Architectures Don't Matter

Making Executables Universally Portable with QEMU

Amy Iris Parker
University of California, Irvine
SCaLE 23x - Pasadena, CA

About Your Presenter

PhD Student, UC Irvine

Distributed and embedded systems, computer architecture, networking, social impacts

Dutt Research Group - proactive compaction

Formerly at CSU Fullerton

Research on IR pipelines and profiling QEMU - led to this presentation

Also censorship evasion (Friday, 17:00)

Secretary, Orange County DSA

Non-profit tech stack management - voting, transparency, identification, contacts

FastRequest, OSSIGINT

Lifelong *nix user

Decade-long history of using Linux; 21x

Currently running a Proxmox homelab, task automation, NixOS systems

Prior Art

Significant portions of this presentation derive from

ICECET '25

Boosting Cross-Architectural Emulation Performance by Foregoing the Intermediate Representation Model
July 4, 2025 - Paris, France (presented online)

DebianWiki

QemuUserEmu & EmDebian/CrossDebootstrap

Overview

1. What is QEMU and what is user mode?
2. Running cross-arch user binaries/apps
3. Binfmt, multilib, and LD_PRELOAD
4. CrossDebootstrap and other chroots
5. Enterprise software and modern architectures
6. Universal Binaries!

Today’s Presentation

This **is not a workshop.**

We unfortunately don’t have the time today to all set up QEMU user-mode, binfmt, etc. Attendees are still encouraged to do so when able however!

This **is** informational and a place to start!

Today’s talk aims to show the potential for QEMU user mode as it is almost entirely unknown within the community. We’re all hackers here — ***hack!!***

1

What is QEMU?
What is user mode?

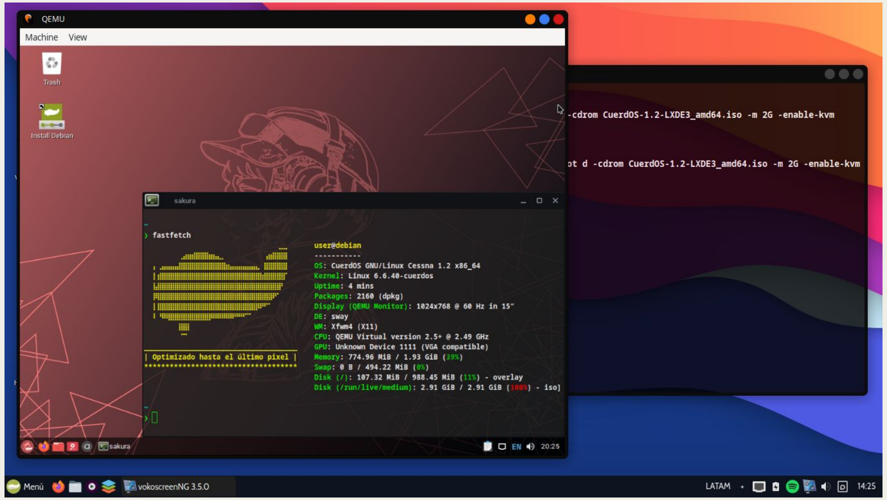
Many such questions…

What is QEMU?

QEMU is the current standard on Linux and several other platforms for both system-level and user-level emulation.

Most VMs, testing systems run through QEMU. Examples: nix boot-vm, buildroot.

QEMU supports 30 different ISAs and thousands of different hardware modules, CPU configurations, BIOS/UEFI versions, etc - and is extensible.



```
-cdrom CuerdOS-1.2-LXDE3_amd64.iso -m 2G -enable-kvm

ot d -cdrom CuerdOS-1.2-LXDE3_amd64.iso -m 2G -enable-kvm
```

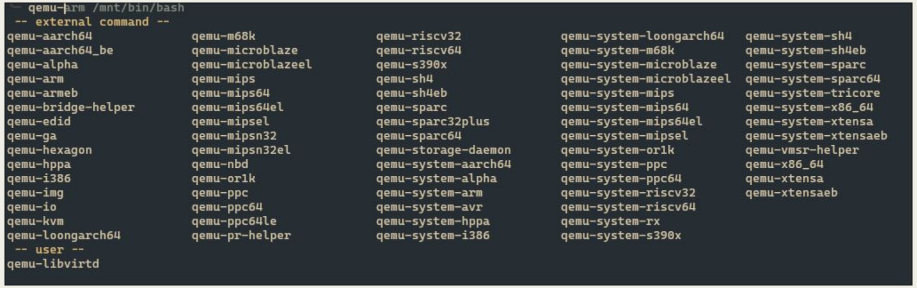
```
~
❯ fastfetch
                                        user@debian
                                        -----------
                                        OS: CuerdOS GNU/Linux Cessna 1.2 x86_64
                                        Kernel: Linux 6.6.40-cuerdos
                                        Uptime: 4 mins
                                        Packages: 2160 (dpkg)
                                        Display (QEMU Monitor): 1024x768 @ 60 Hz in 15"
                                        DE: sway
                                        WM: Xfwm4 (X11)
                                        CPU: QEMU Virtual version 2.5+ @ 2.49 GHz
                                        GPU: Unknown Device 1111 (VGA compatible)
| Optimizado hasta el último pixel |    Memory: 774.96 MiB / 1.93 GiB (39%)
************************************    Swap: 0 B / 494.22 MiB (0%)
                                        Disk (/): 107.32 MiB / 988.45 MiB (11%) - overlay
                                        Disk (/run/live/medium): 2.91 GiB / 2.91 GiB (100%) - iso]

~
❯
```

