

《ARM Linux内核源码剖析》

图书基本信息

书名：《ARM Linux内核源码剖析》

13位ISBN编号：9787115359105

出版时间：2014-7-1

作者：尹锡训

页数：518

译者：崔范松

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：www.tushu111.com

《ARM Linux内核源码剖析》

内容概要

内核源代码构建系统

ARM处理器结构

构建高效分析环境

汇编级启动过程

内核分析常用API、ARM指令、GAS关键词

发生中断到调用处理器的详细过程

本书是多位作者在3年Linux内核分析经验和庞大资料基础上写成的，收录了其他同类书未曾讲解的内容并进行逐行分析，一扫当前市场中其他理论书带给读者的郁闷。书中详细的代码分析与大量插图能够使读者对Linux内核及ARM获得正确认识，自然而然习得如何有效分析定期发布的Linux内核。

本书适合想从Linux内核启动开始透彻分析全部启动过程的读者，因Linux代码量庞大而束手无策的人、想要了解Linux实际运行过程的人、渴求OS实操理论的人，本书必将成为他们不可或缺的参考书。

《ARM Linux内核源码剖析》

作者简介

作者简介：

尹锡训

mindwave@nate.com

所在公司的主营业务是在Linux、安卓系统上制作无线终端，担任工程师已有3年。一直用C语言、Python、Java开发各种产品。最近把对未来的期待、展望与精力集中到Linux内核以及创业上，并不断为之努力。

崔范松

吉林人，毕业于长春工业大学法学专业。大量接触并翻译过各类计算机图书及相关资料，并从事过游戏策划及软件测试工作。喜欢散步、旅游等户外运动，梦想成为一名自由职业者。

书籍目录

第一部分 ARM Linux内核——分析内核前需要做的准备

第1章 内核介绍及2.6版和3.2版之间的差异	2
1.1 内核的诞生、作用以及内部结构	2
1.1.1 Linus创造的Linux	2
1.1.2 由多种子系统集成运行的单内核	3
1.1.3 全世界最著名的通用操作系统	5
1.2 内核2.6版和3.2版之间的差异	5
第2章 内核构建系统	8
2.1 内核初始化	8
2.2 内核配置	9
2.3 内核构建	11
2.4 内核安装	17
第3章 了解ARM处理器	19
3.1 处理器概要和特征	19
3.2 处理器架构与核心	19
3.3 处理器命名规则	21
3.4 处理器内部结构	21
3.5 处理器模式和寄存器	23
3.6 处理器异常	25
3.7 硬件扩展功能	26
3.7.1 缓存	26
3.7.2 内存管理装置	26
3.7.3 协处理器	26
第4章 构建分析环境	28
4.1 下载并安装Linux源内核	28
4.1.1 下载源内核	28
4.1.2 安装源内核	30
4.2 安装ctags+cscope	31
4.2.1 用ctags制作源代码标签	31
4.2.2 制作cscope标签数据库	33
4.3 vim插件下载及环境设置	34
4.3.1 下载vim插件	34
4.3.2 vim+plugin的环境结构	37
4.3.3 vim环境设置	38
4.4 查看源码分析环境工具	40
第二部分 内核的启动——start_kernel调用方法	
第5章 准备解压内核	48
5.1 进入启动加载后结束首个启动——start 标签	49
5.2 BSS系统域初始化——not_relocated 标签	50
5.3 激活缓存——cache_on 标签	53
5.4 页目录项初始化——__setup_mmu 标签	56
5.5 指令缓存激活及缓存策略适用——__common_mmu_cache_on 标签	58
第6章 从压缩的内核zImage 还原内核映像	60
6.1 解压内核并避免覆写——wont_overwrite、decompress_kernel标签	61
6.2 调用已解压内核——call_kernel标签	62
6.3 缓存清理及清除——cache_clean_flush 标签	62
6.4 缓存禁用——cache_off 标签	64

