

Linux Performance Tools Brendan Gregg

Implementing Service Level Objectives
 Systems Performance
 Performance Analysis and Tuning on Modern CPUs
 Hands-On System Programming with Linux
 BPF Performance Tools
 Problem-solving in High Performance Computing
 Linux Observability with BPF
 The Art of Capacity Planning
 Understanding Software Dynamics
 Performance Solutions
 High Performance Web Sites
 Optimizing Oracle Performance
 Essential Linux Device Drivers
 Linux System Programming
 Cloud Native DevOps with Kubernetes
 Database Internals
 Solaris" Performance And Tools: Dtrace And Mdb Techniques For Solaris 10 And Opensolaris
 Lions' Commentary on UNIX 6th Edition with Source Code
 Linux Kernel Programming
 Solaris Internals
 Mastering KVM Virtualization
 Data Plane Development Kit (DPDK)
 Mastering Embedded Linux Programming
 DTrace
 Cybersecurity Attacks - Red Team Strategies
 BPF Performance Tools
 Systems Performance
 Hello, Startup
 High Performance Browser Networking
 Optimizing Java
 Java Performance Tuning
 Pro .NET Benchmarking
 Programming Rust
 Camel Design Patterns
 Debugging Linux Systems (Digital Short Cut)
 BPF Performance Tools
 The Art of Linux Kernel Design
 Modern X86 Assembly Language Programming
 Linkers and Loaders

Linux Performance Tools Brendan Gregg

Downloaded from blog.gmercyu.edu by guest

ISABEL JAZMYN

Implementing Service Level Objectives Addison-Wesley Professional
 Problem-Solving in High Performance Computing: A Situational Awareness Approach with Linux focuses on understanding giant computing grids as cohesive systems. Unlike other titles on general problem-solving or system administration, this book offers a cohesive approach to complex, layered environments, highlighting the difference between standalone system troubleshooting and complex problem-solving in large, mission critical environments, and addressing the pitfalls of information overload, micro, and macro symptoms, also including methods for managing problems in large computing ecosystems. The authors offer perspective gained from years of developing Intel-based systems that lead the industry in the number of hosts, software tools, and licenses used in chip design. The book offers unique, real-life examples that emphasize the magnitude and operational complexity of high performance computer systems. Provides insider perspectives on challenges in high performance environments with thousands of

servers, millions of cores, distributed data centers, and petabytes of shared data Covers analysis, troubleshooting, and system optimization, from initial diagnostics to deep dives into kernel crash dumps Presents macro principles that appeal to a wide range of users and various real-life, complex problems Includes examples from 24/7 mission-critical environments with specific HPC operational constraints
Systems Performance "O'Reilly Media, Inc."
 Performance tuning is becoming more important than it has been for the last 40 years. Read this book to understand your application's performance that runs on a modern CPU and learn how you can improve it. The 170+ page guide combines the knowledge of many optimization experts from different industries.
Performance Analysis and Tuning on Modern CPUs Pearson Education
 UNIX, UNIX LINUX & UNIX TCL/TK. Write software that makes the most effective use of the Linux system, including the kernel and core system libraries. The majority of both Unix and Linux code is still written at the system level, and this book helps you focus on everything above the kernel, where applications such as Apache, bash, cp, vim, Emacs, gcc, gdb, glibc, ls, mv, and X exist.

Written primarily for engineers looking to program at the low level, this updated edition of Linux System Programming gives you an understanding of core internals that makes for better code, no matter where it appears in the stack. -- Provided by publisher.
Hands-On System Programming with Linux O'Reilly Media
 Performance tuning is an experimental science, but that doesn't mean engineers should resort to guesswork and folklore to get the job done. Yet that's often the case. With this practical book, intermediate to advanced Java technologists working with complex technology stacks will learn how to tune Java applications for performance using a quantitative, verifiable approach. Most resources on performance tend to discuss the theory and internals of Java virtual machines, but this book focuses on the practicalities of performance tuning by examining a wide range of aspects. There are no simple recipes, tips and tricks, or algorithms to learn. Performance tuning is a process of defining and determining desired outcomes. And it requires diligence. Learn how Java principles and technology make the best use of modern hardware and operating systems Explore several performance tests and common anti-patterns that can vex your team Understand the pitfalls of measuring Java performance numbers and the drawbacks of microbenchmarking Dive

into JVM garbage collection logging, monitoring, tuning, and tools Explore JIT compilation and Java language performance techniques Learn performance aspects of the Java Collections API and get an overview of Java concurrency

BPF Performance Tools "O'Reilly Media, Inc."