What is user mode?

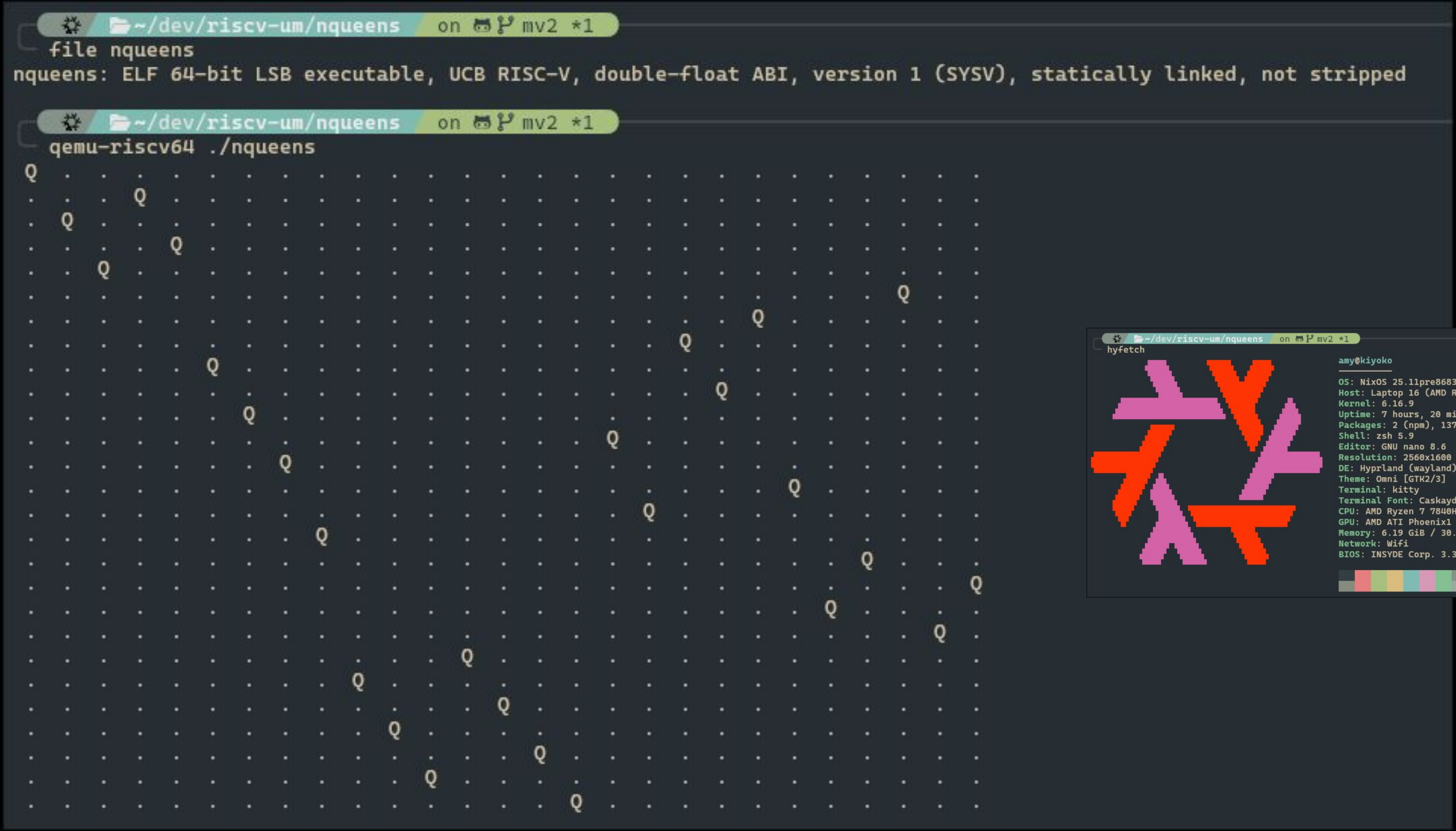
QEMU primarily runs in two modes: system mode (emulating an entire system — essentially a VM) and user mode (just running a process).

