東京 2025

LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025

A Linux VM on Android via AVF

Jeongik Cha <jeongik@google.com>

東京 2025 Linux Plumbers Conference
TOKYO, JAPAN / DEC. 11-13, 2025

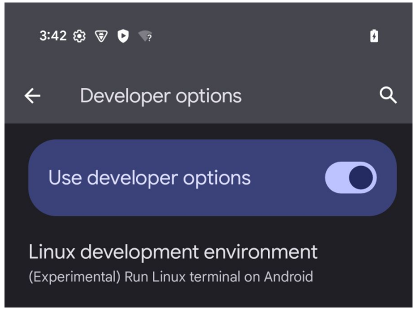
Background & Motivation

- AVF(Android Virtualization Framework)'s Standard Use Case
  - AVF is designed primarily for Protected VMs (pKVM).
  - Focuses on isolation for sensitive workloads (e.g., Biometrics, KeyMint, DRM).
  - VM payload and resource usage are typically minimal and strictly controlled.
- The Experiment: A Linux VM on Android
  - Objective: Evaluate AVF for General-Purpose Computing with a standard OS.
  - Key Evaluation Points:
    - Feasibility: Can AVF host a full Linux distribution on Android?
    - Performance: Is the pKVM/crosvm stack efficient enough to run heavy desktop-level applications (e.g., IDE, Office, Games)?

Background & Motivation

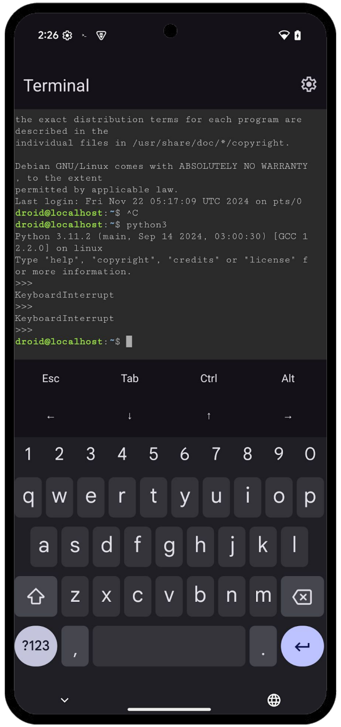
- Why Virtualization? (vs Userspace container-based solutions)
  - Incompatibility:
    - Android uses Bionic Libc, preventing standard glibc-based Linux binaries from running natively.
  - Performance Penalty:
    - Workarounds like PRoot rely on ptrace for syscall translation, causing significant overhead.
  - Shared Kernel Limitations
  - Decision: A full Virtual Machine is required to provide an unmodified, standard Linux environment with its kernel.

