

Robert Love

Third Edition



Linux Kernel Development

A thorough guide to the design and implementation of the Linux kernel

Developer's Library



Linux Kernel Development

Third Edition

Linux Kernel Development

Table of Contents

Table of Contents

1 Introduction to the Linux Kernel

- History of Unix

- Along Came Linus: Introduction to Linux

- Overview of Operating Systems and Kernels

- Linux Versus Classic Unix Kernels

- Linux Kernel Versions

- The Linux Kernel Development Community

- Before We Begin

2 Getting Started with the Kernel

- Obtaining the Kernel Source

 - Using Git

 - Installing the Kernel Source

 - Using Patches

- The Kernel Source Tree

- Building the Kernel

 - Configuring the Kernel

 - Minimizing Build Noise

 - Spawning Multiple Build Jobs

 - Installing the New Kernel

- A Beast of a Different Nature

 - No libc or Standard Headers

 - GNU C

Table of Contents

- No Memory Protection
- No (Easy) Use of Floating Point
- Small, Fixed-Size Stack
- Synchronization and Concurrency
- Importance of Portability

Conclusion

3 Process Management

The Process

Process Descriptor and the Task Structure

- Allocating the Process Descriptor
- Storing the Process Descriptor
- Process State
- Manipulating the Current Process State
- Process Context
- The Process Family Tree

Process Creation

- Copy-on-Write
- Forking
- vfork()

The Linux Implementation of Threads

- Creating Threads
- Kernel Threads

Process Termination

- Removing the Process Descriptor
- The Dilemma of the Parentless Task

Conclusion

4 Process Scheduling

Table of Contents

Multitasking

Linuxs Process Scheduler

Policy

I/O-Bound Versus Processor-Bound Processes

Process Priority

Timeslice

The Scheduling Policy in Action

The Linux Scheduling Algorithm

Scheduler Classes

Process Scheduling in Unix Systems

Fair Scheduling

The Linux Scheduling Implementation

Time Accounting

Process Selection

The Scheduler Entry Point

Sleeping and Waking Up

Preemption and Context Switching

User Preemption

Kernel Preemption

Real-Time Scheduling Policies

Scheduler-Related System Calls

Scheduling Policy and Priority-Related System Calls

Processor Affinity System Calls

Yielding Processor Time

Conclusion

5 System Calls

Communicating with the Kernel

APIs, POSIX, and the C Library

Table of Contents

Syscalls

- System Call Numbers

- System Call Performance

System Call Handler

- Denoting the Correct System Call

- Parameter Passing

System Call Implementation

- Implementing System Calls

- Verifying the Parameters

System Call Context

- Final Steps in Binding a System Call

- Accessing the System Call from User-Space

- Why Not to Implement a System Call

Conclusion

6 Kernel Data Structures

Linked Lists

- Singly and Doubly Linked Lists

- Circular Linked Lists

- Moving Through a Linked List

- The Linux Kernels Implementation

- Manipulating Linked Lists

- Traversing Linked Lists

Queues

- kfifo

- Creating a Queue

- Enqueuing Data

- Dequeuing Data

- Obtaining the Size of a Queue

Table of Contents

Resetting and Destroying the Queue

Example Queue Usage

Maps

Initializing an idr

Allocating a New UID

Looking Up a UID

Removing a UID

Destroying an idr

Binary Trees

Binary Search Trees

Self-Balancing Binary Search Trees

What Data Structure to Use, When

Algorithmic Complexity

Algorithms

Big-O Notation

Big Theta Notation

Time Complexity

Conclusion

7 Interrupts and Interrupt Handlers

Interrupts

Interrupt Handlers

Top Halves Versus Bottom Halves

Registering an Interrupt Handler

Interrupt Handler Flags

An Interrupt Example

Freeing an Interrupt Handler

Writing an Interrupt Handler

Table of Contents

Shared Handlers

A Real-Life Interrupt Handler

Interrupt Context

Implementing Interrupt Handlers

/proc/interrupts

Interrupt Control

Disabling and Enabling Interrupts

Disabling a Specific Interrupt Line

Status of the Interrupt System

Conclusion

8 Bottom Halves and Deferring Work

Bottom Halves

Why Bottom Halves?

A World of Bottom Halves

Softirqs

Implementing Softirqs

Using Softirqs

Tasklets

Implementing Tasklets

Using Tasklets

ksoftirqd

The Old BH Mechanism

Work Queues

Implementing Work Queues

Using Work Queues

The Old Task Queue Mechanism

Which Bottom Half Should I Use?

Locking Between the Bottom Halves

Table of Contents

Disabling Bottom Halves

Conclusion

9 An Introduction to Kernel Synchronization

Critical Regions and Race Conditions

Why Do We Need Protection?