Assuming all the libraries needed are present (or the executable is statically linked!), QEMU can run Linux binaries designed for one ISA for any within its realm of support.

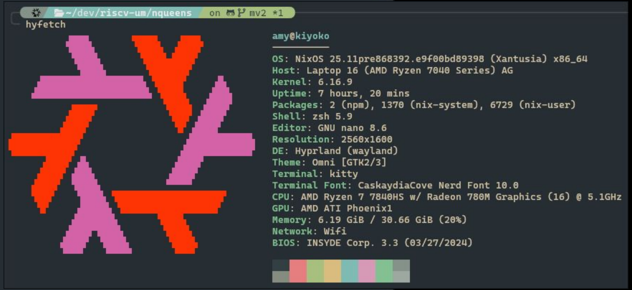
Want to run an Alpha binary that launches riscv32 binaries on a PPC system? You can, just as easy as running x86 binaries on x86!



```
   qemu-arm /mnt/bin/bash
 -- external command --
qemu-aarch64              qemu-m68k                 qemu-riscv32              qemu-system-loongarch64   qemu-system-sh4
qemu-aarch64_be           qemu-microblaze           qemu-riscv64              qemu-system-m68k          qemu-system-sh4eb
qemu-alpha                qemu-microblazeel         qemu-s390x                qemu-system-microblaze    qemu-system-sparc
qemu-arm                  qemu-mips                 qemu-sh4                  qemu-system-microblazeel  qemu-system-sparc64
qemu-armeb                qemu-mips64               qemu-sh4eb                qemu-system-mips          qemu-system-tricore
qemu-bridge-helper        qemu-mips64el             qemu-sparc                qemu-system-mips64        qemu-system-x86_64
qemu-edid                 qemu-mipsel               qemu-sparc32plus          qemu-system-mips64el      qemu-system-xtensa
qemu-ga                   qemu-mipsn32              qemu-sparc64              qemu-system-mipsel        qemu-system-xtensaeb
qemu-hexagon              qemu-mipsn32el            qemu-storage-daemon       qemu-system-or1k          qemu-vmsr-helper
qemu-hppa                 qemu-nbd                  qemu-system-aarch64       qemu-system-ppc           qemu-x86_64
qemu-i386                 qemu-or1k                 qemu-system-alpha         qemu-system-ppc64         qemu-xtensa
qemu-img                  qemu-ppc                  qemu-system-arm           qemu-system-riscv32       qemu-xtensaeb
qemu-io                   qemu-ppc64                qemu-system-avr           qemu-system-riscv64
qemu-kvm                  qemu-ppc64le              qemu-system-hppa          qemu-system-rx
qemu-loongarch64          qemu-pr-helper            qemu-system-i386          qemu-system-s390x
 -- user --
qemu-libvirtd
```



```
~/dev/riscv-um/nqueens  on  mv2 *1
file nqueens
nqueens: ELF 64-bit LSB executable, UCB RISC-V, double-float ABI, version 1 (SYSV), statically linked, not stripped

~/dev/riscv-um/nqueens  on  mv2 *1
qemu-riscv64 ./nqueens
Q . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . Q . . . . . . . . . . . . . . . . . . . . . . .
. Q . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . Q . . . . . . . . . . . . . . . . . . . . . .
. . Q . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . Q . .
. . . . . . . . . . . . . . . . . . . . Q . . . . . .
. . . . . . . . . . . . . . . . . . Q . . . . . . . .
. . . . . Q . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . Q . . . . . . .
. . . . . . Q . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . Q . . . . . . . . . .
. . . . . . . Q . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . Q . . . . .
. . . . . . . . . . . . . . . . . Q . . . . . . . . .
. . . . . . . . Q . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . Q . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . Q
. . . . . . . . . . . . . . . . . . . . . . Q . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . Q .
. . . . . . . . . . . . Q . . . . . . . . . . . . . .
. . . . . . . . . Q . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . Q . . . . . . . . . . . . .
. . . . . . . . . . Q . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . Q . . . . . . . . . . . .
. . . . . . . . . . . Q . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . Q . . . . . . . . . . .
```

