Waydroid solution research

Waydroid description

Waydroid uses a **container-based** approach to boot a **full Android system** on a **regular GNU/Linux** system like Ubuntu.

waydroid offical link: https://waydro.id/

waydroid refer to AnboxLineageOSAndroid-x86spurvLXC

What is Waydroid

Waydroid is a GPL licensed project to provide Android container solution for embedded and desktop linux environment.

Requirement & status

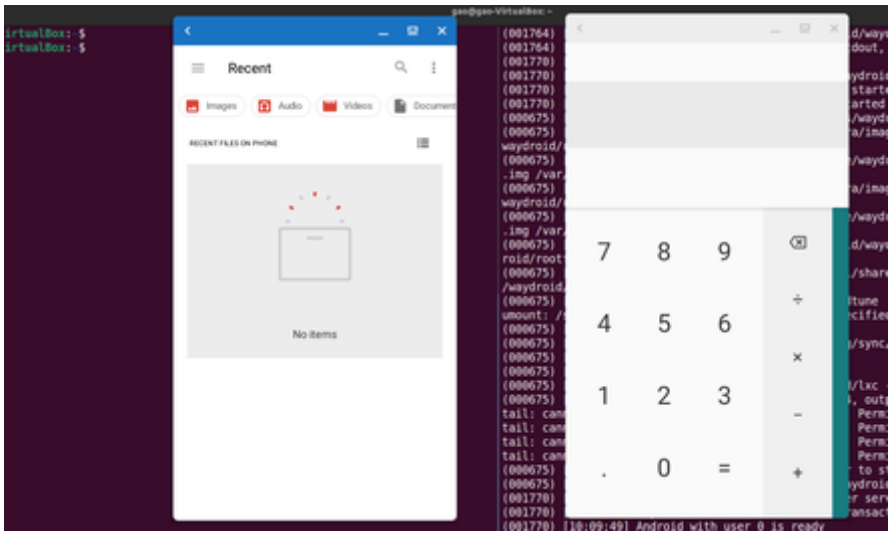
Kernel	lxc support
Linux	lxc 4.0.6 wayland 1.18.0 wayland + gnome support
Android	Lineage OS (Based on android 10)
Develop Period	2021/9 - now (still active, under developing)
Target system	Ubuntu phone (Ubuntu touch) Ubuntu desktop

Bugs	Act Not good in x86 Ubuntu system (ex. Android applications may exit accidentally. sometimes blurred screen. multi-window mode and single window mode cannot fully support. Many Android applications cannot support because of x86 architecture) App Compatibility Universal:✓ ARM32:X ARM64:X

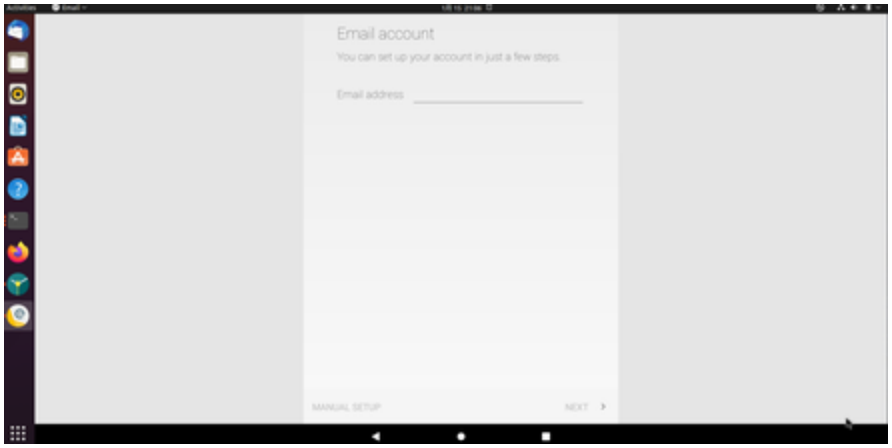
Features

Auto Sync Android applications to Ubuntu.

