华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

网络空间安全学院

2.3 Linux系统内存管理

网络空间安全学院　慕冬亮
Email : dzm91@hust.edu.cn

为什么需要内存管理？

- 早期程序直接运行于物理内存，直接利用物理内存进行寻址
- 然而，对于多任务系统，其中存在很多问题：
  - 地址空间不隔离。所有的程序都直接访问物理地址，可直接更改内容
  - 内存使用缺乏有效管理机制，效率低下
  - 程序运行地址不确定

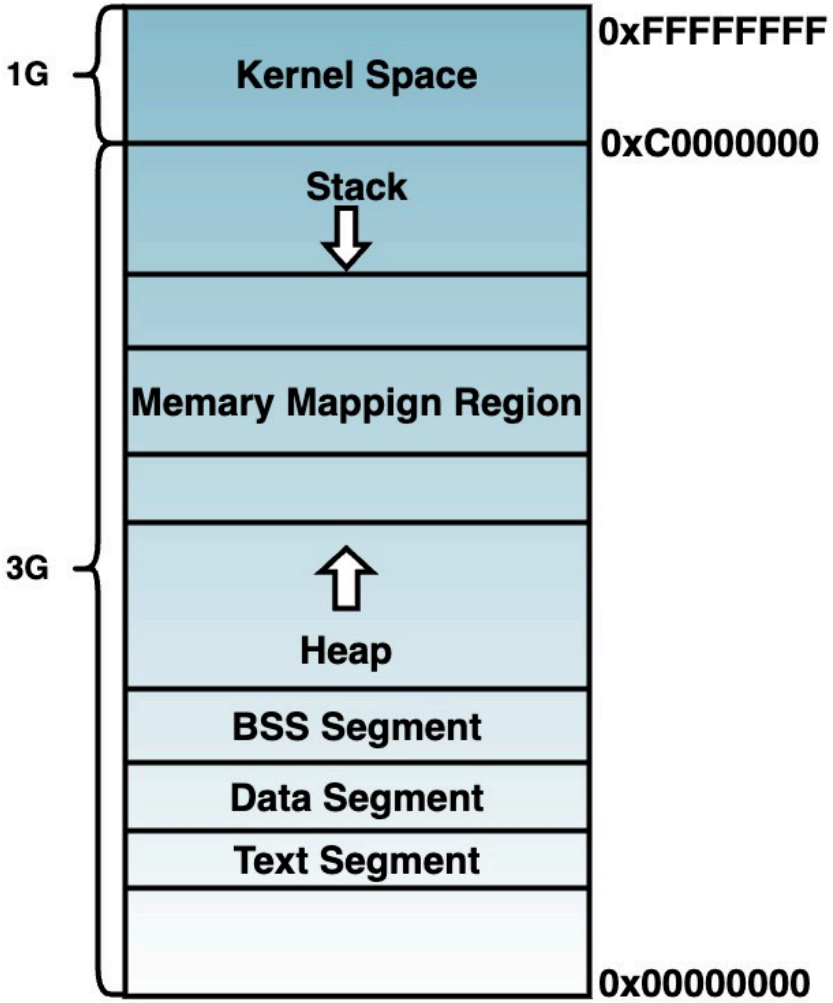
