

코드로 알아보는
ARM 리눅스 커널

코딩 알아보는

ARM 리눅스 커널

초판 1쇄 발행 2012년 8월 22일

지은이 노서영, 윤석훈, 강진성, 송원준, 이윤재, 임윤재

펴낸이 장성두 | 책임편집 안주연

본문디자인 디자인콤마 | 표지디자인 미디어픽스

주소 경기도 파주시 문발동 파주출판도시 530-1 뮤즈빌딩 403호

전화 070-8201-9010 | 팩스 02-6280-0405

홈페이지 www.jpub.kr | 펴낸곳 제이펍

출판신고 2009년 11월 10일 제406-2009-000087호

용지 신승지류유통 | 인쇄 해외정판 | 제본 동호문화

ISBN 978-89-94506-49-6 (93560)

값 35,000원

- ※ 이 책은 저작권법에 따라 보호를 받는 저작물이므로 무단전재와 무단복제를 금지하며, 이 책 내용의 전부 또는 일부를 이용하려면 반드시 저작권자와 제이펍의 서면동의를 받아야 합니다.
- ※ 잘못된 책은 구입하신 서점에서 바꾸어 드립니다.

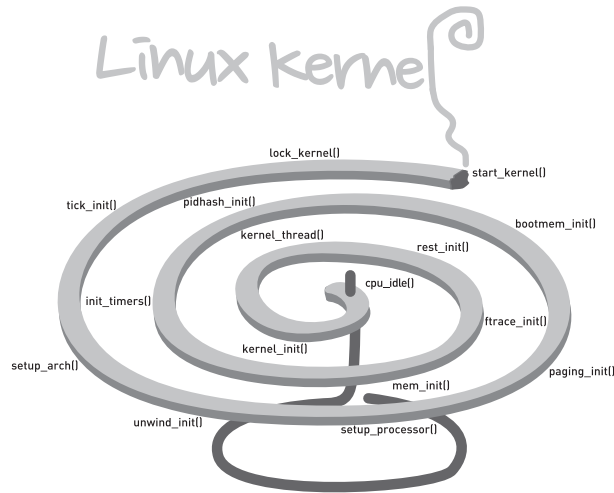
제이펍은 독자 여러분의 책에 관한 아이디어와 원고 투고를 기다리고 있습니다. 책으로 펴내고자 하는 아이디어나 원고가 있으신 분께서는 책에 대한 간단한 개요와 차례, 구성과 저(역)자 약력 등을 메일로 보내주세요.

(보내실 곳: jeipub@gmail.com)

코드로 알아보는

ARM 리눅스 커널

2년간의 코드 분석, 1년간의 집필로 집대성한 최초의 ARM 리눅스 커널 분석서!



노서영, 윤석훈, 강진성, 송원준, 이윤재, 임윤재 지음

감수자의 글 XVII
머리말 XIX
질펀 후기 XXI

PART I ARM 리눅스 커널 - 커널 분석을 위해 어떤 준비가 필요할까?

Chapter 1 커널에 대한 소개 그리고 2.6과 3.2의 차이 002

- 1.1 커널의 탄생과 역할 그리고 내부 구조 002
 - 1.1.1 리누스에 의해 탄생한 리눅스 002
 - 1.1.2 다양한 서브시스템이 모여 동작하는 모노리딕 커널 003
 - 1.1.3 전 세계에서 가장 유명한 범용 운영체제 005
- 1.2 커널 2.6과 3.2의 차이 006

Chapter 2 커널 빌딩 시스템 008

- 2.1 커널 초기화 008
- 2.2 커널 설정 009
- 2.3 커널 빌딩 011
- 2.4 커널 설치 019

Chapter 3 ARM 프로세서 알아보기 021

- 3.1 프로세서 개요와 특징 021
- 3.2 프로세서 아키텍처와 코어 022
- 3.3 프로세서 명명법 023
- 3.4 프로세서 내부 구조 024
- 3.5 프로세서 모드와 레지스터 026

| | | |
|-----------------------------|-----------------------|------------|
| 3.6 | 프로세서 익셉션 | 028 |
| 3.7 | 하드웨어 확장 기능 | 030 |
| 3.7.1 | 캐시 | 030 |
| 3.7.2 | 메모리 관리 장치 | 031 |
| 3.7.3 | 코프로세서 | 031 |
| Chapter 4 분석 환경 구축하기 | | 032 |
| 4.1 | 리눅스 커널 소스 다운로드하여 설치하기 | 032 |
| 4.1.1 | 커널 소스 다운로드하기 | 032 |
| 4.1.2 | 소스 설치하기 | 034 |
| 4.2 | ctags + cscope 설치하기 | 035 |
| 4.2.1 | ctags로 소스 코드 태그 만들기 | 035 |
| 4.2.2 | cscope 태그 데이터베이스 만들기 | 037 |
| 4.3 | vim 플러그인 다운로드 및 환경설정 | 038 |
| 4.3.1 | vim 플러그인 다운로드하기 | 039 |
| 4.3.2 | vim + plugin 환경구성 | 043 |
| 4.3.3 | vim 환경설정하기 | 044 |
| 4.4 | 소스 분석 환경 툴 둘러보기 | 047 |

PART II

커널의 시작 – start_kernel은 어떻게 호출될까?

