

Fun With Alternative Mobile (GNU/)Linux Operating Systems

Heng Ye

Massachusetts Institute of Technology
Student Information Processing Board

January 15, 2026



The problem

- Cell phones are everywhere
- Basically as powerful as computers
- Can they do everything a computer can, technically?
- Can they do everything a computer can, practically?
- What can they do that computers can't?



The power of cell phones

The same RAM and storage of a mid-range office laptop...

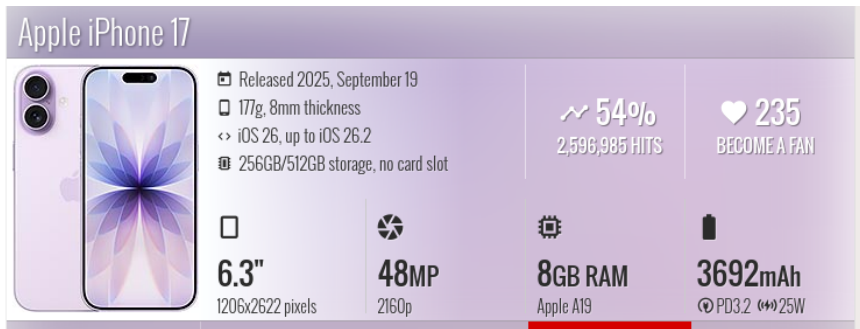


Figure: current mobile phone specs: GSMArena



Networking and sensor capabilities

- What networking and sensors do **laptops** have?
- What networking and sensors do **cell phones** have?
- Answer: a lot more!
 - Modem w/ telephony
 - Multiple cameras
 - Accelerometer
 - Very accurate location
 - Proximity sensor
 - Compass
- Do we have a computer, but better?



The privacy perspective

- We have a bunch of sensors that laptops don't have, yay!
- How much data can those sensors collect (with or without permission) ?

3144

IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 20, NO. 4, JULY/AUGUST 2023

Device-Independent Smartphone Eavesdropping Jointly Using Accelerometer and Gyroscope

Figure: Don't even need the microphone

I would like some more control over the situation. Remember, the OS has access to all that...



The software side

- How many choices of OS do you think we have on mobile?
- Basically a duopoly of Android (Google) and iOS (Apple)
- Android is OSS at its core, but Google is essentially an ad company
- Different manufacturers ship their own Android ROMs
- You need Apple's permission to install apps on iOS
- Both companies trying to do AI, opinion on this is left to you



Trivial solution

- Let's try this to fix some of the concerns
- Well, Android is FOSS, let's fork it...
- Fix security and privacy issues, `rm -rf '*google*',...`
- Give the user some more control about rooting/bootloader unlocking...
- Compile and have people install!
- Well, not so trivial but usually gives you a working system



FOSS ROMs

- This is the approach taken by GrapheneOS, LineageOS, CopperheadOS, etc
- Provides sufficient security (GrapheneOS is **very good** in this regard)
- You can run Android apps (well, most of them)
- You get updates on old devices (especially w/ LineageOS)
- So why do we still need to think about mobile Linux?



Let's read some newspapers

- Still dependent on Google Android architecture and base. Let's read some headlines.

Google moves all Android development behind closed doors

👤 Thom Holwerda 📅 2025-03-26 📱 Android 💬 14 Comments

Up until now, Google developed several components of Android out in the open, as part of AOSP, while developing everything else behind closed doors, only releasing the source code once the final new Android version was released. This meant that Google had to merge the two branches, which lead to problems and issues, so Google decided it's now moving *all* development of Android behind closed doors.

Figure: OSnews. kind of scary, no?



More recent news

Rumour: Google intends to discontinue the Android Open Source Project

👤 Thom Holwerda 📅 2025-06-12 📱 Android 💬 26 Comments

With the release of Android 16, [Google changed how it developed Android](#). Development is now taking place behind closed doors, with the code dropped after the corresponding version has been released to Pixel devices. Well, it turns out this wasn't the only thing Google has changed about Android development. As the developers of CalyxOS, a popular de-Googled Android ROM, dove into the Android 16 AOSP source code, they realised something very important was missing: the device-specific source code for modern Pixel devices.

Figure: OSnews just a few months later



News from just a few days ago

Breaking: Google will now only release Android source code twice a year

Google has announced that it will publish Android source code to AOSP in Q2 and Q4 of each year.

