RAI 1

🧩 quiz

tiny VMs for kernel development

Rob Norris

Hello!

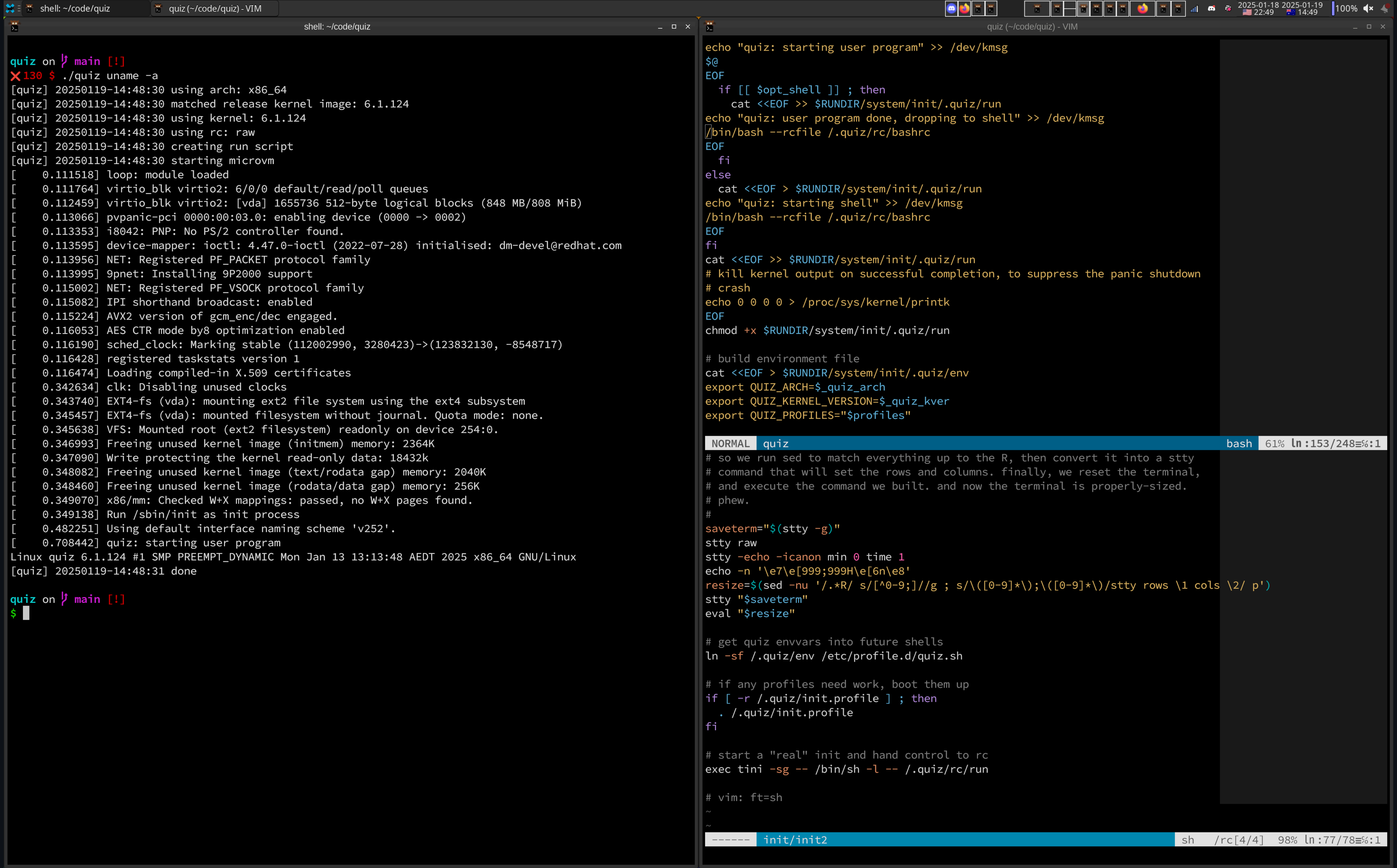
- OpenZFS developer
- Recovering Linux sysadmin
- FreeBSD non-committer