| | | |
|--|---|------------|
| Chapter 5 커널 압축 해제 준비하기 | | 056 |
| 5.1 | 부트로더에 이어 첫 스타트 끊기 – start 레이블 | 057 |
| 5.2 | BSS 영역 초기화하기 – not_relocated 레이블 | 059 |
| 5.3 | 캐시 활성화하기 – cache_on 레이블 | 062 |
| 5.4 | 페이지 디렉터리 엔트리 초기화하기 – __setup_mmu 레이블 | 065 |
| 5.5 | I-Cache 활성화 및 캐시 정책 적용하기 – __common_mmu_cache_on 레이블 | 069 |
| Chapter 6 압축된 커널 이미지인 zimage로부터 커널 이미지 복원하기 | | 070 |
| 6.1 | 덮어쓰지 않게 커널 압축 풀기 – wont_overwrite, decompress_kernel 레이블 | 071 |

| | | |
|--------------------------------------|---|------------|
| 6.2 | 압축 해제된 커널 호출하기 – call_kernel 레이블 | 072 |
| 6.3 | 캐시클린 및 플러시 – cache_clean_flush 레이블 | 073 |
| 6.4 | 캐시 비활성화하기 – cache_off 레이블 | 075 |
| Chapter 7 start_kernel() 호출하기 | | 076 |
| 7.1 | 초기화 조망하기 – stext 레이블 | 076 |
| 7.2 | 프로세서 정보 찾기 – __lookup_processor_type | 081 |
| 7.2.1 | __lookup_processor_type 레이블 | 081 |
| 7.2.2 | __proc_info_begin과 __proc_info_end에 저장된 정보 | 083 |
| 7.2.3 | MMU 비활성화 상태에서 가상 주소를 물리 주소로 변경하기 | 085 |
| 7.2.4 | proc_info_list 구조체를 찾아 프로세서 정보 비교하기 | 087 |
| 7.3 | 내 머신 타입 찾기 – __lookup_machine_type | 088 |
| 7.3.1 | __lookup_machine_type 레이블 | 088 |
| 7.3.2 | __arch_info_begin과 __arch_info_end에 저장된 machine_desc 정보와 접근 방법 | 090 |
| 7.3.3 | machine_desc 구조체를 찾아 머신 정보 비교하기 | 091 |
| 7.4 | 부트로더에서 온 atags – __vet_atags 레이블 | 092 |
| 7.5 | 가상 메모리를 사용하기 위해 기초 공사하기 – __create_page_table 레이블 | 095 |
| 7.6 | 코어를 설정하자 – v6_setup 레이블 | 101 |
| 7.7 | MMU를 켜고 가상 주소 사용하기 – __enable_mmu / __turn_mmu_on 레이블 | 102 |
| 7.8 | start_kernel로 점프 – __mmap_switched 레이블 | 107 |

PART III

커널의 실행 – 커널의 시작과 끝은 어디인가?

| | | |
|---|--|------------|
| Chapter 8 smp_setup_processor_id() ~ lock_kernel() | | 112 |
| 8.1 | smp_setup_processor_id(), lockdep_init(), debug_objects_early_init() | 113 |
| 8.1.1 | smp_setup_processor_id() | 113 |
| 8.1.2 | lockdep_init() | 113 |
| 8.1.3 | debug_objects_early_init() | 115 |
| 8.2 | 스택 오버플로우 감지하기 – boot_init_stack_canary() | 117 |

| | | |
|---|--|-----|
| 8.3 | 프로세스를 그룹화하는 방법을 제공하는 cgroup 초기화하기 - cgroup_init_early() | 118 |
| 8.3.1 | cgroupfs_root와 cgroup의 관계 초기화하기 - init_cgroup_root() | 123 |
| 8.3.2 | 서브시스템 초기화하기 - cgroup_init_subsys() | 123 |
| 8.4 | IRQ 비활성화하기 - local_irq_disable() | 125 |
| 8.5 | early_boot_irqs_off(), early_init_irq_lock_class() | 125 |
| 8.6 | 빅 커널 락 - lock_kernel() | 127 |
| Chapter 9 클럭 이벤트에 대한 핸들러 등록하기 | | 134 |
| 9.1 | 함수의 선언과 정의 - tick_init() | 134 |
| 9.2 | 이벤트 처리 핸들러 등록하기 - clockevents_register_notifier() | 136 |
| 9.2.1 | clockevents_lock에 대한 스핀 락 걸기 | 137 |
| 9.2.2 | clockevents_chain이 생성되는 원리 | 139 |
| 9.2.3 | clockevents_chain에 tick_notifier를 등록하는 방법 | 140 |
| 9.2.4 | clockevents_lock에 대한 스핀 락을 해제하는 원리 | 142 |
| Chapter 10 CPU 비트맵에 수행 중인 CPU 등록과 HIGHMEM 관리를 위한 초기화 | | 144 |
| 10.1 | 핫플러그 정보를 담고 있는 비트맵에 init_task를 수행하는 CPU 추가하기 - boot_cpu_init() | 144 |
| 10.2 | 하이 메모리 관리하기 - page_address_init() | 146 |
| Chapter 11 전체 조망하기 - setup_arch() | | 149 |
| Chapter 12 unwind_init() ~ early_trap_init() | | 152 |
| 12.1 | 스택 역추적하기 - unwind_init() | 152 |
| 12.2 | 머신 정보를 담고 있는 machine_desc 구조체 구하기 - setup_machine() | 153 |
| 12.3 | ATAG 정보 처리하기 - setup_arch() | 154 |
| 12.4 | 부팅 파라미터 처리하기 - parse_cmdline() | 156 |
| 12.5 | 리소스 트리 구성하기 - request_standard_resources() | 158 |
| 12.6 | cpu possible 비트맵 초기화하기 - smp_init_cpus() | 164 |
| 12.7 | ARM 예외 모드마다 스택 지정해주기 - cpu_init() | 165 |
| 12.8 | 익셉션 핸들링을 위해 초기화하기 - early_trap_init() | 167 |