“Probably the most wide ranging and complete Linux device driver book I’ve read.” --Alan Cox, Linux Guru and Key Kernel Developer “Very comprehensive and detailed, covering almost every single Linux device driver type.” --Theodore Ts’o, First Linux Kernel Developer in North America and Chief Platform Strategist of the Linux Foundation The Most Practical Guide to Writing Linux Device Drivers Linux now offers an exceptionally robust environment for driver development: with today’s kernels, what once required years of development time can be accomplished in days. In this practical, example-driven book, one of the world’s most experienced Linux driver developers systematically demonstrates how to develop reliable Linux drivers for virtually any device. Essential Linux Device Drivers is for any programmer with a working knowledge of operating systems and C, including programmers who have never written drivers before. Sreekrishnan Venkateswaran focuses on the essentials, bringing together all the concepts and techniques you need, while avoiding topics that only matter in highly specialized situations. Venkateswaran begins by reviewing the Linux 2.6 kernel capabilities that are most relevant to driver developers. He introduces simple device classes; then turns to serial buses such as I2C and SPI; external buses such as PCMCIA, PCI, and USB; video, audio, block, network, and wireless device drivers; user-space drivers; and drivers for embedded Linux—one of today’s fastest growing areas of Linux development. For each, Venkateswaran explains the technology, inspects relevant kernel source files, and walks through developing a complete example. • Addresses drivers discussed in no other book, including drivers for I2C, video, sound, PCMCIA, and different types of flash memory • Demystifies essential kernel services and facilities, including kernel threads and helper interfaces • Teaches polling, asynchronous notification, and I/O control • Introduces the Inter-Integrated Circuit Protocol for embedded Linux drivers • Covers multimedia device drivers using the Linux-Video subsystem and Linux-Audio framework • Shows how Linux implements support for wireless technologies such as Bluetooth, Infrared, WiFi, and cellular networking • Describes the entire driver development lifecycle, through debugging and maintenance • Includes reference appendixes covering Linux assembly, BIOS calls, and Seq files

Problem-solving in High Performance Computing Addison-Wesley Professional

Develop your red team skills by learning essential foundational tactics, techniques, and procedures, and boost the overall security posture of your organization by leveraging the homefield advantage Key Features Build, manage, and measure an offensive red team program Leverage the homefield advantage to stay ahead of your adversaries Understand core adversarial tactics and techniques, and protect pentesters and pentesting assets Book Description It's now more important than ever for organizations to be ready to detect and respond to security events and breaches. Preventive measures alone are not enough for dealing with adversaries. A well-rounded prevention, detection, and response program is required. This book will guide you through the stages of building a red team program, including strategies and homefield advantage opportunities to boost security. The book starts by guiding you through establishing, managing, and measuring a red team program, including effective ways for sharing results and findings to raise awareness. Gradually, you'll learn about progressive operations such as cryptocurrency mining, focused privacy testing, targeting telemetry, and even blue team tooling. Later, you'll discover knowledge graphs and how to build them, then become well-versed with basic to advanced techniques related to hunting for credentials, and learn to automate Microsoft Office and browsers to your advantage. Finally, you'll get to grips with protecting assets using decoys, auditing, and alerting with examples for major operating systems. By the end of this book, you'll have learned how to build, manage, and measure a red team program effectively and be well-versed with the fundamental operational techniques required to enhance your existing skills. What you will learn Understand the risks associated with security breaches Implement strategies for building an effective penetration testing team Map out the homefield using knowledge graphs Hunt credentials using indexing and other practical techniques Gain blue team tooling insights to enhance your red team skills Communicate results and influence decision makers with appropriate data Who this book is for This is one of the few detailed cybersecurity books for penetration testers, cybersecurity analysts, security leaders and strategists, as well as red team members and chief information security officers (CISOs) looking to secure their organizations from adversaries. The program management part of this book will also be useful for beginners in the cybersecurity

domain. To get the most out of this book, some penetration testing experience, and software engineering and debugging skills are necessary.