💸

Support independent software development

🌏

exploratory programming



```
quiz on ᚠ main [!]
✖130 $ ./quiz uname -a
[quiz] 20250119-14:48:30 using arch: x86_64
[quiz] 20250119-14:48:30 matched release kernel image: 6.1.124
[quiz] 20250119-14:48:30 using kernel: 6.1.124
[quiz] 20250119-14:48:30 using rc: raw
[quiz] 20250119-14:48:30 creating run script
[quiz] 20250119-14:48:30 starting microvm
[    0.111518] loop: module loaded
[    0.111764] virtio_blk virtio2: 6/0/0 default/read/poll queues
[    0.112459] virtio_blk virtio2: [vda] 1655736 512-byte logical blocks (848 MB/808 MiB)
[    0.113066] pvpanic-pci 0000:00:03.0: enabling device (0000 -> 0002)
[    0.113353] i8042: PNP: No PS/2 controller found.
[    0.113595] device-mapper: ioctl: 4.47.0-ioctl (2022-07-28) initialised: dm-devel@redhat.com
[    0.113956] NET: Registered PF_PACKET protocol family
[    0.113995] 9pnet: Installing 9P2000 support
[    0.115002] NET: Registered PF_VSOCK protocol family
[    0.115082] IPI shorthand broadcast: enabled
[    0.115224] AVX2 version of gcm_enc/dec engaged.
[    0.116053] AES CTR mode by8 optimization enabled
[    0.116190] sched_clock: Marking stable (112002990, 3280423)->(123832130, -8548717)
[    0.116428] registered taskstats version 1
[    0.116474] Loading compiled-in X.509 certificates
[    0.342634] clk: Disabling unused clocks
[    0.343740] EXT4-fs (vda): mounting ext2 file system using the ext4 subsystem
[    0.345457] EXT4-fs (vda): mounted filesystem without journal. Quota mode: none.
[    0.345638] VFS: Mounted root (ext2 filesystem) readonly on device 254:0.
[    0.346993] Freeing unused kernel image (initmem) memory: 2364K
[    0.347090] Write protecting the kernel read-only data: 18432k
[    0.348082] Freeing unused kernel image (text/rodata gap) memory: 2040K
[    0.348460] Freeing unused kernel image (rodata/data gap) memory: 256K
[    0.349070] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[    0.349138] Run /sbin/init as init process
[    0.482251] Using default interface naming scheme 'v252'.
[    0.708442] quiz: starting user program
Linux quiz 6.1.124 #1 SMP PREEMPT_DYNAMIC Mon Jan 13 13:13:48 AEDT 2025 x86_64 GNU/Linux
[quiz] 20250119-14:48:31 done

quiz on ᚠ main [!]
$
```

```
echo "quiz: starting user program" >> /dev/kmsg
$@
EOF
  if [[ $opt_shell ]] ; then
    cat <<EOF >> $RUNDIR/system/init/.quiz/run
echo "quiz: user program done, dropping to shell" >> /dev/kmsg
/bin/bash --rcfile /.quiz/rc/bashrc
EOF
  fi
else
  cat <<EOF > $RUNDIR/system/init/.quiz/run
echo "quiz: starting shell" >> /dev/kmsg
/bin/bash --rcfile /.quiz/rc/bashrc
EOF
fi
cat <<EOF >> $RUNDIR/system/init/.quiz/run
# kill kernel output on successful completion, to suppress the panic shutdown
# crash
echo 0 0 0 0 > /proc/sys/kernel/printk
EOF
chmod +x $RUNDIR/system/init/.quiz/run

# build environment file
cat <<EOF > $RUNDIR/system/init/.quiz/env
export QUIZ_ARCH=$_quiz_arch
export QUIZ_KERNEL_VERSION=$_quiz_kver
export QUIZ_PROFILES="$profiles"

 NORMAL  quiz                                                        bash  61%  ln :153/248≡%:1
# so we run sed to match everything up to the R, then convert it into a stty
# command that will set the rows and columns. finally, we reset the terminal,
# and execute the command we built. and now the terminal is properly-sized.
# phew.
#
saveterm="$(stty -g)"
stty raw
stty -echo -icanon min 0 time 1
echo -n '\e7\e[999;999H\e[6n\e8'
resize=$(sed -nu '/.*R/ s/[^0-9;]//g ; s/\([0-9]*\);\([0-9]*\)/stty rows \1 cols \2/ p')
stty "$saveterm"
eval "$resize"

# get quiz envvars into future shells
ln -sf /.quiz/env /etc/profile.d/quiz.sh

# if any profiles need work, boot them up
if [ -r /.quiz/init.profile ] ; then
  . /.quiz/init.profile
fi

# start a "real" init and hand control to rc
exec tini -sg -- /bin/sh -l -- /.quiz/rc/run

# vim: ft=sh
~
~
 ------  init/init2                                          sh   /rc[4/4]  98%  ln :77/78≡%:1
```

🌏 exploratory kernel programming

- every crash is a reboot
- every deadlock is a reboot
- boot times are slow
- unclean shutdown damages filesystems
- traditional VMs are a pain to manage if you're blowing them up all the time
- I get bored and distracted very easily

🧠 Big thoughts

- We run programs in modified environments all the time:
  - alternate environment: `env VAR=val /some/program`
  - alternate filesystem: `chroot /some/path /some/program`
  - alternate language: `bash /some/program.sh`, `perl /some/program.pl`
- If you squint:
  - a VM (hypervisor) is just a program that runs a kernel
  - a kernel is just a program that runs a program called `init`
  - `init` is just a program that runs another program

