

# Оглавление

<b>Об авторе .....</b>	<b>18</b>
<b>О рецензенте .....</b>	<b>19</b>
<b>Предисловие от издательства .....</b>	<b>20</b>
<b>Предисловие.....</b>	<b>21</b>
Для кого предназначена эта книга.....	22
Что добавлено во втором издании.....	22
Краткое содержание книги.....	24
Как извлечь максимум из этой книги.....	28
Скачивание примеров кода .....	28
Скачивание цветных иллюстраций .....	29
Графические выделения .....	29
Обратная связь.....	30
Отзывы .....	30
<b>Глава 1. Программирование ядра Linux – краткое введение ...</b>	<b>31</b>
Подготовка рабочего пространства ядра .....	32
Технические требования.....	33
Клонирование репозитория кода.....	33
<b>Глава 2. Сборка ядра Linux из исходного кода – часть 1 .....</b>	<b>35</b>
Технические требования.....	36
Предварительные условия для сборки ядра .....	36
Нумерация версий ядра Linux .....	37
Нумерация по пальцам на руках и ногах.....	38
Цикл разработки ядра – основы .....	39
Просмотр журнала ядра в Git из командной строки.....	40
Просмотр журнала ядра в Git на странице GitHub.....	42
Цикл разработки ядра конспективно .....	43
Упражнение .....	44
Типы деревьев исходного кода.....	44
LTS-ядра – новый мандат.....	46
Какое ядро мне использовать? .....	47
Шаги сборки ядра из исходного кода .....	48
Шаг 1 – получение исходного кода ядра Linux .....	49
Скачивание дерева конкретного ядра .....	49
Клонирование дерева Git .....	51

Шаг 2 – распаковка исходного кода ядра .....	52
Краткий обзор дерева исходного кода ядра .....	53
Шаг 3 – конфигурирование ядра Linux .....	59
Минимальные сведения о системе сборки Kconfig/Kbuild .....	60
Как работает система Kconfig+Kbuild – минимальные сведения .....	62
Получение конфигурации по умолчанию .....	63
Получение хорошей отправной точки для конфигурации ядра .....	64
Конфигурирование ядра как в дистрибутиве .....	65
Получение оптимизированной конфигурации ядра с помощью localmodconfig .....	66
Конфигурирование ядра для типичной встраиваемой Linux-системы .....	66
Просмотр всех доступных конфигурационных параметров .....	70
Применение подхода localmodconfig .....	74
Донастройка конфигурации ядра с помощью графического интерфейса make menuconfig .....	76
Примеры работы с интерфейсом make menuconfig .....	79
Конфигурирование ядра – дополнительные вопросы .....	84
Поиск по меню .....	84
Поиск различий в конфигурациях .....	86
Использование скрипта config для просмотра и редактирования конфигурации ядра .....	86
Конфигурирование безопасности ядра .....	87
Настройка меню Kconfig и добавление своего собственного пункта .....	90
О файлах Kconfig* .....	90
Создание нового пункта в меню General Setup .....	93
Несколько слово о языке Kconfig .....	96
Резюме .....	99
Упражнение .....	99
Вопросы .....	100
Для дополнительного чтения .....	100

## **Глава 3. Сборка ядра Linux из исходного кода – часть 2 ..... 101**

Технические требования .....	101
Шаг 4 – сборка образа ядра и модулей .....	101
Решение проблемы сертификата в Ubuntu .....	105
Шаг 5 – установка модулей ядра .....	109
Нахождение модулей ядра среди исходного кода .....	109
Устанавливаем модули ядра .....	110
Задание другого места установки модулей по умолчанию .....	111
Шаг 6 – генерирование образа initramfs и подготовка начального загрузчика .....	111
Генерирование образа initramfs – что под капотом .....	113
О назначении каркаса initramfs .....	115
Зачем нужен каркас initramfs? .....	115
Основы процесса начальной загрузки в x86 .....	117
Еще о каркасе initramfs .....	120
Заглянем внутрь образа initramfs .....	120