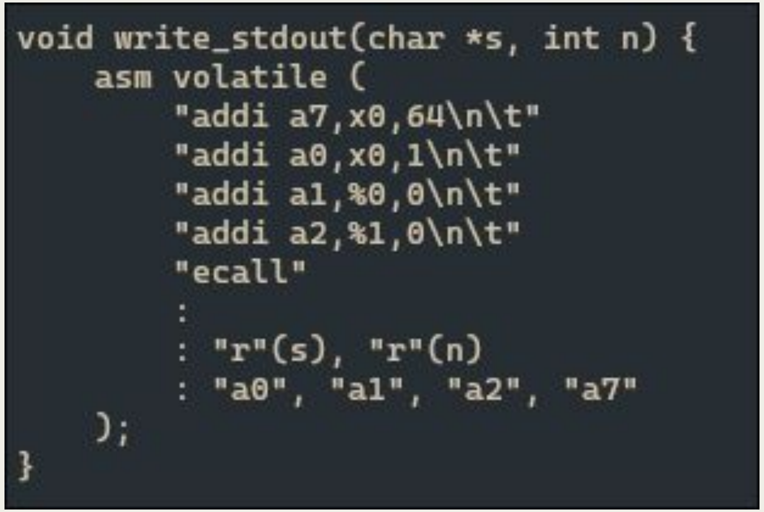


```
~/dev/riscv-um/nqueens  on  mv2 *1
hyfetch
                                        amy@kiyoko
                                        ----------
                                        OS: NixOS 25.11pre868392.e9f00bd89398 (Xantusia) x86_64
                                        Host: Laptop 16 (AMD Ryzen 7040 Series) AG
                                        Kernel: 6.16.9
                                        Uptime: 7 hours, 20 mins
                                        Packages: 2 (npm), 1370 (nix-system), 6729 (nix-user)
                                        Shell: zsh 5.9
                                        Editor: GNU nano 8.6
                                        Resolution: 2560x1600
                                        DE: Hyprland (wayland)
                                        Theme: Omni [GTK2/3]
                                        Terminal: kitty
                                        Terminal Font: CaskaydiaCove Nerd Font 10.0
                                        CPU: AMD Ryzen 7 7840HS w/ Radeon 780M Graphics (16) @ 5.1GHz
                                        GPU: AMD ATI Phoenix1
                                        Memory: 6.19 GiB / 30.66 GiB (20%)
                                        Network: Wifi
                                        BIOS: INSYDE Corp. 3.3 (03/27/2024)
```

User-Mode Ops

QEMU intercepts and translates syscalls, like WINE, except it translates directly to existing syscalls - ex, write translates from 0x40 (aarch64) to 0x1 (amd64).

The real memory state is maintained, but with instruction sections executed directly in host form from a mapped translation cache.



```
void write_stdout(char *s, int n) {
    asm volatile (
        "addi a7,x0,64\n\t"
        "addi a0,x0,1\n\t"
        "addi a1,%0,0\n\t"
        "addi a2,%1,0\n\t"
        "ecall"
        :
        : "r"(s), "r"(n)
        : "a0", "a1", "a2", "a7"
    );
}
```

2

Running cross-arch applications

You’re not supposed to be here!

Basics of Running Cross-Architecture Programs

In general, to run a binary $BIN of some architecture $ARCH, execute:

```
$ qemu-$ARCH $BIN
```

That's it! As long as the binary is a valid Linux binary for that architecture, and all libraries are present, it will run. You can check the architecture and OS target of a binary with the `file` command. Example:

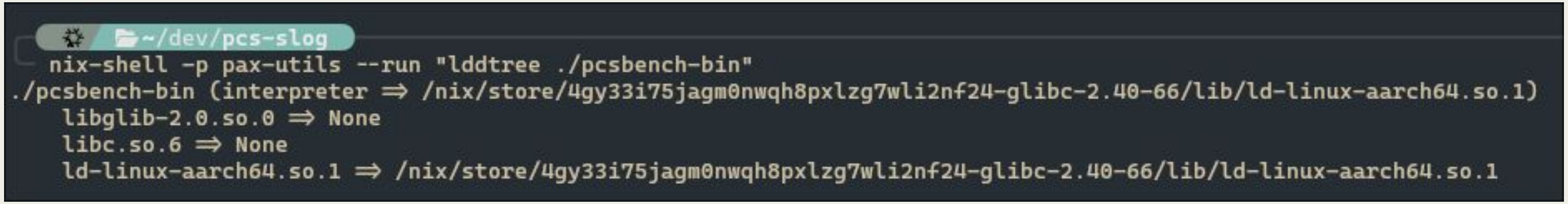
```
$ file ./nqueens
./nqueens: ELF 64-bit LSB executable, UCB RISC-V, double-float ABI, version 1
(SYSV), statically linked, not stripped
```

Additional options are available for some architectures, such as for connecting to GDB. See qemu-$ARCH –help.

Libraries

If a binary relies on shared libraries, you need to have up-to-date versions of those libraries either installed (see multilib) or in the same directory. These don’t need to be QEMU-specific; you can yoink them off almost any Linux system or use vendored versions.

