

# UC Day03

预习课

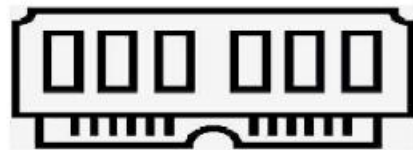
# 预习 内容

虚拟地址

# 虚拟地址

# 我们的程序是如何跑起来的

- 我们写好代码，经过编译后，得到可执行文件a.out。那么a.out是如何被执行起来的呢？
- a.out文件存在于磁盘上，是二进制指令和数据的集合，其内容经操作系统的管理调度被加载到物理内存，CPU再从物理内存中读取指令执行。我们写的程序就这样跑起来啦？



# 电脑为什么会变的很卡

- 电脑上的软件开多了，电脑会变得很“卡”，这是为什么？
  - 程序被加载到物理内存，才能被CPU所执行。物理内存的空间有限，当难以满足使用需求时，操作系统会把一些长期闲置的代码和数据从物理内存缓存到磁盘上，这叫页面换出，一旦需要使用那些代码和数据，再把它们从磁盘上恢复到物理内存中，这叫页面的换入。
  - 磁盘上用来缓存物理内存数据的部分称为交换分区
  - 磁盘的读写速率要远慢于物理内存，大量的数据换入和换出，会导致电脑变得很“卡”



# 地址的“真假”

- 我们再程序中看到的地址，真的就是物理内存的地址吗？
  - 我们在程序中所看到的或者使用的地址，并非真实的物理内存的地址，而是经由系统内核的内存管理系统管理后所看到的虚拟地址。
  - 虚拟地址和物理地址之间存在映射对应关系。
  - 内存管理系统一方面保证物理内存的安全，避免物理内存被直接操作，同时也降低了程序员的编码难度。





# 直播课见

# UC

## C/C++教学体系



# 目录

虚拟地址空间

虚拟地址空间布局

内存映射的建立与解除

# 虚拟地址空间

# 虚拟地址空间

- 对于32位操作系统而言，每个进程所得到的虚拟地址都在一个固定的范围内，不会超出这个范围，我们把这个范围称为虚拟地址空间。
- 所谓的虚拟地址空间本质就是一个地址范围，表示程序的寻址能力。我们所看到的虚拟地址，都是在这个范围内。
- 对于32位操作系统而言，其虚拟地址空间范围从0x00000000到0xFFFFFFFF
- 其中0 ~ 3G-1的范围归用户所使用，称为用户地址空间，3G ~ 4G-1的范围归内核使用，称为内核地址空间。



# 虚拟地址空间

- 对于64位的操作系统而言，目前应用程序没有这么大的内存需求，所以不支持完全的64位虚拟地址。
- 64位系统上，其用户地址空间范围是：
  - 0x0000 0000 0000 0000 ~ 0x0000 FFFF FFFF FFFF
- 64位系统上，其内核地址空间范围是：
  - 0xFFFF 0000 0000 0000 ~ 0xFFFF FFFF FFFF FFFF
- 内核地址空间 and 用户地址空间之间是不规范地址空间，不允许使用。
- 用户地址空间的代码不能直接访问内核空间的代码和数据，但可以通过系统调用进入内核态，间接与系统内核交互




# 虚拟地址空间布局



# 静态库的制作和使用

- 程序中不同性质的数据，加载到内存后，其虚拟地址会被映射到虚拟地址空间中不同区域
- 用户地址空间中从低地址到高地址布局如图：

|   |                                  |
|---|----------------------------------|
| 参数和环境区  | 命令行参数和环境变量                       |
| 栈区(stack)   | 非静态局部变量                          |
|  |                                  |
| 堆区(heap)  | 堆栈增长的预留空间<br>共享库、共享内存等           |
| BSS区(bss)   | 动态内存分配                           |
| 数据区(data)   | 未被初始化的全局和静态局部变量                  |
| 代码区(text)   | 不具常属性且被初始化的全局和静态局部变量             |
|   | 可执行指令、字面值常量、具有常属性且被初始化的全局和静态局部变量 |

# 内存映射的建立与解除



# 内存映射的建立与解除

- 没有与物理地址建立起映射关系的虚拟地址，无法直接使用，我们可以通过系统调用mmap函数手动建立虚拟地址和物理地址之间的映射关系。
- `#include <sys/mman.h>`
- `void* mmap(void* start, size_t length, int prot, int flags, int fd, off_t offset);`
  - 功能：建立虚拟内存到物理内存或磁盘文件的映射：
  - 参数：  
start：映射区虚拟内存的起始地址，NULL系统自动选定后返回。  
length：映射区字节数，自动按页圆整。  
prot：映射区操作权限，可取以下值：
    - PROT\_READ - 映射区可读
    - PROT\_WRITE - 映射区可写
    - PROT\_EXEC - 映射区可执行
    - PROT\_NONE - 映射区不可访问



# 内存映射的建立与解除

- 参数: flags: 映射标志, 可取以下值:
  - MAP\_ANONYMOUS - 匿名映射, 将虚拟内存映射到物理内存而非文件, 忽略fd和offset参数
  - MAP\_PRIVATE - 对映射区的写操作只反映到缓冲区中并不会真正写入文件
  - MAP\_SHARED - 对映射区的写操作直接反映到文件中
  - MAP\_DENYWRITE - 拒绝其它对文件的写操作
  - MAP\_FIXED - 若在start上无法创建映射, 则失败 (无此标志系统会自动调整)
- fd: 文件描述符
- offset: 文件偏移量, 自动按页(4K)对齐
- 返回值: 成功返回映射区虚拟内存的起始地址, 失败返回MAP\_FAILED(-1)



# 内存映射的建立与解除

- `#include <sys/mman.h>`
- `int munmap(void* start, size_t length);`
  - 功能：解除虚拟内存到物理内存或磁盘文件的映射：
  - 参数：start： 映射区虚拟内存的起始地址。  
length：映射区字节数，自动按页圆整。
  - 返回值：成功返回0，失败返回-1。
- munmap允许对映射区的一部分解映射，但必须按页处理



谢谢

# UC Day03

复习课

# 段错误



# 段错误

- 一切对虚拟内存的越权访问，都会导致段错误
  - 试图访问没有映射到物理内存的虚拟内存
  - 试图以非法方式访问虚拟内存，如对只读内存做写操作等





下节课见