Linux Observability with BPF Independently Published

Troubleshoot and Optimize Complex, Time-Constrained Software From mobile and cloud apps to video games to driverless vehicle control, more and more software is time-constrained: It must deliver reliable results seamlessly, consistently, and virtually instantaneously. If it doesn't, customers are unhappy--and sometimes lives are put at risk. When time-constrained software underperforms or fails, software professionals must quickly identify and address the root causes. This is difficult and, historically, few tools have been available to help. In Understanding Software Dynamics, performance expert Richard L. Sites tackles the problem head on, offering expert methods and advanced tools for understanding complex time-constrained software dynamics, improving reliability, and troubleshooting challenging performance problems. Sites draws on several decades of experience pioneering software performance optimization, as well as extensive experience teaching graduate-level developers. He introduces principles and techniques for use in any environment, from embedded devices to datacenters, illuminating them with examples based on x86 or ARM processors running Linux and linked by Ethernet. He also guides readers through building and applying a powerful, new, extremely low-overhead open-source software tool, KUTrace, to precisely trace executions on every CPU core. Using insights gleaned from this tool, readers can apply nuanced solutions--not merely brute-force techniques such as turning off caches or cores. Measure and address issues associated with CPUs, memory, disk/SSD, networks, and their interactions Fix programs that are always too slow, and those that sometimes lag for no apparent reason Design useful observability, logging, and time-stamping capabilities into your code Reason more effectively about performance data to see why reality differs from expectations Identify problems such as excess execution, slow instruction execution, waiting for resources, and software locks Understanding Software Dynamics will be valuable to experienced software professionals, including application and OS developers, hardware and system architects, real-time system designers, and game developers, as well as advanced students. Register your book for convenient access to downloads, updates, and/or corrections as they become available. See inside book for details.

The Art of Capacity Planning "O'Reilly Media, Inc."

Embedded Android is for Developers wanting to create embedded systems based on Android and for those wanting to port Android to new hardware, or creating a custom development environment. Hackers and moders will also find this an indispensable guide to how Android works. [Understanding Software Dynamics](#) Prentice Hall

Debugging Linux Systems discusses the main tools available today to debug 2.6 Linux Kernels. We start by exploring the seemingly esoteric operations of the Kernel Debugger (KDB), Kernel GNU DeBugger (KGDB), the plain GNU DeBugger (GDB), and JTAG debuggers. We then investigate Kernel Probes, a feature that lets you intrude into a kernel function and extract debug information or apply a medicated patch. Analyzing a crash dump can yield clues for postmortem analysis of kernel crashes or hangs, so we take a look at Kdump, a serviceability tool that collects a system dump after spawning a new kernel. Profiling points you to code regions that burn more CPU cycles, so we learn to use the OProfile kernel profiler and the gprof application profiler to sense the presence of code bottlenecks. Because tracing provides insight into behavioral problems that manifest during interactions between different code modules, we delve into the Linux Trace Toolkit, a system designed for high-volume trace capture. The section “Debugging Embedded Linux” takes a tour of the I/O interfaces commonly found on embedded hardware, such as flash memory, serial port, PCMCIA, Secure Digital media, USB, RTC, audio, video, touch screen, and Bluetooth, and provides pointers to debug the associated device drivers. We also pick up some board-level debugging skills with the help of a case study. The section “Debugging Network Throughput” takes you through some device driver design issues and protocol implementation characteristics that can affect the horsepower of your network interface card. We end the shortcut by examining several options available in the kernel configuration menu that can emit valuable debug information.

Performance Solutions "O'Reilly Media, Inc."

Oracle system performance inefficiencies often go undetected for months or even years--even under intense scrutiny--because traditional Oracle performance analysis methods and tools are fundamentally flawed. They're unreliable and inefficient.Oracle DBAs and developers are all too familiar with the outlay of time and resources, blown budgets, missed deadlines, and marginally

