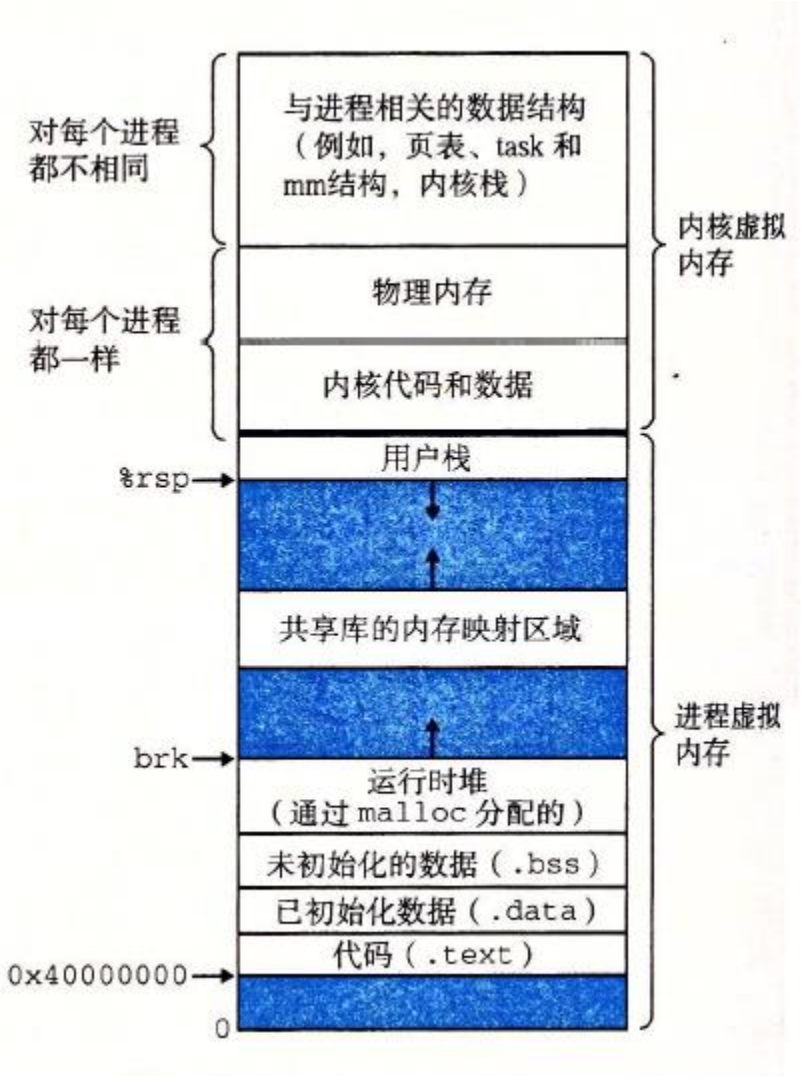
# Virtual Memory & DMM

# Virtual Memory: System

# Linux地址空间

物理内存区域只是整个物理内存空间的一段，一般用于内核与I/O设备之间的交互

.bss除了未初始化的数据，还有初始化成0的数据



# 虚拟内存区域

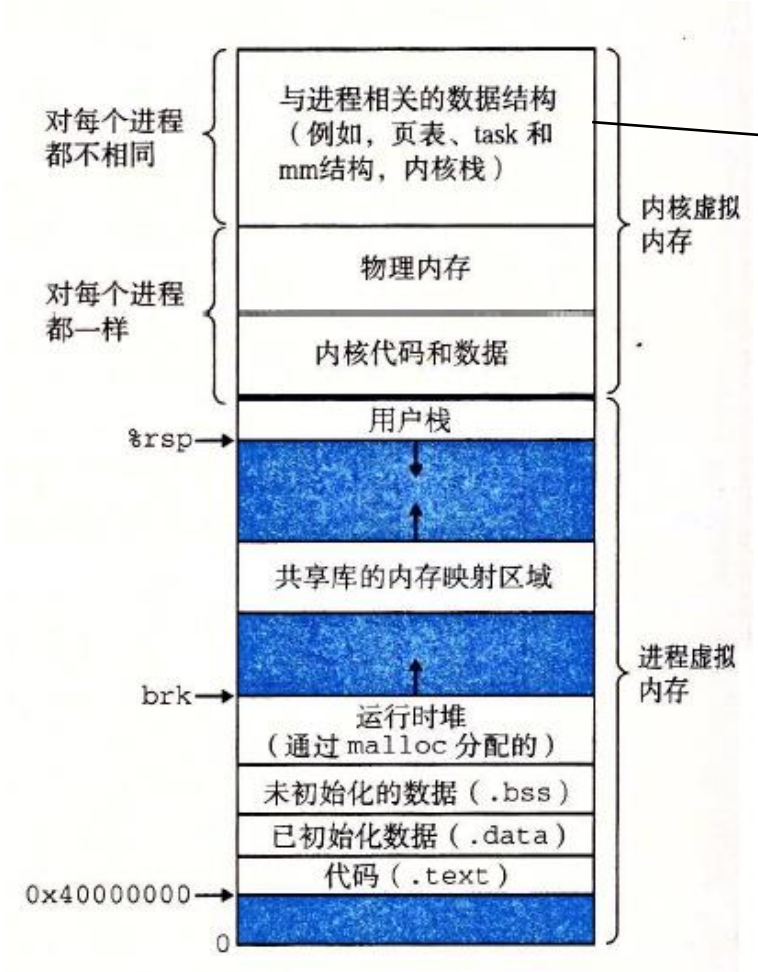
Linux将虚拟内存组织成一些区域（也叫**段**）的集合

