

Ядро Linux

Описание процесса
разработки

Linux Kernel Development

Third Edition

Robert Love



Addison
Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

Ядро Linux

Описание процесса разработки

Третье издание

Роберт Лав



Москва • Санкт-Петербург • Киев
2013

ББК 32.973.26-018.2.75

Л13

УДК 681.3.07

Издательский дом “Вильямс”

Зав. редакцией *С.Н. Тригуб*

Перевод с английского и редакция канд. физ.-мат. наук *С.Г. Тригуб*

По общим вопросам обращайтесь в Издательский дом “Вильямс” по адресу:
info@williamspublishing.com, <http://www.williamspublishing.com>

Лав, Роберт.

Л13 Ядро Linux: описание процесса разработки, 3-е изд. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2013. — 496 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-1779-9 (рус.)

ББК 32.973.26-018.2.75

Все названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм.

Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотокопирование и запись на магнитный носитель, если на это нет письменного разрешения издательства Addison-Wesley Publishing Company, Inc.

Authorized translation from the English language edition published by Addison-Wesley Publishing Company, Inc, Copyright © 2010

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise.

Russian language edition published by Williams Publishing House according to the Agreement with R&I Enterprises International, Copyright © 2013

Научно-популярное издание

Роберт Лав

Ядро Linux: описание процесса разработки

3-е издание

Литературный редактор *И.А. Попова*

Верстка *М.А. Удалов*

Художественный редактор *В.Г. Павлютин*

Корректор *Л.А. Гордиенко*

Подписано в печать 29.06.2012. Формат 70x100/16.

Гарнитура Times. Печать офсетная.

Усл. печ. л. 39,9. Уч.-изд. л. 32,1.

Тираж 1000 экз. Заказ № 0000.

Первая Академическая типография “Наука”

199034, Санкт-Петербург, 9-я линия, 12/28

ООО “И. Д. Вильямс”, 127055, г. Москва, ул. Лесная, д. 43, стр. 1

ISBN 978-5-8459-1779-9 (рус.)

ISBN 978-0-672-32946-3 (англ.)

© Издательский дом “Вильямс”, 2013

© Pearson Education, Inc., 2010

Оглавление

| | |
|---|------------|
| Введение | 19 |
| Об авторе | 23 |
| Глава 1. Понятие о ядре Linux | 25 |
| Глава 2. Начальные сведения о ядре Linux | 37 |
| Глава 3. Управление процессами | 51 |
| Глава 4. Системный планировщик и диспетчеризация процессов | 73 |
| Глава 5. Системные функции | 103 |
| Глава 6. Структуры данных ядра | 119 |
| Глава 7. Прерывания и их обработка | 147 |
| Глава 8. Нижняя половина обработчика и отложенные действия | 169 |
| Глава 9. Общие сведения о синхронизации кода ядра | 201 |
| Глава 10. Средства синхронизации ядра | 217 |
| Глава 11. Таймеры и управление временем | 253 |
| Глава 12. Управление памятью | 279 |
| Глава 13. Виртуальная файловая система | 311 |
| Глава 14. Уровень блочного ввода-вывода | 341 |
| Глава 15. Адресное пространство процесса | 359 |
| Глава 16. Страничный кеш и отложенная запись страниц | 379 |
| Глава 17. Устройства и модули | 395 |
| Глава 18. Отладка | 425 |
| Глава 19. Переносимость | 443 |
| Глава 20. Заплаты, хакерство и сообщество | 461 |
| Список литературы | 475 |
| Предметный указатель | 479 |

