Snapcrafting on a tablet
A little adventure

Alfred Neumayer, 2023

Agenda

- Attempts on the iPad
- Attempts on Ubuntu Touch
- Device integration for Ubuntu Touch
- Improving Snap integration

What to expect

- C++ tooling
- Snapcrafting
- WebAssembly
- Tablets!
- iPadOS
- Ubuntu Touch
- Multi-platform app development
- System integration

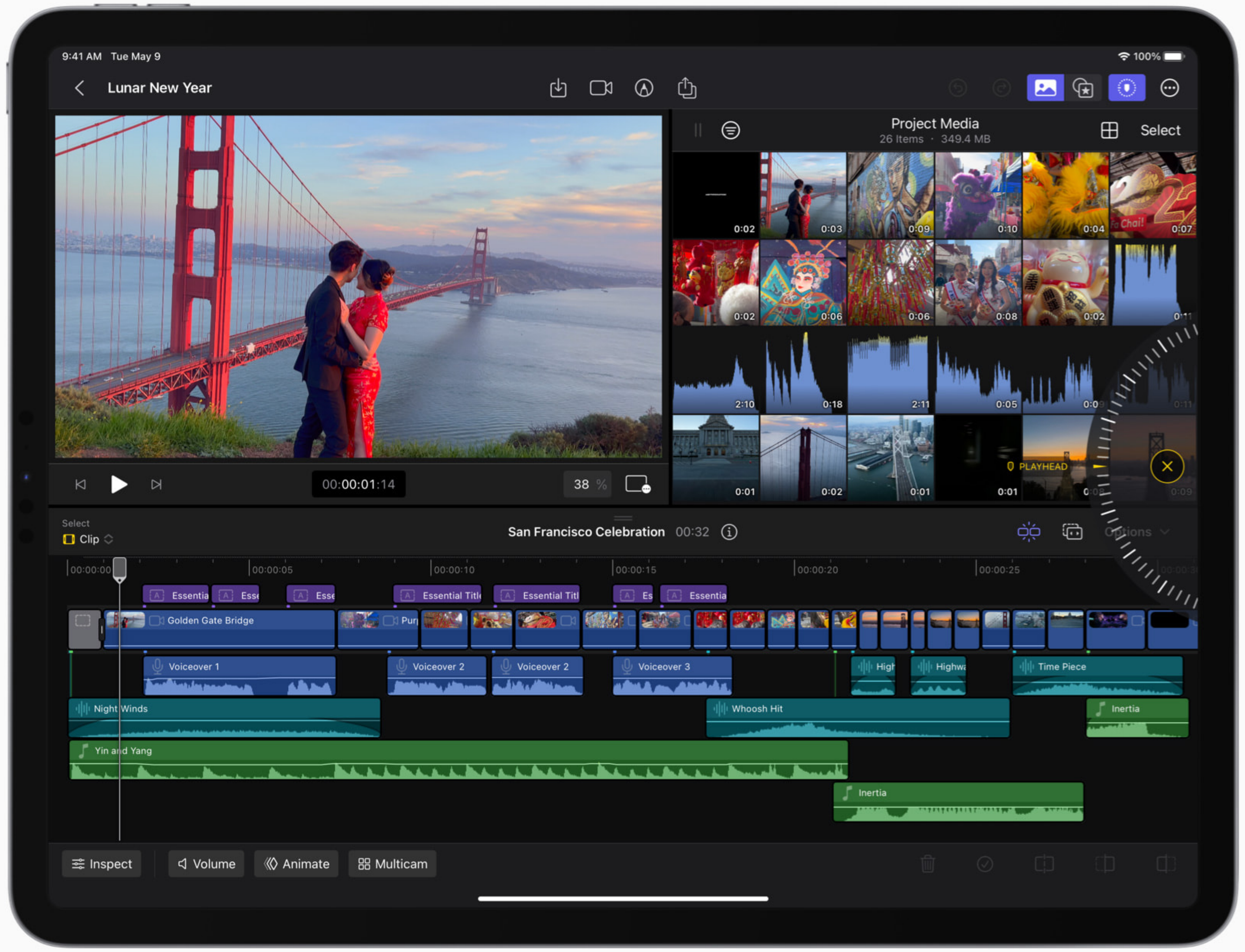
What to gain

- Comparison between two different beasts
  - Where are we versus the popular offerings?
  - Technical differences
  - Differences in policies
- Building apps for an ecosystem
  - Apple
  - Ubuntu & the broader FLOSS world
- "Wins and losses"
  - Learning from failures

Why a tablet?

