Linux Kernel on Reddit

- I just did a cold boot attack on my own system... (2025/12/05 00:27)
  I used an old x60 IBM thinkpad that has 1 stick of 1GB RAM. so this RAM is old because it is DDR2. the hard disk is entirely encrypted with LUKS2 running slackware 15.0. i ran a series of different tests divided into 2 main parts: with the default generic 5.15.19 kernel and a recompiled kernel of the same version with a couple hardened features. the only difference is that i hardcoded modules and specifically enabled these two: CONFIG_INIT_ON_ALLOC_DEFAULT_ON=y CONFIG_INIT_ON_FREE_DEFAULT_ON=y i also explicitly enabled init_on_free=1 init_on_alloc=1 in my boot kernel parameters just to be sure. apparently, page_poison has been overrided if these 2 are set so it has the same effect of doing that. basically it will zero out the pages of memory when the process is killed. therefore, when one does a graceful shutdown, and all processed are killed, the kernel shall zero out those pages which shall include the pages of memory where the LUKS encrypted key resides. I used findaes and aeskeyfind and they returned keys instantly. i used this key to mount the drive without the passphrase! i also used foremost and that returned a few broken images. i ran about 5 tests. Test 1: the typical attack with the default kernel. this is a simulation of the target system being seized while powered on. i sprayed RAM first, then pressed the power off button. i kept the RAM frozen the entire 4 minutes. result: keys were found as expected Test 2: default kernel but graceful init 0 shutdown. there was about a 1-2 second grace period after shutdown from when i began freezing the RAM. result: nothing from any of the 3 programs Test 3: default kernel. same graceful shutdown. froze RAM just after typing init 0result: keys were found Test 4: hardened kernel. same graceful shutdown. froze RAM after system turned off. 1-2 second grace period. result: nothing from any of the 3 programs Test 5: hardened kernel. same graceful shutdown. froze RAM just after typing init 0 result: KEYS WERE FOUND! It was devastating to find out the keys were actually found with my hardened kernel when performing a graceful shutdown!! I conclude that the hardened kernel parameters I used had no effect on actually zeroing out the pages of RAM because the key was indeed found instantly. the only thing that ensured that the LUKS key was not captured was simply having the machine off for even just a couple seconds. of course anyone initiating this attack will begin freezing the RAM while in a powered on state, or suspended to RAM. then cut the power instantly by removing the battery. so...why did Test 5 result in my keys being found? what other kernel configurations should be implemented to prevent this attack? submitted by /u/Klutzy_Scheme_9871 [link] [comments]
- Journey to 2004: Linux 2.4 Environment Setup (2025/12/02 06:26)
  Preface Writing this title on December 1, 2025, the current kernel has evolved to version 6.18, with increasingly complete kernel functionality, support for more and more architectures, and better and better performance. I once tried to read the kernel source code of version 5.10, but found myself utterly confused—I could understand the code itself, but not the meaning behind it. I could only know the what, but not the why. I came across a netizen's recommendation of Teacher Mao's "Linux Kernel Source Code Scenario Analysis." I casually flipped through it, thinking this book is so old, is it still worth reading? But unable to resist the strong recommendations from fellow netizens, I suppressed my impatient mindset and patiently finished one section. After reading it, I felt like I had found a treasure—this was exactly the book I was looking for. Teacher Mao analyzes the source code through various scenarios, helping readers understand the meaning of each conditional judgment in the code, which is infinitely superior to those books that merely list source code. So, is there still significance in reading the 2.4 source code now? I believe