Содержание

| | |
|--|----|
| Предисловие | 17 |
| Введение | 19 |
| Итак... | 20 |
| Версия ядра | 20 |
| Читательская аудитория | 20 |
| Благодарности | 22 |
| Об авторе | 23 |
| От издательства | 24 |
| Глава 1. Понятие о ядре Linux | 25 |
| История систем Unix | 25 |
| Потом пришел Линус: введение в Linux | 27 |
| Обзор операционных систем и ядер | 29 |
| Отличие ядра Linux от классических ядер Unix | 31 |
| Версии ядра Linux | 34 |
| Сообщество разработчиков ядра Linux | 35 |
| Перед тем как начать | 36 |
| Глава 2. Начальные сведения о ядре Linux | 37 |
| Где взять исходный код ядра | 37 |
| Использование Git | 37 |
| Инсталляция исходного кода ядра | 38 |
| Использование заплат | 38 |
| Дерево каталогов исходных кодов ядра | 39 |
| Сборка ядра | 40 |
| Конфигурирование ядра | 40 |
| Уменьшение количества выводимых сообщений | 42 |
| Порождение нескольких параллельных задач сборки | 42 |
| Инсталляция нового ядра | 42 |
| Отличия от обычных приложений | 43 |
| Отсутствие библиотеки libc и стандартных заголовков | 44 |
| Компилятор GNU C | 45 |
| Отсутствие защиты памяти | 47 |
| Нельзя просто использовать вычисления с плавающей точкой | 47 |
| Системная стек-память небольшого фиксированного размера | 47 |
| Синхронизация и параллельное выполнение | 48 |
| Переносимость — это важно | 48 |
| Резюме | 49 |

| | |
|---|-----|
| Глава 3. Управление процессами | 51 |
| Понятие процесса | 51 |
| Дескриптор процесса и структура <code>task_struct</code> | 53 |
| Распределение памяти под дескриптор процесса | 53 |
| Сохранение дескриптора процесса | 55 |
| Состояние процесса | 56 |
| Изменение текущего состояния процесса | 58 |
| Контекст процесса | 58 |
| Дерево семейства процессов | 58 |
| Создание нового процесса | 60 |
| Копирование при записи | 60 |
| Функция <code>fork()</code> | 61 |
| Функция <code>vfork()</code> | 62 |
| Реализация потоков в ядре Linux | 63 |
| Создание потоков | 64 |
| Потоки в ядре | 65 |
| Завершение процесса | 66 |
| Удаление дескриптора процесса | 68 |
| Дилемма “безпризорного” процесса | 68 |
| Резюме | 70 |
| | |
| Глава 4. Системный планировщик и диспетчеризация процессов | 73 |
| Мультипрограммный режим работы | 73 |
| Системный планировщик Linux | 75 |
| Стратегия планирования | 76 |
| Процессы, ориентированные на ввод-вывод и на вычисления | 76 |
| Приоритет процессов | 77 |
| Кванты времени | 78 |
| Стратегия планирования в действии | 79 |
| Алгоритм работы планировщика системы Linux | 80 |
| Классы планировщика | 80 |
| Планирование процессов в системах Unix | 81 |
| Справедливое планирование задач | 83 |
| Реализация планировщика в системе Linux | 85 |
| Учет времени | 85 |
| Выбор процесса | 87 |
| Точка входа в планировщик | 91 |
| Замораживание и активизация процессов | 92 |
| Вытеснение и переключение контекста | 96 |
| Вытеснение пространства пользователя | 97 |
| Вытеснение пространства ядра | 98 |
| Стратегии планирования в режиме реального времени | 99 |
| Системные функции для управления планировщиком | 100 |
| Системные функции для изменения стратегии и приоритета | 101 |
| Системные функции для изменения привязки к процессору | 101 |
| Передача процессорного времени другим задачам | 102 |
| Резюме | 102 |