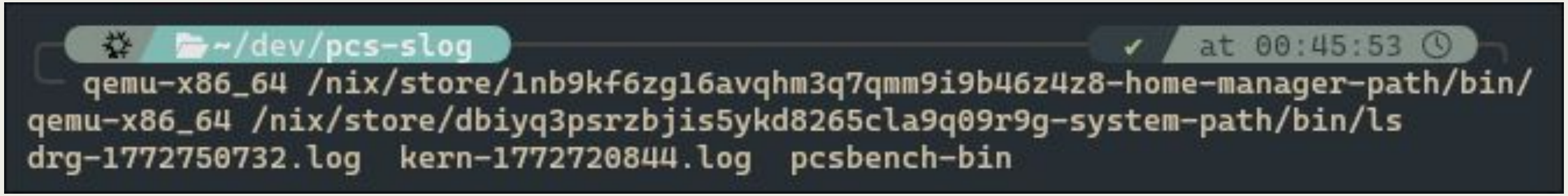
Where possible, use statically linked versions of executables - it saves a lot of trouble! Otherwise, you’ll need to manage lots of libraries — but there are tools to help with that.



```
~/dev/pcs-slog
  nix-shell -p pax-utils --run "lddtree ./pcsbench-bin"
./pcsbench-bin (interpreter ⇒ /nix/store/4gy33i75jagm0nwqh8pxlzg7wli2nf24-glibc-2.40-66/lib/ld-linux-aarch64.so.1)
    libglib-2.0.so.0 ⇒ None
    libc.so.6 ⇒ None
    ld-linux-aarch64.so.1 ⇒ /nix/store/4gy33i75jagm0nwqh8pxlzg7wli2nf24-glibc-2.40-66/lib/ld-linux-aarch64.so.1
```

Practical transparency

As long as the binaries are there, you can do basically anything with QEMU as long as it runs. It runs everything fully transparently, and you can even test it against your normal system libraries by running the same architecture! QEMU will still re-emulate your host architecture in that case.



```
~/dev/pcs-slog                                                    ✔  at 00:45:53
  qemu-x86_64 /nix/store/1nb9kf6zg16avqhm3q7qmm9i9b46z4z8-home-manager-path/bin/
qemu-x86_64 /nix/store/dbiyq3psrzbjis5ykd8265cla9q09r9g-system-path/bin/ls
drg-1772750732.log  kern-1772720844.log  pcsbench-bin
```

3

Binfmt, multilib, and LD_PRELOAD

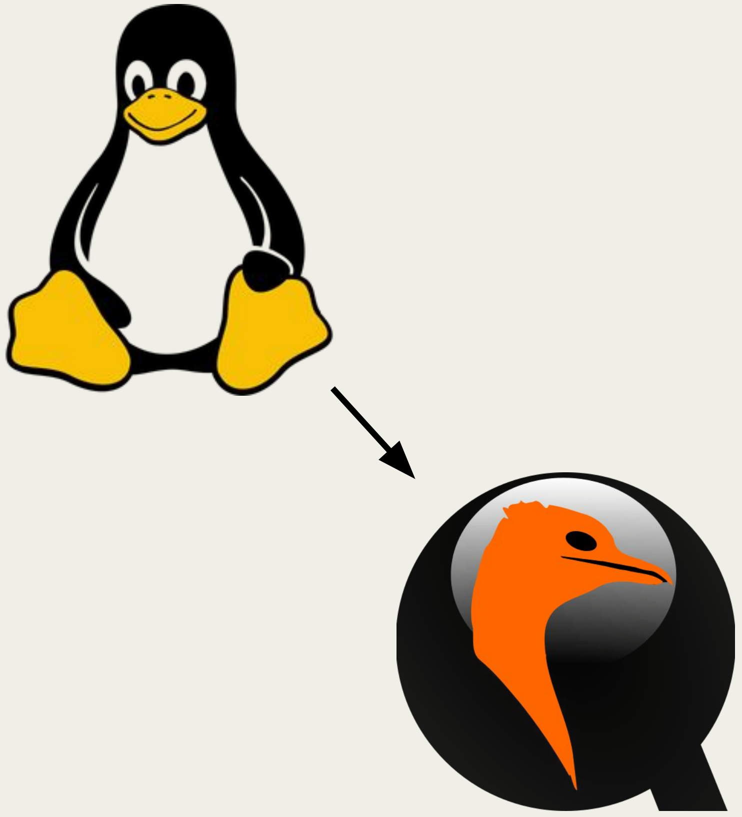
Every program the light touches is executable

Why not *always* run QEMU?

Problem: every single program invocation must be prefixed with qemu-$ARCH /fullpath, which is unreasonable, prohibits running programs in the PATH, and will cause applications that invoke other programs to crash.

Solution: tell the kernel to let QEMU handle unidentified architectures!