there is. First, the 2.4 kernel code is not yet mature, and precisely because of this, the barrier to entry is relatively low. Second, although subsequent code architectures have huge differences, the core ideas remain unchanged. Third, we can attempt to gradually track certain features from 2.4 to newer versions—as the saying goes, Rome wasn't built in a day. Through historical changes, we can better conduct an in-depth analysis of the kernel, this massive project. Environment Setup Because the 2.4 kernel is too old, the biggest obstacle is compiler version incompatibility—modern GCC cannot compile code from 2001. To solve this problem, we use Docker to create a Debian Sarge distribution, compile the kernel with GCC 3.3 inside it, and then run it with QEMU on the host machine. Note: "Linux Kernel Source Code Scenario Analysis" is based on 2.4.0 for analysis, but many errors occur during compilation, so this experiment is based on version 2.4.37. Start an Old Version Debian Container > git clone git@github.com:hlleng/kernel2.4-lab.git > cd kernel2.4-lab > docker run --platform linux/386 -it -v $(pwd):/code debian/eol:sarge /bin/bash After entering docker, first update the software sources and install the necessary software. > apt-get update > apt-get install -y \\ gcc make binutils libncurses5-dev wget bzip2 Next, compile the kernel > make ARCH=i386 menuconfig # Just select "Exit" -> "Yes" (Save). > make ARCH=i386 dep # Must first generate the dependency tree > make ARCH=i386 bzImage The process of creating the filesystem is quite complicated, so we'll skip the details here and directly use the filesystem I prepared in advance. Execute QEMU on the host machine to load the kernel and filesystem. > qemu-system-i386\\ -kernel ./arch/i386/boot/bzImage \\ -hda hda.img \\ -append "root=/dev/hda init=/init console=ttyS0" \\ -nographic If all goes well, you should be able to successfully enter the system. If you need to copy files from the host machine into qemu, you can try the following steps, operating on the host machine > mkdir -p tmp/mnt > sudo mount -o loop hda.img tmp/mnt # Perform operations > sudo umount tmp/mnt At this point, we have completed the environment setup for the 2.4 kernel and can begin studying the 2.4 kernel. submitted by /u/Infinite-Feed-3904 [link] [comments]

- Introducing Riptides Conditional Access: Fine-Grained, Time-Aware Security Policies (2025/12/01 13:32)
  submitted by /u/baluchicken [link] [comments]
- Copy-On-Write (2025/11/30 00:43)
  Copy-On-Write is a technique used in Linux kernel to delay the writing of the process pages until they are written to. This helps with faster process execution and avoids overhead. Want to know more? Read my blog here https://linux-kernel.hashnode.dev/process-management submitted by /u/Junior_Mango9596 [link] [comments]
- Asking for career advice (2025/11/29 07:45)
  1) Is it possible to still get hired in some kernel development field with a degree in Digital System Design? 2) Would DSD be viewed as less relevant degree than CS or Computer Engineering by potential employers? Assuming I do my minor in CS and self-study to get some more relevant experience. submitted by /u/Suspicious_Diet_8774 [link] [comments]
- The Input Stack on Linux: An End-To-End Architecture Overview (2025/11/27 17:16)
  submitted by /u/lynob [link] [comments]
- Advice on Learning Linux Kernel/Firmware Development for Embedded Security Engineers (2025/11/26 18:28)
  Hi everyone, I’m a Cyber Security Engineer with experience in embedded security, ARM TrustZone, and Trusted Execution Environments (TEEs). I’ve worked on Trusted Applications, Secure Boot, HSMs, and privacy-preserving workflows. I’m looking to start learning Linux kernel development and eventually transition my career from embedded security to firmware/kernel development. I’m comfortable with C and want to leverage my embedded systems and security experience to understand kernel concepts more deeply. I also have some Rockchip SoCs that I can

use for hands-on projects. I feel that learning by doing projects is much more effective than just studying theory, and I want to build practical experience while learning. Are there any recommended resources, projects, or learning paths that could help me bridge my current skills into kernel-level programming and firmware development? Any advice would be much appreciated! submitted by /u/Plastic_Extreme_266 [link] [comments]

