Setup Linux Debugging in VS Code

A Fast ARM64 Workflow

Anantha Ramayya Kandrapu

Linux Kernel & eBPF Meetup

The Core Problem: The Kernel Setup Cliff

Why New Developers Give Up on Kernel Hacking

> *"The 'Discovery Gap' is fatal: Most contributors vanish before ever seeing their first line of code execute in a debugger."*

Traditional Pain Points

- 3+ days wrestling with complex toolchains
- 40-minute compilation cycles on local hardware
- Infinite 'printk' debug loops due to lack of visibility
- Silent failures and opaque kernel panics

< 5 MINS

Git clone to stepping in code

Fully automated ARM64 dev environment

High-Level Architecture — The Two-Process Model

Decoupling Execution from Symbol Resolution

Minimal Userspace

Fast Testing Without Distribution Overhead

Initramfs & Static BusyBox

An Initramfs provides a lightweight userspace loaded during early boot. By using a statically-linked BusyBox binary, we bypass the need for a full root filesystem and shared libraries.

/init Script Responsibilities:

- Mount essential virtual filesystems: proc, sys, dev
- Set device IP and bring up loopback networking
- Spawn a standalone shell (sh) as the initial process

⚠ Critical Gotcha

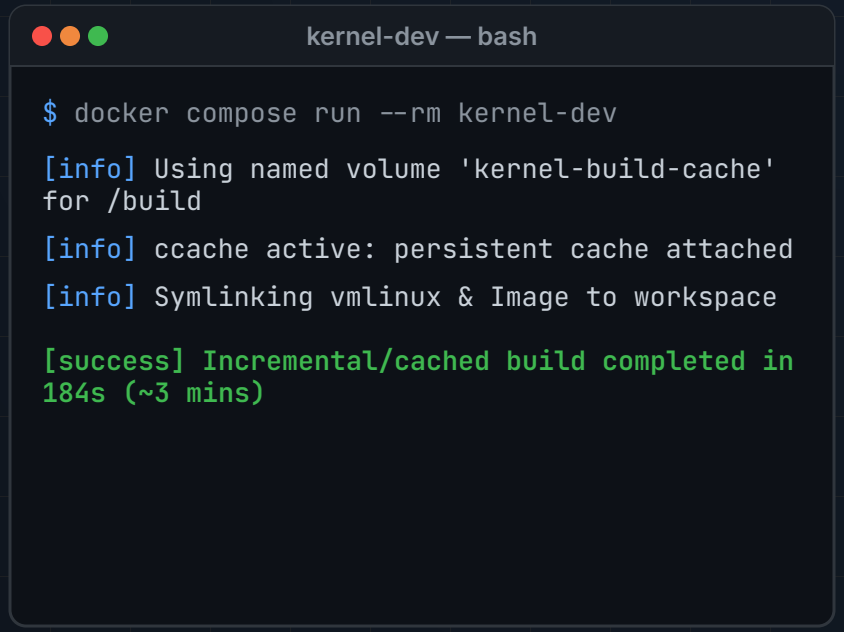
Relative Symlinks

BusyBox applets (ls, cat, etc.) are often symlinks to /bin/busybox. If your archive build script uses absolute symlinks, the kernel won't resolve them inside the initramfs. Always use **correct relative offsets** within the archive structure.

Build Speed Fix

Slashing Mac + Docker Builds from 20+ Mins to 3 Mins

- **Named Volume (/build):** Keeps intermediate object files in Linux VM's native filesystem, bypassing macOS VirtioFS I/O bottlenecks.
- **ccache Integration:** Persistent compiler cache volume serves unchanged translation units across container restarts.
- **Auto-Symlink:** Build script automatically symlinks generated vmlinux and Image back into workspace for instant VS Code symbol resolution.



The Proof — Real-World Developer Loop Benchmarks

Measuring the Time Saved for New Developers

CLEAN BASELINE

~20 Mins

Full uncached kernel build

CACHED REBUILD

~3 Mins

ccache + Named Volume

Demo : DevContainer opening

VS Code Integration : Dual launch.json Config

```
"configurations": [
  {
    "name": "Linux Kernel Debug (Direct GDB)",
    "type": "cppdbg",
    "request": "launch",
    "preLaunchTask": "Build Initramfs",
    "program": "${workspaceFolder}/linux/vmlinux",
    "cwd": "${workspaceFolder}/linux",
    "miDebuggerServerAddress": "localhost:1234",
    "miDebuggerPath": "gdb-multiarch",
    "setupCommands": [
      {
        "description": "Set architecture to aarch64",
        "text": "set architecture aarch64",
        "ignoreFailures": false
      },
      {
        "description": "Disable pagination",
        "text": "set pagination off",
        "ignoreFailures": true
      },
      {
        "description": "Map Linux source root",
        "text": "directory ${workspaceFolder}/linux",
        "ignoreFailures": true
      }
    ],
    "stopAtEntry": false
  },
```

```
{
  "name": "Linux Kernel Debug (mi2gdb Command Viewer)",
  "type": "cppdbg",
  "request": "launch",
  "preLaunchTask": "Build Initramfs",
  "program": "${workspaceFolder}/linux/vmlinux",
  "cwd": "${workspaceFolder}/linux",
  "miDebuggerServerAddress": "localhost:1234",
  "miDebuggerPath": "${workspaceFolder}/scripts/
  mi2gdb",
  "setupCommands": [
    {
      "description": "Set architecture to aarch64",
      "text": "set architecture aarch64",
      "ignoreFailures": false
    },
    {
      "description": "Disable pagination",
      "text": "set pagination off",
      "ignoreFailures": true
    },
    {
      "description": "Map Linux source root",
      "text": "directory ${workspaceFolder}/linux",
      "ignoreFailures": true
    }
  ],
  "stopAtEntry": false
}
```

Fatal Gotchas Solved Upfront

⚠ KASLR

Problem: Kernel randomizes memory addresses at boot, making breakpoints silently fail.

