Episode Book

EPISODE #6186

A Virtual Disk Inside Your Linux Filesystem

PUBLISHED	21 October 2026 at 16:49 UTC
DURATION	20:03
LISTEN	myweirdprompts.com/episode/linux-loop-device-virtual-disks-31688/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	anthropic/claude-sonnet-5.5
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.12
BOOK GENERATED	22 October 2026 at 07:46 UTC

SYNOPSIS

Daniel wants a pocket of storage inside his ext4 home server — a virtual disk that lives as a single file, mounts from fstab, and behaves like a real partition. No VM required. The kernel already knows how to pretend a file is a disk, and that trick is called a loop device. This episode walks through building image files with truncate versus fallocate, why sparse files can fail at 3am, how LUKS encryption and Squashfs read-only images ride on top of the loop, and why swap files take a completely different path through the kernel. Plus: project quotas as the boring alternative, fscrypt for per-directory encryption, and what happens when you try to hand a laptop swap space over the network.

TRANSCRIPT



Corn

The standard advice for sealing off a chunk of storage on a Linux machine is to repartition the disk, or if that sounds like too much trouble, stand up a virtual machine and let it own the space. Daniel wants to know if there's a third way. He's got a home server with an ordinary ext four filesystem, and he wants a pocket inside it, a virtual disk that lives as one file, has its own filesystem or its own job, and mounts from fstab like a real partition. His favourite example is a slab of server storage that a laptop across the network uses as swap, and encrypted containers are the other obvious use.



Herman

The short answer is yes, and no VM is involved. A virtual machine is a whole second computer. What Daniel wants is much smaller than that, because the kernel already knows how to pretend that a plain file is a disk. That one trick is called the loop device, and most of what he's asking about hangs off it.



Corn

So the file is the disk.



Herman

The file is the disk as far as the rest of the system can tell. You create a file of whatever size you want, you tell the kernel to present it as a block device, and from then on you can format it, mount it, partition it, encrypt it. The tool is losetup. It finds the first free loop device, attaches your file, and prints the name, something like loop seven. And the kernel doesn't care that underneath there's an ext four filesystem holding the file.



Corn

Which is a filesystem sitting inside a filesystem. I assume that costs something.



Herman

It does, and we'll get to the bill. First the good news, which is the fstab part. You make the image, you put a filesystem in it with mkfs, and then in fstab you give it the path of the file, a mount point, the filesystem type, and the word loop in the options. At boot, mount sets up the loop device for you and mounts it. It sits in the fstab next to your real



Herman (continued)

partitions and nobody can tell the difference except by looking at what the source is.



Corn

And making the file in the first place?



Herman

Two schools. You can use truncate, which just sets the length. Say it's a hundred gigabytes, and the file instantly claims to be a hundred gigabytes, but it's a hole. Nothing is allocated. The blocks get handed out as you write to them. That's a sparse file.



Corn

Instant and free. There's a catch coming.



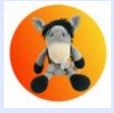
Herman

There's a catch. The image says a hundred gigabytes and the host has fifty left. Everything works until the guest filesystem has written fifty gigabytes of real data, and then the host runs out of space underneath it. The filesystem inside has no idea. It thinks it has plenty of room. So it tries to write, the host can't allocate, and you get I/O errors in the middle of what the inner filesystem believed was a healthy disk.



Corn

So it's an overdraft facility. Fine until everyone turns up at the bank.



Herman

For anything you intend to keep mounted for years, preallocate. Make the host commit the space up front, with fallocate or by just writing zeros with dd, so the failure shows up when you create the image and not at three in the morning. Sparse is great for scratch images and test rigs. For the pocket in Daniel's fstab, I'd pay for the space.



Corn

Now, the swap file. Is that the same thing? A swap file is a file the system treats as swap, so I'd guess you attach it to a loop device and off you go.