Шаг 7 – конфигурирование начального загрузчика GRUB .....	124
Конфигурирование GRUB – основы .....	124
Выбор ядра, загружаемого по умолчанию.....	125
Загрузка ВМ с помощью начального загрузчика GRUB .....	127
Эксперименты с приглашением GRUB .....	129
Проверка конфигурации нового ядра.....	131
Сборка ядра для Raspberry Pi .....	132
Шаг 1 – клонирование дерева исходного кода ядра Raspberry Pi.....	133
Шаг 2 – установка инструментов кросс-разработки x86_64-в-AArch64.....	134
Шаг 3 – конфигурирование и сборка ядра AArch64 для Raspberry Pi.....	136
Разные замечания по сборке ядра .....	138
Минимальные требования к версии .....	139
Сборка ядра для другой машины .....	139
Наблюдение за ходом сборки ядра .....	141
Синтаксис сокращенного выполнения в применении к процедуре сборки.....	143
Отсутствующие заголовочные файлы для разработки с участием OpenSSL.....	143
Как проверить, какие дистрибутивные ядра установлены?.....	144
Резюме.....	145
Вопросы.....	145
Для дополнительного чтения .....	145

## **Глава 4. Написание первого модуля ядра – часть 1.....146**

Технические требования .....	146
Архитектура ядра – часть 1 .....	147
Пространство пользователя и пространство ядра .....	148
Библиотечные функции и системные вызовы .....	148
Компоненты ядра .....	150
Исследование структуры LKM.....	152
Каркас LKM.....	152
Место модулей ядра в дереве исходного кода.....	154
Написание первого модуля ядра .....	156
Знакомство с С-кодом модуля «Hello, world» .....	156
Пояснения к коду.....	158
Заголовочные файлы ядра .....	158
Модульные макросы.....	159
Точки входа и выхода .....	159
Возвращаемые значения .....	160
Типичные операции над модулями ядра .....	163
Сборка модуля ядра.....	163
Выполнение модуля ядра.....	165
Первое знакомство с функцией printk().....	166
Получение списка активных модулей ядра.....	168
Выгрузка модуля из памяти ядра .....	168
Наш вспомогательный скрипт lkm.....	169

Запись в журнал ядра и функция printk.....	172
Работа с кольцевым буфером в памяти ядра .....	172
Протоколирование ядра и программа journalctl.....	173
Уровни протоколирования printk.....	175
Макросы pr_<foo> .....	177
Запись на консоль.....	179
Вывод на консоль Raspberry Pi.....	182
Включение отладочных сообщений ядра .....	185
Введение в динамическую отладку ядра .....	187
Ограничение частоты сообщений printk .....	191
Макросы с ограничением частоты.....	192
Генерирование сообщений ядра из пространства пользователя.....	193
Стандартизация вывода printk с помощью макроса pr_fmt .....	195
Переносимость и спецификаторы формата в printk .....	196
Что такое индексирование printk? .....	197
Основы Makefile для сборки модуля ядра .....	198
Резюме.....	202
Вопросы.....	202
Для дополнительного чтения .....	203

## **Глава 5. Написание первого модуля ядра – часть 2..... 204**

Технические требования.....	204
«Улучшенный» шаблон Makefile для модулей ядра.....	205
Конфигурирование «отладочного» ядра .....	207
Кросс-компиляция модуля ядра.....	209
Подготовка системы к кросс-компиляции .....	209
Попытка 1 – установка переменных окружения ARCH и CROSS_COMPILE .....	210
Попытка 2 – Makefile указывает на дерево исходного кода ядра для целевой платформы .....	212
Попытка 3 – кросс-компиляция модуля ядра.....	214
О совместимости ABI ядра Linux.....	215
Попытка 4 – кросс-компиляция нашего модуля ядра .....	216
Сводка ошибок, допущенных нами при кросс-компиляции, сборке и загрузке модулей, и способов их исправления .....	217
Сбор минимальной информации о системе .....	218
Лицензирование модулей ядра .....	223
Лицензирование внутреннего кода ядра.....	223
Лицензирование внешних модулей.....	224
Эмуляция «библиотек» для модулей ядра .....	225
Эмуляция библиотек путем компоновки нескольких исходных файлов .....	226
Область видимости функций и переменных в модуле ядра .....	227
Стек модулей.....	230
Эксперименты со стеком модулей.....	231
Эмуляция «библиотек» для модулей ядра – резюме и выводы.....	237
Передача параметров модулю ядра .....	238

