**如何学习linux 内核**

**对于刚刚接触linux 爱好者，可以尝试在自己的电脑中安装一个linux 发行片，如优麒麟Linux 18.04，并尝试使用Linux 作为操作系统，另外建立读者一些常用的命令，熟悉如何使用vim和git等工具，尝试去编译和更换优麒麟linux 的内核核心。**

**然后，开始在Linux机器上做一些编程和调试的练习，如何使用QEMU+GDB+Esclipe来单步调试内核，熟悉GDB的使用等。**

**接下来，从一个简单的设备驱动程序开始，选择一个简单的字符设备驱动，如触摸屏驱动等。从编写和调试设备驱动深入Linux内核的一些核心API的实现**

**对Linux驱动有深刻的理解之后，就可以研究Linux内核的一些核心API的实现，如malloc()和中断线程化等。**

**学习Linux 内核的过程是枯燥的，但是又那么吸引人，它的魅力只有你深入后才能体会到**

## 第2章 内存管理

**2.1 物理内存初始化**

1. 在系统启动时，ARM linux 内核如何知道系统中有多大的内存空间？

2. 在32bit linux 内核中，用户空间和内核空间的比例通常是3:1； 可以修改成2:2吗?

3. 物理内存页面如何添加到伙伴系统中，是一页一页添加，还是以2的几次幂加入呢?

**2.2 页表的映射过程**

4. 内核的一级页表存放在什么地方 ，内核空间的二级页表又存在什么地方

5. 用户进程的一级页表存放在什么地方？二级页表又存放在什么地方

6. 在ARM32系统中，页表如何映射的，在ARM64系统中，页表又是如何映射的

**2.3 内核内存的布局图**

* 在32bit Linux 中，内核空间的线性映射的虚拟地址和物理地址是如何换算的？
* 在32bit linux 中，高端内存的起始地址是如计算出来的。
* 请画出ARM32 linux 内核的内存布局图。

**2.4 分配物理页面**

7. 请简述linux内核在理想情况下页面分配器(page allocator)是如何分配出连续页面的；

8. 在页面分配器中，如何从分配掩码(gfp\_mask)中确定可以从哪些zone中分配内存?

9. 页面分配器是按照什么方向来扫描zone的

10.为用户进程分配物理内存，分配应该选用GFP\_kernel,还是GFP\_HIGHUSER\_MOVABLE呢

**2.5 slab分配器**

11.slab分配器是如何分配和释放小内存块的

12.slab 分配器中一个着色的概念(cache color)，着色有什么作用？

13.slab 分配器中的slab 对象有没有根据per-cpu做一些优化

14.slab 增长并导致大量不用空闲对象，该如何解决

**2.6 vmalloc**

15.请问kmalloc,vmalloc和malloc之间有什么区别以及实现上的差异

**2.8 malloc**

* 函数malloc() 函数返回的内存是否马上就分配物理内存？testA和testB分别在何时分配物理内存？
* 使用用户态的API函数malloc分配内存时，会马上为其分配物理内存吗？
* 假设不考虑libc因素，malloc分配的100byte,那么实际上内存是为其分配100Byte吗
* 假设两个用户进程打印的malloc分配的虚拟地址是一样的。那么在内核中这两块虚拟内存是否打架呢
* vm\_normal\_page()函数返回的是什么样页面的stuct page数据结构？为什么内存管理代码中需要这个函数
* 请简述get\_user\_page函数的作用和实现流程
* 请简述follow\_page()函数的作用和实现流程

**2.9 mmap**

* 请简述私有映射和共享映射的区别。
* 为什么第二次调用mmap时，linux内核没有捕捉到地址重叠并返回失败呢

**2.11 Page 引用计数**

* struct page 数据结构中\_count和\_mapcount有什么区别
* 匿名页面和page cache 页面有什么区别
* struct page 数据结构有一个锁，请问trylock\_page()和lock\_page有什么区别