第7章	调用start_kernel()	65
7.1	初始化指向——stext标签	65
7.2	处理器信息搜寻——__lookup_processor_type	69
7.2.1	__lookup_processor_type标签	69
7.2.2	__proc_info_begin和__proc_info_end中保存的信息	71
7.2.3	在MMU禁用状态下将虚拟地址转换为物理地址	73
7.2.4	查找proc_info_list结构体并比较处理器信息	74
7.3	搜寻我的机型——__lookup_machine_type	75
7.3.1	__lookup_machine_type标签	75
7.3.2	保存在__arch_info_begin和__arch_info_end中的machine_desc信息及访问路径	76
7.3.3	查找machine_desc结构体并比较机器信息	77
7.4	源自启动加载项的atags——__vet_atags标签	78
7.5	对虚拟内存进行基础创建——__create_page_tables标签	81
7.6	设置核心(core)——v6_setup标签	85
7.7	打开MMU并使用虚拟地址——__enable_mmu/__turn_mmu_on标签	86
7.8	跳转至start_kernel——__mmap_switched标签	90
第三部分 内核的执行——内核的起始与结束位置		
第8章	start_setup_processor_id()~lock_kernel()	94
8.1	smp_setup_processor_id()、lockdep_init()、debug_objects_early_init()	95
8.1.1	smp_setup_processor_id()	95
8.1.2	lockdep_init()	95
8.1.3	debug_objects_early_init()	96
8.2	栈溢出感应——__boot_init_stack_canary	98
8.3	初始化提供进程集成方法的cgroup——__cgroup_init_early()	98
8.3.1	cgroupfs_root和cgroup的关联初始化——init_cgroup_root()	102
8.3.2	初始化子系统——cgroup_init_subsys()	103
8.4	禁用IRQ	104
8.5	early_boot_irqs_off()、early_init_irq_lock_class()	104
8.6	大内核锁——lock_kernel()	106
第9章	注册针对时钟事件的处理器	111
9.1	函数的声明和定义——tick_init()	111
9.2	注册处理事件的处理器——__clockevents_register_notifier()	113
9.2.1	为clockevents_lock添加自旋锁	114
9.2.2	clockevents_chain生成原理	115
9.2.3	在clockevents_chain中注册tick_notifier的方法	116
9.2.4	对clockevents_lock解除自旋锁的原理	117
第10章	在CPU位图中注册当前运行CPU/初始化HIGHMEM管理	119
10.1	在包含热插拔信息的位图上添加执行init_task的CPU——boot_cpu_init()	119
10.2	管理高端内存——page_address_init()	121
第11章	整体指向——setup_arch	123
第12章	unwind_init()~early_trap_init()	126
12.1	栈回溯——unwind_init()	126
12.2	求出包含机器信息的machine_desc结构体——setup_machine()	126
12.3	处理ATAG信息——setup_arch()	127
12.4	处理启动参数——parse_cmdline()	129
12.5	构建源代码树——request_standard_resources()	131
12.6	初始化cpu possible位图——smp_init_cpus()	136
12.7	用栈指定各ARM异常模式——cpu_init()	137
12.8	初始化以处理异常——early_trap_init()	138