effective performance fiddling that is commonplace with traditional methods of Oracle performance tuning. In this crucial book, Cary Millsap, former VP of Oracle's System Performance Group, clearly and concisely explains how to use Oracle's response time statistics to diagnose and repair performance problems. Cary also shows how "queueing theory" can be applied to response time statistics to predict the impact of upgrades and other system changes.Optimizing Oracle Performance eliminates the time-consuming, trial-and-error guesswork inherent in most conventional approaches to tuning. You can determine exactly where a system's performance problem is, and with equal importance, where it is not, in just a few minutes--even if the problem is several years old.Optimizing Oracle Performance cuts a path through the complexity of current tuning methods, and streamlines an approach that focuses on optimization techniques that any DBA can use quickly and successfully to make noticeable--even dramatic--improvements.For example, the one thing database users care most about is response time. Naturally, DBAs focus much of their time and effort towards improving response time. But it is entirely too easy to spend hundreds of hours to improve important system metrics such as hit ratios, average latencies, and wait times, only to find users are unable to perceive the difference. And an expensive hardware upgrade may not help either.It doesn't have to be that way. Technological advances have added impact, efficiency, measurability, predictive capacity, reliability, speed, and practicality to the science of Oracle performance optimization. Optimizing Oracle Performance shows you how to slash the frustration and expense associated with unraveling the true root cause of any type of performance problem, and reliably predict future performance.The price of this essential book will be paid back in hours saved the first time its methods are used.

High Performance Web Sites Peer to Peer Communications

BPF and related observability tools give software professionals unprecedented visibility into software, helping them analyze operating system and application performance, troubleshoot code, and strengthen security. BPF Performance Tools: Linux System and Application Observability is the industry’s most comprehensive guide to using these tools for observability. Brendan Gregg, author of the industry’s definitive guide to system performance, introduces powerful new methods and tools for doing analysis that leads to more robust, reliable, and safer code. This authoritative guide: Explores a wide spectrum of software and hardware targets Thoroughly covers open source BPF tools from the Linux Foundation iovisor project’s bcc and bpftrace repositories Summarizes performance engineering and kernel internals you need to understand Provides and discusses 150+ bpftrace tools, including 80 written specifically for this book: tools you can run as-is, without programming — or customize and develop further, using diverse interfaces and the bpftrace front-end You’ll learn how to use BPF (eBPF) tracing tools to analyze CPUs, memory, disks, file systems, networking, languages, applications, containers, hypervisors, security, and the Linux kernel. You’ll move from basic to advanced tools and techniques, producing new metrics, stack traces, custom latency histograms, and more. It’s like having a superpower: with Gregg’s guidance and tools, you can analyze virtually everything that impacts system performance, so you can improve virtually any Linux operating system or application.

Optimizing Oracle Performance Packt Publishing Ltd

How prepared are you to build fast and efficient web applications? This eloquent book provides what every web developer should know about the network, from fundamental limitations that affect performance to major innovations for building even more powerful browser applications—including HTTP 2.0 and XHR improvements, Server-Sent Events (SSE), WebSocket, and WebRTC. Author Ilya Grigorik, a web performance engineer at Google, demonstrates performance optimization best practices for TCP, UDP, and TLS protocols, and explains unique wireless and mobile network optimization requirements. You’ll then dive into performance characteristics of technologies such as HTTP 2.0, client-side network scripting with XHR, real-time streaming with SSE and WebSocket, and P2P communication with WebRTC. Deliver superlative TCP, UDP, and TLS performance Speed up network performance over 3G/4G mobile networks Develop fast and energy-efficient mobile applications Address bottlenecks in HTTP 1.x and other browser protocols Plan for and deliver the best HTTP 2.0 performance Enable efficient real-time streaming in the browser Create efficient peer-to-peer videoconferencing and low-latency applications with real-time WebRTC transports

Essential Linux Device Drivers Packt Publishing Ltd

Although service-level objectives (SLOs) continue to grow in importance, there’s a distinct lack of information about how to implement them. Practical advice that does exist usually assumes that your team already has the infrastructure, tooling, and culture in place. In this book, recognized SLO

expert Alex Hidalgo explains how to build an SLO culture from the ground up. Ideal as a primer and daily reference for anyone creating both the culture and tooling necessary for SLO-based approaches to reliability, this guide provides detailed analysis of advanced SLO and service-level indicator (SLI) techniques. Armed with mathematical models and statistical knowledge to help you get the most out of an SLO-based approach, you'll learn how to build systems capable of measuring meaningful SLIs with buy-in across all departments of your organization. Define SLIs that meaningfully measure the reliability of a service from a user's perspective Choose appropriate SLO targets, including how to perform statistical and probabilistic analysis Use error budgets to help your team have better discussions and make better data-driven decisions Build supportive tooling and resources required for an SLO-based approach Use SLO data to present meaningful reports to leadership and your users

Linux System Programming BPF Performance Tools

Helps readers eliminate performance problems, covering topics including bottlenecks, profiling tools, strings, algorithms, distributed systems, and servlets.