- "I have my beefy machine at home!"
- "They’re clunky to use."
- "Software is limited."
- "Capabilities intentionally held back."

Whenever I hold an iPad I wonder how to develop **with** it, not **for** it.


Lunar New Year
Project Media
26 Items · 349.4 MB
Select
00:00:01:14
San Francisco Celebration 00:32
Golden Gate Bridge
Voiceover 1
Voiceover 2
Voiceover 3
Night Winds
Whoosh Hit
Yin and Yang
Inertia
Time Piece
Inspect
Volume
Animate
Multicam

An Evening Song
Soul Fire
Drums
Synth Lead
Strings
Pads
Alchemy
Channel EQ
ChromaVerb
Compressor
Decay Time
Wet
Threshold
Attack
Sustain
Play
Scale
C2
C3
C4

Can we have professional development tooling on a tablet, too?

Why an iPad?

- Raw horsepower
    - Same M2 chip in the iPad Pro & MacBook Air
- Opinion: Great touch-first experience for ordinary users
- Accessories making it work, for work
- Developer tools in the App Store
    - Git app
    - Documentation viewer
    - (mostly online) IDEs

Why Ubuntu Touch?

- Goals similar to the iPad
  - Cover average consumer-grade needs
- Low-to-mid tier selection of hardware
- Terminal + Desktop Mode → powers unlocked
  - Regular git
  - Development tool galore
  - VSCodium
  - Maybe my own IDE too?

So, why a tablet?

- They arguably look cool!
- Different way of thinking → galaxy brain powers
- Light to carry for on-the-go tasks
- A "fresh start" for Personal Computing devices
- New ways of chaining apps together for completing tasks

The rules

- ARM64 tablet
- Do everything offline
    - Development
    - Compilation/Snapcrafting
    - Debugging
- Focus on creating CLI apps first

The app to achieve this

- Tide IDE
- Multi-platform development environment
- WebAssembly
- Snapcrafting, Click Packaging

The Snaps to craft

- git-confined
  - All-in-one Git flavor for confined environments
  - Full featureset (https, ssh, man)
  - Around 40MB .snap file
- Clickable
  - Ubuntu Touch packaging and development tool
  - "Building on the tablet for the tablet"
  - Requires Docker

The boundaries

- iPadOS has limits
    - No app-allocated executable memory
    - No system(), fork() or exec()
    - Otherwise mostly legitimate sandboxing techniques

The boundaries

- iPadOS has limits
  - Executable memory needs all code to be signed
  - Manual review of apps by Apple employees
  - Cannot "fire and forget" a release

The boundaries

- Ubuntu Touch
- 2 models of supplying apps
  - Confined
  - Unconfined
- Confined apps can be released immediately
- Unconfined apps need to be FLOSS & reviewed