计算机科学领域的任何问题都可以通过增加一个间接的中间层来解决

方案：程序使用虚拟地址，运用某种映射方法，转换为物理地址

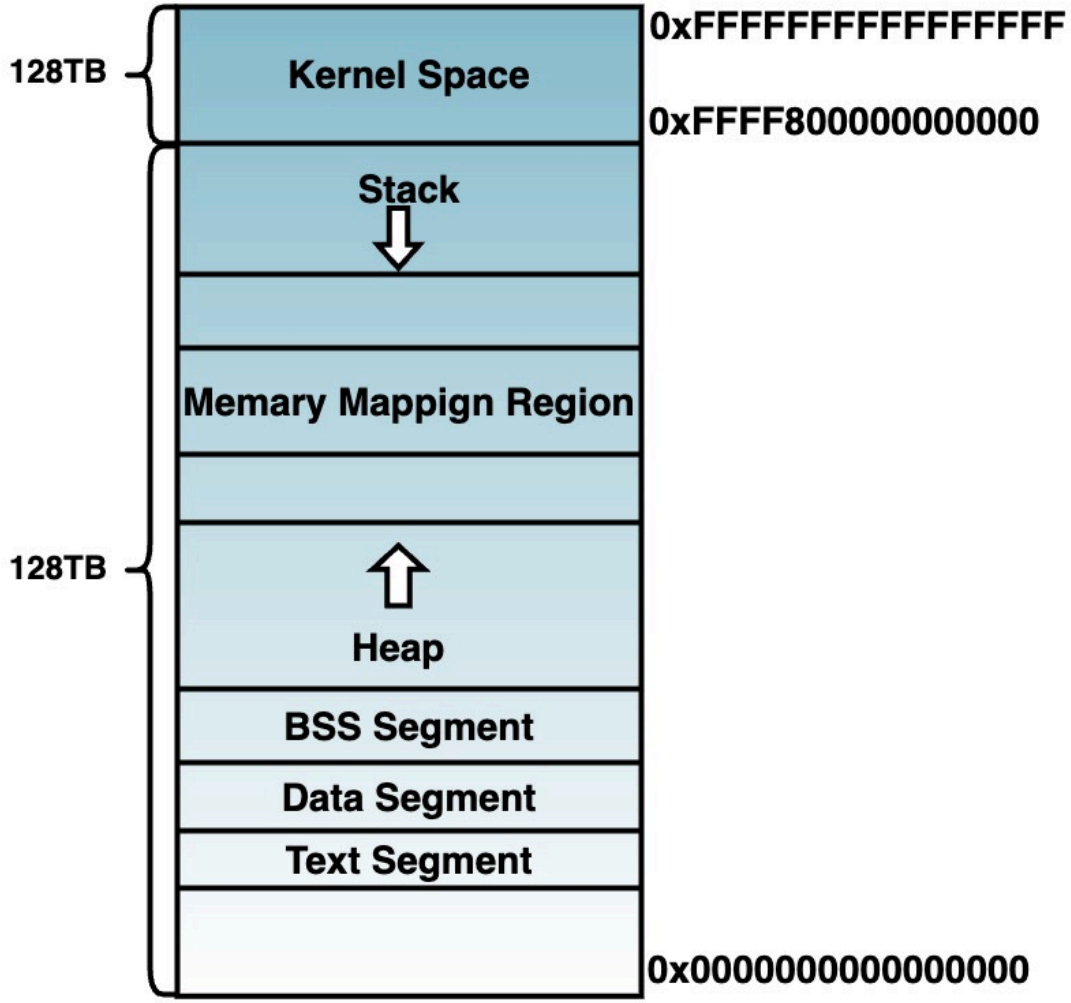
进程地址空间

- 程序运行使用的单一地址空间，称为“进程地址空间”
- 进程地址空间，抽象概念
  - 可以想象为一个大数组，数组大小由地址空间长度决定
- 虚拟地址空间
  - 虚拟的，想象出来的地址空间，并不真实存在
  - 每个进程都有自己独立的虚拟空间，内存地址可固定
  - 每个进程只能访问自己的地址空间，有效进行进程隔离。
- 物理地址空间
  - 实实在在存在于计算机中，可想象为物理内存