Объявление и использование параметров модуля .....	239
Получение и изменение параметров после вставки модуля.....	241
Допустимые типы параметров ядра и их проверка.....	243
Проверка параметров модуля .....	244
Переопределение имени параметра модуля.....	245
Параметры ядра, относящиеся к оборудованию.....	246
Операции с плавающей точкой в ядре запрещены .....	246
Как заставить ядро выполнять FP .....	247
Автозагрузка модулей на этапе начальной загрузки системы .....	249
Автозагрузка модулей – дополнительные сведения.....	254
Модули ядра и безопасность – краткий обзор .....	256
Параметры в файловой системе проcs, влияющие на системный журнал.....	257
О системном элементе dmesg_restrict.....	257
О системном элементе kptr_restrict .....	258
Криптографическое подписание модулей ядра.....	260
Два режима подписания модулей .....	262
Полный запрет модулей ядра .....	263
LSM блокировки ядра – введение.....	263
Наставление по стилю кодирования для разработчиков ядра.....	264
Предложение своего кода в основную ветвь ядра.....	264
С чего начать? .....	265
Резюме.....	266
Вопросы.....	266
Для дополнительного чтения .....	266

## **Глава 6. Основы внутреннего устройства ядра – процессы и потоки..... 267**

Технические требования.....	268
Контексты процесса и прерывания.....	268
Основы виртуального адресного пространства (ВАП) процесса .....	270
Организация процессов, потоков и их стеков – в пространстве пользователя и в пространстве ядра.....	273
Скрипт для получения числа процессов и потоков .....	274
Организация пространства пользователя .....	276
Организация пространства ядра .....	277
Сводка сведений, относящихся к потокам, структурам задач и стекам .....	279
Просмотр стеков в пространстве пользователя и ядра .....	280
Традиционный подход к просмотру стеков .....	281
Взгляд на ВАП процесса с высоты птичьего полета.....	287
Структура задачи в ядре и доступ к ней .....	288
Знакомство со структурой задачи.....	290
Доступ к структуре задачи с помощью current.....	292
Определение контекста .....	293
Работа со структурой задачи с помощью current .....	294
Встроенные в ядро вспомогательные функции и оптимизации .....	296

Использование нашего модуля ядра для печати информации о контексте процесса.....	297
Доказательство монолитности ОС Linux .....	298
Использование <code>printk</code> с учетом безопасности.....	299
Обход списков задач ядра.....	299
Обход списка задач I – отображение всех процессов.....	300
Обход списка задач II – отображение всех потоков .....	301
Как отличить процесс от потока – <code>TGID</code> и <code>PID</code> .....	302
Обход списка задач III – код .....	303
Резюме.....	306
Вопросы.....	307
Для дополнительного чтения .....	307

## Глава 7. Внутреннее устройство управления

### памятью – основы ..... 308

Технические требования.....	308
Разделение VM.....	309
Заглянем под капот – программа <code>Hello, world</code> .....	309
Не только <code>printf()</code> .....	311
Виртуальная адресация и трансляция адресов .....	313
Переход от виртуального адреса к физическому – очень краткий обзор .....	315
Разделение VM в 64-разрядных системах Linux .....	318
Типичные разделения VM .....	320
ВАП процесса – полная картина.....	322
Исследование ВАП процесса.....	323
Детальное исследование ВАП пользователя .....	324
Непосредственный просмотр карты памяти процесса с помощью <code>procsfs</code> .....	324
Интерфейсы для просмотра карты памяти процесса.....	326
Основы VMA.....	332
Исследование ВАП ядра .....	333
Верхняя память в 32-разрядных системах .....	335
Модуль, показывающий информацию о ВАП ядра.....	336
Макросы и переменные, описывающие структуру ВАП ядра.....	336
Практикум – просмотр сведений о ВАП ядра.....	340
ВАП ядра глазами <code>prostar</code> .....	345
Практикум – сегмент пользователя .....	349
Страница ловушки нуля .....	352
Документация ядра по структуре памяти .....	352
Рандомизация структуры памяти – <code>KASLR</code> .....	353
Рандомизация памяти пользователя с помощью <code>ASLR</code> .....	354
Рандомизация памяти ядра с помощью <code>KASLR</code> .....	355
Опрос и изменение состояния <code>KASLR</code> с помощью скрипта .....	355
Организация физической памяти.....	358
Как организована физическая память.....	358
Узлы и <code>NUMA</code> .....	359
Зоны внутри узла.....	362