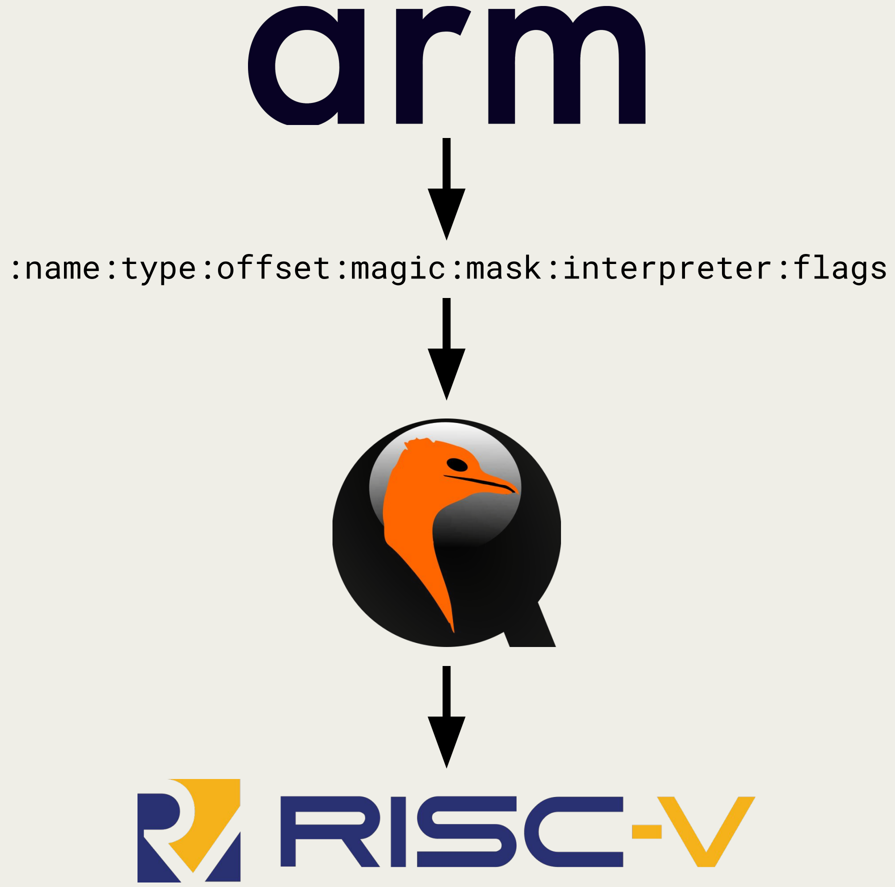
(Also done for .NET, WINE, Java)



binfmt-misc

Linux exposes an interface called binfmt-misc, which allows information in file headers — including Linux ELF binaries — to be used to call an “interpreter”, which will system-wide assume responsibility for all calls to relevant programs.

Upon encountering a non-host binary, Linux will call on the appropriate QEMU instance to handle it - which does so fully in the background, making it appear like **cross-architecture applications natively run on your system!**



```
:name:type:offset:magic:mask:interpreter:flags
```

Multilib Support

Some distributions, like Debian and its derivatives, explicitly have support for *multilib configurations* — not just i386/amd64, but between fundamentally different architectures.

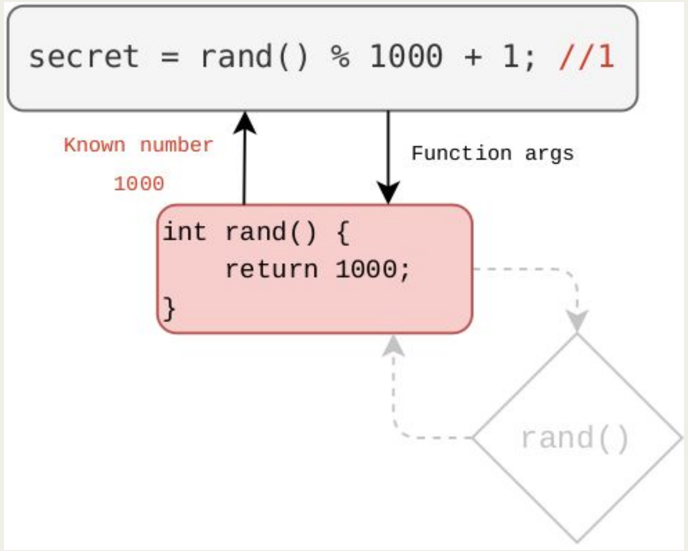
While having a thorough, well-packaged multilib setup is doable on any distribution, Debian has had the longest consistent user experience for doing so, being directly integrated into dpkg. APT commands can pull cross-architecture packages and treat them as native ones.

```
sudo dpkg --add-architecture <architecture>
sudo apt update
sudo apt install libc6:<architecture>

# Install a test package:
sudo apt install hello:<architecture>
# Should print "Hello, world!":
hello
```

LD_PRELOAD/LD_LIBRARY_PATH

Dynamic executables will link exactly the same under QEMU as they do normally — and that means invoking linux-ld, giving the opportunity to link additional libraries with LD_LIBRARY_PATH (if a dependency is unresolved) and overriding libraries with LD_PRELOAD.