- From Build to Root Cause: How Riptides Debugs Its Kernel Module in Real Clusters (2025/11/24 13:27)
  submitted by /u/baluchicken [link] [comments]
- Ineed help to join kernel development (2025/11/24 00:32)
  submitted by /u/Mr_Error01 [link] [comments]
- Can't compile kernel after Fedora 41 to 42 upgrade (2025/11/23 18:36)
  submitted by /u/MakeTopSite [link] [comments]
- I only know what field I'm truly interested in as a junior in college. Should I pursue my new interest or stay with the original plan? (I'm an international student) (2025/11/17 22:52)
  Hi, I'm currently junior in college pursuing a CS major. To be completely honest, the main reason why I chose CS in the beginning is the huge but extremely competitive job market for software engineers. I already had my projects, an internship for a data analyst position back in my home country and some experiences as an undergraduate lab assistant listed in my resume. However, I took my first Operating Systems class this semester and this was the very first time I've ever felt truly interested in this field (huge thanks to my professor). Half a semester went by and I am still enjoying this class very much. This feels very new and different compared to other programming classes where I felt mediocre and leetcoding drains my soul (but I did it anyways). I have great respect for my OS class' professor and I always wanted to ask questions in class and build a connection with him. But most of the time I just don't know what to ask (I think it's because I don't have a deep understanding of the materials that was being taught at that time yet). There are just so many doubts and I don't know how to solve them. I am trying to attend his office hours more often for advice regarding my career choice but I always stumbled on the right questions that should be asked. Also, would it be a good idea to ask him about research assistant opportunities? I am torn between two choices, to keep aiming to be an software engineer (most likely backends) where there might be more opportunities, or to dive deeper into OS (kernel, virtualization, embedded, etc) and having to redo my resume almost from scratch? Should I stay with the safer choice or take the risk? submitted by /u/i_am_not_a_potat0 [link] [comments]
- Where can I take a solid Linux Kernel course (2025/11/17 04:04)
  I work in a storage industry in a support role and I really want to become more professional and confident, I would like to understand how to identify kernel problems and maybe suggest improvements. I want to become really good at this, this is my goal/dream. Also, is there any online/on-demand course? I think I learn more when I interact with people. submitted by /u/Euporia1 [link] [comments]
- I'd like to see Linux be competitive with FreeBSD performance-wise (2025/11/12 23:12)
  Update - the latency comparison favors Linux.
  https://preview.redd.it/2n32iug0741g1.png?width=640&format=png&auto=webp&s=f0539a4eb35fdb195be0d1d52ba96e94b9294a03 Long time Linux user here, recently started testing FreeBSD in connection with Maia Mailguard. Along the way, I tested identical VMs (4 GB RAM, 2 cores) running Debian 13, FreeBSD 14 and FreeBSD 15. With default, out of the box installs all around, FreeBSD seems to perform considerably faster.
  https://preview.redd.it/g9e4jl82741g1.png?width=640&format=png&auto=webp&s=91df4fad89f445c8bc8fae67182868c0a179a249 Is this

expected? Are there some tuning secrets that will bring Debian 13 anywhere near same the ballpark as FreeBSD? submitted by /u/amazingrosie123 [link] [comments]