The boundaries

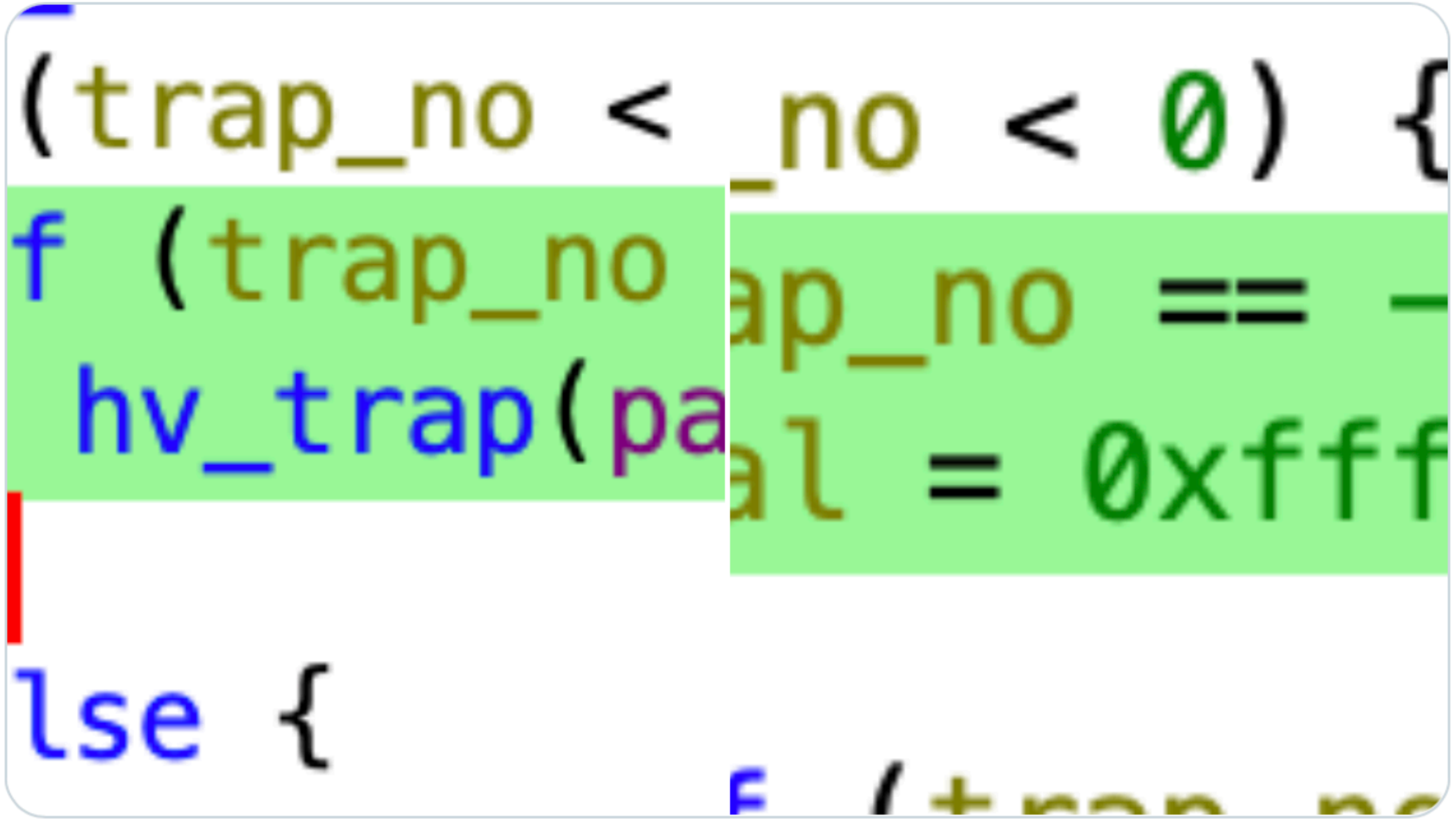
- Performance matters
  - No executable memory means no on-device JIT, no on-device AOT
- Licenses matter
  - GPL cannot enter the iPadOS App Store
    - The exceptions: LGPLv2.1, or other linking exceptions
  - Disputable
  - Good luck with that

More rules

- "Let virtualization technologies handle that"
  - UTM
  - Pocket VMs

UTM
@UTMapp

BAD NEWS: Apple removed Hypervisor support from XNU in iOS 16.4. Here is a diff of iOS 16.3.1 and iOS 16.4. What this means is that even if a jailbreak/TrollStore comes out for iOS 16.6.1/17.0, there will not be UTM virtualization support, even on M1/M2 iPads.



```
(trap_no <
f (trap_no
 hv_trap(pa

lse {
```

```
_no < 0) {
ap_no == -
al = 0xfff

f (trap_no
```