8 Содержание

| | |
|--|-----|
| Глава 5. Системные функции | 103 |
| Взаимодействие с ядром | 103 |
| API, POSIX и библиотека C | 104 |
| Системные функции | 105 |
| Номера системных функций | 106 |
| Быстродействие системных функций | 107 |
| Обработчик вызова системных функций | 107 |
| Как определить, какую системную функцию вызвать | 108 |
| Передача параметров | 109 |
| Реализация системных функций | 109 |
| Разработка системных функций | 109 |
| Проверка параметров | 110 |
| Контекст системной функции | 113 |
| Завершающие этапы регистрации системной функции | 114 |
| Доступ к системным функциям из пользовательских приложений | 116 |
| Почему не нужно создавать системные функции | 117 |
| Резюме | 118 |
| Глава 6. Структуры данных ядра | 119 |
| Связанные списки | 119 |
| Однонаправленный и двунаправленный связанный список | 120 |
| Циклически связанные списки | 120 |
| Перемещение по элементам связанного списка | 121 |
| Реализация в ядре Linux | 122 |
| Работа со связанными списками | 124 |
| Обход элементов связанного списка | 127 |
| Очереди | 130 |
| Система kfifo | 131 |
| Создание очереди | 132 |
| Постановка в очередь | 132 |
| Выборка из очереди | 132 |
| Определение размера очереди | 133 |
| Очистка и удаление очереди | 133 |
| Примеры использования очередей | 134 |
| Таблицы отображения | 134 |
| Инициализация структуры idr | 135 |
| Выделение нового UID | 135 |
| Поиск UID | 137 |
| Удаление UID | 137 |
| Аннулирование idr | 137 |
| Двоичные деревья | 138 |
| Двоичные деревья поиска | 138 |
| Самобалансирующиеся двоичные деревья поиска | 139 |
| Красно-черные деревья | 139 |
| Реализация в Linux | 140 |
| Какие структуры данных следует использовать, если... | 142 |
| Алгоритмическая сложность | 143 |
| Что такое алгоритм? | 143 |

| | |
|---|------------|
| Понятие большого “О” | 144 |
| Понятие большой “тета” | 144 |
| Временная сложность алгоритма | 145 |
| Резюме | 146 |
| Глава 7. Прерывания и их обработка | 147 |
| Прерывания | 147 |
| Обработчики прерываний | 149 |
| Верхняя и нижняя половины | 150 |
| Регистрация обработчика прерывания | 150 |
| Флаги обработчика прерываний | 151 |
| Остальные параметры функции обработки прерывания | 152 |
| Пример обработчика прерывания | 153 |
| Освобождение обработчика прерывания | 153 |
| Написание обработчика прерывания | 154 |
| Обработчики общих запросов на прерывание | 155 |
| Пример настоящего обработчика прерывания | 156 |
| Контекст прерывания | 158 |
| Реализация системы обработки прерываний | 159 |
| Интерфейс /proc/interrupts | 162 |
| Управление прерываниями | 163 |
| Запрещение и разрешение прерываний | 164 |
| Запрещение заданной линии IRQ | 165 |
| Состояние системы обработки прерываний | 166 |
| Резюме | 167 |
| Глава 8. Нижняя половина обработчика и отложенные действия | 169 |
| Нижняя половина | 170 |
| Когда используется нижняя половина обработчика | 171 |
| Многообразие нижних половин | 171 |
| Отложенные прерывания | 174 |
| Реализация механизма отложенных прерываний | 175 |
| Использование отложенных прерываний | 177 |
| Тасклеты | 179 |
| Реализация тасклетов | 179 |
| Использование тасклетов | 182 |
| Демон ksoftirqd | 184 |
| Старый механизм ВН | 186 |
| Очереди отложенных действий | 187 |
| Реализация очередей отложенных действий | 188 |
| Использование очередей отложенных действий | 191 |
| Старый механизм очередей задач | 194 |
| Какие механизмы обработчиков нижних половин следует использовать | 195 |
| Блокировки между нижними половинами обработчиков | 196 |
| Запрещение обработки нижних половин | 197 |
| Резюме | 199 |