| | | |
|--|---|-----|
| 12.9 | 인터럽트 핸들러 함수 살펴보기 | 174 |
| 12.9.1 | IRQ 핸들러 호출하기 - asm_do_IRQ() | 177 |
| 12.9.2 | 인터럽트 이전으로 돌아가기 - ret_to_user 레이블 | 179 |
| Chapter 13 프로세서 셋업하기 - setup_processor() | | 181 |
| 13.1 | setup_processor() 구조 알아보기 | 181 |
| 13.2 | CPU ID 찾기 - read_cpuid_id() | 183 |
| 13.3 | 프로세서 정보 찾기 - lookup_processor_type() | 184 |
| 13.4 | 프로세서 아키텍처 정보 찾기 - cpu_architecture() | 185 |
| 13.5 | 프로세서 캐시 타입 찾기 - cacheid_init() | 189 |
| 13.6 | 프로세서 초기화 함수 호출하기 - cpu_proc_init() | 193 |
| Chapter 14 메모리 페이징 준비하기 - paging_init() | | 196 |
| 14.1 | paging_init() 전체 구조 살펴보기 | 196 |
| 14.2 | 메모리 타입 테이블 설정해주기 - build_mem_type_table() | 198 |
| 14.3 | 메모리 정보 점검하기 - sanity_check_meminfo() | 201 |
| 14.4 | 페이지 테이블 준비하기 - prepare_page_table() | 203 |
| 14.4.1 | prepare_page_table() | 203 |
| 14.4.2 | 리눅스의 페이징 구조 | 205 |
| 14.4.3 | 페이지 디렉터리 엔트리 구하기 | 206 |
| 14.4.4 | pmd_clear() | 208 |
| 14.5 | 디바이스 영역 매핑 준비하기 - devicemaps_init() | 210 |
| 14.6 | 하이 메모리 사용 준비하기 - kmap_init() | 215 |
| 14.7 | 제로 페이지 초기화하기 | 216 |
| 14.7.1 | 메모리 할당하기 - __alloc_bootmem_nopanic() | 217 |
| 14.7.2 | 지정된 노드에서 fallback을 사용하여 메모리 할당받기 - alloc_bootmem_core() | 218 |
| 14.7.3 | 가상 주소를 page 구조체로 변환하기 - virt_to_page() | 220 |
| 14.8 | D-Cache의 일관성 유지하기 - flush_dcache_page() | 221 |
| Chapter 15 부트 타임 시 메모리 할당자 초기화하기 - bootmem_init() | | 223 |
| 15.1 | bootmem의 함수 흐름과 자료구조들 | 224 |
| 15.2 | bootmem_init() 구조 알아보기 | 229 |

| | | |
|---|---|-----|
| 15.3 | 램디스크 위치 찾기 – check_initrd() | 230 |
| 15.4 | 노드의 बैं크 정보를 페이지 디렉터리에 반영하기 – bootmem_init_node() | 231 |
| 15.4.1 | map_memory_bank() | 233 |
| 15.4.2 | bootmem_bootmap_pages() | 236 |
| 15.4.3 | find_bootmap_pfn() | 238 |
| 15.4.4 | node_set_online() | 239 |
| 15.4.5 | NODE_DATA() 매크로 | 240 |
| 15.4.6 | init_bootmem_node() | 242 |
| 15.4.7 | free_bootmem_node() | 244 |
| 15.4.8 | reserve_bootmem_node() | 245 |
| 15.5 | 0번 노드 제외시키기 – reserve_node_zero() | 246 |
| 15.6 | 램디스크 노드 제외시키기 – bootmem_reserve_initrd() | 248 |
| 15.7 | 가용 페이지 없다고 설정하기 – bootmem_free_node() | 248 |
| 15.8 | free_area 영역 초기화 | 251 |
| 15.8.1 | free_area 구조체 | 251 |
| 15.8.2 | free_area_init_node() | 253 |
| 15.8.3 | free_area_init_core() | 254 |
| 15.8.4 | init_currently_empty_zone() | 256 |
| 15.8.5 | memmap_init() | 257 |
| Chapter 16 mm_init_owner() ~ preempt_disable() | | 264 |
| 16.1 | 메모리 소유자 설정하기 – mm_init_owner() | 264 |
| 16.2 | 명령어 라인 저장해두기 – setup_command_line() | 265 |
| 16.3 | per-cpu 데이터 초기화하기 – setup_per_cpu_areas() | 266 |
| 16.4 | CPU 개수 구하기 – setup_nr_cpu_ids() | 269 |
| 16.5 | SMP 상의 부팅 프로세스 등록하기 – smp_prepare_boot_cpu() | 270 |
| 16.6 | 스케줄러를 사용하기 위해 자료구조 초기화하기 – sched_init() | 272 |
| 16.6.1 | 그룹 스케줄링 시 사용되는 task_group의 sched_entity 구조체와 runqueue 구조체를 위한 메모리 할당 | 273 |
| 16.6.2 | root_domain, rt_bandwidth, task_group 관련 자료구조 초기화 | 276 |
| 16.6.3 | 시스템의 모든 possible cpu의 런큐 초기화 | 278 |
| 16.6.4 | 현재 태스크의 스케줄링 관련 값 초기화 및 로드밸런싱을 위한 인터럽트 핸들러 등록 | 280 |
| 16.7 | 커널 선점 허용하기와 선점 막기 – preempt_enable()/preempt_disable() | 281 |