Terminal app



Linux development environment

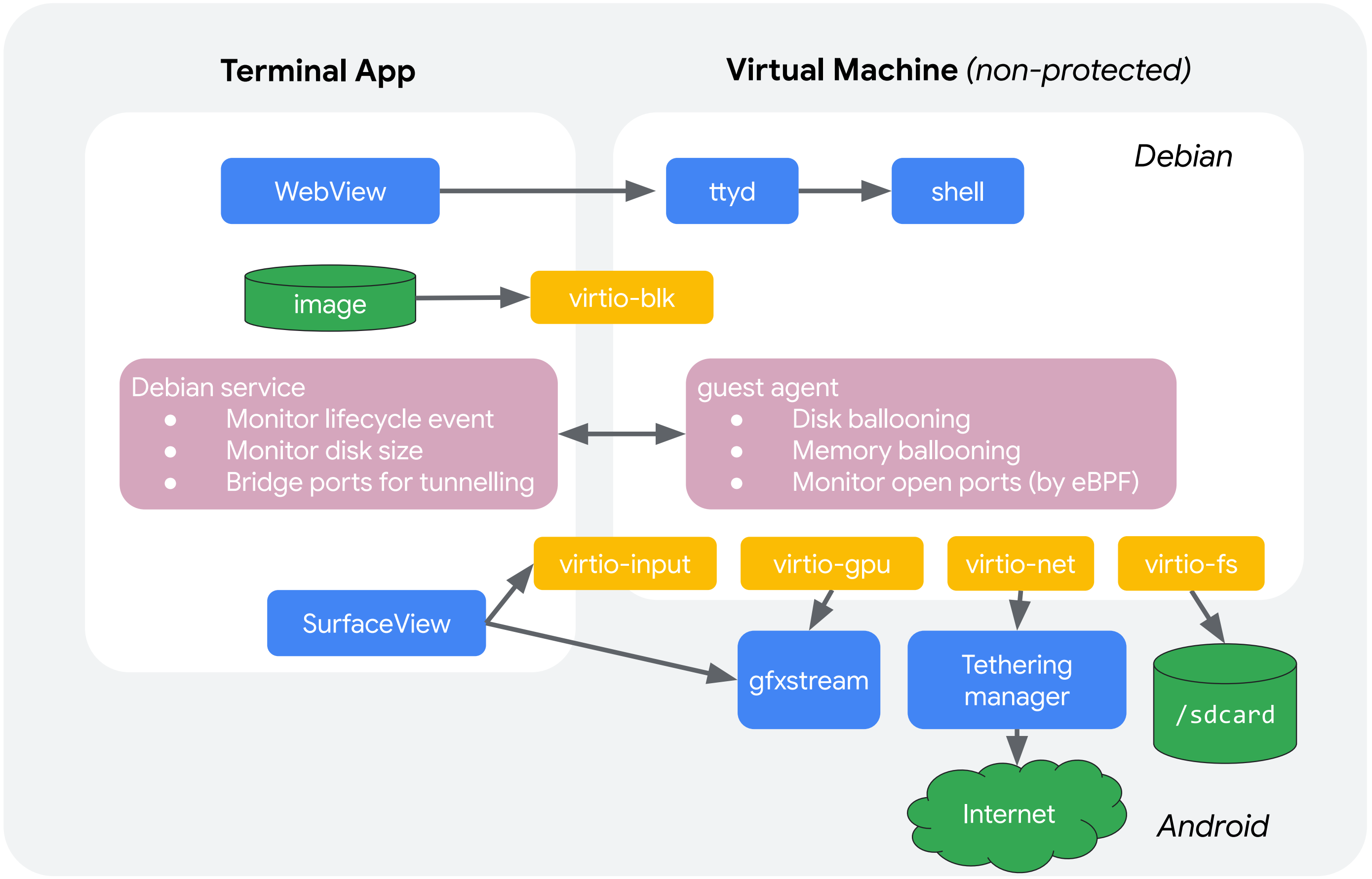
(Experimental) Run Linux terminal on Android



```
the exact distribution terms for each program are
described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY
, to the extent
permitted by applicable law.
Last login: Fri Nov 22 05:17:09 UTC 2024 on pts/0
droid@localhost:~$ ^C
droid@localhost:~$ python3
Python 3.11.2 (main, Sep 14 2024, 03:00:30) [GCC 1
2.2.0] on linux
Type "help", "copyright", "credits" or "license" f
or more information.
>>>
KeyboardInterrupt
>>>
KeyboardInterrupt
>>>
droid@localhost:~$
```

東京 2025 LINUX PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025


Terminal App
Virtual Machine (non-protected)
Debian
WebView
ttyd
shell
image
virtio-blk
Debian service
Monitor lifecycle event
Monitor disk size
Bridge ports for tunnelling
guest agent
Disk ballooning
Memory ballooning
Monitor open ports (by eBPF)
virtio-input
virtio-gpu
virtio-net
virtio-fs
SurfaceView
gfxstream
Tethering manager
/sdcard
Internet
Android

Architecture

- Secure Web-based Terminal Interface
    - Strategy: "Don't Reinvent the Wheel." Used xterm.js (WebView) & ttyd (Guest) instead of native UI.
    - Security: Enforced Mutual TLS (mTLS). Host injects one-time client certs; ttyd rejects unauthorized connections.
- Lifecycle-Aware Resource Management
    - Memory Ballooning: Integrated with Android App Lifecycle. Proactively reclaims Guest memory when the Host app is backgrounded, preventing LMK (Low Memory Killer).
    - Storage Ballooning: Dynamically adjusts virtual disk limits based on the Host's available storage space.
- Seamless Integration Features
    - File Sharing: virtio-fs maps Android’s /sdcard directly into the Guest.
    - Auto-Tunneling (eBPF): In-guest eBPF probe detects bind() syscalls and triggers the Host to auto-create Vsock tunnels (e.g., for localhost:8080).