10 Содержание

| | |
|--|-----|
| Глава 9. Общие сведения о синхронизации кода ядра | 201 |
| Критические участки и конфликты из-за доступа к системным ресурсам | 202 |
| Зачем вообще нужно что-то защищать? | 202 |
| Общая переменная | 204 |
| Блокировки | 205 |
| Причины возникновения параллелизма | 207 |
| Что нужно защищать? | 209 |
| Взаимоблокировки | 210 |
| Конфликт при блокировке и масштабируемость | 213 |
| Резюме | 215 |
| | |
| Глава 10. Средства синхронизации ядра | 217 |
| Неделимые операции | 217 |
| Неделимые целочисленные операции | 218 |
| 64-разрядные неделимые операции | 222 |
| Неделимые битовые операции | 223 |
| Спин-блокировки | 225 |
| Функции для спин-блокировки | 227 |
| Другие средства работы со спин-блокировками | 229 |
| Спин-блокировки и нижние половины обработчиков прерываний | 230 |
| Спин-блокировки по чтению-записи | 230 |
| Семафоры | 233 |
| Счетные и бинарные семафоры | 234 |
| Создание и инициализация семафоров | 235 |
| Использование семафоров | 236 |
| Семафоры для чтения-записи | 237 |
| Мьютексы | 238 |
| Сравнение семафоров и мьютексов | 240 |
| Сравнение спин-блокировок и мьютексов | 240 |
| Условные переменные | 241 |
| ВКЛ: большая блокировка ядра | 242 |
| Последовательные блокировки | 243 |
| Отключение мультипрограммного режима работы ядра | 245 |
| Порядок выполнения операций и барьеры | 247 |
| Резюме | 251 |
| | |
| Глава 11. Таймеры и управление временем | 253 |
| Основная идея учета времени в ядре | 254 |
| Частота импульсов таймера: директива HZ | 255 |
| Идеальное значение параметра HZ | 256 |
| Преимущества больших значений параметра HZ | 257 |
| Недостатки больших значений параметра HZ | 258 |
| Переменная jiffies | 259 |
| Внутреннее представление переменной jiffies | 260 |
| Переполнение переменной jiffies | 261 |
| Пользовательские программы и параметр HZ | 263 |
| Аппаратные часы и таймеры | 264 |
| Часы реального времени | 264 |

| | |
|---|------------|
| Системный таймер | 264 |
| Обработчик прерываний от таймера | 265 |
| Абсолютное время | 267 |
| Таймеры | 269 |
| Использование таймеров | 270 |
| Конфликты из-за доступа к ресурсам при использовании таймеров | 272 |
| Реализация таймеров | 272 |
| Задержка выполнения | 273 |
| Задержка с помощью цикла | 273 |
| Короткие задержки | 274 |
| Функция <code>schedule_timeout()</code> | 276 |
| Резюме | 278 |
| Глава 12. Управление памятью | 279 |
| Страничная организация памяти | 279 |
| Зоны | 281 |
| Выделение страниц памяти | 284 |
| Выделение обнуленных страниц памяти | 284 |
| Освобождение страниц памяти | 285 |
| Функция <code>kmalloc()</code> | 286 |
| Флаги <code>gfp_mask</code> | 287 |
| Функция <code>kfree()</code> | 292 |
| Функция <code>vmalloc()</code> | 292 |
| Уровень блочного распределения памяти | 294 |
| Структура уровня блочного распределения памяти | 295 |
| Интерфейс блочного распределителя памяти | 298 |
| Пример использования блочного распределителя памяти | 300 |
| Статическое выделение памяти в стеке | 301 |
| Одностраничные стеки ядра | 302 |
| Справедливое использование стека | 303 |
| Отображение верхней памяти | 303 |
| Постоянное отображение | 303 |
| Временное отображение | 304 |
| Выделение памяти для конкретного процессора | 305 |
| Новый интерфейс <code>regsrcu</code> | 306 |
| Работа с процессорными данными на этапе компиляции | 306 |
| Работа с процессорными данными на этапе выполнения | 307 |
| Когда лучше использовать данные, связанные с процессорами | 308 |
| Выбор способа выделения памяти | 309 |
| Резюме | 310 |
| Глава 13. Виртуальная файловая система | 311 |
| Общий интерфейс файловых систем | 312 |
| Абстрактный уровень файловой системы | 312 |
| Файловые системы Unix | 314 |
| Объекты VFS и их структуры данных | 315 |
| Объект суперблок | 317 |
| Операции суперблока | 318 |