🧠 Big thoughts

```
$ zfs-kernel-runner my-zfs-test-script.sh
```

🥅 Goals

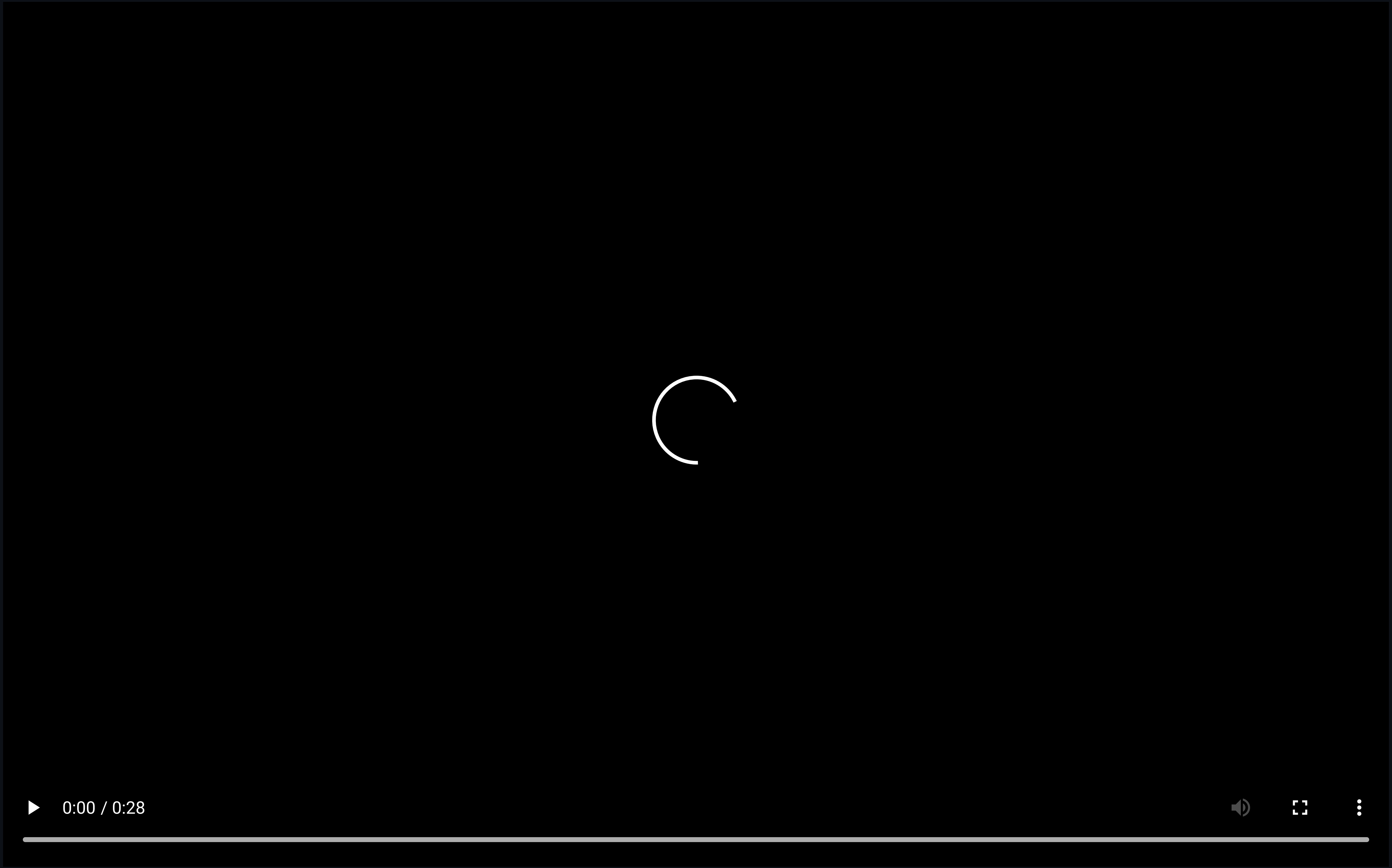
- Feels just like another program
    - Output to stdout, so we can grep it
    - Ctrl-C will kill it
- Gets into the test program in a couple of seconds
- Completely gone without a trace when it completes
- Minimal extra typing
- Get new code and test programs direct from the host filesystem