12.9	查看中断处理器函数	143	
12.9.1	调用IRQ处理器——asm_do_IRQ()	147	
12.9.2	返回中断之前——ret_to_user标签	147	
第13章	设置处理器——setup_processor()	150	
13.1	查看setup_processor()结构	150	
13.2	查找CPU ID——read_cpuid_id()	151	
13.3	查找处理器信息——lookup_processor_type()	153	
13.4	查找处理器结构信息——cpu_architecture()	153	
13.5	查找处理器缓存类型_cacheid_init()	156	
13.6	调用处理器初始化函数——cpu_proc_init()	160	
第14章	准备内存分页——paging_init()	163	
14.1	查看paging_init()的整体结构	163	
14.2	设置内存类型表——build_mem_type_table()	165	
14.3	检验内存信息——sanity_check_meminfo()	166	
14.4	准备页表——prepare_page_table()	168	
14.4.1	prepare_page_table()	168	
14.4.2	Linux的分页结构	170	
14.4.3	求出页目录项	170	
14.4.4	pmd_clear()	172	
14.5	设备区域映射准备——devicemaps_init()	174	
14.6	准备使用高端内存——kmap_init()	177	
14.7	初始化零页	178	
14.7.1	分配内存——__alloc_bootmem_nopanic()	179	
14.7.2	在指定节点使用fallback分配内存——alloc_bootmem_core	180	
14.7.3	将虚拟地址变换为page结构体——virt_to_page	182	
14.8	保持数据缓存一致性——flush_dcachepage()	182	
第15章	在启动时初始化内存分配器	184	
15.1	bootmem函数流和数据结构	185	
15.2	查看bootmem_init()结构	188	
15.3	查找虚拟内存盘位置——check_initrd()	189	
15.4	将节点的BANK信息反映到页目录——bootmem_init_node()	191	
15.4.1	map_memory_bank()	192	
15.4.2	bootmem_bootmap_pages()	195	
15.4.3	find_bootmap_pfn()	196	
15.4.4	node_set_online()	197	
15.4.5	NODE_DATA宏	198	
15.4.6	init_bootmem_node()	200	
15.4.7	free_bootmem_node()	202	
15.4.8	reserve_bootmem_node()	202	
15.5	排除0号节点——reserve_node_zero()	203	
15.6	排除虚拟内存盘节点——bootmem_reserve_initrd()	204	
15.7	设置为无可页——bootmem_free_node()	205	
15.8	初始化free_area区域	207	
15.8.1	free_area结构体	207	
15.8.2	free_area_init_node()	208	
15.8.3	free_area_init_core()	209	
15.8.4	init_currently_empty_zone()	211	
15.8.5	memmap_init()	212	
第16章	mm_init_owner()~preempt_disable()	217	

16.1	设置内存所有者——mm_init_owner()	217
16.2	保存命令行——setup_command_line()	218
16.3	初始化per-cpu数据——setup_per_cpu_areas()	219
16.4	求CPU个数——setup_nr_cpu_ids()	221
16.5	注册SMP上的启动进程——smp_prepare_boot_cpu()	222
16.6	初始化数据结构以使用调度程序——sched_init()	224
16.6.1	为集合调度中使用的task_group的sched_entity结构体和runqueue结构体分配内存	224
16.6.2	初始化root_domain、rt_bandwidth、task_group相关数据结构	227
16.6.3	初始化系统上所有可用CPU的就绪队列	229
16.6.4	初始化当前任务的调度相关值与注册针对负载均衡的中断处理器	230
16.7	允许内核抢占和阻止抢占——preempt_enable()/preempt_disable()	231
第17章 构建借用内存的后台 233		
17.1	在build_all_zonelists()中操作的一些数据结构	233
17.2	查看build_all_zonelists()结构	235
17.3	决定zone的列表方式——set_zonelist_order()	236
17.4	构建备用列表和备用位图——__build_all_zonelists()	238
17.4.1	build_zonelists()	239
17.4.2	build_zonelist_in_node_order()	241
17.4.3	build_zonelists_in_zone_order()	243
17.4.4	build_thisnode_zonelists()	244
17.4.5	build_zonelists_cache()	244
17.5	输出备用列表信息——mminit_verify_zonelist()	246
17.6	指定处理页分配请求的节点——cpuset_init_current_mems_allowed()	246
17.7	求空页数——nr_free_pagecache_pages()	247
17.8	页移动性	250
第18章 page_alloc_init()~pidhash_init() 253		
18.1	处理用于热插拔CPU的页——page_alloc_init()	253
18.2	处理console参数——parse_early_param()	255
18.3	处理特殊参数——parse_args()	257
18.4	确认中断处理是否激活——irqs_disable()	260
18.5	内核异常列表定义——sort_main_extable()	261
18.6	初始化RCU机制——rcu_init()	263
18.7	准备使用IRQ——early_irq_init()	266
18.8	初始化中断——init_IRQ()	269
18.9	构建迅速搜寻进程信息的结构——pidhash_init()	271
第19章 init_timers()~page_cgroup_init() 273		
19.1	初始化计时器——init_timers()	274
19.1.1	timers_cpu_notify()	275
19.1.2	register_cpu_notifier()	275
19.1.3	open_softirq()	276
19.2	初始化高分辨率计时器——hrtimers_init()	277
19.3	注册softirq的回调函数——softirq_init()	280
19.4	设置xtime——timekeeping_init()	282
19.5	初始化硬件计时器——time_init()	285
19.6	初始化时钟时间——sched_clock_init()	286
19.7	激活CPU的中断处理——local_irq_enable()	288
19.8	检测用作根文件系统的init虚拟内存盘	288
19.9	初始化以分配动态内存——vmalloc_init()	289
19.10	预先初始化目录项和索引节点缓存——vfs_caches_init_early()	290