Solution: Pass `'nokaslr'` in QEMU boot arguments.

⚠ ARM64 Serial Console

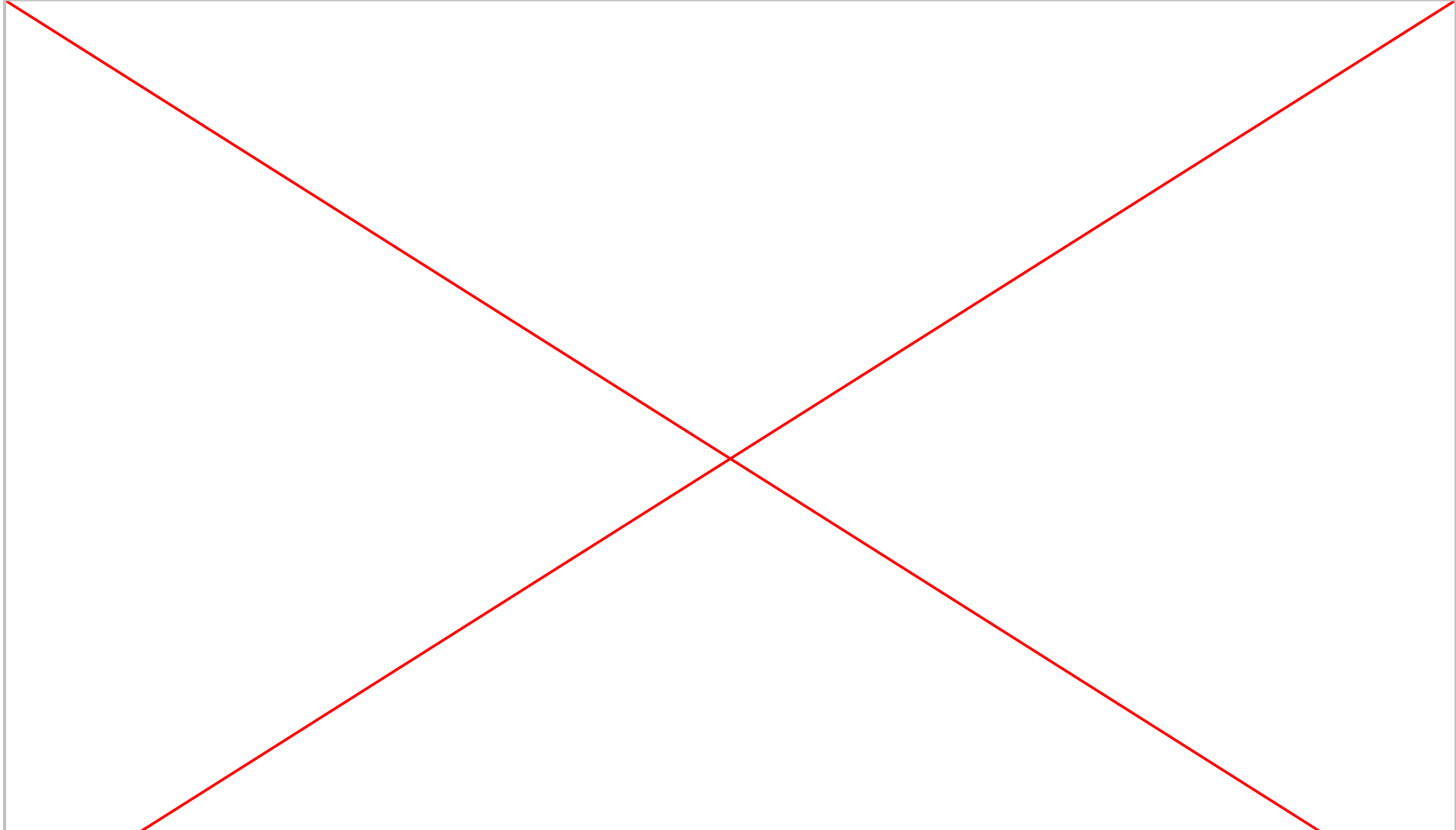
Problem: QEMU graphic window shows a black screen, hiding critical panic logs.

Solution: Set `'console=ttyAMA0'` (ARM64 PL011 UART) instead of x86 'ttyS0' to redirect output to the terminal.

⚠ make defconfig Trap

Problem: `'make defconfig'` silently wipes debug symbols (CONFIG_DEBUG_INFO) and re-enables KASLR.

Solution: Use `'make olddefconfig'` with pinned config fragments and assert debug flags before building.



Repo

Clone, Build, and Step in Under 5 Minutes

https://github.com/Anantha-Kandrapu/kernel_setup