Herman

No, and this is a place where the intuition leads you wrong. A swap file doesn't go through a loop device. The kernel's swap code reads the file's block mapping, where on the real disk each piece of the file sits, and writes straight there. It doesn't go through the filesystem for each page, because swapping out a page has to work when memory is already short and you can't afford to start allocating things.



Corn

So no holes.



Herman

No holes. If the file is sparse, swapon refuses and tells you the file has holes. That's why the swapon man page says the portable way is to build the file with dd from dev zero. Fallocate may or may not leave you hole-free depending on the filesystem. Preallocated swap files are documented as supported on XFS from kernel four eighteen. I'm not certain how every ext four version treats a fallocate file, so I'd create it, run swapon, and believe whatever it says. If it complains, fall back to dd. Btrfs has its own rules. It supports swap files, but the file has to be fully preallocated, copy-on-write is switched off for it, so no checksums and no compression, and you can't snapshot that subvolume while the swap is active.



Corn

That's a lot of conditions for a feature that is supposed to be a file.



Herman

It's the price of the kernel not being able to be clever at the worst possible moment.



Corn

Herman, I'll ask you something. Say I put a swap file on my laptop's own disk. Why would I ever prefer that to a swap partition? I like partitions. They're old, they're boring, and they've never once tried to surprise me.



Herman

Because you can resize a file with a command and a swapoff, and a partition means a rescue stick and some nerve. I know you'll defend the partition to the grave, but the performance is the same once it's mapped. The file is just easier to change your mind about.



Corn

I'll keep the partition for sentimental reasons. Next pocket.



Herman

Encryption, the one Daniel mentioned. LUKS works on any block device, and a loop device is a block device, so you can make a file, put a LUKS header on it, and open it, which gives you a mapped device with a normal filesystem on top. As far as I know cryptsetup will attach the loop device itself if you hand it a file path, but I'd check that against your own version before relying on it. Close it, and what's left on the disk is a file full of noise. You can copy it to a USB stick, put it in a cloud bucket, mail it. The encrypted pocket travels as one object. Corn, would you put everything you own in one file like that?



Corn

I'd put the tax records in it and nothing I'd miss. One file to back up, one file to lose, and I know which of those I'd worry about.



Herman

Then read-only pockets. Squashfs is a compressed read-only filesystem. You build an image from a directory, and you mount it with the loop option. Live installers use it, and it's lovely for things you want frozen, like an archive of a project or a bundle of tools. Nothing can change it, it's compressed, and it's one file.



Corn

And that you can mount on four machines without any of them being able to ruin it.



Herman

And there's a heavier option, qcow two, the image format qemu uses. It supports snapshots and growing on demand. You can attach one to the host without running a VM, using a tool called qemu-nbd, which exposes the image as a network block device on the same machine, and then you mount the partitions in it. I haven't walked through every step recently, so treat that as a pointer, not a recipe. Most people won't need it unless they already have qcow files lying around.



Corn

Does the host run out of loop devices if you go mad with these?



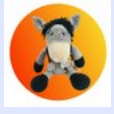
Herman

There's a cap on loop devices in some setups. Eight, historically, raisable with a kernel parameter. If you plan a dozen pockets it's worth knowing.



Corn

And now let's hear the voicemail from Siobhan.
Siobhan: Hi lads, Siobhan from Cork, I'm on the forty five minute bus into town, sitting upstairs, and the whole top deck smells of a bag of chips someone is after opening, and I've the phone up against the window. I think the topic today is about making a pretend hard drive inside the real hard drive, sort of like a pocket sewn into the lining of a coat, which I think is a lovely idea, because my granny used to sew coins into the hem of her good cardigan before every wedding, and I suppose my question is, if you carve out that pocket and then the pocket gets sick, does the whole coat know, or does the rest of the coat just carry on as if nothing happened? Anyway, that's me, bye bye.