**2.12 反向映射RMAP**

* 在linux 2.4.x 内核中，如何从一个page找到映射该页面的vma,反射映射可以带来哪些使得
* 阅读linux4.0 内核ramp机制的代码，画出父子进程之间vma,avc,anon\_vma和page等数据结构之间的关系图
* 在linux 2.6.43 中，RMAP机制采用的新的实现，在linux 2.6.33 和之前的版本中称为旧版本的rmap机制，那么在旧版本rmap机制中，如果父进程有1000个子进程，每个子进程都有一个vma,这个vma里面有1000个匿名页面，当所有的子进程的vma同时发生写复制时会是什么情况呢

2.13 回收页面

* 当page 加入lru链接中，被其他线程释放了这个page,那么lru链接如何知道这个page已经被释放了
* kswapd内核线程何时被会唤醒
* LRU链接如何知道page的活动频繁程度
* kswapd 按照什么原则来换出页面
* kswapd 按照什么方向来扫描zone
* kswapd 以什么标准来退出扫描LRU
* 手持设备例如Android 系统，没有swap分区或者swap文件，kswap会扫描匿名页面LRU吗
* swappiness的含义是什么？kswapd如何计算匿名页面和page cache 之间的扫描比重？
* 当系统充斥大量只访问一次的文件访问(user one streaming IO)时，kswapd 如何规避这种风暴。
* 在回收page cache时，对于dirty 的page cache,kswapd会马上回写的吗
* 内核有那些页面会被kswapd写回交换分区

41. ARM32 linux如何模拟这个linux版本的L\_TYPE\_YOUNG比特位呢

42如何理解refault Distance算法

**2.14 匿名页面生命周期**

* 请简述匿名页面的生命周期，在什么情况下会产生匿名页面，在什么条件下会释放匿名页面

**2.17 KSM**

* KSM是基于什么原理来合并页面的
* 在ksm机制里，合并过程中把page设置成写write\_protect\_page()有这样一个判断
* 如果多个vma的虚拟页面同时映射了同一个匿名页面，那么此时page->index应该等于多少？

**2.18 Dirty COW内存漏洞**

* 为什么Dirty cow小程序可以修改一个只读文件的内容。
* 在dirty cow内存漏洞中，如何dirty cow 程序没有madvise thread 线程，即有procselfmem thread线程，能否修改foo文件内容
* 假设在内核空间获取了某个文件对应page cache 页面struct page 数据结构，而对应vma属性是只读，那么内核空间是否可以成功修改该文件呢
* 如果用户进程使用只读属性(PROT\_READ)来mmap映射一个文件到用户空间，然后memcpy来写这段内存空间，会是什么样的情况

**2.19 总结内存管理数据结构和API**

* 请画出内存管理中常用的数据结构的关系图：如mm\_struct, vma, vaddr, page, pfn, pte, zone, paddr和pg\_data等，并思考如下转换关系。

(1) 如何由mm数据结构和虚拟地址adddr找到对应vma

(2) 如何由page和vma找到虚拟地址addr

(3) 如何由page 找到所的映射的vma

(4) 如何由vma和虚拟地址addr 找出相应的page数据结构

(5) page 和pfn之间互换

(6) pfn 和paddr之间互换

(7) page 和pte之间互换

(8) zone 和page 之间互换

(9) zone 和pg\_data之间互换

* 52 请画出在最糟糕的情况下分配若干个连续物理页面的流程图
* 53 在Android 中新加了LMK(low memory killer )请描述LMK 和OOM killer 之间的关系
* 54 请描述一致性DMA映射dma\_alloc\_coherent()函数在ARM中是如何管理cache一致性
* 55.请描述流式DMA 映射 dma\_map\_single()函数在ARM中是如何管理cache一致性的
* 56. 为什么在linux 4.8 内核中要把基于zone的LRU链表机制迁移到基于node 呢？

## 第3章 进程管理

**3.1 进程的诞生**

1. 在内核中如何获取当前进程的task\_struct 数据结构

2. 下面程序 会打印几个"\_"