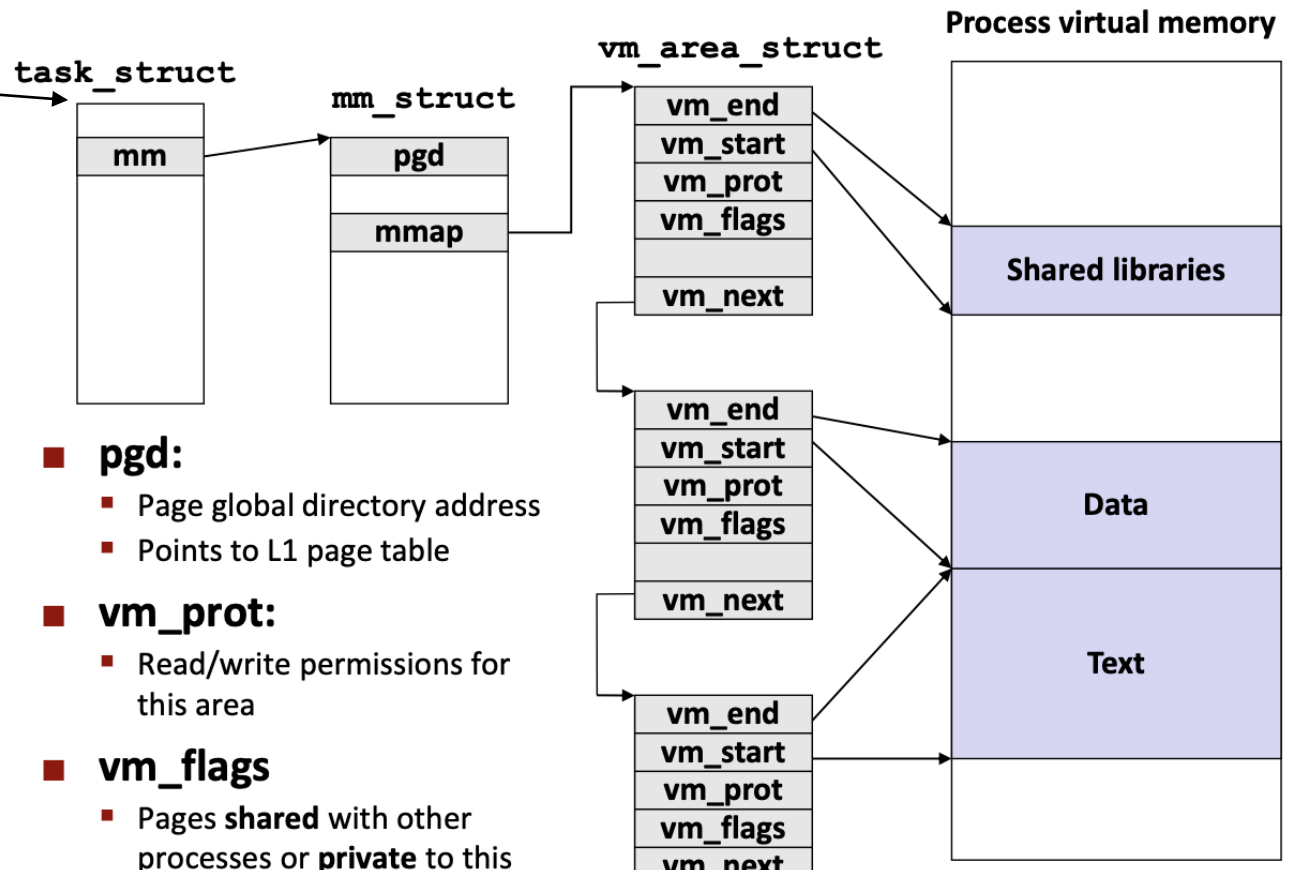
但是，硬件是不用段来管理的，这只是操作系统管理的一个方式

地址空间可以分页管理，也可以分段管理。但是，分段管理和这里的虚拟内存区域是不一样的，分段管理需要硬件和操作系统共同实现。

# 内存管理的数据结构

从程序员的角度来说，有两个重要的数据结构




- **pgd:**
  - Page global directory address
  - Points to L1 page table
- **vm_prot:**
  - Read/write permissions for this area
- **vm_flags**
  - Pages **shared** with other processes or **private** to this process

Each process has own **`task_struct`**, etc

# 页故障的种类（basic）

地址无效：如NULL (0)