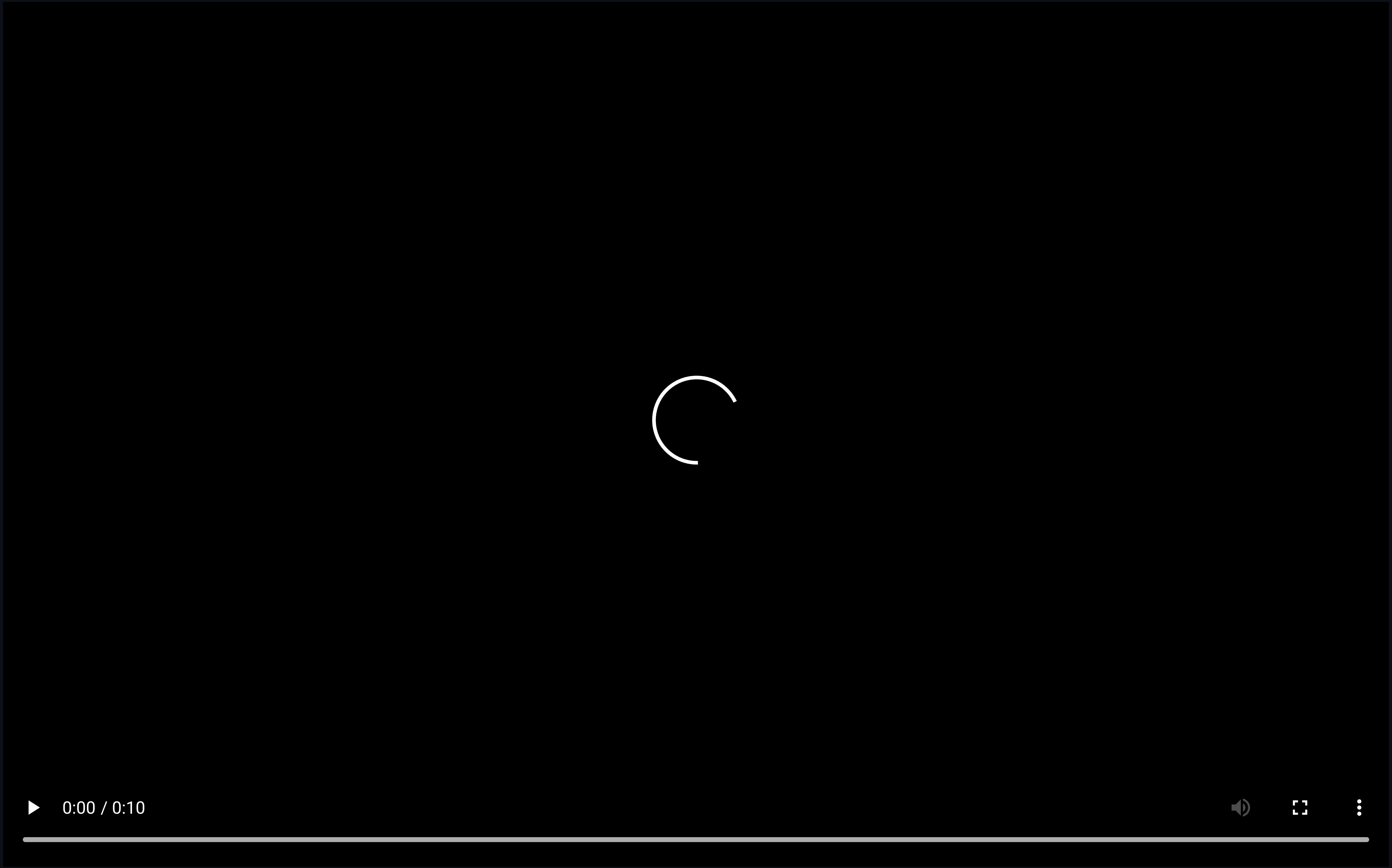
quiz

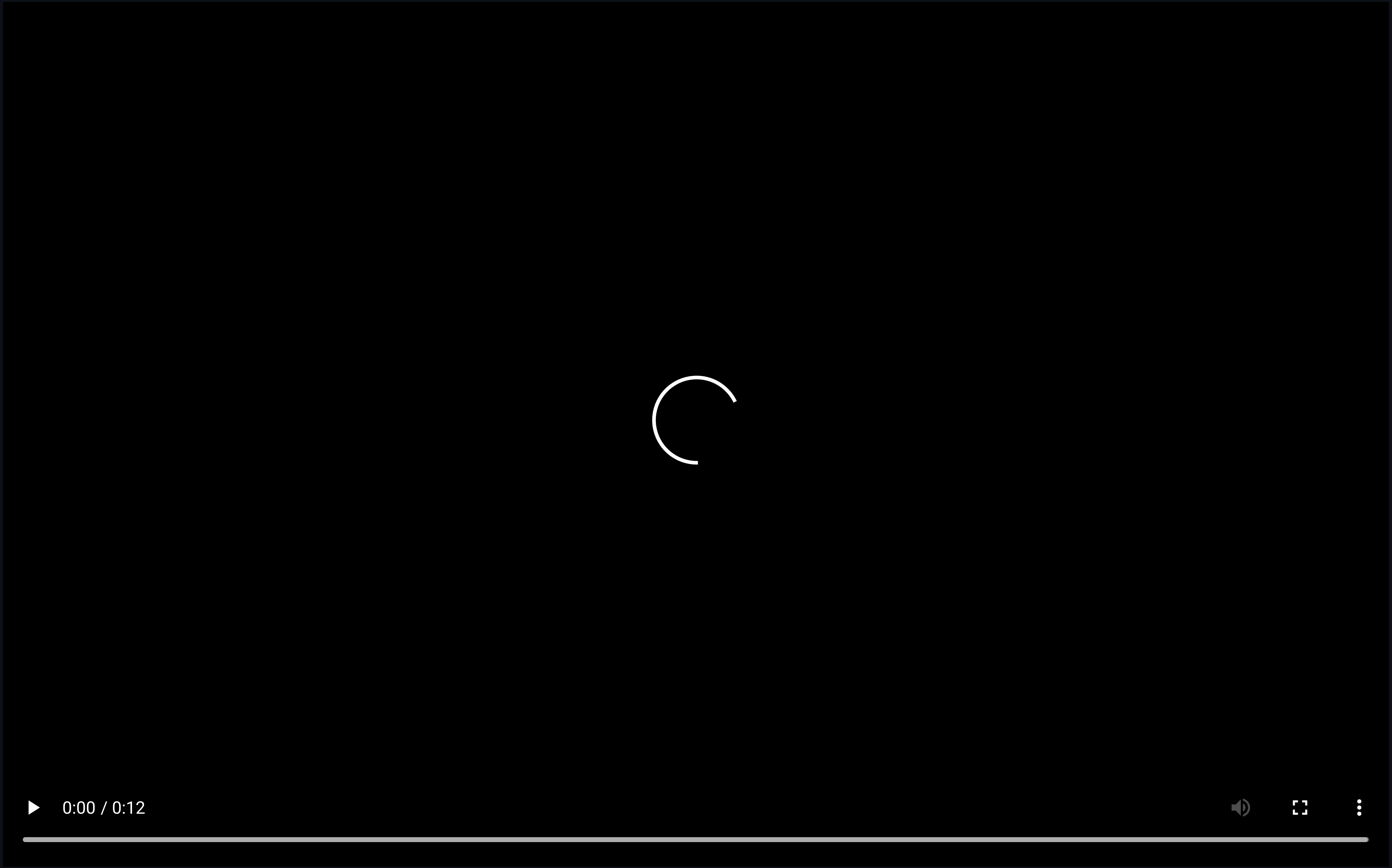
https://github.com/robn/quiz

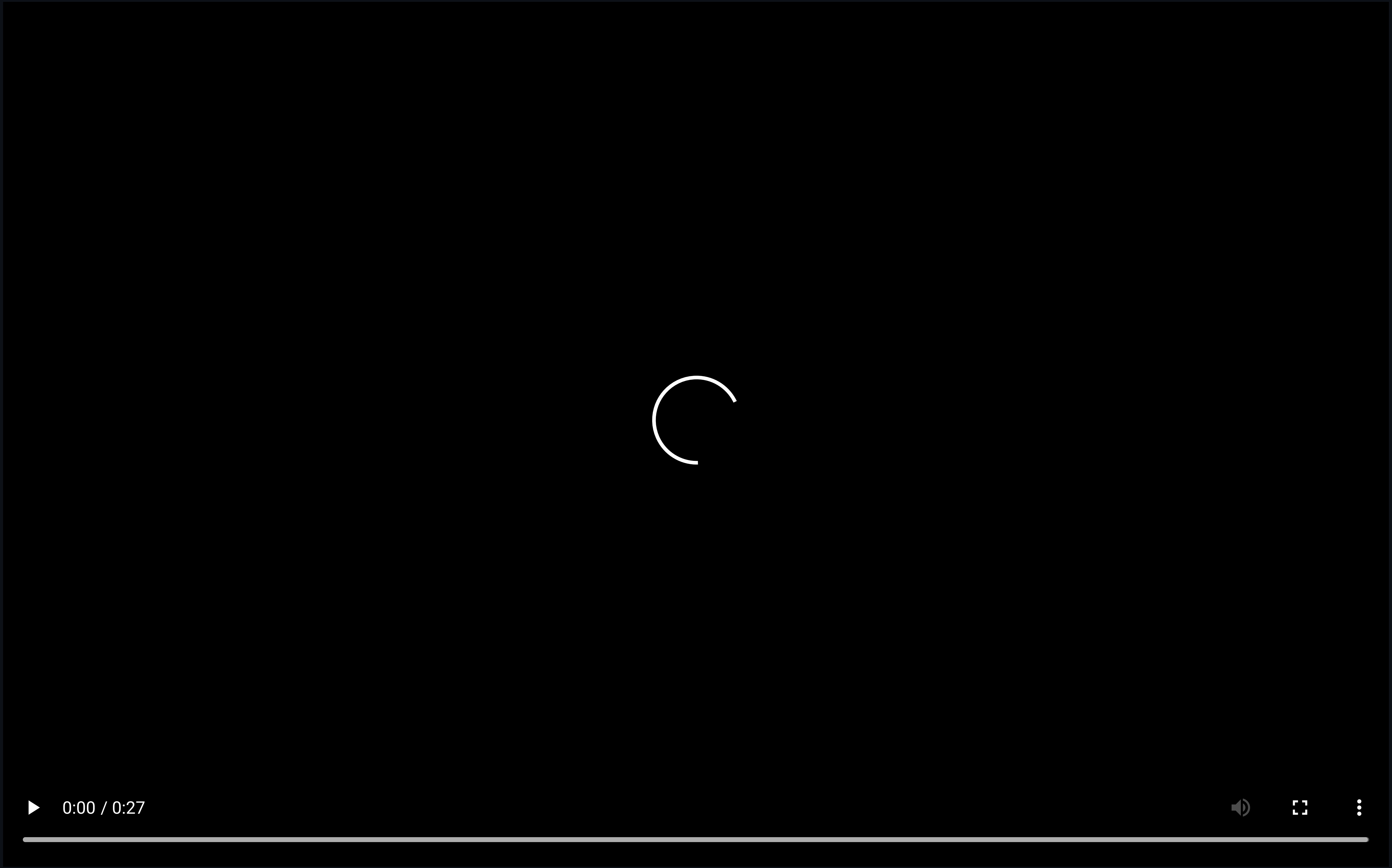
🧩 demo

basic operation


0:00 / 0:28


0:00 / 0:10


0:00 / 0:12


0:00 / 0:27

🧩 quiz

- QEMU `microvm` machine model
- Custom build of Linux kernel
- Minimal Debian userspace
- Custom boot process
- 9pfs+overlayfs to build the root filesystem
- Profiles to add devices or facilities to this run
- OpenZFS build support
- Kernel build support

🧩 MicroVM?

- QEMU "machine model"
  - the kind of "whole computer" being emulated
  - Architecture, CPU model, board type, core devices and controllers
- `microvm`: A minimalist `x86_64` / `amd64` machine model
  - yes: PCI bus, ISA bus, LAPIC, IOAPIC, clock, virtio-mmio/pci slots
  - no: BIOS, ACPI, option ROMs, ISA serial, PIC, PIT, RTC
- Fast boot: nothing to discover, nothing to initialise
  - Known, fixed, minimal set of devices

🧩 Custom kernel build

- Bare minimum device support
    - No need to initialise devices that aren't there
    - Or enumerate buses that aren't there
    - Or discover devices when we already know where they are
- All drivers built into kernel, no modules
    - No initrd required to boot!

🧩 Minimal Debian userspace

- `minbase` variant: "required" packages + package manager
- plus useful tools for this task:
  - performance and profiling: `perf`, `bpftrace`, `fio`, `gdb`, `strace`, `blktrace` …
  - block device construction: `gdisk`, `dmsetup`, `cryptsetup`, …
  - OpenZFS test suite support: `ksh`, …
  - Boot support: `tini`, `udev`, `kmod`, …

🧩 Custom boot process

- `init1`: first stage; build the root filesystem, pivot
- `init2`: second stage; prepare userspace, profile init
- `tini`: bare-minimum PID 1
- `rc`: run control, the "user interface" to the run