**3.2 CFS调度器**

3. 用户空间进程的页表是什么分配的，其中一级页表什么时候分配 二级页表呢

4. 请简述对进程调度器的理解，早期linux 内核调度器(包括O(N)和O(1) 是如何工作的。

5. 请简述进程优先级，nice值和权重之间的关系

6. 请简述CFS调度器是如何工作的。

7. CFS调度器中vruntime是如何计算的

8. vruntime是何时更新的。

9. CFS调度器的min\_vruntime有什么作用

10. CFS调度器对新创建的进程和刚唤醒的进程有何关照

11. 如何计算普通进程的平均负载 load\_avg\_contrib?runnable\_avg\_sum和runnable\_avg\_period分别是什么含义

12. 内核代码中定义了若干个表，请分别说出它们的含义，比如prio\_to\_weight prio\_to\_wmult,runable\_avg\_yN\_inv,runable\_avg\_yN\_sum

13. 如果一个普通进程在就绪队列等待了很长时间才被调度，那么它的平均负载该如何计算

**3.2 SMP调度器**

14. 一个4核处理器中的每个物理CPU拥有独立 L1 cache且不支持超线程技术，

分成两个簇cluser0 和cluster1,每个簇包含两个物理CPU核，族中的CPU核共享

L2 cache.请画出该处理器在linux内核里调度域和调度组的拓扑关系图。

15. 假设CPU0和CPU1同属于一个调度域中且它们都不是ilde cpu,那么CPU1可以做负载负均衡吗？

16. 如何查找出一个调度域 里最繁忙的调度组？

17. 如果一个调度域负载不均衡，请问如何计算需要迁移多少负载量呢？

18. 使用内核提供的唤醒的进程API，比如wake\_up\_process来唤醒一个进程，

那么进程唤醒后应该在哪个CPU上运行呢？是调用wake\_up\_process的那个CPU，还是之前运行的那个CPU，或者是其他CPU呢？

**3.6 EAS绿色节级调度器**

19. 请问WALT算法是如何计算进程期望运行时间的。

20. EAS调度器如何衡量一个进程的计算能力？

21. 当一个进程被唤醒时，EAS调度器如何选择哪个CPU上运行

22. EAS调度器是否做CPU间的负载均衡呢

23. 目前在linux 4.0内核中，CPU动态调频调压模块CPUfreq和进程调度器之间如何协同工作的。有什么优缺点

24. 在EAS调度器中，WALT算法计算出来的负载等信息有什么作用？

## 第4章 并发与同步

**4.1 原子操作与内存屏障**

* 在ARM处理器中，如何实现独占访问内存
* atomic\_cmpxchg()和atomic\_xchg()分别表示什么含义

4.2 spinlock

* 为什么spinlock的临界区不能睡眠(不考虑RT-linux的情况）
* Linux内核中经典spinlock的实现有什么缺点？
* 为什么spinlock临界区不允许发生抢占？
* ticket-based的spinlock机制是如何实现的
* 如果在spin\_lock()和spin\_unlock()的临界区中发生了中断，并且中断处理程序也恰巧修改了该临界资源，那么会发生什么后果？该如何避免呢？

**信号量**

* 与spinlock相比，信号时有那些特点？
* 请简述信号量是如何实现的。

**4.4 Mutex 互斥体**

* 12. Linux内核已经实现了信号量机制，为何要单独设置一个mutex机制呢？
* 13. 请简述MCSe锁机制实现原理
* 14. 在编写内核代码时，该如何选择信号量和mutex?

**4.5 读写锁**

* 什么时候使用读者信号量，什么时候使用写者信号量，由什么来判断？
* 读写信号 量使用的自旋等机制(optimistic spinning)是如何实现的？

**4.6 RCU**

15. RCU相比读写锁有那些优势？

16. 请解释quiescent state和grace period

17. 请简述RCU实现的基本原理

18. 在大型系统中，经典RCU遇到了什么问题？Tree RCU又是如何解决该问题的

19. 在RCU实现中，为什么要使用ULONG\_CMP\_GE()和ULONG\_CMP\_LT()宏来比较两个数的大小，而不直接使用大于号或者小于号来比较。

20. 请简述一个grace period 的生命周期及其状态机的变化

**4.7 内存管理中的锁**

* 请总结原子操作，spinlock,信号量，读写信号量，mutex和RCU等Linux内核常用锁的特点和使用规则
* 在KSM中扫描某个VMA寻找有效的匿名页面，假设此VMA恰巧恰巧被其他CPU销毁了，会不会有问题呢？
* 请简述页锁PG\_locked的常用使用方法
* 在mm/rmap.c文件中 page\_get\_anon\_vma()函数中，为什么要使用rcu\_read\_lock()什么时候注册RCU回调函数呢
* 在mm/oom\_kill.c的select\_bad\_process()函数中，为什么要使用rcu\_read\_lock()什么时候注册RCU回调函数 呢

