x40a3	0x00a0	0	0	1
0x40a4	0x0a00	1	1	0
0x40a5	0x0300	1	0	1

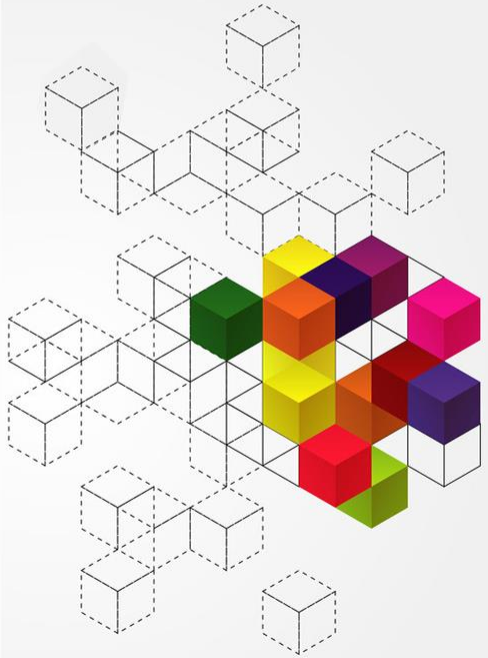


- 可以为每个页提供足够保护信息

四、分页机制下的内存共享

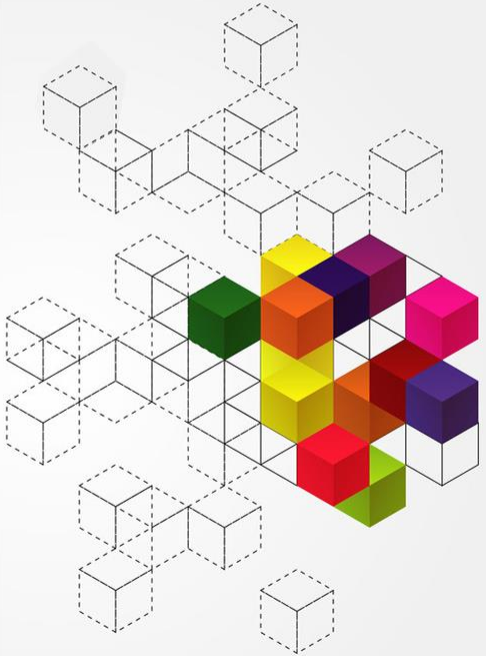
- **分页机制下内存共享示意图**





8.3 小结

- 引入分页的意义
- 分页原理
- 分页机制下的内存保护
- 分页机制下的内存共享



三、可变分区分配

- 慕课堂-讨论3

分页有没有彻底解决内存碎片问题