Figure: Android Authority Jan 2026



Goals of Mobile Linux

- **Goal:** shake off Big Tech dependence for using mobile hardware
- Microsoft code is not strictly necessary to operate a laptop
- Other perks: running desktop Linux software (Blender, KiCad, whatever)
- hmm...what were the connections on modern phones?...did I see USB-C display out?...we'll come back to this



But what is "Mobile Linux"?

We have already come up with some constraints, but let's come up with a complete definition.

- Not reliant on closed source/Big Tech software
- GNU-like/glibc userspace (basically, able to run Linux programs)
- Linux kernel
- Follows desktop Linux conventions (package manager, init system, sudo, etc)
- Tries to get the phone parts to work like a phone (somewhat optional)



Why is this important now?

If we don't do these projects:

- Phones software increasingly becomes closed
- User control of phones decreases, and they become more of a "thin-client"
- Importantly, no alternative OS ecosystem, so nothing to resort to if Google/Apple do something you don't agree with



Android is Linux?

- Concession: Android has a Linux kernel.
- Userspace is completely different and not compatible. Bionic vs glibc, etc.
- Different underlying libraries for basic tasks: desktop Linux software cannot run
- Structured in a way that is much harder to dig deep



Why is Mobile Linux nontrivial?

So if Android userspace works with the Linux kernel, why can't we swap out the userspace?

- Some vendors don't even allow unlocking bootloader
- Each vendor uses a different custom kernel
- Each vendor uses different firmware that is not intercompatible (camera, modem, etc)
- Much of the firmware, unlike in conventional Linux setups, is intertwined with userspace parts.
- Kernel options assume Android userspace



Compromises

- Much of the firmware is **closed source**.
- If the firmware doesn't work with our userspace, the phone components will not work.
- Surprise! They do not work with our userspace (in most cases, caveat later)
- Any ideas on how to solve this?



Libhybris and Halium

Let's run a very small Android container using LXC (similar to Docker) to deal with this.

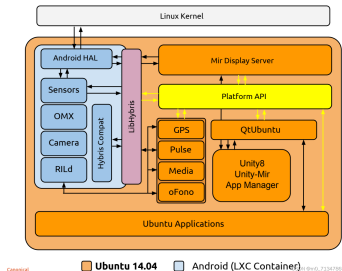


Figure: Yes, I know the caption says Ubuntu 14.04. This has existed for some time.

*note: Halium is libhybris + some scripts to manage the Android parts + some other glue. 

Now let's write the userspace

- Now we just slap desktop Linux onto the userspace, as libhybris took care of the low level stuff, right?
- Well, if you just want a server with ssh access, sure
- But we want actual phone stuff on our phone.
- This is hard.*

* Note: everything is hard.



Mobile Linux distributions

These distributions can use libhybris/Halium for the hardware abstractions:

- SailfishOS
- Ubuntu Touch
- Droidian



Ubuntu Touch History

Ubuntu Touch has a long and turbulent history.

- Started development within Canonical in 2011
- First released in 2013
- Abandoned in 2017 by Canonical
- Taken over by community (UBports Foundation)
- Continues today: great successes last year with VoLTE and new devices!



Ubuntu Touch user interface

- UI/UX is a hard* part of any OSS project
- Ubuntu Touch has a UI that is somewhat similar to desktop Linux but is nicely gesture/swipe oriented.
- This is the **Lomiri desktop environment**

* As mentioned prior, everything is hard.



Lomiri screenshots

If I did not run out of time to flash my phone, demo available at front after presentation. If not, look at these screenshots :)

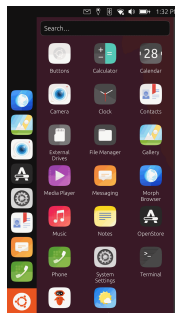


Figure: Ubuntu Touch app drawer



More Lomiri screenshots



Great Expectations

- The concept of **convergence** was pioneered by the early visions of Ubuntu Touch.
- Convergence meant plugging in the phone to a monitor and keyboard and using it as desktop
- Currently, some smartphones have this (Samsung DeX) but UT would allow for running desktop apps
- It works on some devices now, but **very experimental still** (after > 10 years)



Convergence working on the Nexus5 numerous years ago

Pictured is not a laptop: it is a portable monitor+keyboard+touchpad.



Figure: from <https://ko.ubunlog.com/>



Convergence working on the Fxtec Pro1 (3 years ago)

Notice smartphone screen acting as touchpad.