19.11	初始化cpuset子系统——cpuset_init_early()	293
19.12	初始化内存子系统——page_cgroup_init()	294
第20章	终止bootmem分配器并替换为伙伴系统	297
20.1	mem_init()函数的调用关系及其与数据结构的相互关系	297
20.2	查看mem_init()结构	298
20.3	记录到不存在的内存位图——free_unused_mmap_node()	300
20.4	移交至普通空白页伙伴系统——free_all_bootmem_node()	301
20.4.1	register_page_bootmem_info_node()	301
20.4.2	free_all_bootmem_core()	303
20.4.3	__free_pages_bootmem()	305
20.4.4	__free_pages()	308
20.4.5	free_hot_cold_page()	308
20.4.6	__free_pages_ok()	309
20.5	移交到高端内存空白页伙伴系统——free_area()	314
第21章	初始化以支持CPU热插拔	315
21.1	初始化cpu_hotplug成员变量——cpu_hotplug_init()	315
21.2	CPU的联机 脱机转换处理	316
第22章	激活slab内存分配器——kmem_cache_init()	318
22.1	slab分配器的概念及结构体	318
22.2	slab分配器的重要结构体——kmem_cache和kmem_list3	319
22.3	查看kmem_cache_init()结构	322
22.4	初始化initkmem_list3[]、cache_cache、nodelist[]	326
22.5	连接kmem_list3数组并决定cache压缩时间——set_up_list3s()	328
22.6	求出用于cache扩展/压缩的页顺序——cache_estimate()	329
22.7	malloc_sizes和cache_names	332
22.8	生成cache——kmem_cache_create()	334
22.8.1	kmem_cache_zalloc()	336
22.8.2	calculate_slab_order()	337
22.8.3	setup_cpu_cache()	337
22.8.4	enable_cpucache()	339
22.9	生成arraycache_init,kmem_list3 cache	340
22.10	用kmalloc()函数分配的内存替代静态分配的内存	342
第23章	kmem_trace_init()~security_init()	344
23.1	生成ID allocator缓存——idr_init_cache()	345
23.2	初始化pageset——setup_per_cpu_pageset()	345
23.3	指定交叉节点——numa_policy_init()	350
23.4	结束计时器初始化——late_time_init()	353
23.5	测定BogoMIPS——calibrate_delay()	353
23.6	制作位图以分配进程识别符 (ID) ——pidmap_init()	354
23.7	初始化优先树的数据结构——prio_tree_init()	356
23.8	生成anon_vma slab缓存——anon_vma_init()	356
23.9	为对象的每个用户赋予资格——cred_init()	357
23.10	初始化数据结构以使用fork()函数——fork_init()	358
23.11	初始化生成进程的缓存——proc_caches_init()	359
23.12	初始化缓冲缓存——buffer_init()	361
23.13	准备密钥——key_init()	364
第24章	初始化VFS中使用的多种缓存——vfs_cache_init()	367
第25章	radix_tree_init()~ftrace_init()	382
25.1	基数树相关数据结构初始化——radix_tree_init()	383