## 第5章 中断管理

**5.1 Linux 中断管理机制**

* 发生硬件中断后，ARM处理器做了哪些事情？
* 硬件中断号和Linux 内核的IRQ中断号是如何映射的？
* 一个硬件中断发生后，Linux内核如何响应并处理该中断？
* 为什么说中断上下文不能执行睡眠操作

**5.2 软中断和tasklet**

* 软中断的回调函数执行过程中是否允许响应本趣中断
* 同一类型的软件中断是否允许多个CPU并行执行
* 软件中断上下文件包括那几种情况
* 软中断上下文件和进程上下那个优先级高？为什么呢
* 是否允许一个tasklet在多个CPU上并行执行？

**5.3 workqueue 工作队列**

* workqueue是运行在中断上下文，还是进程上下文？其回调函数允许睡眠呢？
* 旧版本(linux 2.6.25) 的workqueue机制在实际过程中遇到那些问题和挑战
* CMWQ 机制如何动态管理工作线程池的线程呢
* 如果有多个work 挂入一个工作线程中执行，当某个work的回调函数执行了阻塞操作，那么剩下的work该怎么办？

并发和同步

(1) 中断和异常：中断发生后，中断处理程序和被中断的进程之间有可能产生并发访问。

(2) 软中断和tasklet：软件中断或者tasklet随时可能会被调度执行，从而打断当前

正在执行的进程上下文。

(3) 内核抢占：调度器支持可抢占特性，会导致进程和进程之间的并发访问

(4) 多处理器并发执行：多处理上可以同时运行多个进程

概念

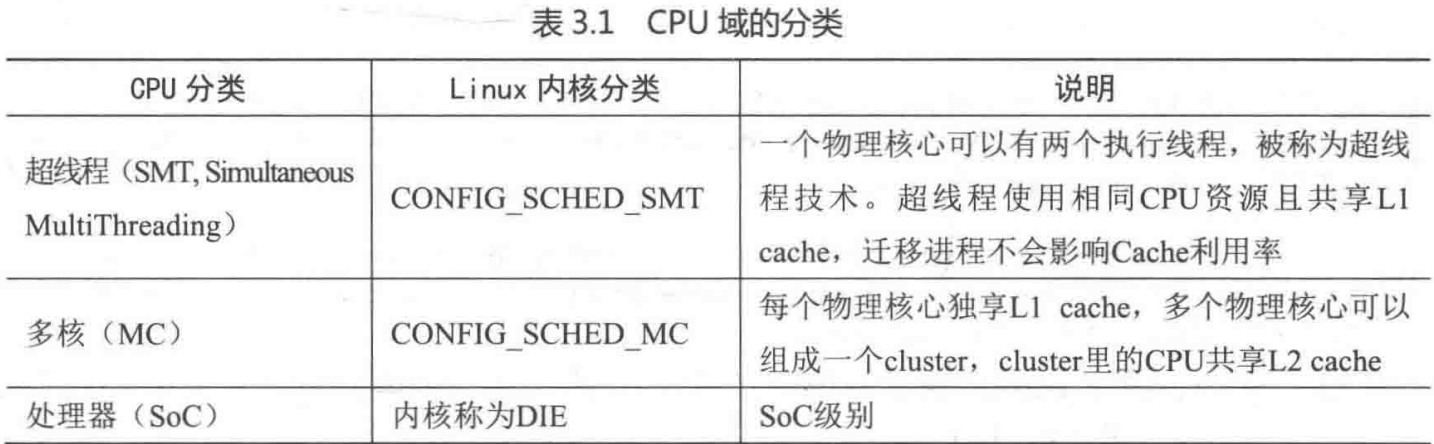
**临界区**是指访问和操作共享数据的代码段，这些资源无法同时被多个执行线程访问。访问临界区的执行或者代码路径为并发源。 为了避免临界区的中并发访问，开发者必须保证访问临界区的原子性，也就是说临界区不能有多个并发源同时执行。整个临界区就像一个不可分割的整体。