LINUX PC IN
YOUR POCKET



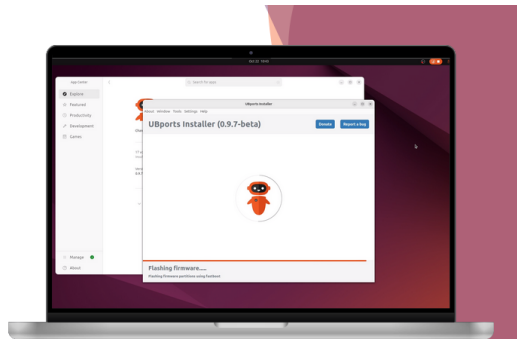
Some drawbacks

- Desktop apps can't run directly due to compositor—rather, "Libertine" is used
- Updates of the base (Ubuntu) are difficult. Only recently were we able to get it up to 20.04.



Installer

UBports developed this a few years ago purely in the community. This they did **100% right!**



The UBports Installer.

A simple tool to Ubuntu Touch on your device with just a few clicks.



Install Ubuntu Touch

Contribute to the Installer



Droidian: experimental Debian on Halium

- Droidian is a rather pure version of Debian
- Can run desktop apps directly, use Debian repos (yay!)
- Runs **phosh** – UI similar to GNOME, uses GTK, etc



SailfishOS

- Development led by Jolla in Finland (a small company)
- Jolla Phone started sales in 2013 running first version of SailfishOS
- Pioneered gesture-based navigation before Android/iOS (in 2013, iPhone was version 5!)
- Based on openSUSE
- Probably the most reliable for daily use



Superior gesture-based UI

- Jolla was founded by disgruntled Nokia employees when Nokia switched to Android
- Before the switch, Nokia used a Debian-like OS for their phones
- The professionalism of the UI can be easily seen compared to contemporary FOSS projects.
- Demo phone up at front, see after lecture.



Licensing model

The company earns money by selling their Android app compatibility layer (Alien Dalvik).

Android Apps for Linux Platform

Sailfish OS supports Android™ apps

While the primary focus of Sailfish OS is to enable the development of native Linux applications, it has been deemed necessary to support also existing key apps (such as WhatsApp, WeChat, Slack and Office). The Android Apps for Linux Platforms software enables applications natively developed for Android OS to run on non-Android-based mobile devices, such as Sailfish OS.

The screenshot for Android Apps for Linux Platforms



Common challenges

We will now discuss major challenges faced by these distributions.

- App ecosystem
- Device porting
- Funding and community efforts



App ecosystem

- Naturally, there is a smaller app ecosystem.
- Community is responsible for writing **all** the apps
- For Lomiri/silica (UT and SFOS respectively), customized Qt framework is used
- Many mainstream apps are not available



Android App compatibility

- Similar to WINE on desktop Linux
- Waydroid: Android container that can run on conventional Linux.
- You can try Waydroid on desktop Linux too!
- Compatibility is not 100% there
- Also the proprietary option from Jolla, which is very well integrated into SFOS and works very well



Porting difficulties

- The first step in booting another OS is booting.
- Many vendors have bootloader locks. This obliterates any possibility of running Linux, excepts if someone finds an exploit.
- Any carrier locked phone will not work either



The porting process

- Did it seem like we generalized a solution above with hybris to run Linux userspace on Android phones?
- This is only in the ideal case.
- Each phone has so many quirks (standardization does not exist) so a developer needs to persuade all the parts to work together
- This takes weeks to months for a cooperative device!
- Even after porting, some features may be deemed too hard to fix
- All volunteer effort



Perils of community projects

I **love** FOSS, but...

- projects (apps, ports) get abandoned if the dev is tired of them
- things progress slowly but surely. Older devices often have better support for this reason
- not everything is "polished" – I love this part, but daily driving might not be best
- **Be the change you want to see in these projects!**



Case in point: VoLTE

- Modern telephony over data connections requires lots of software
- In their infinite wisdom, this software is a bit higher-level than the firmware
- Very difficult to reimplement (is in userspace)
- SailfishOS and UT have recently succeeded in getting this to work, but only on 3-5 devices each.



Android devices

- As discussed, through **porting**, Android-running phones can be made to run a Halium-based OS.
- Examples include Sony Xperia 10 series, Samsung Galaxy Note Tab, etc.
- These are the easiest to manufacture for Linux phone companies, because they are essentially mainstream devices
- However, they are not running **mainline Linux kernel**. They use a heavily customized **vendor kernel and firmware**.