| | |
|--|----------------|
| Chapter 17 빌려줄 후원자 구성하기 | 283 |
| 17.1 build_all_zonelists()에서 다루는 자료구조들 | 283 |
| 17.2 build_all_zonelists() 구조 알아보기 | 285 |
| 17.3 존의 리스트 방식 결정하기 – set_zonelist_order() | 287 |
| 17.4 폴백 리스트와 폴백 비트맵 구성하기 – __build_all_zonelists() | 290 |
| 17.4.1 build_zonelists() | 291 |
| 17.4.2 build_zonelists_in_node_order() | 293 |
| 17.4.3 build_zonelists_in_zone_order() | 296 |
| 17.4.4 build_thisnode_zonelists() | 297 |
| 17.4.5 build_zonelist_cache() | 297 |
| 17.5 폴백 리스트 정보 출력하기 – mminit_verify_zonelist() | 299 |
| 17.6 페이지 할당 요청을 처리할 노드 정하기 – cpuset_init_current_mems_allowed() | 300 |
| 17.7 프리 페이지 개수 구하기 – nr_free_pagecache_pages() | 300 |
| 17.8 페이지 모빌리티 | 304 |
| Chapter 18 page_alloc_init() ~ pidhash_init() | 308 |
| 18.1 핫플러그 CPU를 위한 page 처리하기 – page_alloc_init() | 309 |
| 18.2 console 파라미터 처리하기 – parse_early_param() | 311 |
| 18.3 특별한 파라미터 처리하기 – parse_args() | 313 |
| 18.4 인터럽트 활성화 여부 확인하기 – irq_disabled() | 317 |
| 18.5 커널 예외 테이블 정의하기 – sort_main_extable() | 318 |
| 18.6 RCU 메커니즘 초기화하기 – rcu_init() | 320 |
| 18.7 IRQ를 사용하기 위한 준비하기 – early_irq_init() | 323 |
| 18.8 인터럽트 초기화하기 – init_IRQ() | 328 |
| 18.9 프로세스 정보를 빠르게 찾기 위한 구조 만들기 – pidhash_init() | 331 |
| Chapter 19 init_timers() ~ page_cgroup_init() | 333 |
| 19.1 타이머 초기화하기 – init_timers() | 334 |
| 19.1.1 timer_cpu_notify() | 335 |
| 19.1.2 register_cpu_notifier() | 336 |
| 19.1.3 open_softirq() | 337 |
| 19.2 High Resolution 타이머 초기화하기 – hrtimers_init() | 338 |

| | | |
|---|---|------------|
| 19.3 | softirq의 콜백 함수 등록하기 – softirq_init() | 341 |
| 19.4 | xtime 설정하기 – timekeeping_init() | 345 |
| 19.5 | 하드웨어 타이머 초기화하기 – time_init() | 348 |
| 19.6 | clock 타임 초기화하기 – sched_clock_init() | 350 |
| 19.7 | CPU의 인터럽트 활성화하기 – local_irq_enable() | 352 |
| 19.8 | 루트 파일시스템으로 사용되는 init 램디스크 검사하기 | 353 |
| 19.9 | 동적 메모리 할당을 위한 초기화 작업하기 – vmalloc_init() | 353 |
| 19.10 | 덴트리와 아이노드 캐시에 대한 초기화 미리 수행하기 – vfs_caches_init_early() | 355 |
| 19.11 | cpuset 서브시스템 초기화하기 – cpuset_init_early() | 358 |
| 19.12 | memory 서브시스템 초기화하기 – page_cgroup_init() | 360 |
| Chapter 20 bootmem 할당자를 종료하고 버디 시스템으로 교체 | | 363 |
| 20.1 | mem_init() 함수의 호출 관계와 자료구조의 상관 관계 | 363 |
| 20.2 | mem_init() 구조 알아보기 | 364 |
| 20.3 | 존재하지 않는 메모리 비트맵에 기록하기 – free_unused_memmap_node() | 366 |
| 20.4 | 일반 프리 페이지 버디 시스템으로 이관하기 – free_all_bootmem_node() | 368 |
| 20.4.1 | register_page_bootmem_info_node() | 368 |
| 20.4.2 | free_all_bootmem_core() | 370 |
| 20.4.3 | __free_pages_bootmem() | 372 |
| 20.4.4 | __free_pages() | 376 |
| 20.4.5 | free_hot_cold_page() | 376 |
| 20.4.6 | __free_pages_ok() | 378 |
| 20.5 | 하이 메모리 프리 페이지 버디시스템으로 이관하기 – free_area() | 383 |
| Chapter 21 CPU 핫플러그 지원을 위한 초기화 | | 385 |
| 21.1 | cpu_hotplug 멤버 변수 초기화하기 – cpu_hotplug_init() | 385 |
| 21.2 | CPU의 온라인 → 오프라인으로 변경 시 처리 | 386 |

| | |
|--|------------|
| Chapter 22 슬랩 메모리 할당자 활성화하기 – kmem_cache_init() | 389 |
| 22.1 슬랩 할당자 개념과 구조체들 | 389 |
| 22.2 슬랩 할당자의 중요 구조체 – kmem_cache와 kmem_list3 | 391 |
| 22.3 kmem_cache_init() 구조 알아보기 | 394 |
| 22.4 initkmem_list3[], cache_cache.nodelists[] 초기화 | 399 |
| 22.5 kmem_list3 배열에 연결하고 cache 축소시간 정하기 – set_up_list3s() | 401 |
| 22.6 cache의 확장 및 축소에서 사용될 페이지 오더 구하기 – cache_estimate() | 402 |
| 22.7 malloc_sizes와 cache_names | 406 |
| 22.8 cache 생성하기 – kmem_cache_create() | 408 |
| 22.8.1 kmem_cache_zalloc() | 410 |
| 22.8.2 calculate_slab_order() | 411 |
| 22.8.3 setup_cpu_cache() | 412 |
| 22.8.4 enable_cpucache() | 414 |
| 22.9 arraycache_init, kmem_list3 cache 생성 | 414 |
| 22.10 정적으로 할당받은 메모리를 kmalloc()을 통해 할당받은 메모리로 대체하기 | 417 |
| Chapter 23 kmem_trace_init() ~ security_init() | 420 |
| 23.1 ID allocator 캐시 생성하기 – idr_init_cache() | 421 |
| 23.2 pageset 초기화하기 – setup_per_cpu_pageset() | 421 |
| 23.3 인터리브 노드 지정하기 – numa_policy_init() | 427 |
| 23.4 타이머 초기화 마무리하기 – late_time_init() | 431 |
| 23.5 BogoMIPS 측정하기 – calibrate_delay() | 431 |
| 23.6 프로세스 식별자(ID) 할당을 위한 비트맵 만들기 – pidmap_init() | 433 |
| 23.7 우선순위 트리의 자료구조 초기화하기 – prio_tree_init() | 435 |
| 23.8 anon_vma 슬랩 캐시 생성하기 – anon_vma_init() | 435 |