Память прямого отображения и трансляция адресов .....	363
Введение в модели физической памяти .....	367
Краткое описание модели sparsemem[-vmemmap] .....	368
Резюме .....	369
Вопросы .....	370
Для дополнительного чтения .....	370

## Глава 8. Выделение памяти ядра для авторов

### модулей – часть 1 ..... 371

Технические требования .....	371
Введение в распределители памяти ядра .....	372
Страничный распределитель и его использование .....	373
Принципы работы страничного распределителя .....	373
Организация списка свободных в страничном распределителе .....	373
Как работает страничный распределитель .....	376
Проработка нескольких сценариев .....	377
Внутреннее устройство страничного распределителя – дополнительные детали .....	378
Как использовать API страничного распределителя .....	379
Флаги GFP .....	381
Освобождение памяти с помощью страничного распределителя .....	382
Рекомендации по выделению и освобождению памяти ядра .....	383
Написание модуля ядра для демонстрации API страничного распределителя .....	384
Страничный распределитель и внутренняя фрагментация .....	389
Еще о флагах GFP .....	391
Никогда не засыпайте в атомарных контекстах .....	391
Страничный распределитель – за и против .....	394
Слябовый распределитель и его использование .....	394
Идея кеширования объектов .....	394
Часто задаваемые вопросы об использовании слябовой памяти .....	395
Как использовать API слябового распределителя .....	398
Выделение слябовой памяти .....	398
Освобождение слябовой памяти .....	399
Функция kfree_sensitive() (ранее kzfree()) .....	400
Структуры данных – замечания о проектировании .....	401
Слябовые кеши и kmalloc .....	402
Модуль ядра, демонстрирующий использование слябового API .....	404
Размерные ограничения функции kmalloc .....	406
Тестируем пределы выделения памяти за один вызов .....	407
Слябовый распределитель – дополнительные детали .....	411
Управляемые API выделения памяти .....	411
Дополнительные слябовые API .....	412
Контрольные группы и память .....	412
Подводные камни при использовании слябового распределителя .....	413
Базовые положения и выводы .....	413

Проверка выделения слябовой памяти с помощью ksize() – случай 1 .....	414
Проверка выделения слябовой памяти с помощью ksize() – случай 2 ...	415
Интерпретация вывода в случае 2.....	417
Графическое представление .....	418
Нахождение внутренней фрагментации в ядре.....	419
Простой способ с применением slabinfo .....	419
Получение дополнительных сведений с помощью alloc_traces и скрипта .....	420
Слой слябового распределителя – плюсы и минусы.....	424
Слой слябового распределителя – несколько слов о реализации в ядре ...	424
Резюме.....	425
Вопросы.....	425
Для дополнительного чтения .....	425

## Глава 9. Выделение памяти ядра для авторов

<b>модулей – часть 2 .....</b>	<b>426</b>
Технические требования.....	426
Создание специального слябового кеша .....	426
Создание и использование специального слябового кеша в модуле ядра .....	427
Шаг 1 – создание специального слябового кеша .....	427
Шаг 2 – использование памяти из специального слябового кеша.....	429
Шаг 3 – уничтожение специального кеша .....	430
Специальный слябовый кеш – пример модуля ядра .....	430
Получение полезной информации о слябовых кешах.....	434
Резчики слябов .....	435
Сводка плюсов и минусов слябового распределителя.....	436
Отладка проблем с памятью в ядре – краткий обзор .....	436
API выделения памяти vmalloc.....	438
Как использовать функции семейства vmalloc .....	439
Пример использования vmalloc() .....	440
Краткое замечание о выделении памяти в пространстве пользователя и подкачке страниц по запросу .....	443
Родственники vmalloc().....	445
Эта память выделена vmalloc (или принадлежит модулю)? .....	445
Не уверены, какую функцию использовать? Попробуйте kvmalloc().....	445
Различные вспомогательные функции – vmalloc_exec() и vmalloc_user() .....	449
Задание защиты памяти .....	449
Сравнение функций kmalloc() и vmalloc() .....	450
Выделение памяти в ядре – какой API использовать .....	451
Наглядное представление API выделения памяти ядра.....	451
Выбор подходящей функции для выделения памяти ядра .....	452
Пара слов о DMA и CMA .....	455
Восстановление памяти – важная служебная задача ядра .....	455
Зонные предельные уровни и kswarp.....	456