12 Содержание

| | |
|---|------------|
| Объект <code>inode</code> | 320 |
| Операции с файловыми индексами | 322 |
| Объект элемента каталога (<code>dentry</code>) | 325 |
| Состояние элементов каталога | 326 |
| Кеш объектов <code>dentry</code> | 327 |
| Операции с элементами каталогов | 328 |
| Файловый объект | 329 |
| Файловые операции | 330 |
| Структуры данных, связанные с файловыми системами | 335 |
| Структуры данных, связанные с процессом | 337 |
| Резюме | 339 |
| | |
| Глава 14. Уровень блочного ввода-вывода | 341 |
| Структура блочного устройства | 342 |
| Буферы и их заголовки | 343 |
| Структура <code>bio</code> | 346 |
| Векторы ввода-вывода | 347 |
| Сравнение старой и новой реализаций | 348 |
| Очереди запросов | 349 |
| Планировщики ввода-вывода | 350 |
| Задачи планировщика ввода-вывода | 350 |
| Лифт имени Линуса | 351 |
| Планировщик ввода-вывода с ограничением по времени | 353 |
| Прогнозирующий планировщик ввода-вывода | 355 |
| Планировщик ввода-вывода с полностью равноправными очередями | 356 |
| Планировщик ввода-вывода с отсутствием операций (<code>Noop</code>) | 357 |
| Выбор планировщика ввода-вывода | 357 |
| Резюме | 358 |
| | |
| Глава 15. Адресное пространство процесса | 359 |
| Адресные пространства | 359 |
| Дескриптор памяти | 361 |
| Выделение дескриптора памяти | 363 |
| Удаление дескриптора памяти | 363 |
| Структура <code>mm_struct</code> и потоки ядра | 364 |
| Области виртуальной памяти | 364 |
| Флаги областей <code>VMA</code> | 365 |
| Операции с областями <code>VMA</code> | 367 |
| Списки и деревья областей памяти | 368 |
| Области памяти в реальных приложениях | 369 |
| Работа с областями памяти | 371 |
| Функция <code>find_vma()</code> | 371 |
| Функция <code>find_vma_prev()</code> | 372 |
| Функция <code>find_VMA_intersection()</code> | 372 |
| Функции <code>mmap()</code> и <code>do_mmap()</code> : создание диапазона адресов | 373 |
| Функции <code>munmap()</code> и <code>do_munmap()</code> : удаление диапазона адресов | 375 |
| Таблицы страниц | 375 |
| Резюме | 377 |

| | |
|--|-----|
| Глава 16. Страничный кеш и отложенная запись страниц | 379 |
| Методики кеширования | 380 |
| Кеширование при записи | 380 |
| Вытеснение данных из кеша | 381 |
| Реализация страничного кеша в ОС Linux | 383 |
| Объект <code>address_space</code> | 383 |
| Операции объекта <code>address_space</code> | 385 |
| Базисное дерево | 387 |
| Старая хеш-таблица страниц | 387 |
| Буферный кеш | 388 |
| Потоки синхронизатора | 388 |
| Режим ноутбука | 390 |
| Экскурс в историю: <code>bdflush</code> , <code>kupdated</code> и <code>pdflush</code> | 391 |
| Предотвращение перегрузки с помощью нескольких потоков | 392 |
| Резюме | 393 |
| | |
| Глава 17. Устройства и модули | 395 |
| Типы устройств | 395 |
| Модули | 396 |
| Модуль “Hello, World!” | 397 |
| Сборка модулей | 398 |
| Установка модулей | 401 |
| Генерация зависимостей между модулями | 401 |
| Загрузка модулей | 401 |
| Поддержка параметров конфигурации | 402 |
| Параметры модулей | 404 |
| Экспортируемые символы | 406 |
| Модель представления устройств | 407 |
| Объекты <code>kobject</code> | 408 |
| Типы <code>ktype</code> | 409 |
| Множества объектов <code>kset</code> | 410 |
| Взаимосвязь <code>kobject</code> , <code>ktype</code> и <code>kset</code> | 410 |
| Управление и работа с объектами <code>kobject</code> | 411 |
| Счетчики ссылок | 412 |
| Файловая система <code>sysfs</code> | 414 |
| Добавление и удаление объектов файловой системы <code>sysfs</code> | 417 |
| Добавление файлов в файловую систему <code>sysfs</code> | 418 |
| Уровень событий ядра | 421 |
| Резюме | 423 |
| | |
| Глава 18. Отладка | 425 |
| Начало работы | 425 |
| Ошибки ядра | 426 |
| Отладка с помощью вывода диагностических сообщений | 427 |
| Устойчивость | 427 |
| Уровни вывода сообщений ядра | 428 |
| Буфер сообщений ядра | 429 |
| Демоны <code>syslogd</code> и <code>klogd</code> | 429 |