在支持SMP对称多处理器的Linux内核里，并发运行不同CPU中内核线程完全有可能同一时刻并发访问共享数据并发访问随时都可能发生。 特别是现在的linux内核早已经支持内核抢占。调度器可能抢占正在运行的进程。

重新调度其他进程来执行。

3.3 负载均衡

3.3.1 CPU域初始化



CPU分类：

超线程(SMT) :CONFIG\_SCHED\_SMT 一个物理核心可以有两个执行线程，被称为超线程技术。超线程使用相CPU资源且共享L1 cache,迁移进程 不会影响cache利用率。

多核 MC CONFIG\_SCHED\_MC 每个物理核心 独享L1 cache ,多个物理核心可以组成一个cluster,cluster里的CPU共享L2 cache

Kernel/cpu.c

* cpu\_possible\_mask :表示系统中有多少个可以行（现在运行或者将来某个时间运行）的CPU核心
* cpu\_online\_mask表示系统中有多少个正在处于运行状态 online 的CPU核心
* cpu\_present\_mask：表示系统中有多少个具备onlinet条件的CPU核心，它们不定处于online 状态，有的cpu 核心可能被热插拔。
* cpu\_active\_mask：表示系统中有多少个活跃的CPU核心。

如何构建 CPU拓扑关系，在系统启动时即开可构建CPU拓扑关系：

[start\_kernel()->reset\_init()->kernel\_init()->kernel\_init\_freeable()->sched\_init\_smp()>init\_sched\_domains()]

[start\_kernel()->setup\_arch()->arm\_dt\_init\_cpu\_maps()

在系统启动时，arm\_dt\_init\_cpu\_maps()函数通过查询DTS来获取CPU核心的数量，然后通过set\_cpu\_possible()函数设置到

cpu\_possible\_bits 位图中，从而设置cpu\_possible\_mask变量

[start\_kernel()->reset\_init()->kernel\_init()->kernel\_init\_freeable()->smp\_prepare\_cpus()

3.3.2 负载均衡

SMP 负载均衡机制从注册软中断开始，每次系统处理调度tick时会检查当前是否需要处理SMP负载均衡

start\_kernel() ->sched\_init()->init\_sched\_fair\_class()

rebalance\_domains()函数是负载均衡的核心入口。

4.4.1 MCS锁机制

假设在一个锁争用激烈的系统中。所有自旋等待锁的线程都在同时一个共享变量上自旋，申请和释放锁都 在同时一个变量上修改。由于cache一致性原理。导致参与自旋的CPU中cacheline变得无效。在锁争用激烈过程中导致严重的CPU高速缓存行颠簸现象（cpu cacheling bouncing)现象，即多个CPU上的cache line反复失效。大大降低系统整体性能。

MCS 算法可以解决自旋锁遇到的问题。

**同步管理与锁机制**

*互斥 （mutual exclusion）*

在软件中保证一个时刻只有一个线程执行读写操作，那么这种在微观上确保一个时刻只允许一个线程执行特定的代码段的问题，称为互斥问题

**临界区**

临界区：一个访问共用资源（例如：共用设备或是共用存储器）的程序片段，而这些共用资源又无法同时被多个线程访问的特性。

**Linux内核编程中并发访问**

 单CPU以及单进程的系统中，会不会有并发访问？

 单CPU多进程的系统中，有哪些因素会产生并发访问？

 **(1) 中断处理程**序可以打断软中断、 tasklet和进程上下文的执行。 **(2)** 软中断和tasklet之间不会并发，但是可以打断进程上下文的执行。(3)在支持抢占的内核中，进程上下文之间会并发。**(4)在不支持抢占**的内核中，进程上下文之间不会产生并发。