Community Android devices

- Some Linux phone companies advertise "Linux phones" than run Halium
- Essentially, they sell an Android phone that has nice Halium compatibility
- ...and of course SailfishOS/UT/etc Linux is preinstalled



Volla Phone

Volla Phone - series of German phones running Ubuntu Touch! Some sold in physical stores in Germany already, yay! The first device that runs Ubuntu Touch that supports VoLTE.



FuriLabs

FuriLabs FLX1: runs something similar to Droidian



Fxtec Pro1(X)

Runs LineageOS, SailfishOS, Ubuntu Touch, Droidian (all the previous ones can too, but this one shipped with a choice). Unfortunately discontinued. Demo at front of room.



Planet Computers series

Runs "Gemian," which is similar to Droidian. Unfortunately discontinued.



Jolla Phone C2

Retails with SailfishOS!

The Linux Phone that actually works

Jolla C2 is a true Linux smartphone that works!
Thanks to its 4G and VoLTE support, it is ready to be
your trusted phone for everyday use.



New Jolla Phone

On Preorder, design still to be set (with community feedback!)



Can I daily drive a Linux phone?

Halium/libhybris OSes are somewhat daily drivable!

- Most phone features will work
- Basic and nerdy apps exist (email, Matrix, Signal, code editor, terminal emulator, etc)
- Webapps must work for the rest (banking, etc)



Compatibility matrix

When selecting a phone to play with, check on the distribution website first, or ask a maintainer!

Fairphone 5 Ubuntu Touch compatibility matrix:

Actors:	✓ 24+ hours battery lifetime	✓ Hotspot
✓ Manual brightness	✓ 7+ days stability	✓ NFC
✓ Torchlight		✓ WiFi
✓ Vibration	CPU:	
	✓ Boot into UI	Sensors:
Camera:	✓ Hardware video playback	✓ Automatic brightness
✓ Flashlight		✓ Fingerprint reader
✓ Photo	Misc:	✓  GPS
✓ Video	✓ AppArmor patches	✓ Proximity
✓ Switching between cameras	✓ Battery percentage	✓ Rotation
	✓ Offline charging	✓ Touchscreen
Cellular:	✓ Online charging	✓ Double touch to wake
✓ Carrier info, signal strength	✓ Recovery image	
✓ Data connection	✓ Reset to factory defaults	Sound:
✓  <u>Incoming, outgoing calls</u>	✓ SD card storage	✓ Earphones
✓ MMS in, out	✓ RTC time	✓ Loudspeaker
✓ PIN unlock	✓ Shutdown / Reboot	✓ Microphone
✓ SMS in, out	✓ Wireless External monitor	✓ Volume control
✓ Change audio routings	✓ Waydroid	
✓ Voice in calls	Network:	USB:
✓ Volume control in calls	✓ Bluetooth	✓ MTP access
Endurance:	✓ Flight mode	✓ ADB access
		✓ Wired External monitor



How to get started

- ① Find a supported phone model
- ② Buy it (usually on the used market for a lower cost, porting takes time and when porting is done, usually the phone is outdated by modern "trends")
- ③ Install OS of your choice using tools provided by distribution
- ④ Have fun!



More-or-less mainline devices

- Let's pivot to **mainline** devices running even more "pure Linux"
- No Haliu/Android glue, just Linux ARM!
- Most Android phones won't work due to proprietary firmware
- So we made our own hardware!
- Much less stable than Haliu for now, but more experimental fun!
- Note: sometimes even with our own stuff it doesn't get merged into the Linux kernel as fast as we would like. So, it's "more-or-less mainline."



Why mainline?

Halium works well enough to daily drive, why do this?

- Mainline Linux is closer to desktop: faster patches and packages
- Less moving parts: Halium has dependence on Android firmware
- More pure: everything that can run on ARM Linux desktop can run on mainline phones. No exceptions.
- More auditable: Minimal vendor blobs and closed-source components



The earliest ones

OpenMoko Neo Freerunner (2007) – ran Linux, almost completely FOSS software! Didn't do so well in the market due to many bugs, but it was a great first effort.



OpenMoko headline recently

Linux 6.17 To Finish Clearing Out Old Code For OpenMoko Devices

Written by [Michael Larabel](#) in [Hardware](#) on 3 July 2025 at 06:52 AM EDT. [15 Comments](#)



Linux 6.17 is expected to clear out some final remnants of the OpenMoko Neo 1973 and Neo FreeRunner smartphone support from that Linux smartphone effort from two decades ago.