8:09 PM · Oct 2, 2023 · **96.2K** Views

TAKESHI'S CASTLE HD

WebAssembly

- wasm32 is a target a compiler can generate code against
- Clang supports that
- Existing Clang forks within the App Store
    - Build a Linux environment on your iPad
    - Put Snapcraft into that
    - ?
    - Profit

WebAssembly

- container2wasm
  - Docker pre-packed into a virtual filesystem
  - Preconfigured TinyEMU
  - RISCV or X86_64 emulation, no virtualization
  - Snapcraft inside of that
- Would allow for theoretical functionality, with lower performance
- Takes around 14 minutes for "snapcraft version" to return.
- Reduced down to around 1.5 minutes
  - .pyc creation
  - Reduction of shipped files

Quick end

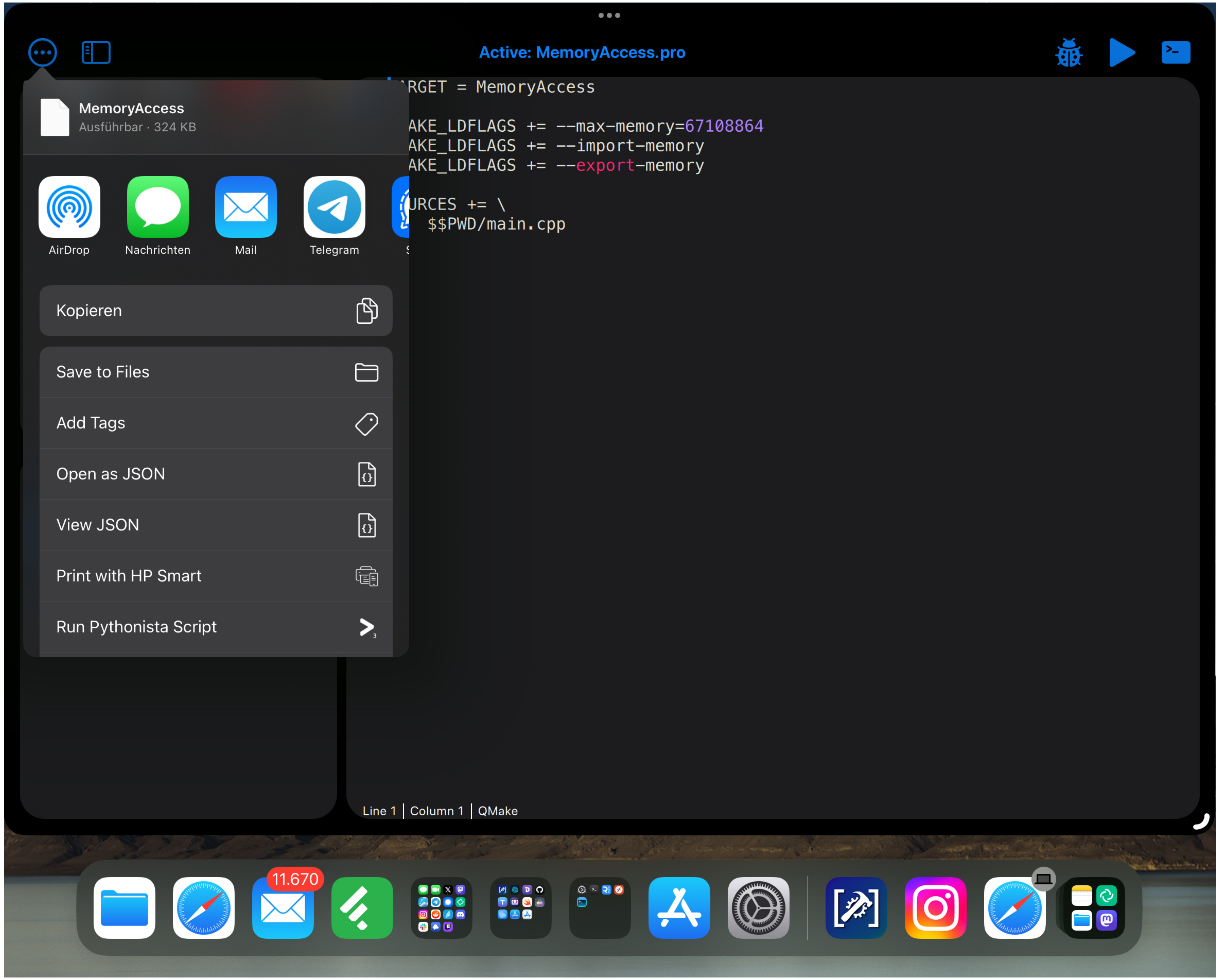
- The iPad cannot do it yet
  - Limitations don’t do the hardware justice
- Enriching the respective ecosystems
  - iPad can build for itself and iPhone using Swift
  - Ubuntu Desktop can build for it and Ubuntu Touch
  - How about Ubuntu Desktop & Touch building interchangeably?