14 Содержание

| | |
|--|-----|
| Взаимозаменяемость функций printf() и printk() | 430 |
| Сообщения Oops | 430 |
| Утилита ksymoops | 431 |
| Функция kallsyms | 432 |
| Параметры конфигурации для отладки ядра | 433 |
| Объявление об ошибках и выдача информации | 433 |
| “Магическая” клавиша <SysRq> | 434 |
| Сага об отладчике ядра | 435 |
| Отладчик gdb | 436 |
| Отладчик kgdb | 436 |
| Исследование и тестирование системы | 437 |
| Использование идентификатора UID в качестве условия | 437 |
| Использование условных переменных | 437 |
| Использование статистики | 438 |
| Ограничение частоты следования и общего количества событий при отладке | 438 |
| Поиск методом половинного деления изменений, приводящим к ошибкам | 439 |
| Поиск с помощью git | 440 |
| Если ничто не помогает — обратитесь к сообществу | 441 |
| Резюме | 441 |

Глава 19. Переносимость 443

| | |
|--|-----|
| Переносимые операционные системы | 443 |
| История переносимости Linux | 445 |
| Размер машинного слова и типы данных | 446 |
| Скрытые типы данных | 449 |
| Специальные типы данных | 449 |
| Типы с явным указанием размера | 450 |
| Знаковые и беззнаковые типы char | 451 |
| Выравнивание данных | 451 |
| Как избежать проблем с выравниванием | 452 |
| Выравнивание нестандартных типов данных | 452 |
| Пустые поля структур | 453 |
| Порядок следования байтов | 454 |
| Учет времени | 456 |
| Размер страницы памяти | 457 |
| Порядок выполнения операций процессором | 458 |
| Многопроцессорность, мультипрограммирование и верхняя память | 458 |
| Резюме | 459 |

Глава 20. Заплаты, хакерство и сообщество 461

| | |
|--------------------------------|-----|
| Сообщество | 461 |
| Стиль написания исходного кода | 462 |
| Отступы | 462 |
| Оператор switch | 463 |
| Пробелы | 463 |
| Фигурные скобки | 464 |
| Длина строки исходного кода | 465 |
| Соглашения о присвоении имен | 466 |

| | Содержание | 15 |
|---|-------------------|------------|
| Функции | | 466 |
| Комментарии | | 466 |
| Использование директивы typedef | | 467 |
| Использование того, что уже есть | | 468 |
| Избегайте директив ifdef в исходном коде | | 468 |
| Инициализация структур | | 468 |
| Исправление ранее написанного кода | | 469 |
| Организация команды разработчиков | | 469 |
| Отправка сообщений об ошибках | | 470 |
| Заплаты | | 470 |
| Генерация заплат | | 470 |
| Генерирование заплат с помощью программы Git | | 471 |
| Публикация заплат | | 472 |
| Резюме | | 473 |
| Список литературы | | 475 |
| Книги по основам построения операционных систем | | 475 |
| Книги о ядре Unix | | 476 |
| Книги о ядре Linux | | 476 |
| Книги о ядрах других операционных систем | | 476 |
| Книги по API Unix | | 477 |
| Книги по программированию на языке C | | 477 |
| Другие работы | | 477 |
| Веб-сайты | | 478 |
| Предметный указатель | | 479 |

◆
Посвящается Дорис (Doris) и Элен (Helen)
◆

Предисловие

В связи с тем, что ядро и приложения операционной системы Linux используются все более широко, возрастает число разработчиков системного программного обеспечения, желающих заняться разработкой и поддержкой операционной системы Linux. Часть из этих разработчиков руководствуется исключительно собственным интересом, часть — работает в компаниях, которые занимаются операционной системой Linux, часть — работает на производителей компьютерных аппаратных средств, часть — занята в проектах по разработке программного обеспечения на дому.

Однако все они сталкиваются с общей проблемой: кривая затрат на изучение ядра становится все длиннее и круче. Система становится все более сложной и, кроме того, очень большой по объему. Годы проходят, и нынешние члены команды разработчиков ядра приобретают все более широкие и глубокие знания, что увеличивает разрыв между ними и разработчиками-новичками.