Новые многопоколенческие LRU-списки .....	457
Эксперимент – гистограмма данных MGLRU .....	458
Краткое введение в DAMON – мониторинг доступа к данным .....	460
Выполнение рабочей нагрузки и наблюдение за ней с помощью интерфейса damo.....	461
Остаться в живых – палач процессов .....	464
Намеренный вызов палача процессов .....	465
Вызов палача процессов с помощью магической SysRq.....	465
Вызов палача процессов с помощью бешеного распределителя памяти .....	466
Три политики перезаказа памяти .....	467
Перезаказ виртуальной памяти с точки зрения функции _vm_enough_memory() .....	468
Случай 1: vm.overcommit_memory == 0 (режим по умолчанию, OVERCOMMIT_GUESS).....	471
Случай 2: vm.overcommit_memory == 2 (перезаказ VM выключен, OVERCOMMIT_NEVER) и vm.overcommit_ratio == 50 .....	473
Подкачка страниц по запросу и палач процессов.....	475
Оптимизированное чтение (неотображенной памяти) .....	481
Что такое оценка OOM .....	482
Заключительные мысли о палаче процессов и контрольных группах...	482
Замечание о контрольных группах и полосе пропускания памяти .....	483
Резюме.....	484
Вопросы.....	485
Для дополнительного чтения .....	485

## **Глава 10. Планировщик CPU – часть 1.....486**

Технические требования .....	487
Внутреннее устройство планирования CPU, часть 1 – основы.....	487
Что такое KSE в Linux?.....	487
Конечный автомат процесса в Linux.....	489
Политики планирования POSIX .....	491
Приоритеты потоков.....	493
Визуализация хода выполнения.....	494
Использование gnome-system-monitor для визуализации хода выполнения.....	495
Использование perf для визуализации хода выполнения .....	496
Практическое занятие – командная строка .....	498
Практическое занятие – графический интерфейс.....	500
Другие подходы к визуализации потока выполнения.....	502
Внутреннее устройство планирования CPU, часть 2.....	504
Модульные классы планирования .....	504
Концептуальный пример для понимания классов планирования .....	509
Опрос класса планирования .....	510
Краткое описание работы класса вполне справедливого планирования (CFS).....	513
Статистика планирования .....	517

Запрос политики и приоритета планирования данного потока.....	518
Внутреннее устройство планирования CPU, часть 3.....	521
Вытесняемое ядро .....	522
Динамическое задание режима вытеснения .....	523
Кто вызывает код планировщика? .....	524
Когда вызывается schedule()? .....	525
Минимально необходимые сведения о структуре thread_info .....	525
Обслуживание прерывания от таймера – установка TIF_NEED_	
RESCHED.....	527
Контекст процесса – проверка TIF_NEED_RESCHED .....	529
Точки входа в планировщик CPU – итоги.....	531
Краткий обзор кода планировщика .....	533
Резюме.....	534
Вопросы.....	535
Для дополнительного чтения .....	535

## **Глава 11. Планировщик CPU – часть 2..... 536**

Маска привязки к CPU, ее получение и установка .....	536
Запрос и установка маски привязки потока к CPU .....	537
Использование taskset для работы с маской привязки к CPU.....	540
Задание маски привязки к CPU для потока ядра .....	541
Запрос и установка политики и приоритета планирования потока .....	542
Задание политики и приоритета для потока ядра.....	543
Реальный пример – поточные обработчики прерываний .....	545
Введение в контрольные группы .....	546
Групповые контроллеры .....	547
Исследование иерархии контрольных групп версии v2 .....	549
Активация и деактивация контроллеров .....	550
Контрольные группы внутри иерархии .....	552
Systemd и контрольные группы .....	556
Наш скрипт для изучения контрольных групп .....	563
Практическое занятие – ограничение потребления CPU	
с помощью контрольных групп версии v2 .....	565
Использование systemd для задания ограничений	
на потребление ресурсов службой .....	566
Введение в эксплуатацию Linux в качестве OCPB.....	577
О сборке стандартного ядра 6.x с заплатами RTL (для x86_64).....	579
Разные вопросы, относящиеся к планированию .....	580
Несколько функций ядра, о которых стоит знать .....	581
OC ghOSt.....	581
Резюме.....	582
Вопросы.....	582
Для дополнительного чтения .....	582