Improving the layers underneath the app

- Effin forget it on Apple platforms
- Ubuntu Touch
  - snapd integration is halfway there
  - Sometimes needs Kernel changes
    - Android vendor kernels
    - AppArmor
  - Further explorations (binfmt, FUSE?)
  - UBports Porting
    - Responsible individuals get to shape the platform

A dance of apps

- Both contenders offer a solution
- iPadOS
  - Share Sheet for passing single files or abstract content
  - Applications can share whole directories with each other
- Ubuntu Touch
  - Content-Hub for passing files or abstract content
  - No sandboxed way of allowing FUSE in the OpenStore

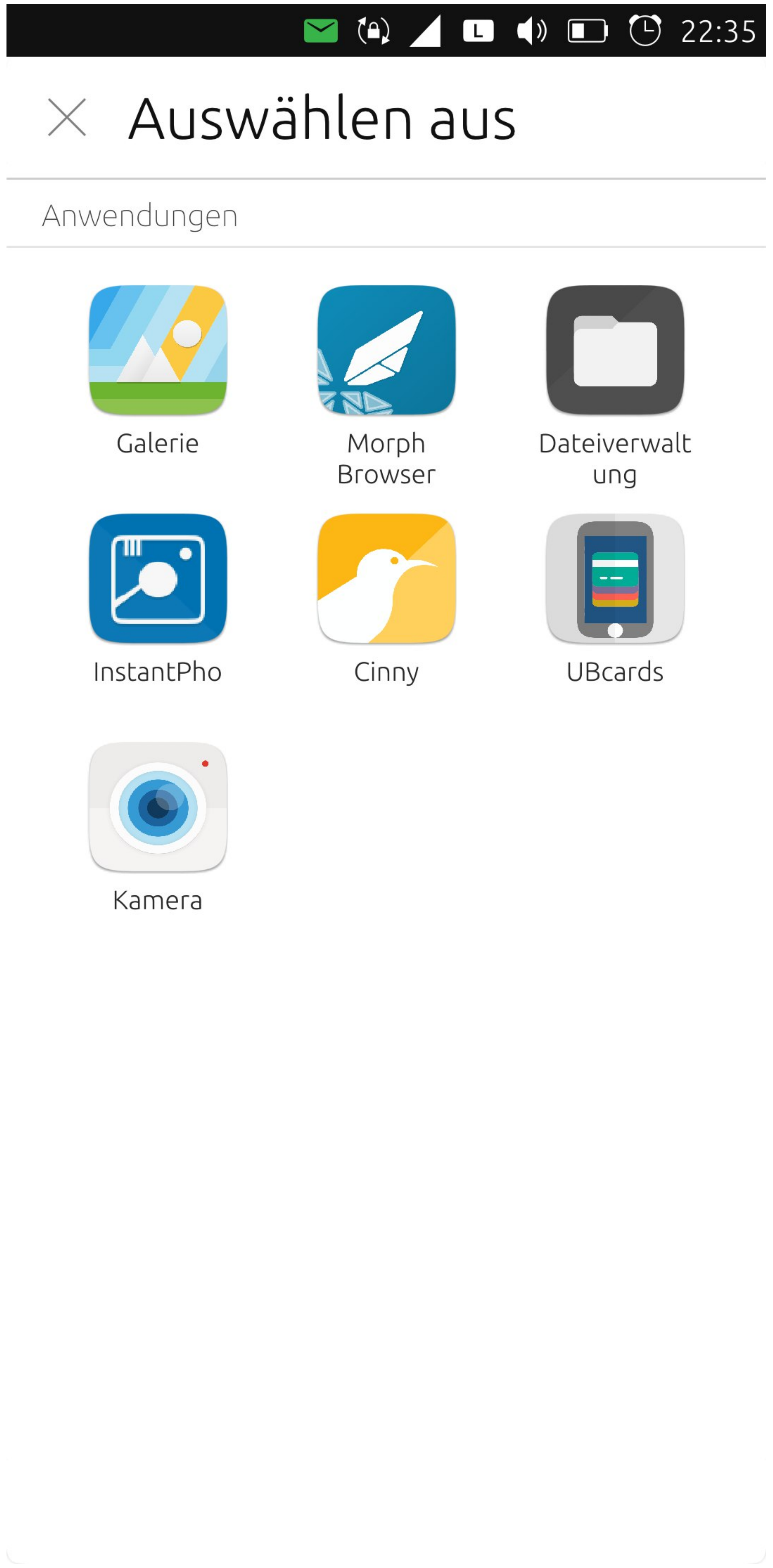


```
RGET = MemoryAccess

AKE_LDFLAGS += --max-memory=67108864
AKE_LDFLAGS += --import-memory
AKE_LDFLAGS += --export-memory

URCES += \
  $$PWD/main.cpp
```

