



# 第0007讲 虚拟地址空间布局架构

User Applications

O/S Services

Linux Kernel

Hardware Controllers



零声学院讲师: Vico老师



一、内存管理架构

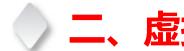
二、虚拟地址空间布局





内存管理子系统架构可以分为:用户空间、内核空间及硬件部分3个层面,具体结构如下图所示:

- 1、用户空间:应用程序使用malloc()申请内存资源/free()释放内存资源。
- 2、内核空间:内核总是驻留在内存中,是操作系统的一部分。内核空间为内核保留, 不允许应用程序读写该区域的内容或直接调用内核代码定义的函数。
- 3、硬件:处理器包含一个内存管理单元(Memory Management Uint, MMU)的部件,负责把虚拟地址转换为物理地址。



# 二、虚拟地址空间布局架构



一切只为渴望更优秀的你!

因为目前应用程序没有那么大的内存需求,所以ARM64处理器不支持完全的64位虚拟地址。

在ARM64架构的Linux内核中,内核虚拟地址和用户虚拟地址的宽度相同。

所有进程共享内核虚拟地址空间,每个进程有独立的用户虚拟地址空间,同一个线程组的用户线程共享用户虚拟地址空间,内核线程没有用户虚拟地址空间。



ARM 64内核/用户虚拟地址空间划分



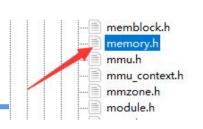
#### 1、用户虚拟地址空间划分

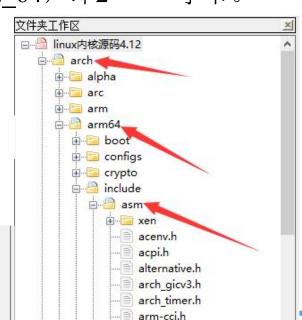
进程的用户虚拟空间的起始地址是0,长度是TASK\_SIZE,由每种处理器架构定义自己的宏TASK\_SIZE。ARM64架构定义的宏TASK\_SIZE如下:

32位用户空间程序: TASK\_SIZE的值是TASK\_SIZE\_32,即0x100000000,等4GB。

64位用户空间程序: TASK SIZE的值是TASK SIZE 64,即2<sup>VA\_BITS</sup>字节。

### 用户虚拟地址空间Linux内核源码分析右图所示:







360

361

362

363

364

365

366

367

368

369

#endif



一切只为渴望更优秀的你!

Linux内核使用内存描述符mm\_struct,描述进程的用户虚拟地址空间,内核源码分析如下图所示:

```
□-- linux内核源码4.12
                                                                 mmu context.h
                                                                                         arch arch
                                                                 mmu notifier.h
                                                                                         i block
                                                                 mmzone.h
                                                                                         di certs
                                                                 mm inline.h
                                                                                         ⊕ a crypto
                                                                 mm types.h
                                                                                         i Documentation
                                                                 mm types task.h
                                                                                         drivers
                                                                 mnt namespace.h
                                                                                         firmware
                                                                 module.h
                                                                                         # fs
                                                                 moduleloader.h
                                                                                         include
                                                                 moduleparam.h
                                                                                           acpi
                                                                 mod devicetable.h
                                                                                           asm-generic
                                                                 mount.h
                                                                                           di clocksource
                                                                 mpage.h
                                                                                           erypto
                                                                 mpi.h
                                                                                           drm drm
                                                                 mpls.h
                                                                                           dt-bindings
                                                                                           ⊕ 📴 keys
struct mm struct {
                                                                                           ± kvm
      struct vm area struct *mmap;
                                                 /* list of VMAs */
                                                                                           ⊟ ⊟ linux ◀
      struct rb root mm rb;
                                                                                             amba
      u32 vmacache seqnum;
                                                     /* per-thread vmacache */
                                                                                             ⊕ 📴 bcma
##ifdef CONFIG MMU
                                                                                             byteorder
                                                                                             🗓 🧧 can
      unsigned long (*get unmapped area) (struct file *filp,
                                                                                             eph ceph
                    unsigned long addr, unsigned long len,
                                                                                             ⊕ . □ clk
                    unsigned long pgoff, unsigned long flags);
```



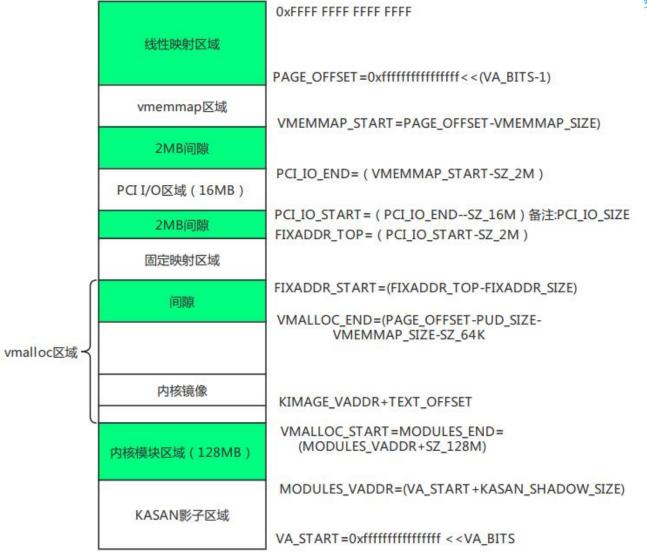
多的你!

## 2、内核地址空间布局

ARM64处理器架构内核地址空

间布局如右图所示:

KASAN: 动态内存错误检查工具









办学宗旨:一切只为渴望更优秀的你

办学愿景: 让技术简单易懂