| | | |
|--|--|---------|
| 23.9 | 객체에 사용자별로 자격 부여하기 - cred_init() | 437 |
| 23.10 | fork()를 사용할 수 있게 자료구조 초기화하기 - fork_init() | 438 |
| 23.11 | 프로세스 생성을 위한 캐시 초기화하기 - proc_caches_init() | 439 |
| 23.12 | 버퍼 캐시 초기화하기 - buffer_init() | 442 |
| 23.13 | 보안 키 준비하기 - key_init() | 445 |
| Chapter 24 VFS에서 사용되는 다양한 캐시들 초기화하기 - vfs_caches_init() | | 449 |
| Chapter 25 radix_tree_init() ~ ftrace_init() | | 468 |
| 25.1 | 래덱스 트리 관련된 자료구조 초기화하기 - radix_tree_init() | 469 |
| 25.2 | 시그널을 사용할 준비하기 - signals_init() | 470 |
| 25.3 | proc 파일시스템을 등록하고 마운트하기 - proc_root_init() | 471 |
| 25.4 | 초기화하지 못한 서브시스템 등록하기 - cgroup_init() | 472 |
| 25.5 | top_cpuset을 재설정하고 cpuset 파일시스템 등록하기 - cpuset_init() | 474 |
| 25.6 | 태스크 통계 정보 인터페이스 초기화하기 - taskstats_init_early() | 475 |
| 25.7 | 지연 정보 관리를 위한 준비 - delayacct_init() | 477 |
| 25.7.1 | 딜레이 어카운팅 | 477 |
| 25.7.2 | dealyacct_init() | 477 |
| 25.7.3 | task_delay_info 구조체와 delayacct_tsk_init() | 478 |
| 25.8 | 쓰기 버퍼의 일관성 검사하기 - check_bugs() | 481 |
| Chapter 26 메모리와 백킹 스토어 싱크하기 - 페이지 라이트백 | | 483 |
| 26.1 | 페이지 라이트백 메커니즘 | 484 |
| 26.2 | 페이지 라이트백 활성화하기 - pdflush_init() | 485 |
| 26.3 | pdflush 스레드 | 487 |
| 26.4 | 페이지 라이트백 함수 지정하기 - pdflush_operation() | 489 |
| 26.5 | 어떻게 주기적 페이지 라이트백, 강제적 페이지 라이트백 콜백 함수가 호출되나? | 490 |
| 26.5.1 | 주기적 페이지 라이트백 함수 - wb_kupdate() | 490 |
| 26.5.2 | 강제적 페이지 라이트백 함수 - background_writeout() | 493 |
| 26.6 | 주기적 페이지 라이트백 초기화하기 - page_writeback_init() | 494 |

| | | |
|-------------------|---|------------|
| Chapter 27 | 커널 부팅의 마지막 함수 구조 알아보기 – rest_init() | 496 |
| Chapter 28 | 함수를 실행할 커널 스레드 생성하기 – kernel_thread() | 499 |
| 28.1 | kernel_thread() 구조 알아보기 | 499 |
| 28.2 | 프로세서를 생성하는 게이트웨이 – do_fork() | 500 |
| 28.3 | 부모 프로세스 복사하기 – copy_process() | 504 |
| Chapter 29 | 새로 생성한 태스크 깨우기 | 514 |
| 29.1 | wake_up_new_task() 구조 알아보기 | 514 |
| 29.2 | 태스크의 런큐 가져오기 – task_rq_lock() | 517 |
| 29.3 | 태스크의 우선순위 개선하기 – effective_prio() | 518 |
| Chapter 30 | 커널 사용할 준비하기 – kernel_init() | 522 |
| 30.1 | 현재 프로세스를 다른 CPU로 이주시키기 – sched_init_smp() | 523 |
| 30.2 | 시스템 전반적으로 초기화 마무리하기 – do_basic_setup() | 526 |
| 30.2.1 | rcu_sched_grace_period()를 실행할 스레드 생성하기 – rcu_init_sched() | 527 |
| 30.2.2 | events 워크 큐 생성하기 – init_workqueues() | 528 |
| 30.2.3 | cpuset 서브시스템의 top_cpuset 초기화하기 – cpuset_init_smp() | 535 |
| 30.2.4 | khelper 워크 큐 생성하기 – usermodehelper_init() | 536 |
| 30.2.5 | 리눅스의 디바이스 모델 초기화하기 – driver_init() | 538 |
| 30.2.6 | irq 정보를 proc 파일시스템에 등록하기 – init_irq_proc() | 548 |
| 30.2.7 | 커널이 모르는 서브시스템 호출하기 – do_initcalls() | 549 |
| 30.3 | 초기화 다음을 준비하기 – init_post() | 554 |
| Chapter 31 | 커널 스레드 데몬 | 556 |
| 31.1 | 커널 스레드 데몬 – kthreadd() | 556 |
| 31.2 | 시그널 무시하기 – ignore_signals() | 559 |
| 31.3 | 나이스 값 설정하기 – set_user_nice() | 562 |
| 31.4 | 태스크를 실행시킬 CPU 찾기 – set_cpus_allowed_ptr() | 567 |
| 31.5 | 리스트를 포함하고 있는 실제 구조체 위치 찾기 – list_entry() | 569 |
| 31.6 | 커널 스레드 생성하기 – create_kthread() | 572 |

| | |
|---|------------|
| Chapter 32 find_task_by_pid_ns() ~ cpu_idle() | 575 |
| 32.1 PID로 태스크 찾기 – find_task_by_pid_ns() | 576 |
| 32.2 BKL 해제하기 – unlock_kernel() | 579 |
| 32.3 스케줄링 클래스를 idle로 변경하기 – init_idle_bootup_task() | 580 |
| 32.4 RCU 메커니즘이 활성화되었음을 알리기 – rcu_scheduler_starting() | 580 |
| 32.5 커널 선점 활성화하기 – preempt_enable_no_resched() | 581 |
| 32.6 프로세스 스케줄링 수행하기 – schedule() | 582 |
| 32.6.1 schedule()의 진짜 알맹이 – __schedule() | 582 |
| 32.7 리눅스 부팅 대장정의 끝 – cpu_idle() | 584 |