Line 1 | Column 1 | QMake



The results!

- iPad
  - Incredible hardware performance
  - Held back by its own software’s capabilities
- Ubuntu Touch
  - Could do it!
  - Needs proper hardware to showcase

How to Snapcraft

- Install snapd on Ubuntu Touch
  - Might need to unmount file overlays along the way
- LXD with privileged container
  - Unprivileged needs better integration
    - Android vendor kernel variations
    - More partially-writable filesystem changes
- Run "snapcraft --destructive-mode" inside LXD
- Throw away later
- Snapcrafting with native performance

System- and Device Integration

- 2 approaches
  - As-Mainline-as-possible kernel
    - Pine devices
    - Otherwise few off-the-shelf devices with adequate performance
  - Halium device integration
    - Android vendor kernels
    - Mini-Android services inside LXC
    - Android libraries in a GNU userland

System- and Device Integration

- Each distribution has their own delivery mechanism
  - Droidian, LuneOS & more
- Ubuntu Touch
  - A generic Halium systemimage built on UBports CI
  - Port maintainer adapts the kernel
  - On UBports GitLab: halium-generic-adaptation-build-tools
  - Easy hardware integration fixups using runtime file overlays
  - Separate from the also generic Ubuntu Touch rootfs
  - Ship it to users using system-image

System- and Device Integration

- AppArmor on Ubuntu Touch
  - 2 approaches:
    - Remove-And-Replace security/apparmor with closest Ubuntu Kernel
    - Take the patches from AppArmor’s GitLab
  - I would welcome coordination.
  - Binder integration one day?

Improving Snap integration

- snapd integration planning and shaping
- Installable "Docker on Ubuntu Touch wen?"
- Many CLI tools work well
- Additional permissions for GNU+Android hybrid userland
- libhybris-based graphics driver support
  - OpenGL ES (!), Vulkan (?)
- Content-Hub
- Other Ubuntu Touch frameworks & services

Thank you!

Resources

- https://ubports.com/
- https://gitlab.com/ubports
- https://halium.org/
- https://lomiri.com/
- https://github.com/fredldotme/Tide