Я уверен, что понимание основного кода ядра Linux уже сейчас является проблемой, приводящей к ухудшению качества ядра, и в будущем эта проблема станет еще более серьезной. Все, кому нравится операционная система Linux, несомненно, заинтересованы в увеличении числа разработчиков, которые смогут внести свой вклад в развитие ядра этой операционной системы.

Один из возможных подходов к решению данной проблемы — ясность исходного кода: удобные интерфейсы, четкая структура, следование принципу “Лучше меньше, да лучше” и т.д. Такое решение предложено Линусом Торвальдсом (Linus Torvalds).

Подход, который предлагаю я, состоит в использовании большего числа комментариев в исходном коде, что поможет читателю понять, чего хотел достичь программист. (Процесс выявления расхождений между поставленной программистом целью и полученной в результате реализацией называется *отладкой*. Этот процесс значительно затрудняется, если не известно, чего хотели достичь.)

Однако комментарии все же не дают представления о том, для чего предназначено большинство подсистем и как разработчики приступали к их реализации. Именно печатное слово лучше всего подходит для отправной точки такого понимания.

Вклад Роберта Лава (Robert Love) состоит в предоставлении возможности, благодаря которой опытные разработчики смогут получить полную информацию о том, какие функции должны выполнять различные подсистемы ядра и каким образом предполагается выполнение этих функций. Этой информации должно быть достаточно для многих людей: для любопытных, для разработчиков прикладного программного обеспечения, для тех, кто хочет ознакомиться с устройством ядра, и т.д.

Кроме того, данная книга является ступенькой, которая поможет начинающим разработчикам перейти на новый уровень, где изменения в ядро вносятся для того, чтобы достичь определенной цели. Я хотел бы посоветовать начинающим разработчикам, чтобы они не боялись испачкать свои руки: наилучший способ понять какую-либо часть ядра —

18 Предисловие

это внести в нее изменения. Внесение изменений повышает понимание разработчика до уровня, которого нельзя достичь простым чтением кода ядра. Серьезный разработчик ядра присоединится к спискам рассылки разработчиков и будет контактировать с другими коллегами. Это основной способ, позволяющий разработчикам учиться и быть на высоком уровне. Роберт очень хорошо осветил механизмы и культуру этой важной части жизни сообщества разработчиков ядра.

Пользуйтесь книгой Роберта и учитесь по ней! Может быть, и вы решите сделать следующий шаг и вступить в сообщество разработчиков ядра, куда мы вас и приглашаем. Людей ценят по важности их дел, поэтому, помогая развитию операционной системы Linux, знайте, что ваша работа — небольшая, но непосредственная помощь десяткам и даже сотням миллионов людей.

*Эндрю Мортон (Andrew Morton)
Open Source Development Labs*

Введение

Сделав первую попытку превратить свой опыт работы с ядром Linux в текст книги, я понял, что не знаю, куда двигаться дальше. Не хотелось просто писать еще одну книгу о ядре операционной системы. Конечно, на эту тему написано *не так уж и много* книг, но все же я хотел сделать что-то такое, благодаря чему моя книга была бы особенной. Как достичь этой цели? Я не могу успокоиться, пока не сделаю что-нибудь особенное, лучшее в своем роде.

Наконец я решил, что смогу предложить достаточно уникальный подход к данной теме. Моя работа — изучение и разработка ядра операционной системы. Мое увлечение — изучение и разработка ядра операционной системы. Моя любовь — ядро операционной системы. Конечно, за многие годы я успел собрать много интересных анекдотов и полезных советов. С моим опытом я смог бы написать книгу о том, как нужно разрабатывать программный код ядра и как этого делать *не нужно*. Прежде всего, эта книга о структуре и практической реализации ядра операционной системы Linux. Информация в ней представлена так, чтобы получить достаточно знаний для решения реальных практических задач и решать эти задачи правильно. Я человек прагматичный, и книга имеет практический уклон. Она должна быть полезной, интересной и легко читаться.

Я надеюсь, что читатели после прочтения этой книги получат хорошее понимание тех правил (писанных и неписанных), которые действуют в ядре операционной системы. Надеюсь также, что читатели сразу после прочтения этой книги смогут начать действовать и писать полезный, правильный и хороший код ядра. Конечно, эту книгу можно читать и просто ради интереса.