- `run`: the actual test program or other thing to run

🧩 `init1`: filesystem construction

🧩 `init1`: filesystem construction

```
Filesystem      1K-blocks  Used Available Use% Mounted on
overlay           8198812   144   8198668   1% /
```

- Linux `overlay` filesystem
  - build a "virtual" filesystem by layering other filesystems over the top
  - file not found at one layer, try the next one

🧩 `init1`: filesystem construction

- base system image: ext2
  - Debian `minbase` + extras + `init1`
- quiz kernel dir: 9pfs (host dir)
  - compiled kernels, debugging symbols, system map
- quiz system dir: 9pfs (host dir)
  - install target for OpenZFS, built outside
- quiz init dir: 9pfs (host dir)
  - script fragments, config, etc created by `quiz` script for this run
- quiz user dir: 9pfs (host dir)
  - test scripts and other random stuff I want inside
- top: tmpfs
  - writes inside the VM go here, and disappear later

`init1`: filesystem construction

```
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/root          820625    753572     25660  97% /
quiz-kernel     440391552 397204608  43186944  91% /mnt/quiz-kernel
quiz-system     440391552 397204608  43186944  91% /mnt/quiz-system
quiz-init       440391552 397204608  43186944  91% /mnt/quiz-init
quiz-user       440391552 397204608  43186944  91% /mnt/quiz-user
tmpfs             8198812         0   8198812   0% /mnt/top
```

🧩 `init1`: filesystem construction

```
mkdir /mnt/top/upper /mnt/top/work

mount --bind / /mnt/lower

mount -t overlay overlay
      -o lowerdir=/mnt/quiz-init:
                  /mnt/quiz-user:
                  /mnt/quiz-kernel:
                  /mnt/quiz-system:
                  /mnt/lower,
         upperdir=/mnt/top/upper,
         workdir=/mnt/top/work
      /mnt/newroot
```