The possibilities are endless — testing new library versions, attempting exploits without native hardware/full VMs, or patching closed-source software with up-to-date libs.



```
secret = rand() % 1000 + 1; //1
```

```
int rand() {
    return 1000;
}
```

4

CrossDebootstrap and chroots

Container-style access, VM-level possibilities

Chroots, generally

A **chroot** changes the current process's environmental context, placing the system root (/) somewhere other than the actual current root.

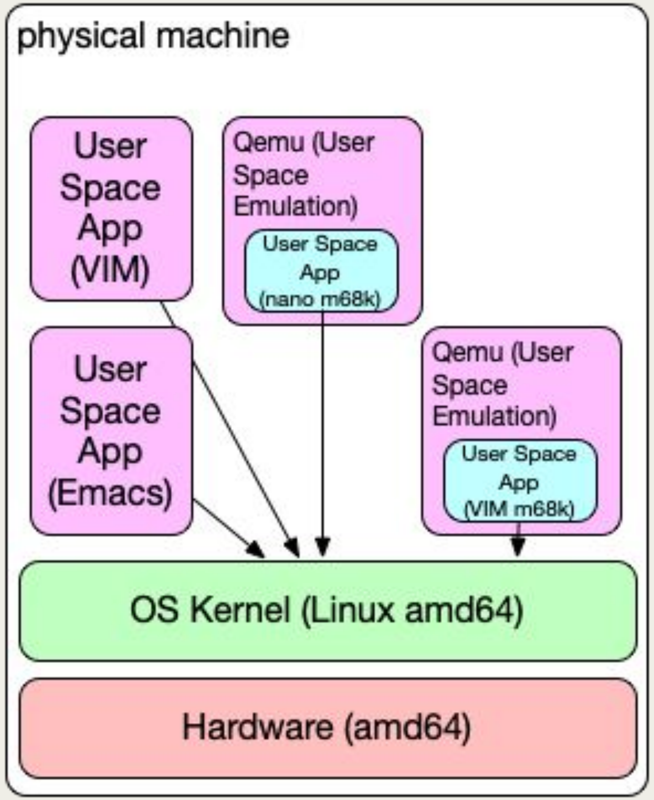
Using a chroot, your currently running kernel and init can act as if they're operating for another system stored on disk. Chroots are an invaluable tool for debugging and restoring systems.

Traditional chroots only work for systems of the same architecture, as the user wouldn't be able to execute any of the tools on the system. However…

QEMU for Chroots

If a system has QEMU user mode set up in binfmt, chrooted programs *work* as if the target system matched the host's architecture!

This gives significant amount of flexibility when debugging and repairing other systems; as with other use cases, QEMU + binfmt essentially *removes architecture-specific execution restrictions* for almost all programs.


physical machine
User Space App (VIM)
Qemu (User Space Emulation)
User Space App (nano m68k)
User Space App (Emacs)
Qemu (User Space Emulation)
User Space App (VIM m68k)
OS Kernel (Linux amd64)
Hardware (amd64)

CrossDebootstrap

Debian uses QEMU user mode with binfmt to do cross-architecture initialization of Debian system environments through Debootstrap.

As long as QEMU user mode and its binfmt wrapper are installed, using Debootstrap for cross-architecture installations just *works* automatically. No additional trickery necessary!

The former Embedded Debian project has been effectively using QEMU for this since ***2006***.

5

Enterprise and Modern Architectures

How to justify QEMU development time as a business expense

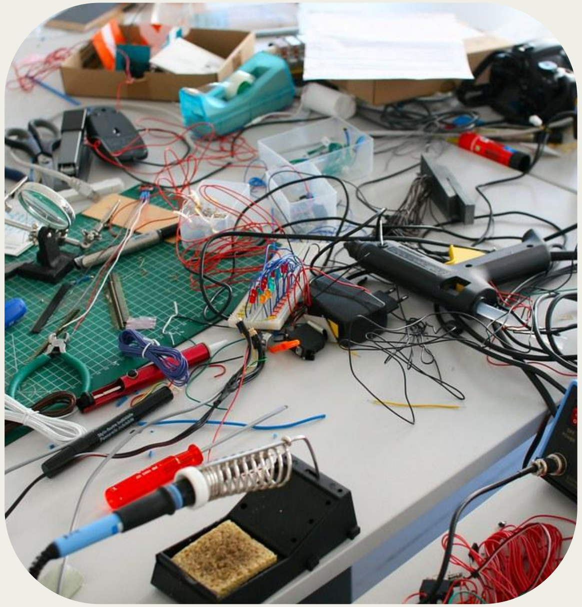
Enterprise Applications

Old Closed-Source Software

While porting is always the better option, it isn't always possible to port old enterprise software, especially if it was closed-source. QEMU can breathe new life into old enterprise software allowing it to run in a native-like environment on modern hardware.