The Single Variable

Locking

Causes of Concurrency

Knowing What to Protect

Deadlocks

Contention and Scalability

Conclusion

10 Kernel Synchronization Methods

Atomic Operations

Atomic Integer Operations

64-Bit Atomic Operations

Atomic Bitwise Operations

Spin Locks

Spin Lock Methods

Other Spin Lock Methods

Spin Locks and Bottom Halves

Reader-Writer Spin Locks

Semaphores

Counting and Binary Semaphores

Creating and Initializing Semaphores

Using Semaphores

Reader-Writer Semaphores

Table of Contents

Mutexes

- Semaphores Versus Mutexes

- Spin Locks Versus Mutexes

Completion Variables

BKL: The Big Kernel Lock

Sequential Locks

Preemption Disabling

Ordering and Barriers

Conclusion

11 Timers and Time Management

Kernel Notion of Time

The Tick Rate: HZ

- The Ideal HZ Value

- Advantages with a Larger HZ

- Disadvantages with a Larger HZ

Jiffies

- Internal Representation of Jiffies

- Jiffies Wraparound

- User-Space and HZ

Hardware Clocks and Timers

- Real-Time Clock

- System Timer

The Timer Interrupt Handler

The Time of Day

Timers

- Using Timers

- Timer Race Conditions

Table of Contents

Timer Implementation

Delaying Execution

Busy Looping

Small Delays

schedule_timeout()

Conclusion

12 Memory Management

Pages

Zones

Getting Pages

Getting Zeroed Pages

Freeing Pages

kmalloc()

gfp_mask Flags

kfree()

vmalloc()

Slab Layer

Design of the Slab Layer

Slab Allocator Interface

Statically Allocating on the Stack

Single-Page Kernel Stacks

Playing Fair on the Stack

High Memory Mappings

Permanent Mappings

Temporary Mappings

Per-CPU Allocations

The New percpu Interface

Table of Contents

Per-CPU Data at Compile-Time

Per-CPU Data at Runtime

Reasons for Using Per-CPU Data

Picking an Allocation Method

Conclusion

13 The Virtual Filesystem

Common Filesystem Interface

Filesystem Abstraction Layer

Unix Filesystems

VFS Objects and Their Data Structures

The Superblock Object

Superblock Operations

The Inode Object

Inode Operations

The Dentry Object

Dentry State

The Dentry Cache

Dentry Operations

The File Object

File Operations

Data Structures Associated with Filesystems

Data Structures Associated with a Process

Conclusion

14 The Block I/O Layer

Anatomy of a Block Device

Buffers and Buffer Heads

Table of Contents

The bio Structure

- I/O vectors

- The Old Versus the New

Request Queues

I/O Schedulers

- The Job of an I/O Scheduler

- The Linus Elevator

- The Deadline I/O Scheduler

- The Anticipatory I/O Scheduler

- The Complete Fair Queuing I/O Scheduler

- The Noop I/O Scheduler

- I/O Scheduler Selection

Conclusion

15 The Process Address Space

Address Spaces

The Memory Descriptor

- Allocating a Memory Descriptor

- Destroying a Memory Descriptor

- The mm_struct and Kernel Threads

Virtual Memory Areas

- VMA Flags

- VMA Operations

- Lists and Trees of Memory Areas

- Memory Areas in Real Life

Manipulating Memory Areas

- find_vma()

- find_vma_prev()

- find_vma_intersection()

Table of Contents

mmap() and do_mmap(): Creating an Address Interval

munmap() and do_munmap(): Removing an Address Interval

Page Tables

Conclusion

16 The Page Cache and Page Writeback

Approaches to Caching

Write Caching

Cache Eviction

The Linux Page Cache

The address_space Object

address_space Operations

Radix Tree

The Old Page Hash Table

The Buffer Cache

The Flusher Threads

Laptop Mode

History: bdflush, kupdated, and pdflush

Avoiding Congestion with Multiple Threads

Conclusion

17 Devices and Modules

Device Types

Modules

Hello, World!

Building Modules

Installing Modules

Generating Module Dependencies

Loading Modules

Table of Contents

Managing Configuration Options

Module Parameters

Exported Symbols

The Device Model

Kobjects

Ktypes

Ksets

Interrelation of Kobjects, Ktypes, and Ksets

Managing and Manipulating Kobjects

Reference Counts

sysfs

Adding and Removing kobjects from sysfs

Adding Files to sysfs

The Kernel Events Layer

Conclusion

18 Debugging

Getting Started

Bugs in the Kernel

Debugging by Printing

Robustness

Loglevels

The Log Buffer

syslogd and klogd

Transposing printf() and printk()

Oops

ksymoops

kallsyms

Kernel Debugging Options

Table of Contents

Asserting Bugs and Dumping Information

Magic SysRq Key

The Saga of a Kernel Debugger

gdb

kgdb

Poking and Probing the System

Using UID as a Conditional

Using Condition Variables

Using Statistics

Rate and Occurrence Limiting Your Debugging

Binary Searching to Find the Culprit Change

Binary Searching with Git

When All Else Fails: The Community

Conclusion

19 Portability

Portable Operating Systems

History of Portability in Linux

Word Size and Data Types

Opaque Types

Special Types

Explicitly Sized Types

Signedness of Chars

Data Alignment

Avoiding Alignment Issues

Alignment of Nonstandard Types

Structure Padding

Byte Order

Table of Contents

Time

Page Size

Processor Ordering

SMP, Kernel Preemption, and High Memory

Conclusion

20 Patches, Hacking, and the Community

The Community

Linux Coding Style

Indention

Switch Statements

Spacing

Braces

Line Length

Naming

Functions

Comments

Typedefs

Use Existing Routines

Minimize ifdefs in the Source

Structure Initializers

Fixing Up Code Ex Post Facto

Chain of Command

Submitting Bug Reports

Patches

Generating Patches

Generating Patches with Git

Submitting Patches

Conclusion

Table of Contents

Bibliography

Index