`init1`: filesystem construction

```
Filesystem     1K-blocks      Used Available Use% Mounted on
/dev/root         820625    753572     25660  97% /
quiz-init      440391552 397204608  43186944  91% /mnt/quiz-init
quiz-user      440391552 397204608  43186944  91% /mnt/quiz-user
quiz-kernel    440391552 397204608  43186944  91% /mnt/quiz-kernel
quiz-system    440391552 397204608  43186944  91% /mnt/quiz-system
tmpfs            8198812         0   8198812   0% /mnt/overlay
overlay          8198812         0   8198812   0% /mnt/newroot
```

🧩 `init2`: prepare environment

- set the hostname
- get `/dev` nodes up (`udev`)
- mount debug filesystems (`tracefs`, `debugfs`, `configfs`, `bpf`, ...)
- do profile init
- exec `tini` as PID 1

🧩 `tini`: the littlest PID 1 that could

https://github.com/krallin/tini

- runs a program
- reaps zombies
- provides default signal handlers
- the "standard" PID 1 for containers

🧩 `rc`: run control

- the "user interface" for a quiz run
- what you see on your screen
- where your keypresses go

- `raw`: kernel and program output on stdio, Ctrl-C kills the VM
- `tmux`: everything inside a tmux session, interactive & exploratory

`run`

profiles

🧩 profiles

- chosen via commandline option
- add extra stuff to this run
- profiles can:
  - run stuff on the host, before the VM starts (`quiz`)
  - run stuff in the guest, before the user program starts (`init2`)
  - provide extra files that will be included in the guest

🧩 profile: `zfs`

- `init2`: install `zfs` module into kernel

profile: `memdev`

- `init2`: create some small (100M) memory-backed block devices (`/dev/loopX`)

🧩 profile: `blockdev`

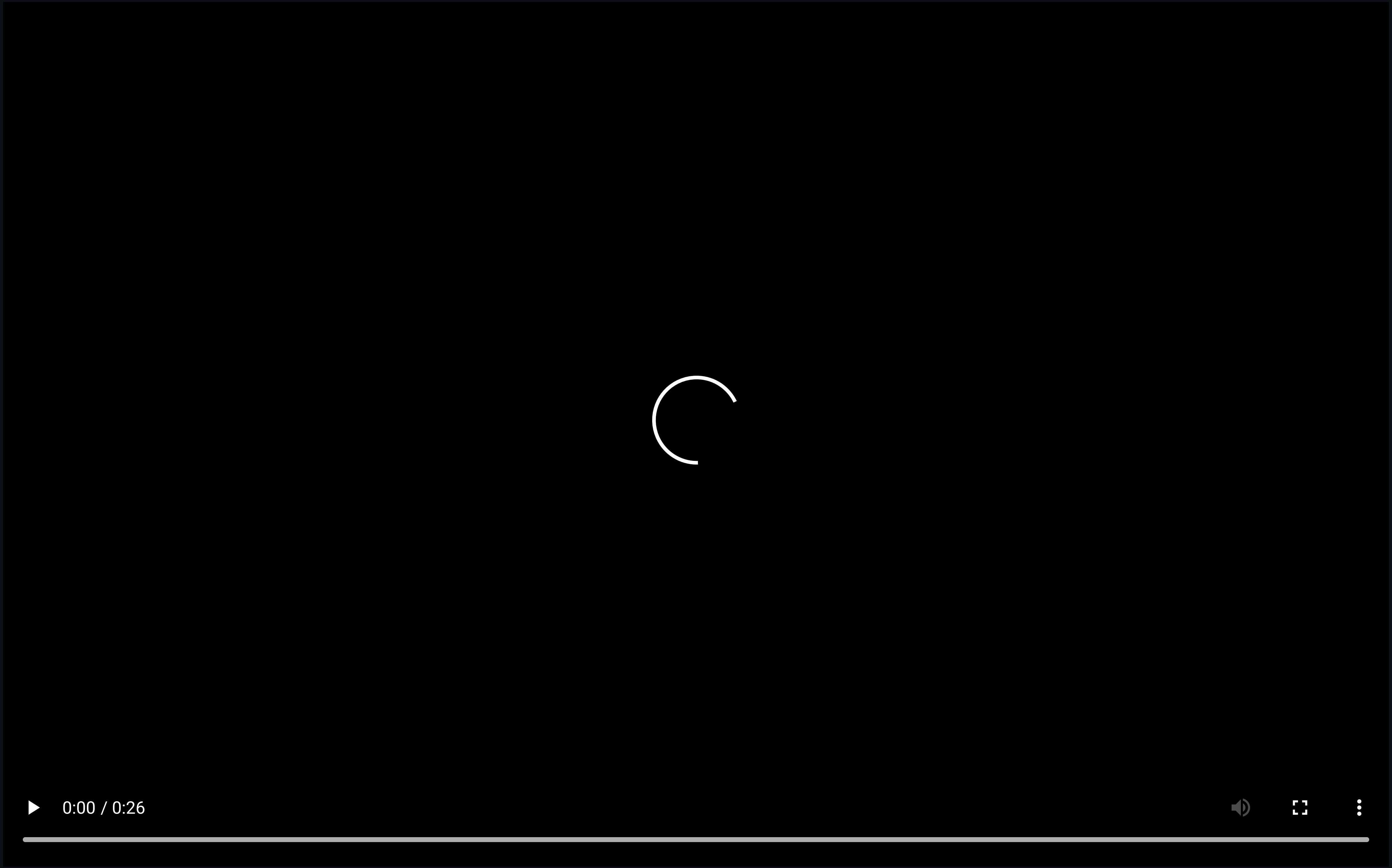
- `quiz`:
  - create some 1G sparse files as block device backing
  - extend `qemu` command line to attach them as virtio-blk devices