Cloud Native DevOps with Kubernetes Packt Publishing Ltd

"I enjoyed reading this useful overview of the techniques and challenges of implementing linkers and loaders. While most of the examples are focused on three computer architectures that are widely used today, there are also many side comments about interesting and quirky computer architectures of the past. I can tell from these war stories that the author really has been there himself and survived to tell the tale." -Guy Steele Whatever your programming language, whatever your platform, you probably tap into linker and loader functions all the time. But do you know how to use them to their greatest possible advantage? Only now, with the publication of *Linkers & Loaders*, is there an authoritative book devoted entirely to these deep-seated compile-time and run-time processes. The book begins with a detailed and comparative account of linking and loading that illustrates the differences among various compilers and operating systems. On top of this foundation, the author presents clear practical advice to help you create faster, cleaner code. You'll learn to avoid the pitfalls associated with Windows DLLs, take advantage of the space-saving, performance-improving techniques supported by many modern linkers, make the best use of the UNIX ELF library scheme, and much more. If you're serious about programming, you'll devour this unique guide to one of the field's least understood topics. *Linkers & Loaders* is also an ideal supplementary text for compiler and operating systems courses. Features: * Includes a linker construction project written in Perl, with project files available for download. * Covers dynamic linking in Windows, UNIX, Linux, BeOS, and other operating systems. * Explains the Java linking model and how it figures in network applets and extensible Java code. * Helps you write more elegant and effective code, and build applications that compile, load, and run more efficiently.

Database Internals Apress

Build your expertise in the BPF virtual machine in the Linux kernel with this practical guide for systems engineers. You'll not only dive into the BPF program lifecycle but also learn to write applications that observe and modify the kernel's behavior; inject code to monitor, trace, and securely observe events in the kernel; and more. Authors David Calavera and Lorenzo Fontana help you harness the power of BPF to make any computing system more observable. Familiarize yourself with the essential concepts you'll use on a day-to-day basis and augment your knowledge about performance optimization, networking, and security. Then see how it all comes together with code examples in C, Go, and Python. Write applications that use BPF to observe and modify the Linux kernel's behavior on demand Inject code to monitor, trace, and observe events in the kernel

in a secure way—no need to recompile the kernel or reboot the system Explore code examples in C, Go, and Python Gain a more thorough understanding of the BPF program lifecycle

Solaris" Performance And Tools: Dtrace And Mdb Techniques For Solaris 10 And Opensolaris "O'Reilly Media, Inc."

Success on the web is measured by usage and growth. Web-based companies live or die by the ability to scale their infrastructure to accommodate increasing demand. This book is a hands-on and practical guide to planning for such growth, with many techniques and considerations to help you plan, deploy, and manage web application infrastructure. The Art of Capacity Planning is written by the manager of data operations for the world-famous photo-sharing site Flickr.com, now owned by Yahoo! John Allspaw combines personal anecdotes from many phases of Flickr's growth with insights from his colleagues in many other industries to give you solid guidelines for measuring your growth, predicting trends, and making cost-effective preparations. Topics include: Evaluating tools for measurement and deployment Capacity analysis and prediction for storage, database, and application servers Designing architectures to easily add and measure capacity Handling sudden spikes Predicting exponential and explosive growth How cloud services such as EC2 can fit into a capacity strategy In this book, Allspaw draws on years of valuable experience, starting from the days when Flickr was relatively small and had to deal with the typical growth pains and cost/performance trade-offs of a typical company with a Web presence. The advice he offers in *The Art of Capacity Planning* will not only help you prepare for explosive growth, it will save you tons of grief.