25.2	准备使用信号——signals_init()	383
25.3	注册并挂载proc文件系统——proc_root_init()	384
25.4	注册未能初始化的子系统——cgroup_init()	385
25.5	重置top_cpuset并注册cpuset文件系统——cpuset_init()	386
25.6	初始化任务统计信息接口——delayacct_init()	387
25.7	为管理延迟信息做准备——delayacct_init()	388
25.7.1	延迟审计	388
25.7.2	delayacct_init	389
25.7.3	task_delay_info结构体和delayacct_tsk_init()	390
25.8	检查写缓冲一致性——check_bugs()	392
第26章	同步内存与后备存储——page write back	394
26.1	页回写机制	394
26.2	激活页回写——pdflush_init()	395
26.3	pdflush线程	397
26.4	指定页回写函数	398
26.5	周期性页回写和强制性页回写回调函数调用方法	399
26.5.1	周期性页回写函数——wb_kupdate()	399
26.5.2	强制性页回写函数——background_writeout()	401
26.6	初始化周期性页回写	403
第27章	查看启动内核的最终函数结构——rest_init()	405
第28章	生成执行函数的内核线程——kernel_thread()	407
28.1	查看kernel_thread()结构	407
28.2	生成处理器的网关——do_fork()	408
28.3	复制父进程——copy_process()	411
第29章	唤醒新生成的任务	419
29.1	查看wake_up_new_task结构	419
29.2	获取任务的就绪队列——task_rq_lock()	421
29.3	改善任务的优先顺序——effective_prio()	422
第30章	准备使用内核	426
30.1	将当前进程转移到其他CPU——sched_init_smp()	427
30.2	结束系统整体初始化——do_basic_setup()	429
30.2.1	生成执行rcu_sched_grace_period()的线程——rcu_init_sched()	430
30.2.2	生成events工作队列——init_workqueues	430
30.2.3	初始化cpuset子系统的top_cpuset——cpuset_init_smp()	437
30.2.4	生成khelper工作队列——usermodehelper_init()	437
30.2.5	初始化Linux的设备模型——driver_init()	439
30.2.6	在proc文件系统注册irq信息——init_irq_proc()	446
30.2.7	调用内核未知子系统——do_initcalls()	448
30.3	为初始化之后的操作做准备——init_post()	451
第31章	内核线程守护进程	453
31.1	内核线程守护进程——kthreadd()	453
31.2	忽略信号——ignore_signals()	456
31.3	设置nice值——set_user_nice()	458
31.4	搜索执行任务的CPU——set_cpus_allowed_ptr()	463
31.5	搜索包含列表的实际结构体位置——list_entry()	464
31.6	生成内核线程——create_kthread()	466
第32章	find_task_by_pid_ns()~cpu_idle()	469
32.1	用PID搜索任务——find_task_by_pid_ns()	470
32.2	解除BKL——unlock_kernel()	472

32.3	将调度类变更为idle——init_idle_bootup_task()	473
32.4	RCU机制激活完成通知——rcu_scheduler_starting()	473
32.5	激活内核抢占——preempt_enable_no_resched()	473
32.6	执行进程调度表——schedule()	474
32.7	Linux启动万里长征的终点——cpu_idle()	476
附 录		
附录A	汇编语言、gas关键词总结	480
附录B	内核分析常见API	485
附录C	浅谈ext2文件系统	487
附录D	Linux线程模型	497
附录E	链接器脚本文件结构	500
后记		510
索引		513

《ARM Linux内核源码剖析》

精彩短评

- 1、译者非IT工作者，各种常识性错误，简直了！！！！书本身内容不错，讲解的很细致！！值得ARM Linux内核的玩家一看！
- 2、只是推荐，翻译一般，内核书籍里非常细致的代码分析，一般推荐先熟读一本内核原理的书籍后再阅读此书会比较有效果。

《ARM Linux内核源码剖析》

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:www.tushu111.com