Herman

The coat carries on, Siobhan. If the filesystem inside the image gets corrupted, the host sees one perfectly healthy file and has no opinion about what's in it. You run a repair on the pocket and the coat never notices. What the coat can't survive is the whole thing tearing, because if the host disk dies, every pocket on it goes with it, so the backups still have to live somewhere else.



Corn

Back to the overhead, then. The inner filesystem keeps a journal, and the outer one keeps a journal for the file holding it. Two journals, two layers of caching.



Herman

The part I can't tell you with confidence is exactly how a flush travels down through the loop device. The loop driver has a direct I/O option that bypasses one layer of caching, and it's off by default. Whether your fsync really reached the platter or only the outer cache depends on how the layers are set up. For a database in a pocket, I'd read the documentation for my own kernel instead of trusting my memory.



Corn

So for the family photographs, fine. For the accounts database, I'd keep it on the real disk.



Herman

I'd agree with that, but my reason would be different. You'd keep it there so there's one fewer layer between it and the bad day.



Corn

There's an irritating alternative, which is no image at all. If all Daniel wants is to stop one directory eating the server, a file isn't necessary.



Herman

Project quotas. ext four supports them. You turn the feature on when you make the filesystem or with tune two fs, mark a directory with a project ID, and set limits per project. A directory can't grow past its allowance and there's no inner filesystem, no loop, no double journal. It's a limit and nothing more, so it's not a pocket in the sense of having its own filesystem, but if the aim is a capped slab, it's the cheapest way to get one.



Corn

And for encryption without a container?



Herman

Fscrypt. Per-directory encryption built into ext four, F two F S and a few others. Unprivileged users can use it and there's no mount step. The limits are real, though. Only the file names and contents are protected. Sizes, permissions and timestamps stay visible. For a stolen laptop it's good. For hiding what you have, an encrypted container hides far more. What does your gut say about the network swap, before I explain it? You've been itching to say.



Corn

That it sounds fine and I'd be surprised if it were. Daniel has a server with a lot of space and a laptop that wants more memory, and he'd like to hand the laptop a slab across the network.



Herman

It's possible, which is the first thing to say. There are three ways. The simplest is a swap file on an NFS mount. That was broken for years, because the swap writes ran through a check that always failed on a swap file, and the swapon man page still says swap over NFS may not work. But the repair series landed, and since Linux five point nineteen, from the summer of two thousand twenty-two, a swap file on NFS is supported again. The man page is stale.



Corn

We did NFS versions and the gotchas in an earlier episode. I don't want to go back into it. We're here for the block device.



Herman

The block-level options are nbd-server on the server, which exports a file as a network block device, and nbd-client on the laptop, which makes it appear as a local disk you can mkswap. Or an iSCSI target, where you make a file-backed backstore with targetcli and export it as a LUN. Or NVMe over TCP, where a namespace can point at a file. I've seen the configuration attribute but I haven't run one end to end, so I won't walk anyone through it.



Corn

And all of these work.



Herman

They work. Under a calm system they work fine. The trouble comes when memory is short, which is the only time swap matters. Think about what has to happen. The kernel needs to push a page out to free memory. To write it over the network, it needs to build a packet and send it. To know the write succeeded, it needs to receive the reply. Receiving the reply takes memory, because the packet has to land somewhere before the kernel can even identify whose reply it is.



Corn

So the system is out of memory, asks the network for help, and the network needs memory to answer.



Herman

Yes. A local disk gets emergency pools set aside for exactly this. The network stack didn't, and for roughly a decade that was an open problem. Patches in the mid two thousands described machines deadlocking while swapping to network storage. A later set of patches tagged the socket used by network block devices so it can draw on the emergency reserve, and that's the mitigation that exists now. It's a mitigation, not a proof. An intensively swapping machine over NBD, before those patches, could deadlock inside minutes.



Corn

So the kernel's position on network swap is that it will try its best.



Herman

Roughly, yes. And that's on a wired connection with a stable link. Daniel said laptop. A laptop on Wi-Fi is going to wander in and out of range, the access point will hiccup, someone will microwave something. If the swap device vanishes for ten seconds, whatever process had pages out there freezes, and sometimes the whole machine.