Linux虚拟内存管理

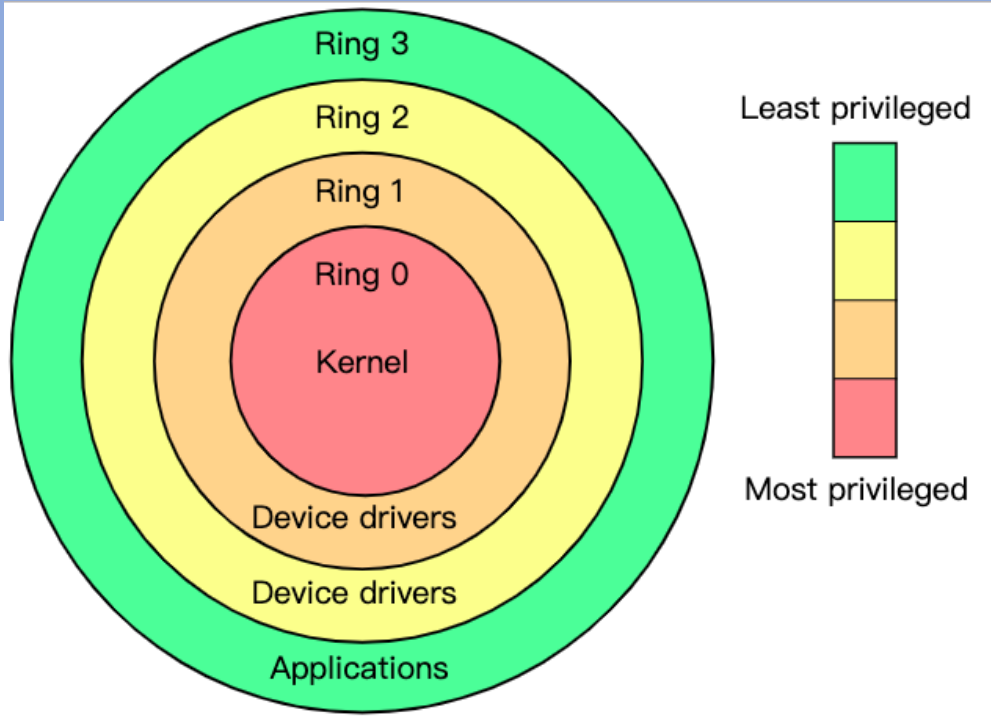


x86_32 进程地址空间



x86_64 进程地址空间

CPU特权级与内存访问



与进程虚拟内存中用户模式区和内核模式区相对应，Linux为了确保系统稳定性，将处理器存取模式划分为用户模式（Ring 3）和内核模式（Ring 0）。

- 用户应用程序一般运行在用户模式。
  - 其访问空间局限于用户区；
- 操作系统内核代码（如系统服务和设备驱动程序等）运行在内核模式。
  - 可以访问所有的内存空间（包括用户模式分区）和硬件，可使用所有处理器指令。

用户态内存区域

- 用户区是每个进程真正独立的可用内存空间，进程中的绝大部分数据都保存在这一区域。
  - 主要包括：应用程序代码、全局变量、所有线程的线程栈以及加载的DLL代码等

- 每个进程的用户区的虚拟内存空间相互独立，一般不可以直接跨进程访问，这使得一个程序直接破坏另一个程序的可能性非常小。

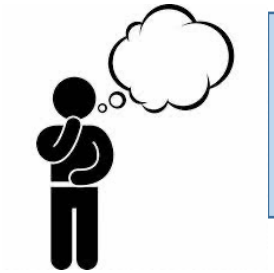
用户态内存Demo

```
seclab@VM-0-11-debian:~$ cat /proc/self/maps
5618ef5df000-5618ef5e1000 r--p 00000000 fe:01 132390                     /usr/bin/cat
5618ef5e1000-5618ef5e6000 r-xp 00002000 fe:01 132390                     /usr/bin/cat
5618ef5e6000-5618ef5e9000 r--p 00007000 fe:01 132390                     /usr/bin/cat
5618ef5e9000-5618ef5ea000 r--p 00009000 fe:01 132390                     /usr/bin/cat
5618ef5ea000-5618ef5eb000 rw-p 0000a000 fe:01 132390                     /usr/bin/cat
5618f14a7000-5618f14c8000 rw-p 00000000 00:00 0                          [heap]
7fccc74e7000-7fccc7509000 rw-p 00000000 00:00 0
7fccc7509000-7fccc77f1000 r--p 00000000 fe:01 138337                     /usr/lib/locale/locale-archive
7fccc77f1000-7fccc7813000 r--p 00000000 fe:01 146811                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7fccc7813000-7fccc796d000 r-xp 00022000 fe:01 146811                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7fccc796d000-7fccc79bc000 r--p 0017c000 fe:01 146811                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7fccc79bc000-7fccc79c0000 r--p 001ca000 fe:01 146811                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7fccc79c0000-7fccc79c2000 rw-p 001ce000 fe:01 146811                     /usr/lib/x86_64-linux-gnu/libc-2.31.so
7fccc79c2000-7fccc79c8000 rw-p 00000000 00:00 0
7fccc79cd000-7fccc79ce000 r--p 00000000 fe:01 146662                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7fccc79ce000-7fccc79ee000 r-xp 00001000 fe:01 146662                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7fccc79ee000-7fccc79f6000 r--p 00021000 fe:01 146662                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7fccc79f7000-7fccc79f8000 r--p 00029000 fe:01 146662                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7fccc79f8000-7fccc79f9000 rw-p 0002a000 fe:01 146662                     /usr/lib/x86_64-linux-gnu/ld-2.31.so
7fccc79f9000-7fccc79fa000 rw-p 00000000 00:00 0
7ffea715f000-7ffea7180000 rw-p 00000000 00:00 0                          [stack]
7ffea71f0000-7ffea71f4000 r--p 00000000 00:00 0                          [vvar]
7ffea71f4000-7ffea71f6000 r-xp 00000000 00:00 0                          [vdso]
```

内核区的内存

- 内存内核区中的所有数据是所用进程共享的，是操作系统代码的驻地。
  - 其中包括：操作系统内核代码，以及与线程调度、内存管理、文件系统支持、网络支持、设备驱动程序相关的代码。

- 该分区中所有代码和数据都被操作系统保护。
  - 用户模式代码无法直接访问和操作：如果应用程序直接对该内存空间内的地址访问，将会发生地址访问违规。



思考题：那么内核是否可以毫无限制地访问用户空间的内存呢？