Это то, что касалось еще первого издания книги. Однако время идет, и снова приходится возвращаться к рассмотренным вопросам. В этом, третьем по счету, издании представлено несколько больше информации по сравнению с двумя остальными: материал серьезно пересмотрен и доработан, появились новые разделы и главы. С момента выхода второго издания в ядро были внесены изменения. Но, что более важно, сообщество разработчиков ядра Linux приняло решение¹ в ближайшем будущем не начинать разработку ядра версии 2.7. Было решено заняться стабилизацией ядра версии 2.6. Стабилизация включает в себя много моментов, тем не менее есть один важный, касающийся данной книги, — книга, которая посвящена ядру серии 2.6, остается актуальной до сих пор. Поскольку изменения происходят не слишком быстро, существует большой шанс, что “моментальный снимок” ядра останется актуальным и в будущем. Эта книга, по сути, стала канонической документацией по ядру, в которой отражены как история, так и взгляд в будущее.

¹ Это решение было принято на саммите разработчиков ядра Linux (Linux Kernel Development Summit), который состоялся летом 2004 года в Оттаве, Канада. Меня пригласили туда в качестве участника.

Итак...

Разработка программного кода ядра операционной системы не требует наличия гениальной, волшебной или густой бороды Unix-хакера. Хотя ядро операционной системы и имеет некоторые свои особенности, оно незначительно отличается от любого большого программного продукта. Так же как и в случае любой сложной программы, здесь есть, что изучать, но в программировании ядра не намного больше священных или непонятных вещей, чем в создании любой другой программы.

Очень важно, чтобы вы читали программный код. Доступность открытого исходного кода операционной системы Linux — это подарок, который встречается очень редко. Однако недостаточно *только* читать исходный код. Необходимо взяться за дело серьезно и изменять этот программный код. Находите ошибки и исправляйте их! Улучшайте драйверы для своего аппаратного обеспечения! Добавляйте новые функциональные возможности в ядро, даже если они на первый взгляд кажутся весьма простыми. Находите слабые места и закрывайте их! У вас все получится, если вы будете сами *писать* программный код.

Версия ядра

В этой книге рассмотрено ядро Linux версии 2.6. Я не стал описывать устаревшие версии ядра, за исключением случаев, представляющих чисто исторический интерес. Например, мы рассмотрим, как были реализованы определенные подсистемы в ядре версии 2.4, поскольку тогда они были намного проще, и сможем легче во всем разобраться. Что касается данной книги, то она была написана на момент, когда актуальным являлось ядро Linux версии 2.6.34. Ядро — это “движущийся объект”, и никакая книга не в состоянии передать динамику во все моменты времени. Тем не менее базовые внутренние структуры ядра уже сформировались, и основные усилия по представлению материала были направлены на то, чтобы этот материал можно было использовать и в будущем.

Несмотря на то что в этой книге описано ядро версии 2.6.34, я постарался сделать так, чтобы в ней также были максимально корректно отражены сведения по ядру версии 2.6.32. Дело в том, что эта устаревшая версия была официально одобрена в качестве “корпоративного” ядра, которое продолжает использоваться во многих дистрибутивах Linux. А это значит, что мы еще долго будем встречать его во многих производственных системах и оно еще много лет будет находиться в состоянии активной разработки. (Существовавшие ранее ядра версий 2.6.9, 2.6.18 и 2.6.27 — аналогичные примеры “долгожителей”.)

Читательская аудитория

Эта книга предназначена для разработчиков программного обеспечения, которые хотят понять, как устроено ядро операционной системы Linux. Тем не менее она *не* является сборником построчных комментариев, извлеченных из исходного кода ядра. Ее также нельзя считать руководством по разработке драйверов или справочником по программному интерфейсу (API) ядра. Целью книги является предоставление достаточной информации о структуре и реализации ядра, чтобы подготовленный программист смог начать разработку программного кода. Разработка ядра может быть увлекательным и полезным занятием, и я хочу ознакомить читателя с этой сферой деятельности по возможности быстро. В книге обсуждаются как вопросы теории, так и практические приложения, она об-