Corn

Herman, I have to ask. Would you do it? You with a fast wired link, a patient server, and a laptop short of memory.



Herman

On wired gigabit, for a machine I could afford to reboot, as an experiment, yes. As the thing I depend on, no. A gigabit link has round trips in the low fractions of a millisecond, and an NVMe drive answers in tens of microseconds. Those are my estimates, not a benchmark. The gap is large, and swap is latency before it's anything else.



Corn

So what do I do with a laptop short of memory?



Herman

Compress in RAM. Zram makes a compressed block device out of memory, you run mkswap on it and turn it on as swap, and pages that would have gone to disk get squeezed and stay in RAM. Zswap does something similar as a cache in front of a real swap device. On a laptop that's usually the better answer, and when swap is in memory like that, people tune the swappiness setting upward, sometimes past a hundred.



Corn

So the network pocket is an answer to a question laptops don't really ask.



Herman

A server with a spare disk and a lot of RAM is more likely to benefit. The laptop wants the network pocket for backups, for the encrypted vault, for the archive. Those are all better uses of a file on the server.



Corn

I'll accept that. I'd just like it noted that my laptop has never complained about swap, partly because I use it for very little.



Herman

You use it for a browser and a text file, Corn.



Corn

Two text files. I'm ambitious.



Hilbert

...and Pettigrew next door says the file can be any size as long as it weighs no more than his ferret. Four hundred and ten grams. So four hundred and ten megabytes, and I've told him that's all the swap that laptop is ever going to want. He keeps a server in his garage that the council has written down as a bird.



Herman

A bird?



Hilbert

Category on the form. Nobody has argued with it since. The laptop is mine and I'm moving its swap into his garage this week. Seventy metres of cable, forty-one pounds from the place on Marsh Road. It goes through the goat shed. Brenda has eaten two cables already.



Corn

The goat is now a component of the storage system.



Hilbert

Pettigrew holds the plug end, because the crimper is made for hands and I'm an anteater. The ferret sits on the connector the whole time so it knows what four hundred and ten grams is.



Herman

A hard cap on the file size makes sense whatever the reason behind it. Four hundred and ten megabytes is plenty if the machine only uses swap as a safety net.



Hilbert

Herman, you took a whole turn to get to the word zram. I'd have said it first. Pettigrew has been running it on the old laptop since Thursday.



Corn

Then the ferret is only the backstop.



Hilbert

He ran that laptop six days on the cable. It used eleven megabytes of the swap. You said the reply needs memory before anyone knows who it's for, Herman. I've stopped taking in Pettigrew's post.



Corn

Four hundred and ten megabytes of cable and a goat, to protect eleven.



Hilbert

The four images are with my brother-in-law Gerald, on a thumb drive in a sock drawer in Leominster. He says the drawer is the backup.



Herman

A sock drawer is a second host disk with nobody watching the temperature.



Hilbert

Nineteen minutes, the crimp took. Pettigrew timed it with the ferret.



Herman

Eleven out of four hundred and ten says the swap was mostly insurance, and that matches what I'd expect from a machine with enough memory. People size swap for the worst day and then find out the machine lives on the best one.



Corn

Which is an argument for keeping it small and local, and putting the network pocket to work on the things a server is good at.



Herman

One more detail, since it's the kind of thing guides get wrong. The swapon manual still says swap over NFS may not work, and the code that fixed it has been in the kernel since five point nineteen. Check the kernel version before you give up on the idea.



Corn

The pocket was never the hard part, then. The hard part is deciding what happens on the day the host fills up or the disk dies, and the answer to that is a backup that lives somewhere else.



Herman

Thanks to our producer Hilbert Flumingtop for keeping the desk running.



Corn

This has been My Weird Prompts. Every episode is at my weird prompts dot com.



Herman

See you tomorrow.