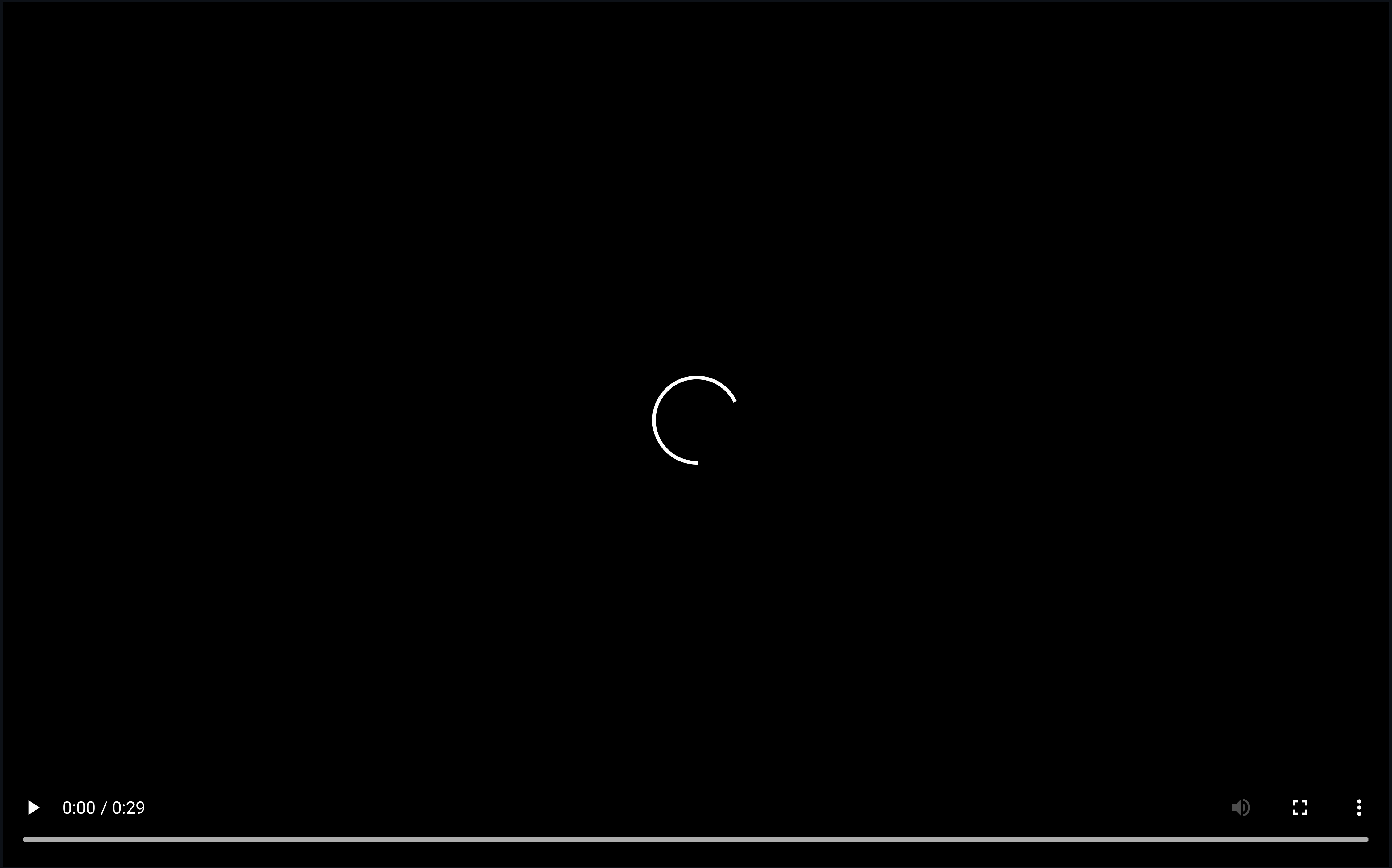
profile: `ztest`

- files: provide "no-op" variants of `sudo` and `id` to work around ZTS assumptions

🧩 demo

profiles


0:00 / 0:26


0:00 / 0:29

OpenZFS build support

OpenZFS build support

The dream:

```
$ ./autogen.sh
$ ./configure --prefix=/path/to/quiz/system
$ make -j6
$ make install
```

OpenZFS build support

The reality:

```
$ ./autogen.sh
$ ./configure \
    --with-linux=/path/to/quiz/system/kernel/x86_64/kbuild/6.1.124 \
    --prefix=/usr/local \
    --disable-sysvinit \
    --disable-systemd \
    --disable-pam \
    --disable-pyzfs \
    --with-mounthelperdir=/usr/local/sbin \
    --with-dracutdir=/usr/local/lib/dracut \
    --with-udevdir=/usr/local/lib/udev \
    'lt_cv_sys_lib_dlsearch_path_spec=/lib /usr/lib /usr/lib/x86_64-linux-gnu'
$ make -j6
$ make install DESTDIR=/path/to/quiz/system
```

🧩 OpenZFS build support: `quiz-zfs`

For now:

```
$ ./autogen.sh
$ quiz-zfs configure
$ make -j6
$ quiz-zfs make install
```

💾 OpenZFS Linux support

- OpenZFS 2.3.0 & 2.2.7
    - Linux 4.18 (August 2018) - 6.12 (November 2024)
    - Red Hat Enterprise Linux 8.10: 4.18+
    - Ubuntu 18.04.5 LTS (HWE): 5.4+

🧩 Kernel build support: `quiz-kernel`

- compiles specific kernels version: `-k 6.1.124`
  - or latest in series: `-k 6.1`
  - or release candidate: `-k 6.13.0-rc7`
  - or nightly build: `-k 6.13.0-next-20250117`
- upgrade all compiled kernels to latest: `-K -U -X`
- rebuild with changed config: `-k ... -e CONFIG_FOO -m CONFIG_BAR`
- rebuild with Clang/LLVM: `-k ... -L`

- run with specific kernel: `quiz -k 6.1.124`
- or any in series: `quiz -k 6.1 ...`