- curious about the cause for an increased amd igpu power consumption and/or impossible voltage present in backported 6.16.3+deb13 that does not occur in 6.16.12+deb14 (2025/11/12 17:31)
  in debian with backported kernel 6.16.3+deb13 there are two odd things. first, when igpu boosts, vddgfx will boost slightly below 1.3V, while vddnb stays at stock 1.020V, which is the vsoc on my board bios. this is odd as according to skatterbencher (https://skatterbencher.com/2023/02/21/skatterbencher-55-amd-radeon-graphics-ryzen-7000-overclocked-to-3100-mhz/#AMD_Radeon_Graphics_Ryzen_7000_Voltage_Topology) the gfx voltage should be derived from the vsoc in the cpu. the power consumption metric also shows an increase, even though the igpu clock stays at the same 2200 MHz as normally. to clarify, in both debian stable kernel and testing 6.16.12 kernel, vddgfx stays slightly below vsoc/vddnb, with the same clock and with lower reported heat. 1.3V happens to be the vsoc safety limit added some years ago as some cpus died, so it is curious that gfx voltage gets capped a bit below that value. what change in the kernel could possibly cause these results? second, in amdgpu code there was an addition a year or two ago that prevented lowering the power limit of the gpu below 70% of the standard value. this was unpopular and many kernel variants have added a patch that restores the ability to reduce the power limit. now, debian should not have this patch, but the backported kernel still happens to allow lower power limits than that. nice, but unexpected. i checked the source code i got from apt source download, and could not see this patch added to it. perhaps not as interesting, but remains a mystery to me. edit: it seems that vddgfx is not necessarily capped by vsoc, so those readings might be correct after all. still odd behaviour. submitted by /u/Niwrats [link] [comments]
- fuck what (2025/11/10 06:13)
  did i code a kernel panic thrower?
  https://preview.redd.it/qc7e5nr5gd0g1.png?width=2880&format=png&auto=webp&s=8146a2f563d7041771d2993f115407c62cbe148f
  https://preview.redd.it/h39xkkc9gd0g1.png?width=272&format=png&auto=webp&s=b434d5e8b848e0e6905d82b52b0753b0d14e8641
  submitted by /u/Weird_Egg_1186 [link] [comments]
- net_rx softirq clarifications (2025/11/09 19:47)
  We have some servers at work exhibiting this problem: 8 CPUs dedicated to softirqs, and under modest packet/sec pressure (400K/sec system-wide), these 8 CPUs go north of 50% occupied in softirq state. (When it's bad, they're 99% occupied.) We've looked at spreading the load around more with RPS/etc, but we also believe that there is something fundamentally whack with our setup, as we've run benchmarks with similar packet sizes pushing 3 Million PPS on a different machine. So I've been trying to zero in on what's occupying the extra CPU. `perf` has showed me indeed that 98% of softirq CPU are spent in net_rx. But in my reading of various blogs/doc I do not understand a few things: 51% of a CPU is reported in `softirq` state. (i.e., `mpstat -P ALL 1` shows 51% on 8 different CPUs.) Yet, `ksoftirqd` shows 1-10% per CPU in top. Does this mean the culprit is mostly in the "inline" portion of the softirq and not the bit that gets deferred to `ksoftirqd`? Other side of the same coin: does work done in `ksoftirqd` show up as `softirq` state when looking at CPU metrics and /proc/stat? Do softirqs work like that- where a fixed amount is executed "inline" and then the rest spills over to ksoftirqd? I found some blogs/talks saying so, but there's a lot of inconsistency out there. And, of course, my chatGPT-assisted investigation has probably led me to a few misleading conclusions. Maybe a read of the code is in order... OK, finally, is there a Slack where such things get discussed? submitted by /u/seizethedave [link] [comments]