APPENDIX

| | |
|--------------------------------------|------------|
| Appendix A 어셈블러, gas 키워드 정리 | 588 |
| A.1 ARM 명령어 기본 | 588 |
| A.1.1 분기 명령어 | 588 |
| A.1.2 산술 연산 명령어 | 588 |
| A.1.3 논리 연산 명령어 | 588 |
| A.1.4 비교 연산 명령어 | 589 |
| A.1.5 데이터 전송 명령어 | 589 |
| A.1.6 상태 레지스터 명령어 | 589 |
| A.1.7 코프로세서 명령어 | 590 |
| A.1.8 기타 명령어 | 591 |
| A.2 조건부 실행 | 591 |
| A.3 gas 지시어 | 592 |
| Appendix B 커널 분석 시 자주 나오는 API | 594 |
| B.1 이중 연결 리스트 | 594 |
| B.2 비트 연산 API | 595 |

| | |
|---|-----|
| Appendix C 가볍게 알아보는 ext2 파일시스템 | 596 |
| C.1 파일을 나타내는 구조: inode | 596 |
| C.2 i_mode 필드를 이루는 비트 구조 | 597 |
| C.3 파일의 데이터를 가지고 있는 필드 - i_block[] | 598 |
| C.4 디스크에 저장되는 구조 | 600 |
| C.4.1 파티션이 가지는 구조 | 600 |
| C.4.2 Block Group 구조 | 601 |
| C.4.3 Inode Table 구조 | 603 |
| C.4.4 디렉터리, 파일, Inode Table, Data Block 간의 관계 | 604 |
| | |
| Appendix D 리눅스 스레드 모델 | 608 |
| D.1 스레드 모델 | 608 |
| D.2 리눅스 스레드 구현 | 609 |
| | |
| Appendix E 링커 스크립트 파일의 구조 | 612 |
| E.1 ARM 링커 스크립트 파일 구성 | 613 |
| E.2 링커 스크립트 구성도 | 615 |
| | |
| Appendix F 코드 리스트 | 623 |
| | |
| Appendix G 그림 리스트 | 632 |
| | |
| Appendix H 알아봅시다! 리스트 | 639 |

감수자의 글

□ 러 3년의 집필 기간……

└ 이 책의 공식적인 집필 기간은 약 1년이다. 하지만 본인은 3년이라고 이야기하고 싶다. 왜냐하면 저자들은 무려 2년간 리눅스 커널 소스를 분석하였고, 분석한 내용을 토대로 다시 1년간의 집필 과정을 거쳤기 때문이다. 이 한 권의 책을 위해 저자들은 3년이란 긴 시간 동안 주말과 밤을 잊은 채 오로지 리눅스 커널이라는 방대한 분야에 실력을 갈고 닦았으며, 마침내 이 한 권의 책을 탄생시켰다.

이러한 사정을 알기에 감수를 위해 원고를 받아본 순간 저자들보다 본인이 더 감개무량했다. 한 땀 한 땀 저자들의 노고가 녹아 있는 책을 보며, ‘드디어 우리나라에도 이러한 책이 나오는구나!’라며 자랑스러워 하기도 했다.

이 책에 대해 간단히 설명하자면, 이 책은 독자들보다는 저자들 자신의 책이라 말하고 싶다. 시중에는 내용도 모르고 번역만 한 번역서가 참으로 많다. 특히나 리눅스 커널 분야에 그 경향은 더 두드러진다. 하지만 이 책은 그렇지 않다. 이 책은 무려 2년이라는 시간 동안 리눅스 커널을 분석하고 그 실력을 연마한 저자들이 직접 자신의 이야기를 쓴 책이다. 그래서 지금까지 나온 그 어떠한 책보다 수준이 높다. 그리고 저자들 자신의 것으로 온전히 소화해낸 책이기에 독자들이 궁금해하는 부분 대부분도 다룰 수 있었다고 본다.

이 책은 마치 예리한 메스와 같은 책이다. 이 책을 읽어보면 알겠지만, 이 책은 모든 부분에 있어 디테일이 살아있다. 이는 2년이라는 시간 동안 자신을 예리한 메스와 같이 유지한 저자들 본인의 실력이 반영되어 있어서 그렇다. 또한 이 책은 막힘이 없다. 이는 처음부터 끝까지 다 알고 있는 사람이 하나씩 하나씩 이야기를 풀었기 때문에 가능한 것이다.

이러한 이유들로 이 책은 지금까지 나온 그 어떤 커널 책보다 돋보이는 수작이라 할 수 있다. 그래서 본인은 한국에 이러한 책이 나오게 된 것을 자랑스럽게 생각한다.

3년이란 긴 시간 동안 최선을 다했을 저자들에게 박수를 보낸다. 그리고 이러한 책으로 공부할 수 있게 된 독자에게는 행운이라고 이야기하고 싶다. 왜냐하면 필자는 이러한 책이 나오기 한참 전에 리눅스 커널을 접했는데, 정말 엄청나게 많은 시행착오를 겪었기 때문이다. 리눅스 커널을 공부하고자 하는 많은 사람들에게 이 책은 정말 좋은 길잡이가 되어줄 것이라 장담한다.

감수자 **백창우**



백창우 · 삼성전자, 삼성SDS, (주)누스코에서 여러 종류의 RTOS를 주도적으로 개발하였다. 삼성종합기술원에서 컴파일러를 개발하였다. (주)누스코에서 디버거와 하이퍼바이저 및 각종 시스템 S/W를 개발하였다. 현재는 (주)누스코의 대표이사로 근무하고 있고, 소프트웨어 마에스트로 멘토로 있으며, 오프라인 시스템 S/W 스터디 그룹인 <http://www.iamroot.org>를 10년째 운영 중에 있다.