Build the image

- Challenge: Custom guest agents are required, but building rootfs from scratch (FAI) was too slow on especially aarch64(requires QEMU).
- Solution: Hybrid Provisioning.
  - Base: Upstream Debian Cloud Images + cloud-init.
  - Optimization: Pre-install heavy dependencies via chroot at build time to ensure instant first boot.

Bridging Input

- The Strategy: Translation
  - Rejected Approach: Passthrough (/dev/input/eventX).
  - Blockers: Android permissions & Focus Trapping (User cannot escape the VM or switch apps).
  - Chosen Path: Event Translation. Capturing events at the Android View layer allows handling Android Lifecycle & Focus correctly.
- Keyboard
  - Sources:
    - Soft IME: Intercepts InputConnection commits.
    - Physical Keyboard: Captures View.onKey() events.
  - Translation: Maps Android Keycodes to Linux Scan Codes.
  - The Meta-Key Problem: Standard Android Soft IMEs lack Ctrl, Alt, Tab, Esc.
  - Solution: Implemented a Custom Toolbar UI to inject these modifier keys explicitly.

Bridging Input

- Mouse:
  - API: Uses requestPointerCapture() to grab raw mouse event.
  - Disables Android system cursor; Guest renders its own cursor.
- Touch (Screen):
  - Coordinate Scaling: Maps Android View coordinates to Guest Absolute coordinates
  - Added Multi-touch support in crosvm.
  - Enables pinch-to-zoom & multi-finger gestures

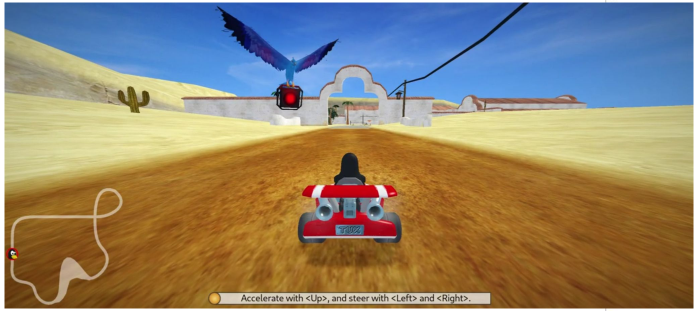
Graphics Stack

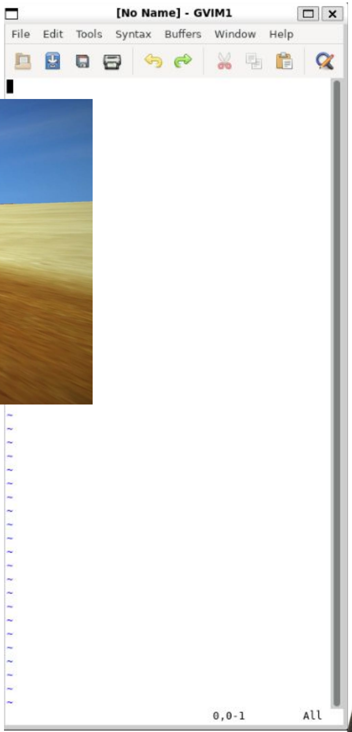
- Windowing Strategy: Weston + Kiosk Shell
  - Constraint: Desktop window managers (GNOME/KDE) are unsuitable for small mobile screens.
  - Decision: Adopt Weston with Kiosk Shell.
    - Forces applications to launch in full-screen mode.
    - Removes window decorations (Title bars) to maximize screen real estate.
  - Legacy Support: Integrated Xwayland for X11 application compatibility.
- Backend Evolution: Enabling Acceleration
  - Initial: Software Rendering (lavapipe + Zink).
    - Verified functionality but suffered from high CPU usage and low frame rates.
  - Current: Hardware Acceleration via gfxstream.
    - Porting Task: gfxstream lacked an Android Host backend.
    - Implementation: Developed a new backend using AHB (Android Hardware Buffer) to map Guest GPU commands/buffer to Android Host memory.