- su -c "apt install [LINUX IS AN OPERATING SYSTEM].dumbass.church.do.sleep.UBUNTU.is.technically.a.windows.just.liek.windows.cuss.windows.dude.clickoOnButtonsInMyWindows....KERNEL_PANIK (2025/11/09 17:01)
  SORRY for the DISTÜRBANZE... i needed to MOOD.OFF.SOME.STEAM just ignore me & please have a nice rest of SunDay kind regards melkanea submitted by /u/ZookeepergameOver701 [link] [comments]
- Kernel stack use-after-free: Exploiting NVIDIA's GPU Linux drivers (2025/11/01 10:44)
  submitted by /u/killjoy_buzzkill [link] [comments]
- VSL-DSP Open Source Driver - PreSonus Audio Interfaces for Linux (2025/10/30 04:48)
  GPL driver for PreSonus audio interfaces using the proprietary VSL (Virtual StudioLive) protocol. Developed through reverse engineering of VSL software using Ghidra, with assembly-level analysis of USB commands and DSP communication. What it does Enables native control of PreSonus interfaces on Linux without relying on proprietary software. Implements direct USB communication with the DSP for routing configuration, latency control, and channel management. Technical stack Kernel: Custom module based on snd-usb-audio with kernel compiled from source to ensure version compatibility. Userspace: C client with low-level USB communication, initialization sequence analysis, and DSP control commands. Reverse Engineering: Ghidra analysis of proprietary VSL binary to extract protocol and command structure. Current status Functional USB communication with successful device writes. Currently debugging response sequences and hardware-specific timings. Why it matters PreSonus interfaces are professional-grade hardware artificially locked by proprietary software. This driver liberates the hardware you already own, enabling full functionality on Linux systems without restrictions. License: GPL Hardware tested: Audiobox 22vsl Presonus Seeking: Beta testers with PreSonus hardware and feedback from ALSA/kernel community Any feedback, testing, or any pull requests you want to make are welcome. If you have any improvements, it would be great to add them to my repo. After all, I did this because I have this thing, I don't use Windows at all anymore, and I couldn't stand that the coolest feature of the motherboard wasn't available on my OS, so I did what any good Linux user would do: I did it myself. https://github.com/grisuno/VSL-DSP https://medium.com/@lazyown.redteam/whe ... 6302d93906 https://medium.com/@lazyown.redteam/%EF ... 414c695740 Upvote4Downvote3Go to comments submitted by /u/Reasonable_Listen888 [link] [comments]
- Unpacking the rocknix KERNEL file for the Orange PI 5 Plus (2025/10/28 14:44)
  What I want to do is extract the linux kernel and the initramfs from the KERNEL file which is in the root / (once you've put the rocknix d/l image onto a sd card etc). I've tried using binwalk and extracting but once extracted I get exactly the same file, its packed somehow... Anyway, to explain what I'm trying to achive is a DUAL book using U-BOOT... First put Armbian on the Orange PI 5 EMMC (works) Before orange pi 5 boots armbian for the first time use a rocknix sd card in the machine to stop it booking ssh into it and create the .rootfs_resize (with the percentage of the emmc to use for armbian leaving enough for rocknix), i use 87% on a 256gb emmc. Create a 2nd partition for rocknix now teh fun bit, convert the ROCKNIX and KERNEL files used by rocknix to the initrd.img etc files needed to boot via armbian the just ssh into the machine when you want rocknix and swap in a armbianEnv.txt to point to the 2nd partition... Convoluted yes, but I have a SLOW SLOW PC that would take over 3 days to build rocknix and to be honest I have no idea how to build rocknix orange pi 5 plus to use uboot.... So hacking the kernel files etc is the only way i know.. and nope haven't found anyone who has build a rocknix opi5+ uboot build :(submitted by /u/TrafficCapital8855 [link] [comments]
- When eBPF Isn't Enough: Why We Went with a Kernel Module (2025/10/27 13:13)

submitted by /u/baluchicken [link] [comments]

- The Linux Boot Process: From Power Button to Kernel (2025/10/26 00:20)
  submitted by /u/swe129 [link] [comments]
- Is it still possible to install 4.3.0 kernel in AMD EPYC-Milan Processor? (2025/10/25 13:34)
  Hi, guys I need to install kernel 4.3.0 to setup the Ingens, I just want to ask you guys that is it still possible this old kernel and some codes to relatively current processor. Also, if you know how to setup Ingens, I'll be really appreciated. Finally my environment: - lscpu CPU op-mode(s): 32-bit, 64-bit Byte Order: Little Endian Address sizes: 40 bits physical, 57 bits virtual CPU(s): 12 On-line CPU(s) list: 0-11 Thread(s) per core: 1 Core(s) per socket: 1 Socket(s): 12 NUMA node(s): 1 Vendor ID: AuthenticAMD CPU family: 25 Model: 1 Model name: AMD EPYC-Milan Processor ... - kernel version and OS 5.4.0-216-generic / Ubuntu 20.04.6 LTS - Architecture x86-64 Thanks in advance. submitted by /u/Itchy-Path94 [link] [comments]
- Custom Android devices with custom OS linux kernel (2025/10/22 08:24)
  Hi everyone! This is actually my first ever post on reddit. Been working on a very big project. Custom OS, fully secure that only installs my apps, connects only to my server and all data is encrypted - i don't see any of the user data. I want to open source the full project. The issue is that I can't find a reliable android oem/odm manufacturer that shares the kernel source code for free to test the OS (Lineage Base). They all ask for money even though they are required by GPL to share it. I am thinking of buying in bulk devices that are on the lineage wiki for the ease of customizing my own OS. Does anyone have any suggestions? submitted by /u/PlusUpstairs2500 [link] [comments]
- core linux kernel initialization (2025/10/21 18:26)
  Appreciate your feedback on my recent blog which details the core kernel initialization process.
  https://linux-kernel.hashnode.dev/core-kernel-initialization submitted by /u/Junior_Mango9596 [link] [comments]