**Linux内核编程中并发访问**

 在多CPU和多进程的系统里，有哪些因素会产生并发访问？

**(1)同一类型**的中断处理程序不会并发，但是不同类型的中断有可能送达到不同的CPU上，因此不同类型的中断处理程序可能会存在并发执行。**(2) 同一类型的软中**断会在不同的CPU上并发执行。**(3)同一类型的tas**klet是串行执行的，不会在多个CPU上并发。(4)不同CPU上的进程上下文会并发

**使用锁机制的要点**

**是保护资源或者数据，而不是保护代码**

**(1)**除了当前内核代码路径外，是否还有其他内核代码路径会访问它？例如中断处理程序、工作者（worker） 处理程序、 tasklet处理程序、软中断处理程序等。**(2)**当前内核代码路径访问该资源数据时发生被抢占，被调度执行的进程会不会访问该数据？**(3)**进程会不会睡眠阻塞等待该资源？

**内存屏障**

* 为啥要有内存屏障？
* 如果没有内存屏障行不行？
* 在什么情况下的CPU不需要内存屏障，而在什么情况下的CPU又需要内存屏障？
* 那内存屏障产生的原因究竟是什么？
* ARM这个体系结构又是如何来解决这个问题？

ARM的内存屏障指令

*  顺序执行的处理器
* 乱序执行的处理器
  + 超标量技术
  + 乱序发射
  + 乱序执行

什么是同步和异步？ ARM芯片手册里关于同步和异步的概念

**中断的类型**

中断：系统计划外发生的事情，类似突发事件。属于异步。

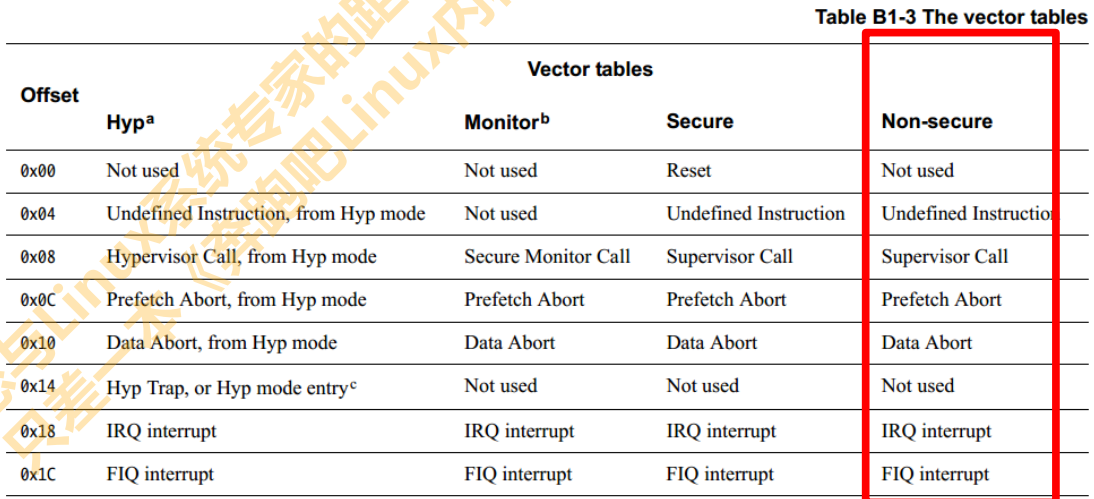
异常：通常是执行到某条非法指令，是计划内的事情，有点类似同步。

自陷：程序自愿请求陷入内核态。比如系统调用。

**ARM v7上中断处理**

ARM v7上的中断向量表

 啥是中断向量表,ARM v7上的中断向量表



ARM v7中断发生硬件做了那些？

* 当硬件异常发生了，这时候处理器要识别出是那个模式的异常。
* 把当前CPSR寄存器的值存放在SPSR里。
* 异常模式的lr寄存器保存着异常发生时候那个地址，这个就是异常返回的链接地址
* 硬件会修改当前模式下的CPSR寄存器，然后进入到对应的异常模式里。
* 然后PC指针指向异常向量表对应的表项里
* 然后执行跳转。