Lions' Commentary on UNIX 6th Edition with Source Code Pearson Education India

Use this in-depth guide to correctly design benchmarks, measure key performance metrics of .NET applications, and analyze results. This book presents dozens of case studies to help you understand complicated benchmarking topics. You will avoid common pitfalls, control the accuracy of your measurements, and improve performance of your software. Author Andrey Akinshin has maintained BenchmarkDotNet (the most popular .NET library for benchmarking) for five years and covers common mistakes that developers usually make in their benchmarks. This book includes not only .NET-specific content but also essential knowledge about performance measurements which can be applied to any language or platform (common benchmarking methodology, statistics, and low-level features of modern hardware). What You'll Learn Be aware of the best practices for writing benchmarks and performance tests Avoid the common benchmarking pitfalls Know the hardware and software factors that affect application performance Analyze performance measurements Who This Book Is For .NET developers concerned with the performance of their applications

Linux Kernel Programming CRC Press

Gain the fundamentals of x86 64-bit assembly language programming and focus on the updated aspects of the x86 instruction set that are most relevant to application software development. This book covers topics including x86 64-bit programming and Advanced Vector Extensions (AVX) programming. The focus in this second edition is exclusively on 64-bit base programming architecture and AVX programming. Modern X86 Assembly Language Programming's structure and sample code are designed to help you quickly understand x86 assembly language programming and the computational capabilities of the x86 platform. After reading and using this book, you'll be able to code performance-enhancing functions and algorithms using x86 64-bit assembly language and the AVX, AVX2 and AVX-512 instruction set extensions. What You Will Learn Discover details of

the x86 64-bit platform including its core architecture, data types, registers, memory addressing modes, and the basic instruction set Use the x86 64-bit instruction set to create performance-enhancing functions that are callable from a high-level language (C++) Employ x86 64-bit assembly language to efficiently manipulate common data types and programming constructs including integers, text strings, arrays, and structures Use the AVX instruction set to perform scalar floating-point arithmetic Exploit the AVX, AVX2, and AVX-512 instruction sets to significantly accelerate the performance of computationally-intense algorithms in problem domains such as image processing, computer graphics, mathematics, and statistics Apply various coding strategies and techniques to optimally exploit the x86 64-bit, AVX, AVX2, and AVX-512 instruction sets for maximum possible performance Who This Book Is For Software developers who want to learn how to write code using x86 64-bit assembly language. It's also ideal for software developers who already have a basic understanding of x86 32-bit or 64-bit assembly language programming and are interested in learning how to exploit the SIMD capabilities of AVX, AVX2 and AVX-512.

Solaris Internals "O'Reilly Media, Inc."

Learn how to write high-quality kernel module code, solve common Linux kernel programming issues, and understand the fundamentals of Linux kernel internals Key Features Discover how to write kernel code using the Loadable Kernel Module framework Explore industry-grade techniques to perform efficient memory allocation and data synchronization within the kernel Understand the essentials of key internals topics such as kernel architecture, memory management, CPU scheduling, and kernel synchronization Book Description Linux Kernel Programming is a comprehensive introduction for those new to Linux kernel and module development. This easy-to-follow guide will have you up and running with writing kernel code in next-to-no time. This book uses the latest 5.4 Long-Term Support (LTS) Linux kernel, which will be maintained from November 2019 through to December 2025. By working with the 5.4 LTS kernel throughout the book, you can be confident that your knowledge will continue to be valid for years to come. You'll start the journey by learning how to build the kernel from the source. Next, you'll write your first kernel module using the powerful Loadable Kernel Module (LKM) framework. The following chapters will cover key kernel internals topics including Linux kernel architecture, memory management, and CPU scheduling. During the course of this book, you'll delve into the fairly complex topic of concurrency within the kernel, understand the issues it can cause, and learn how they can be addressed with various locking technologies (mutexes, spinlocks, atomic, and refcount operators). You'll also benefit from more advanced material on cache effects, a primer on lock-free techniques within the kernel, deadlock avoidance (with lockdep), and kernel lock debugging techniques. By the end of this kernel book, you'll have a detailed understanding of the fundamentals of writing Linux kernel module code for real-world projects and products. What you will learn Write high-quality modular kernel code (LKM framework) for 5.x kernels Configure and build a kernel from source Explore the Linux kernel architecture Get to grips with key internals regarding memory management within the kernel Understand and work with various dynamic kernel memory alloc/dealloc APIs Discover key internals aspects regarding CPU scheduling within the kernel Gain an understanding of kernel concurrency issues Find out how to work with key kernel synchronization primitives Who this book is for This book is for Linux programmers beginning to find their way with Linux kernel development. If you're a Linux kernel and driver developer looking to overcome frequent and common kernel development issues, or understand kernel internals, you'll find plenty of useful information. You'll need a solid foundation of Linux CLI and C programming before you can jump in.

Related with Linux Performance Tools Brendan Gregg:

- Dislyte Expert Course Exam 2 : [click here](#)