## **Глава 12. Синхронизация ядра – часть 1 ..... 583**

Критические секции, монопольное выполнение и атомарность .....	583
Что такое критическая секция? .....	584

Классический случай – i++ для глобальной переменной .....	587
Концепции – блокировка .....	590
Критические секции – основные положения .....	593
Гонки за данные – более формальное определение .....	594
Вопросы конкурентности в ядре Linux .....	597
Многоядерные SMP-системы и гонки за данные .....	597
Вытесняемые ядра, блокирующий ввод-вывод и гонки за данные .....	598
Аппаратные прерывания и гонки за данные .....	599
Наставления по блокировке и взаимоблокировка .....	599
Мьютекс или спин-блокировка?	
Что и когда использовать .....	603
Какую блокировку использовать – теоретически .....	605
Какую блокировку использовать – практически .....	606
Использование мьютексов .....	607
Инициализация мьютекса .....	607
Правильное использование мьютекса .....	608
Функции захвата и освобождения мьютекса .....	609
Захват мьютекса: прерываемый или непрерываемый сон? .....	610
Захват мьютекса – пример драйвера .....	611
Мьютекс – еще несколько замечаний .....	616
Варианты API мьютексов .....	616
Вариант I/O .....	618
Семафор и мьютекс .....	618
Инверсия приоритетов и RT-мьютекс .....	619
Внутреннее устройство .....	620
Использование спин-блокировок .....	621
Спин-блокировка – простое использование .....	621
Спин-блокировка – пример драйвера .....	622
Тест – засыпание в атомарном контексте .....	624
Тестирование модуля с ошибками на отладочном ядре 6.1 .....	625
Блокировка и прерывания .....	630
Сценарий 1 – метод драйвера и обработчик аппаратного прерывания сериализованы .....	632
Сценарий 2 – метод драйвера и обработчик аппаратного прерывания чередуются .....	632
Сценарий 2 в одноядерной системе .....	632
Сценарий 2 в многоядерной SMP-системе .....	633
Решение проблемы в одноядерных и многоядерных системах с помощью функций spin_[un]lock_irq() .....	634
Сценарий 3 – некоторые прерывания замаскированы, метод драйвера и обработчик прерывания чередуются .....	635
Обработка прерывания, нижние половины и блокировка .....	637
Обработка прерываний в Linux – основные положения .....	637
Нижние половины и блокировка .....	638
Использование спин-блокировок – итоги .....	639

Блокировка – типичные ошибки и рекомендации .....	639
Типичные ошибки .....	640
Наставления по работе с блокировками .....	640
Решения .....	642
Резюме .....	642
Вопросы .....	643
Для дополнительного чтения .....	643

## **Глава 13. Синхронизация ядра – часть 2 ..... 644**

Использование типов <code>atomic_t</code> и <code>refcount_t</code> .....	644
Новый тип <code>refcount_t</code> и старый тип <code>atomic_t</code> .....	645
Работа с типами <code>atomic_t</code> и <code>refcount_t</code> .....	646
Примеры использования <code>refcount_t</code> в коде ядра .....	647
Атомарные операторы для 64-разрядных целых .....	650
Замечание о внутренней реализации .....	651
Использование атомарных RMW-операторов .....	652
Атомарные RMW-операции – работа с регистрами устройств .....	653
Использование поразрядных RMW-операторов .....	655
Пример использования атомарных поразрядных RMW-операторов ....	656
Эффективный поиск по битовой маске .....	659
Использование спин-блокировки чтения–записи .....	660
Интерфейсы блокировки чтения–записи .....	661
Применение спин-блокировки чтения–записи на практике .....	662
Проблемы производительности	
спин-блокировок чтения–записи .....	664
Семафор чтения–записи .....	665
Основы кеширования в CPU, эффекты кеширования и ложное	
разделение .....	666
Введение в процессорные кешы .....	666
Риски – когерентность кешей, проблемы производительности	
и ложное разделение .....	668
В чем состоит проблема когерентности кешей? .....	668
Проблема ложного разделения .....	671
Безблокировочное программирование с помощью переменных	
с копиями на каждом процессоре и RCU .....	675
Переменные с копиями на каждом процессоре .....	675
Работа с переменными с копиями на каждом процессоре .....	676
Пример использования переменных с копиями на каждом	
процессоре в модуле ядра .....	680
Примеры использования переменных с копиями	
на каждом процессоре в ядре .....	684
Введение в безблокировочную технологию RCU	
(прочитать–скопировать–обновить) .....	686
Как работает RCU? .....	687
Испытание RCU на практике .....	696
Простой пример – конкурентные читатели и писатели и их защита ....	697
RCU: подробная документация .....	706