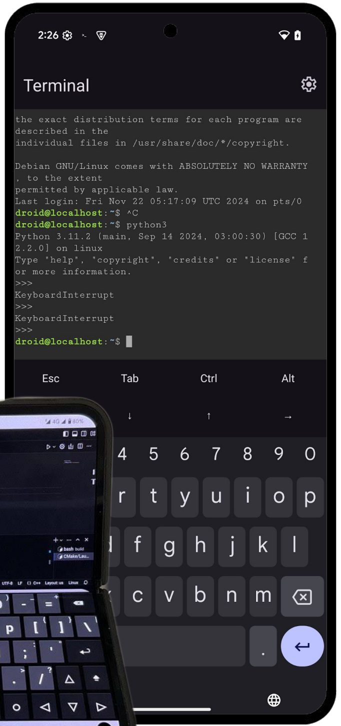
Graphics Stack

- Resolving ARM Cache Coherency
  - Issue: Visual artifacts and texture corruption observed on Android devices.
  - Cause: ARM's weak cache coherency support between host and guest.
  - Fix: Added explicit Cache Flush in gfxstream at necessary points(vkQueueSubmit).
- Zero-Copy Presentation via Surface Transactions
  - Bottleneck: Unnecessary frame copy(during composition in Android and gfxstream) which cause increasing latency.
  - Optimization
    - Remove unnecessary frame copy inside gfxstream during flushing
    - Leveraged Android SurfaceControl Transaction API.
  - Mechanism:
    - Guest renders directly to the mapped AHB and ensure there is no extra copy before Android-side composition.
    - Host calls setBuffer(ahb) to swap buffer handles instead of copying pixels.
  - Result: Achieved a Zero-Copy path from Guest rendering to the physical display.

Demo

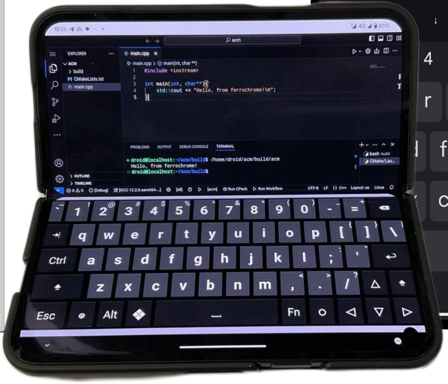






```
the exact distribution terms for each program are
described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY
, to the extent
permitted by applicable law.
Last login: Fri Nov 22 05:17:09 UTC 2024 on pts/0
droid@localhost:~$ ^C
droid@localhost:~$ python3
Python 3.11.2 (main, Sep 14 2024, 03:00:30) [GCC 1
2.2.0] on linux
Type "help", "copyright", "credits" or "license" f
or more information.
>>>
KeyboardInterrupt
>>>
KeyboardInterrupt
>>>
droid@localhost:~$
```


acm
main.cpp
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash build
CMake/Lau...
[Debug] [GCC 12.2.0 aarch64-...] [all] [acm] Run CPack Run Workflow UTF-8 LF C++ Layout: us Linux

```
#include <iostream>

int main(int, char**){
    std::cout << "Hello, from ferrochrome!\n";
}
```

```
droid@localhost:~/acm/build$ /home/droid/acm/build/acm
Hello, from ferrochrome!
droid@localhost:~/acm/build$
```

東京 2025 LINUX PLUMBERS CONFERENCE
TOKYO, JAPAN / DEC. 11-13, 2025

Conclusion

- Summary: Bridging the Gap
  - Scope: Addressed the architectural differences between Mobile(Android) and Desktop(Linux)
  - Implementation Highlights:
    - Input: Bridged Android Touch events to Linux Multi-touch protocol.
    - Graphics: Connected Guest rendering to Android Display via AHB & Zero-Copy.
    - Resources: Memory/Disk ballooning with Android's status(app lifecycle, storage status), storage sharing via virtio-fs
  - Outcome: Confirmed AVF capabilities for General-Purpose Computing.
- Future Work
  - More Optimization: Slim down VM image, optimize boot time
  - GKI adoption
  - More seamless update via cloud-init
  - More Integration: Wayland compositor on Android
  - Any suggested use case / workload via this VM?

Q&A

Jeongik Cha <jeongik@google.com>

東京 2025

LINUX PLUMBERS CONFERENCE

TOKYO, JAPAN / DECEMBER 11-13, 2025