☀️ Sun ZFS

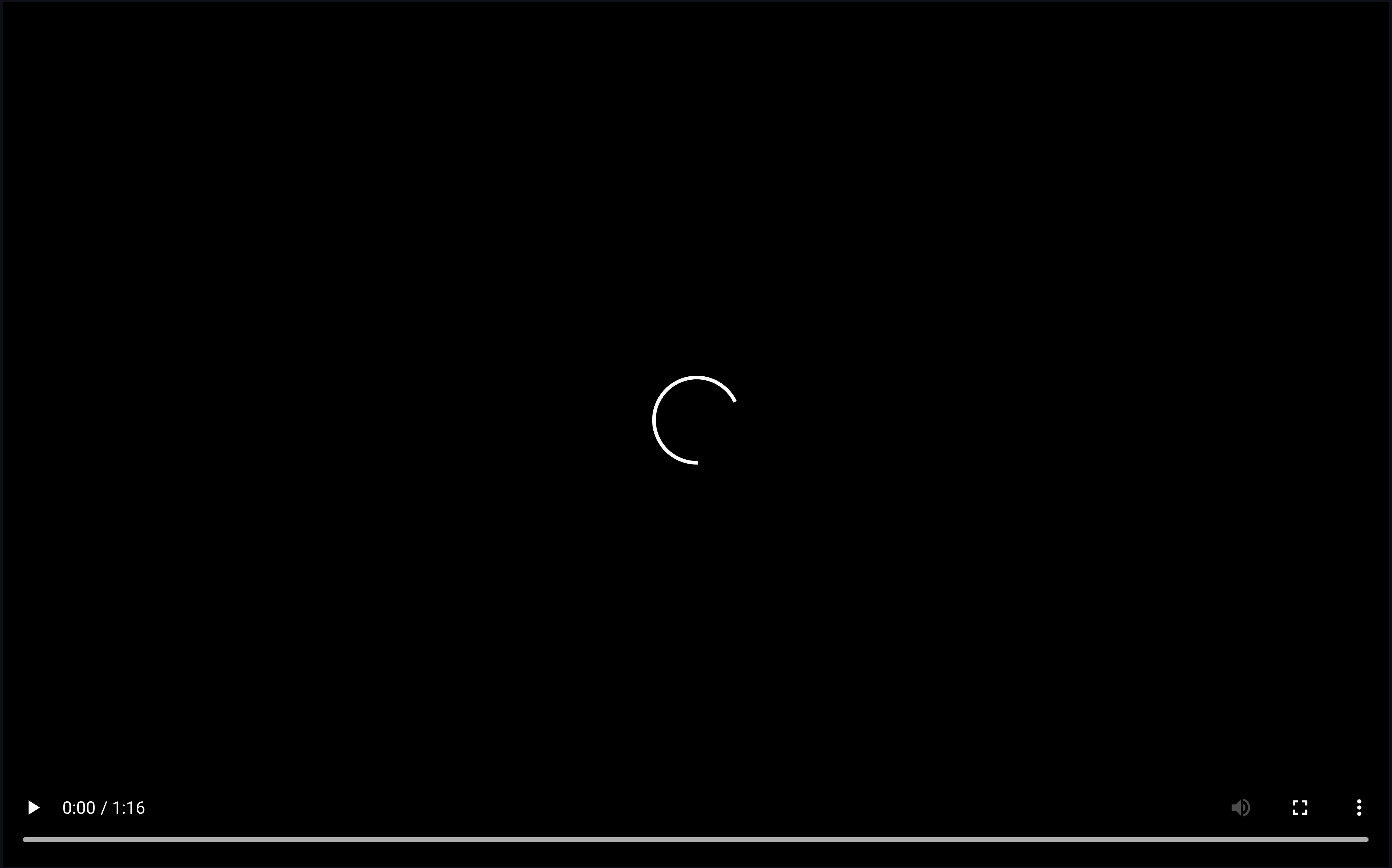
- OpenSolaris build 27 (2005), Solaris 10 6/06 U2 (2006)
    - SPARC (big-endian)
    - i386 (little-endian)
- ZFS is *endian-agnostic*
    - Stores everything in native endianness
    - With a flag indicating big or little
    - Pools imported on "foreign" endianness will be byte-swapped on the fly
- OpenZFS inherits this legacy

🧩 Multiple architectures

- `ppc64` support
  - `qemu` machine emulation
  - cross-compile kernels
  - cross-compile OpenZFS (WIP)
- Everything takes a `-a <arch>` argument

🧩 demo

tmux & multiarch


0:00 / 1:16

🧩 Plans and dreams

- multiple instances, for each OpenZFS checkout
- writable host mount (save logs and build artifacts)
- profiles for building block devices out of dm stacks
- perform the same run over multiple kernel versions

- (get a better puzzle piece for a logo, please help)

🧩 FreeBSD support

- OpenZFS is the default filesystem for FreeBSD

- Build more FreeBSD-specific features
- Make sure my changes work well on both platforms

🧩 FreeBSD support (guest)

- Extract `base.tgz`
- Cross-compile kernel `FIRECRACKER` config
- p9fs available in FreeBSD 15.x (December 2025)
- unionfs in early planning stages
  - use NFS?
  - use symlinks?
  - combine on host side into disk image?

🧩 FreeBSD support (host)

- `bhyve`: FreeBSD-native hypervisor
    - fundamentally different model to KVM
    - anonymous & self-destructing VMs coming in FreeBSD 15.x
    - initial support for Linux kernel direct load
        - (I need to find time to finish it)

🧩 FreeBSD support (host)

- `qemu` works now, but no hardware acceleration
- `bhyve` (`libvmm`) memory model doesn't match
- options still being considered
  - reimplement/extend `libvmm` to support `qemu` model
  - implement KVM (port from Illumos?)
  - port NVMM from NetBSD?
  - add the kitchen sink to `bhyve`?

quiz

https://github.com/robn/quiz

kernels are
just programs
do not listen
to their bulls••t