权限错误

R/W, XD, U/S

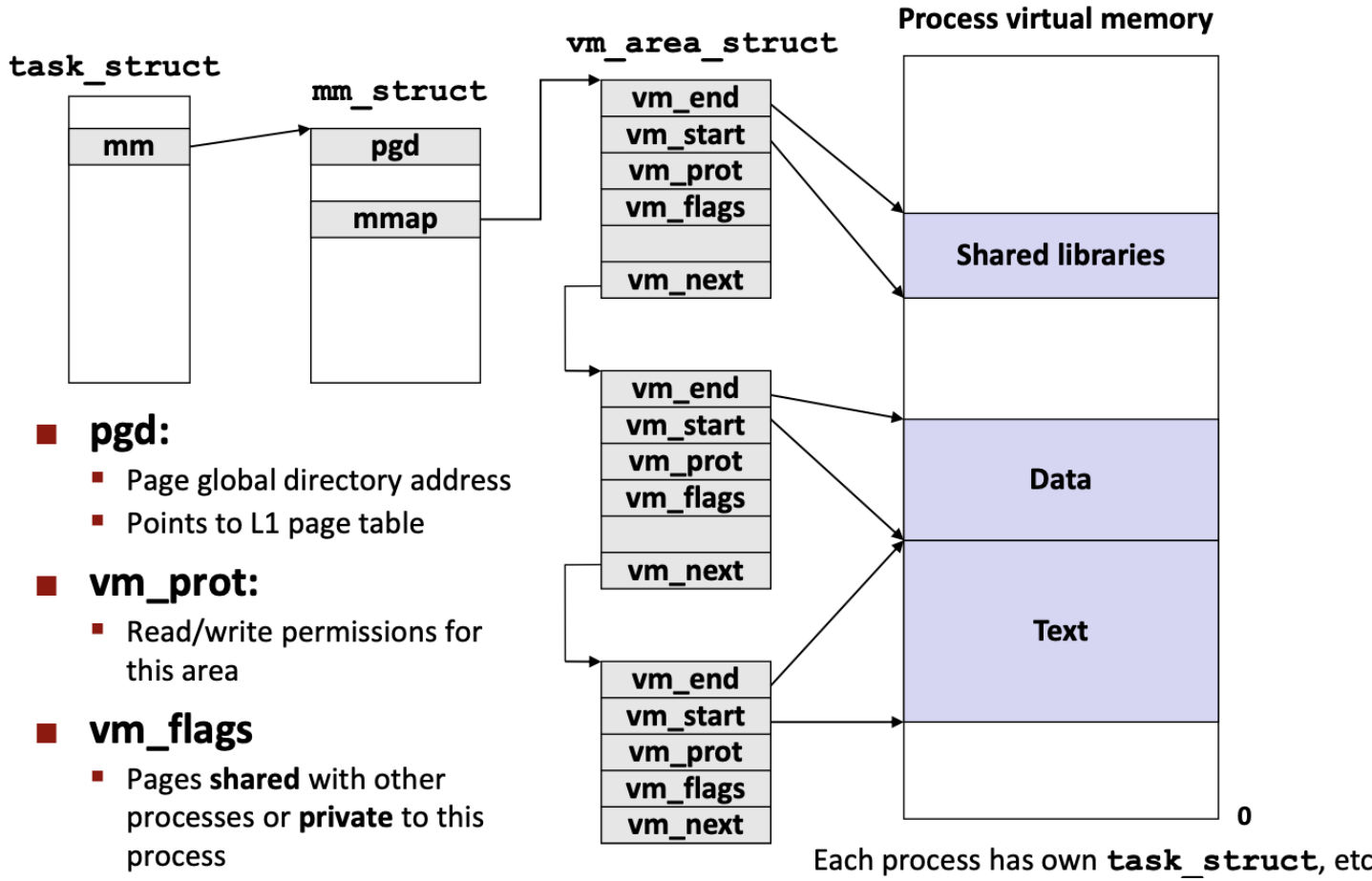
页面不在内存中

操作系统如何决定如何处理Page Fault?

# 操作系统如何决定如何处理Page Fault?

以下只是部分简单的讲解用户状态的PF(page fault)应该如何处理，实际情况会更加复杂

首先，检查引发PF的地址是否在地址映射范围中，如果不在，说明地址非法，杀死进程



# 操作系统如何决定如何处理Page Fault?

如果合法，检查权限：

U/S(是否为内核态)

访问内核态，应该kill

XD(是否可执行)

如果试图执行不可执行的地址，应该kill

R/W(是否可写)

一般情况下，试图写入只读页，应该kill

但如果是因为COW机制导致的PF，应该发生写时复制

P(是否在内存中)

因为该页不在内存而在磁盘上引发PF：从磁盘上读取该页

# 段故障（segmentation fault）

处理页故障时，如果是因为地址无效/权限错误需要杀死进程，实际上操作系统的做法是给进程发送SIGSEGV信号，大家看到的段错误一般是这个原因

地址无效

```
1  int main() {
2      return *(int *)0;
Exception has occurred.
Segmentation fault
3  }
```

权限错误

```
1  int main() {
2      *(int *)main = 0;
Exception has occurred.
Segmentation fault
3  }
```

# 内存映射

普通文件（目录文件不能被mmap映射）

```c
#include "csapp.h"

int main() {
    int dir_fd = Open("./mmap.c", O_RDONLY, S_IRUSR);
    Mmap(NULL, 10, PROT_READ, MAP_SHARED, dir_fd, 0);
}
```

