Having Fun With Emulators

Conflating ARM support, 32 bit macOS and world peace
in one confusing talk

Agenda

- Last Wineconf André and I introduced Project Hangover with the goal of building a version of qemu that can run x86 apps in Wine on ARM
  - This talk provides a status update
- Relatedly, soon there will be no 32 bit macOS support
  - Hangover is one way to handle it
  - This talk explores a few other options

Hangover Status

- https://github.com/andrerh/hangover
- 32 Bit Support!
  - On pure 64 bit host, 64 bit Wine
- Actual applications and games usable
- Performance: Games from late 90’s / early 00’s playable
- Improved build system :-)

DLL status

- Necessary DLLs: 10 / 16
  - ntdll, d3d, ws2_32, …
  - Missing: mmdevapi, opengl, imm32, ...
- Nice-to-have DLLs: 05 / 12
  - dsound, dinput, crypto, ...
  - Missing: dwrite, crypt32, xaudio, windowscodecs
- “Auxiliary” DLLs: 0 / 14
  - gphoto, sane, wpcap, ...

DLL options

- Write an API level wrapper
- Cross compile the Wine DLL for x86, x64
- Cross compile Unix dependencies + wine DLL
  - e.g. libxml2 + libxslt ==> msxml3.dll
  - Freetype ==> dwrite.dll
- Can be decided on a DLL by DLL basis

ARM Performance Issues

- Qemu: Generated code horrible
- Qemu: No hardware floating point support
- Wine: NtCurrentTeb → pthread_getspecific
- Wine: Interlocked* → pthread_mutex_lock
- Emulator entry / exit right now not an issue
  - Will probably change once above issues are fixed

Other TODOs

- Debugger (→ Copy protection)
- Properly announce this on qemu devel list
- Find a good solution for msvcrt
- Exception handling still buggy

Mac 32 bit support

- Apple will stop supporting 32 bit code next year
- No 32 bit libs, no compiler, no 32 bit processes, nada

Option 1

- Use qemu and our thunks
- Works, but slow
- Can we do better?

Idea 2

- The hardware can still run 32 bit code
- Use hardware virtualization support!
- Ken made it work :-)
  - Building on Sergio Gomez Del Real’s GSoC work
- Pro: Running code is FAST
- Con: Jumping in and out is SLOOOOOOW

Where to leave the emulator?

Hardware | OS Kernel | System Libs | Wine | Application

OS Virtualization

Qemu performance :-(
HW Virtualization :-)
Interface size :-)
Host integration :-(

This isn’t what Wine is meant for

Qemu-linux-user

Qemu performance :-(
HW Virtualization ???
Interface size :-)
Host integration :-/

Hangover

Qemu performance :-)
HW Virtualization :-(
Interface size :-(
Host integration :-)

qemu-linux-user on macOS

- Huw investigates this option
- Load ELF binaries inside qemu on macOS, emulate Linux syscalls, but use HW virt
- We pull libc, ntdll, kernel32 into the VM
- Issue 1: winemac.drv, audio, system integration
- Issue 2: GPL code
- Issue 3: Is it fast enough?

qemu-macos-user

- Build a Frankenstein libc that thunks to macOS libc
- No real advantage over previous idea
- Relies on soon unmaintained 32 bit macOS compilers

Pull everything into the emulator

- We can create 32 and 64 bit code segments inside the HW VM
- Call between 32 bit app and 64 bit Wine inside the VM is cheap
- Map 64 bit macOS pages into the VM
- Blindly repeat real syscalls outside the VM
- What could possibly go wrong?

Speed up qemu?

- Code generated by qemu is awful
- Ideally converting x86_32 to x86_64 code is way easier – just put a few operand size prefixes in the right places
- Still has management overhead
- We haven’t investigated this idea yet

Carefully engineer wrapper libs

- E.g. d3d could use the command stream as its “syscall” layer
- How many Win32 calls lead to a Linux syscall?
  - File IO, Network, memory alloc, wineserver calls, sync primitives, x11 calls, opengl, sound, ...
- Having to aggressively optimize here makes the entire task even more difficult
- No guarantee it will be enough

The actual solution?

- We don’t know yet
- There may be none
- Hangover should be fast enough for installers
- It won’t run today’s games, many of which are still 32 bit only
- Likely macOS and ARM solutions will diverge
  - Until macOS switches to ARM some day...