머리말

리눅스 커널을 분석하고 분석한 내용을 책으로 낸다는 것은 우리에게 큰 도전이었다. iamroot.org를 통해 2009년 5월 ARM용 리눅스 커널을 분석하기 위한 첫 모임을 가졌다. 그리고 우리는 1차 분석을 종료하기까지 2년 내내 주말 중 하루를 꼬박 반납해야 했다. 책으로 접했던 운영체제의 코드를 직접 분석한다는 것은 엔지니어로서 많은 희열을 느끼게 해주었지만, 분석하는 과정은 우리에게 적잖은 인내를 요구하였다.

수년 동안 커널 소스 분석 스터디를 통해 정립된 iamroot의 가이드라인을 따라 소스 코드 분석을 하였다. 이 책은 ARM용 리눅스 커널 소스를 다루고 있다. 소스 코드를 라인 단위로 분석해가면서 실제 커널이 어떤 일들을 수행하는지를 분석함으로써 커널의 내부를 좀 더 잘 이해하고자 하는 독자들을 대상으로 하고 있다. 커널 소스를 라인 단위로 분석해간다는 것은 전원이 인가된 후에 커널의 시작 코드가 어떻게 호출되고, 시작 코드 내에서 여러 함수들이 호출되어 프롬프트 셸을 최종적으로 보여주는 일련의 과정을 분석한 것이다. 이런 점에서 이 책은 다른 여타의 리눅스 커널 책과는 다르게 구성될 수밖에 없었으며, 그 어떤 책에서도 언급되지 않는 내용을 다루고 있다. 리눅스 커널에 대한 기본 개념은 많은 자료를 통해 얻을 수 있다. 하지만 커널 소스를 다루지 않는 이상 운영체제의 구현 방법을 알고자 하는 엔지니어의 지적 갈증을 해결해줄 수는 없을 것이다. 바로 이 점이 이 책이 갖는 큰 가치라고 생각한다.

분석을 시작할 때나 1차 분석을 마칠 때까지도 커널 책에 대한 집필은 우리의 로드맵에 없었던 일이다. 커널 분석을 시작할 때는 모두 의기양양했지만 분석하면 할수록 우리의 능력과 지식이 쉽게 바닥을 드러냈고, 그렇기 때문에 책을 집필한다는 것이 우리들의 능력을 훨씬 넘어선 일임을 모두 잘 알고 있었다. 우리 스스로 많이 부족하다는 것을 인정하고 잘 알고 있지만, 우리가 이 책을 내게 된 이유는 분석하면서 고민하고 또 이해한 내용들을 리눅스를 좋아하는 엔지니어들과 커널에 관심을 가지고 있는 여러분들과 공유하고 싶었다. 리눅스 커널에 대한 수많은 엔지니어의 공헌으로 우리가 리눅스 커널 코드를 공부할 수 있었다는 믿음 때문이다.

우리가 이렇게 책을 통해 공유하는 것은 우리가 아는 것이 많아서가 아니다. 또한 이 책에서 다루는 내용이 완벽하다고도 생각하지 않는다. 1991년 리눅스가 수줍은 이메일로 자신의 커널을 세상에 공유했던 것처럼 우리도 누군가에게 우리가 정리한 내용이 도움이 되길 바랄 뿐이고, 또 다른 누군가는 이 책의 내용을 기반으로 더 나은 지적 자산을 완성해주길 바랄 뿐이다. 그래서 결국 우리가 여러분의 도움으로 리눅스 커널을 더 잘 이해하고 싶다. 리눅스 커널을 이해하고자 하는 분들에게 이 책이 조금이나마 도움이 되길 바란다.

2012년 8월

저자 일동

집필 후기

리눅스 커널은 수많은 엔지니어들의 협력과 공헌으로 이뤄낸 값진 결정체다. 리눅스와 같은 수평적 협력을 통해 이처럼 위대한 지적 자산을 인류가 만들어낸 적이 또 있었을까? 우리는 리눅스 커널 소스를 분석하고 배웠지만, 그보다도 리눅스가 가지는 정신을 더 많이 배웠다고 생각한다. 이 책이 커널을 좀 더 깊게 알고자 하는 모든 분들에게 도움이 되길 바란다.

노서영

리눅스 커널은 수많은 엔지니어들의 협력과 공헌으로 이뤄낸 값진 결정체다. 리눅스와 같은 수평적 협력을 통해 인류가 이처럼 위대한 지적 자산을 만들어낸 적이 또 있었을까? 우리는 커널 소스를 분석하였지만, 사실은 리눅스가 품고 있는 정신을 배웠다. 책을 집필하는 중에는 기쁨과 설렘의 연속이었지만, 집필 후기를 쓰는 지금은 아쉬움이 밀려든다. 조금 더 깊이 있는 분석, 조금 더 세밀한 설명, 화려한 문장가의 문필은 아닐지라도 핵심을 짚어 깔끔하게 설명하는 문장력, 이 모든 것이 아쉽기만 하다. 어찌 보면 이러한 부족함들은 저자들의 머리말처럼 여러분들의 몫이 될지도 모른다. 우리들이 채우지 못한 부족한 부분들을 꼭 채워주길 바란다. 끝으로 이 책이 나오기까지 고생한 저자들, 책의 출간을 권유했던 누스코 백창우 대표님, 저자들의 어설픈 원고에도 불구하고 여기까지 이끌어주신 제이펍의 장성두 실장님, 오랜 기간 동안 주말 중 하루를 반납하였지만 아낌없이 지원해준 사랑하는 아내와 을과솔에게 감사의 마음을 전한다.

윤석훈

재미있었다. 힘든 노동과 같던 커널 분석이 멋진 동료들이 함께 하니 재미있는 놀이가 되었다. 놀랍게도 오랜 기간의 스터디 동안 난 버텼고 책이라는 인쇄물에 내 이름을 새길 수 있게 되었다. 겁 많고 소심하던 기존의 나 혼자라면 이런 일이 가능했을까? 함께였기에 가능했던 작업, 전방위로 나에게 많은 영감을 준 멤버들에게 모든 영광을 돌리고 싶다.