Fast, Inexpensive Prototyping

Prior to shipping production-grade software, testing should always be done on physical hardware or provably identical VMs. However, QEMU user mode offers the ability to do in-development testing without the time-sinks of spinning up VMs or flashing new software to testing hardware.



Modern Architectures

New architectures can be easily implemented into QEMU through *plugins*. Because QEMU's Tiny Code Generator (TCG) runs on a frontend-IR-backend model, only a frontend and backend implementation are necessary to make an architecture compatible with all other architectures.

RISC-V's QEMU port took several years to upstream, in part due to the complexity of the standard and focus on system mode — but simple modes can often have a first working example in a few days.

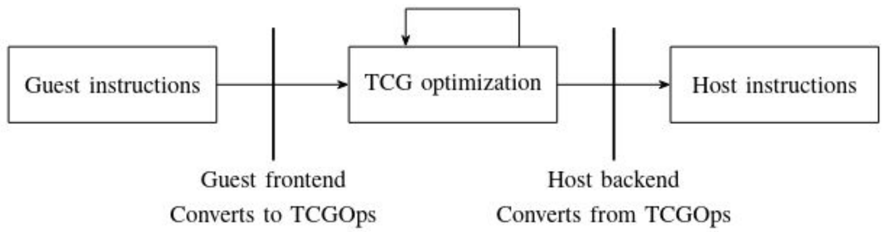


Fig. 3. TCG IR pipeline

Caveat usor

While QEMU is great and a transparent drop-in to run cross-architecture applications, environments where performance is crucial currently suffer due to structural issues in QEMU's pipeline.

However, this may not be a permanent flaw; revisions to QEMU's structure and individual architecture-pair bypasses could significantly increase QEMU's performance.

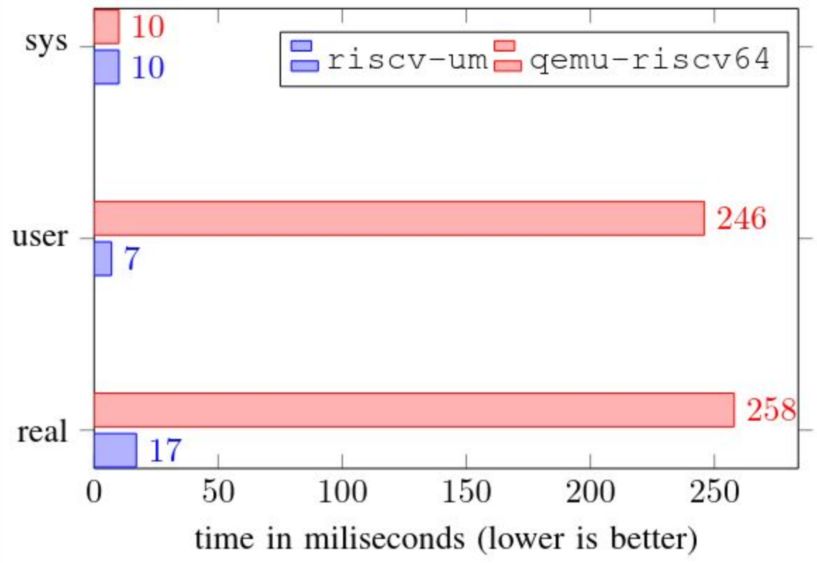


Fig. 6. `riscv-um` vs QEMU results

6

Universal Binaries!

30 > 2 && 2003 < 2020

Prerequisites

QEMU-user-static

QEMU can be compiled entirely statically, even without libc dependence, making it fully portable across Linux systems. Download a static binary, make it executable, and you have QEMU running anywhere!

Executable Wrappers

Plenty of different solutions for wrapping programs exist - as long as one binary can be made to call another self-contained binary, this wrapper requirement is satisfied.

Bundling QEMU Versions

FatELF can already be used to bundle different versions of a binary and should be used when the source code is available and performance is crucial, or distributing

To build a universal binary, structure your application as the following:

- Have either a shell script or FatELF launcher that can detect the current architecture
- Bundle all reasonably foreseeable QEMU builds for your target architecture (qemu-$ARCH)
    - These are a few MB each
- Point the QEMU instances towards the target single-architecture embedded binary

This was done successfully on a system at CSU Fullerton with no root access or ability to install programs.

You can also distribute a bundled/fat QEMU executable and use that as a basis for running any other program.

Thank you!

Are there any questions?

You can also contact me via:
Phone/Signal: +1 (562) 299-8551
Matrix: @amyip.dev:matrix.org
SoCal Mesh: 62ap
Email: amy@amyip.net (keys.openpgp.org)