---

Отладка блокировок в ядре.....	709
Конфигурирование ядра для отладки блокировок .....	710
Валидатор блокировок lockdep – раннее обнаружение ошибок работы с блокировками.....	713
Обнаружение потенциальных взаимоблокировок с помощью lockdep – несколько примеров.....	715
Пример 1 – обнаружение взаимоблокировки с собой .....	716
Пример 2 – обнаружение взаимоблокировки типа АВ-ВА с помощью lockdep .....	722
Краткие замечания о lockdep – аннотации и известные проблемы.....	727
К вопросу об аннотациях lockdep .....	727
К вопросу о lockdep – известные проблемы .....	728
Статистика блокировок ядра .....	729
Просмотр и интерпретация статистики блокировок в ядре .....	730
Введение в барьеры памяти.....	732
Пример использования барьеров памяти в драйвере устройства.....	733
К вопросу о маркированных операциях доступа.....	735
Резюме.....	735
Вопросы.....	736
Для дополнительного чтения .....	736
<b>Предметный указатель .....</b>	<b>737</b>

# Об авторе

Кайван Н. Биллимория учился кодированию на IBM PC своего отца (1983). Он программировал на C и ассемблере для DOS, пока не открыл для себя Unix, а вскоре после того и Linux! Кайван много работал в области системного программирования для Linux, включая драйверы и встраиваемые системы. Он принимал активное участие в нескольких коммерческих и свободных проектах с открытым исходным кодом. Является автором нескольких драйверов для Linux и большого числа проектов поменьше на GitHub. Его бесконечная любовь к Linux помогает ему преподавать эти предметы инженерам, что он успешно делает уже в течение двух десятков лет. Своим основным достижением он считает книги «Hands-On System Programming with Linux», «Linux Kernel Programming» (и ее вторую часть) и «Linux Kernel Debugging». На досуге любит заниматься бегом.

*В первую очередь посвящаю книгу своей чудесной семье: родителям Надсу и Диане, супруге Дилишад, детям Шерой и Данешу, брату Дариусу и всем остальным. Спасибо, что вы есть! Коллектив издательства Packt терпеливо и безупречно помогал на протяжении всего пути – как обычно. Отдельное спасибо Райанне Родригес, Ааарону Танна и Аникет Шетти – за своевременную поддержку от начала и до конца работы!*

# О рецензенте

Чи-Тхан Хоанг в настоящее время работает главным архитектором по ПО радиосвязи в компании Mavenir Systems, которая занимается радиосвязью стандарта O-RAN 5G. За его плечами более тридцати лет опыта разработок преимущественно в области встраиваемых систем (коммутаторы, маршрутизаторы, Wi-Fi и мобильные сети) – от чипсетов до протоколов связи и, конечно, ядра и ОСРВ. Впервые он познакомился с ядром Linux в 1993 году и до сих пор занимается отладкой на уровне ядра. Получил степень бакалавра по электротехнике в Шербрукском университете, Канада. Любит играть в теннис и постоянно возится с электроникой и программами.

# Предисловие от издательства

## Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте [www.dmkpress.com](http://www.dmkpress.com), зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу [dmkpress@gmail.com](mailto:dmkpress@gmail.com); при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу [http://dmkpress.com/authors/publish\\_book/](http://dmkpress.com/authors/publish_book/) или напишите в издательство по адресу [dmkpress@gmail.com](mailto:dmkpress@gmail.com).

## Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в основном тексте или программном коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу [dmkpress@gmail.com](mailto:dmkpress@gmail.com), и мы исправим это в следующих тиражах.

## Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу [dmkpress@gmail.com](mailto:dmkpress@gmail.com).

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.



Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

[e-Univers.ru](http://e-Univers.ru)