mmap error: No such device

匿名文件（全是二进制0）

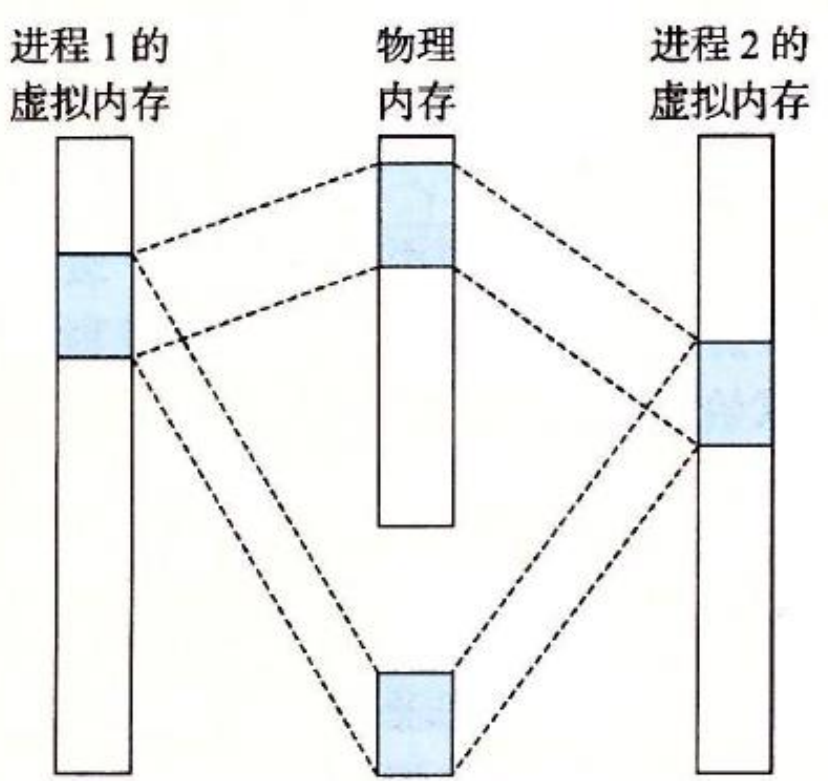
# 进程创建

fork

COW(copy on write)，借助PTE中的R/W位和vm_area_struct中的权限位实现

execve

lazy loading




a）两个进程都映射了私有的写时复制对象之后

12、进程 P1 通过 fork()函数产生一个子进程 P2。假设执行 fork()函数之前，进程 P1 占用了 53 个（用户态的）物理页，则 fork 函数之后，进程 P1 和进程 P2 共占用_____个（用户态的）物理页；假设执行 fork()函数之前进程 P1 中有一个可读写的物理页，则执行 fork()函数之后，进程 P1 对该物理页的页表项权限为_____。上述两个空格对应内容应该是（ B ）

A．53，读写　B．53，只读　C．106，读写　D．106，只读

# Dynamic Memory Management

# 为什么要动态内存分配

最根本的是对象的生命周期，可能比它的创建者的生命周期要长

- 在C语言中有三种类型的生命周期：自动（栈中的对象）、动态（堆中的对象）、静态（.data .bss）

变长对象

- 可以被放在变长栈帧中，但是不高效、不安全
- 大的对象可能会造成栈溢出

# 动态内存管理API

malloc/calloc/realloc

- calloc会初始化
- C++ new会初始化

free

brk/sbrk

# 动态内存管理的目标

最大化吞吐率

最大化内存利用率

- 碎片
  - 内部碎片
  - 外部碎片

# 动态内存管理实现

链表
- 隐式链表
- 显示链表
- 分离链表

适配
- 首次适配
- 下次适配
- 最佳适配

伙伴系统：在固定大小的物理内存分配中很实用

7. 下列与虚拟内存有关的说法中哪些是不对的？

A. 操作系统为每个进程提供一个独立的页表，用于将其虚拟地址空间映射到物理地址空间。

B. MMU 使用页表进行地址翻译时，虚拟地址的虚拟页面偏移与物理地址的物理页面偏移是相同的。

C. 若某个进程的工作集大小超出了物理内存的大小，则可能出现抖动现象。

D. 动态内存分配管理，采用双向链表组织空闲块，使得首次适配的分配与释

答案：D 首次释放不一定是空闲块数量的线性时间

17.动态内存管理中，可能会造成空闲链表中，小空闲块，即"碎片"，比较集中的算法是（ ）

A．首次适配算法　　B．下次适配算法

C．最佳适配算法　　D．以上三种算法无明显区别

**【答案】A**

**【说明】考察空闲链表的不同搜索分配策略的性质。**

# 内存安全

https://en.wikipedia.org/wiki/Memory_safety

分配、赋值、访问、释放

如果任何一个步骤出现错误，或者他们的顺序有错，就会产生内存不安全的情况