Back in 2023 all of the Samsung s3c24xx SoC support was removed from the upstream Linux kernel. This effectively ended the mainline Linux kernel support for the OpenMoko Neo 1973 and Neo FreeRunner devices in the interesting pre-Android era.

But it turns out some other OpenMoko device code has still been lingering in the kernel albeit rather useless with the Samsung SoC support for those OpenMoko devices having been removed. In particular, the input driver for the OpenMoko devices.



Nokia

Nokia had devices that ran their distribution of Debian, which ran bare-metal without Android layer. An argument can be made that this is the ancestor of SailfishOS.

The most iconic is the N900 (released 2009), which can still run postmarketOS (more on that later)



PinePhone

Released 2020 by PINE64 in a business model in which PINE64 did the hardware and the community did the software. Jumpstarted mainline mobile software development on Linux.



PinePhone modem

PinePhone modem (EG25) can run open-source software.

- Modem is essentially a separate computer
- In all other phones, this is closed source
- Reverse engineers have gotten open-source modem software to work on the PinePhone modem, making it the only phone to have open-source modem!



PinePhone Pro

After the commercial success of the PinePhone, the PinePhone Pro was released with better specifications. However, communication was lackluster and firmware issues caused it to be much less of a success. It has been discontinued.

No image because the chassis is basically exactly the same as the PinePhone



Librem 5

Purism Librem 5 is similar to PinePhone in that it ran a (mostly) mainline kernel and conventional Linux software. However, it was much more expensive, causing hobbyists to adopt the PinePhone more, despite the marginally higher performance.



Linux distros for mainline Linux

These are much closer to Linux on ARM distributions, often the difference is just tweaks for the screen size and specific mobile devices.

- Mobian (Debian)
- postmarketOS
- Manjaro
- Ubuntu Touch and SailfishOS (note, they are very dependent on the Halium glue, so sometimes don't work as well as the normal Linux distros)
- ARCH BTW



User interfaces

Just like desktop Linux, the desktop environment can be swapped out!

- Phosh (GNOME-like)
- KDE Plasma Mobile
- Vanilla GNOME (experimental and resource-hogging)
- SXMO (lightweight, based on sway/i3)
- You can just use the framebuffer if you want to



Status of Mainline

- Slow but steady improvement: each piece of the phone may need to be implemented from scratch
- **not** daily drivable yet
- Path for future development of the ideal Linux phone!



What works on mainline now?

- On purpose-built Linux phones (older desktop-like chipsets and tech): most modem/LTE stuff works, and stuff can run for the most part
- However, performance becomes a real issue. You can experience it for yourself with the PinePhone at the front
- Porting originally Android phones to mainline takes much longer than porting to Halium
- On these devices...we're working on it.



Example of compatibility on OnePlus 6

This is one of the ported devices that runs mainline decently well.

Multimedia	
3D Acceleration	Works
Audio ▾	Works
Camera	Partial
Camera Flash	Works
Connectivity	
WiFi	Partial
Bluetooth	Works
GPS	Partial
NFC ▾	Partial
Modem	
Calls	Partial
SMS	Partial
Mobile data	Partial



The State of Things

We see that we still need Halium if we want to use Mobile Linux daily...for now.



Getting involved

As you see, we need **a lot of help**.

- We need programmers to work on apps
- We need firmware people to figure out firmware
- We need people to write documentation
- Most of all...we need testers! Anyone can be a tester!



It's easy!

Some basic steps to get involved...

- ① Pick your poison
- ② Lurk around on the Forums and Matrix channels for a bit
- ③ Jump in when you find something that can be helped!



Porting

One of the easiest ways to get involved is porting. You don't have to do ideation!

- 1 Find old phone in drawers
- 2 See if you can unlock bootloader and find kernel sources
- 3 If so, you can port!

Prerequisites: Patience, Linux command line skills, ability to chat with people online and ask for help



Testing

If you found a phone (or obtained one) that already works, install a distro on it! Give feedback or write documentation, you don't need programming skill at all!



Thank you!

Thanks...and much more fun not covered! :)

Devices are at front of room for playing with. As you now know, stuff may not work perfectly, but that's part of the charm.

E-mail: hng@mit.edu

Course website: www.mit.edu/~hng/cluedump-iap-2026