이 책을 읽는 독자들도 부디 맘이 맞는 동료를 찾아서 책을 벗 삼아 커널을 연구하길 추천한다. 함께 토론하며 학습하는 것이 혼자 하는 것보다 몇 배는 더 좋은 결과를 내는 것을 체험했기 때문이다. vi도 제대로 못쓰던 내가 책을 쓸 정도로 발전한 것처럼 말이다.

여러분! 이제 난 더 멋진 일을 해낼 수 있는 용기가 생겼습니다!

강진성

3년 전 시작된 여정의 끝을 보았다. 아니 어찌 보면 이제 새로운 여정의 시작이라고 할 수도 있겠지만, 지금까지의 과정은 하나의 여행 같았다. vi라는 차를 타고, 주말 반납이라는 요금을 내고, 스터디 멤버들과 함께 달려온 종착역에서 만난 『코드로 알아보는 ARM 리눅스 커널』을 보는 순간 감격과 설렘에 잠을 이룰 수 없었다. 노하우와 노력이 담긴 이 녀석이 이젠 다른 이의 머릿속에 숨쉬길 바라며, 리눅스 커널의 장벽에 놓인 여러분께 한줄기 희망처럼 비취 주길 바란다. 마지막으로, 저의 멘토이신 백창우 님과 도움을 주신 JCDSOFT의 정준호, 권홍재 님께 진심으로 감사드린다.

송원준

3년 전 리눅스 커널을 공부하고 싶다는 목적만으로 30명 가까이 되는 사람들이 모였었다. 그렇게 매주 바쁜 시간을 쪼개어 토요일마다 8시간씩 커널을 보며 토론을 했다. 어느덧 2년이 지났을 땐 지금의 저자들만이 남게 되었다. 이렇게 끈기와 열정 가득한 팀원들과 1년의 시간 동안 정리하여 결과물을 내놓게 되어 매우 기쁘다. 백창우 선생님의 권유로 얼떨결에 책을 쓰기로 결정했지만, 큰 기대와 부푼 꿈을 안고 시작했었다. 하지만 시간이 지날수록 스스로의 부족함에 부끄러워져 갔었다.

후기를 쓰기 위해 길었던 시간들을 돌이켜보지만 떠오르는 건 아쉬움뿐인 것

같다. 커널은 컴퓨터 공학을 전공한 사람에게 있어서 한 번쯤은 공부할 만한 가치가 있는 소프트웨어라고 생각한다. 꼭 자신의 분야와 밀접하게 관련이 있지 않다고 하더라도 어떤 식으로든 큰 배움을 얻을 수 있다고 확신한다. 이번 경험을 통해 느낀 여러 가지 아쉬움들이 앞으로도 꾸준히 리눅스 커널을 공부할 수 있는 원동력이 되길 바란다. 무엇보다도 끈기와 열정 가득한 팀원들 모두에게 감사하는 마음이 크다. 지치지 않는 열정으로 정신적 지주가 되어주신 노서영 박사님을 비롯하여 타고난 엔지니어적 센스로 무장하여 스테디가 지루하지 않도록 분위기를 이끌어주신 이윤재 형님, 특유의 성실함과 과묵함으로 책임과 끈기를 더해주신 임윤재 형님, 팀의 선봉으로 다른 팀원들이 신경 쓰지 못하는 작은 부분까지 신경 써주고 이끌어오신 윤석훈 씨, 대찬 성품으로 항상 파이팅 넘치는 모습을 보여주신 강진성 씨, 모두와의 인연에 감사하고 지난 3년을 함께 보낼 수 있어서 너무 행복했다. 마지막으로, 항상 좋은 환경에서 공부할 수 있도록 도와주시고 지켜봐주시는 어머니, 아버지께 감사와 사랑의 마음을 전하고 싶다.



이윤재

우선 멤버들과 백창우 선생님의 도움, 그리고 회사에서 계속 스테디를 할 수 있게 배려해주신 많은 분들 덕분에 이 책이 나올 수 있었다는 이야기를 하고 싶다. 'ls -al' 명령을 알게 되서 리눅스를 좋아했을 뿐인 청년이 멋진 멤버들을 만나 스테디를 하고 책까지 내게 되었다는 사실이 신기할 뿐이다.

사실 우리가 쓴 책은 어찌 보면 리눅스 커널이라는 넓은 바다에서 선구자들이 노력하여 획득한 수많은 지식의 돌고래를 하나로 엮어낸 데 불과할 수 있지만, 지금껏 리눅스라는 바다에 직접적으로 뛰어들기를 원하거나 이것이 필요했던 사람들에게 이 책이 도움이 될 수 있기를 바란다. 그리고 커널을 넘어 모두의 삶에 작은 보탬이어나마 되고 싶은 소망이 있다.

여러 명의 도움으로 책 하나 쓰고서는 아버지, 어머니 그리고 언젠가 생길 여자친구에게 감사하다는 말을 주제넘게 하고 싶다.



임윤재

리눅스 커널 분석 작업은 리눅스에 대한 지식만이 아니라 이를 통해서 능숙해진 코드 분석 능력, 그리고 커널에 적용된 수많은 자료구조와 알고리즘을 살펴보면서 익힌 지식과 이를 다른 코딩에도 유용하게 응용할 수 있는 능력과 같은 부가적인 큰 선물을 안겨주었다.

회사 업무로 바쁜 중에 오랜 기간에 걸쳐 틈틈이 코드를 분석하고 책으로 써내기까지 정말 쉽지 않은 시간들이었다. 부족한 능력 탓에 좀 더 잘 설명하지 못해 아쉬운 부분이 한두 군데가 아닌데, 마치 물가에 아이를 내놓은 것과 같은 걱정과 옷을 벗고 사람들 앞에 서는 부끄러움으로 이 책을 내놓는다. 부족하지만 커널을 알고자 하는 또는 커널 분석을 통해 내공을 쌓고자 하는 이들에게 조금이나마 도움이 되었으면 하는 바람을 가져본다.