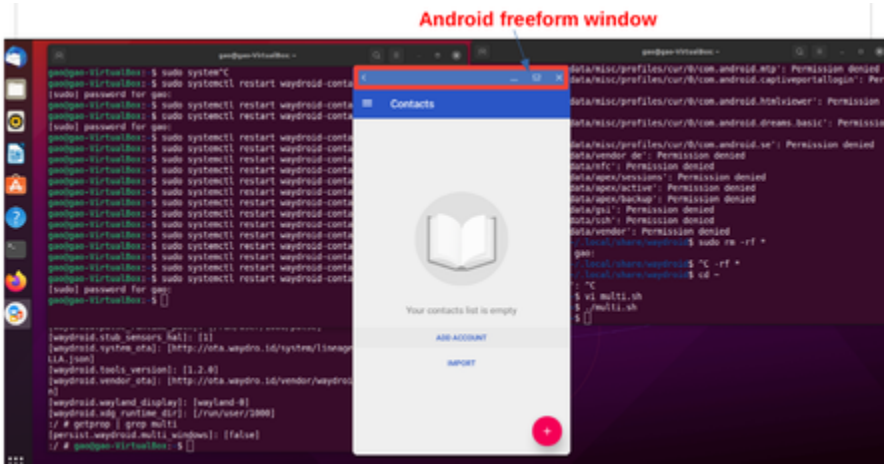
Multi window mode



Single window mode



max/min window using android Freeform mode



Command line LXC container manager

waydroid command

```
action:
  {status,log,init,upgrade,session,container,app,prop,show-full-ui,shell,logcat}
    status              quick check for the waydroid
    log                 follow the waydroid logfile
    init                set up waydroid specific configs and install images
    upgrade             upgrade images
    session             session controller
    container           container controller
    app                 applications controller
    prop                android properties controller
    show-full-ui        show android full screen in window
    shell               run remote shell command
    logcat              show android logcat
```

Where Waydroid come from

Ubuntu has released ubuntu touch phone https://ubuntu-touch.io/.

Waydroid project is targeted to support android applications in ubuntu phone.

How Waydroid work

Linux Host

Waydroid daemon	Manage LXC OTA Log control Install/Uninstall Android apk Set android Prop
python-gbinder	Create/get Android service through binder. Synchronize Android status to Waydroid daemon Clipboard sharing App info sharing

Android

Android Guest	Launched by Waydroid daemon
Display	Shared display buffer to Ubuntu in HWC by wayland protocol

Waydroid technology analysis

LXC control

Waydroid uses python subprocess module to control the LXC.

Waydroid command	**LXC command**
status	lxc-info
start/stop	lxc-start/lxc-stop
freeze/unfreeze	lxc-freeze/lxc-unfreeze
shell	lxc-attach --
logcat	lxc-attach – /system/bin/logcat

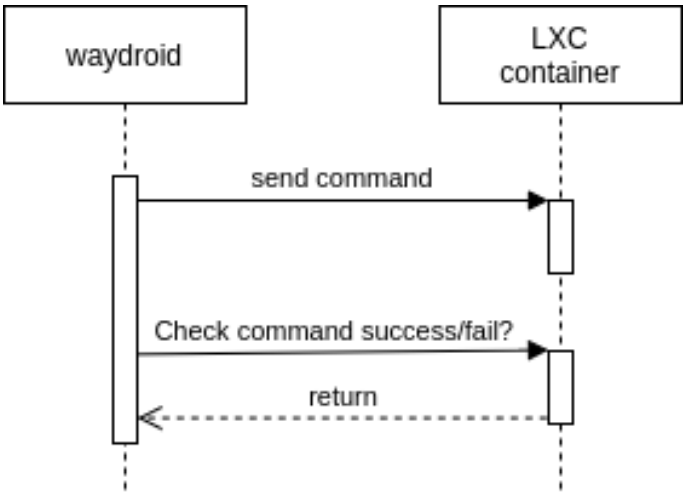
Waydroid divided the concepts of container and session.

There’s only one container in the system.

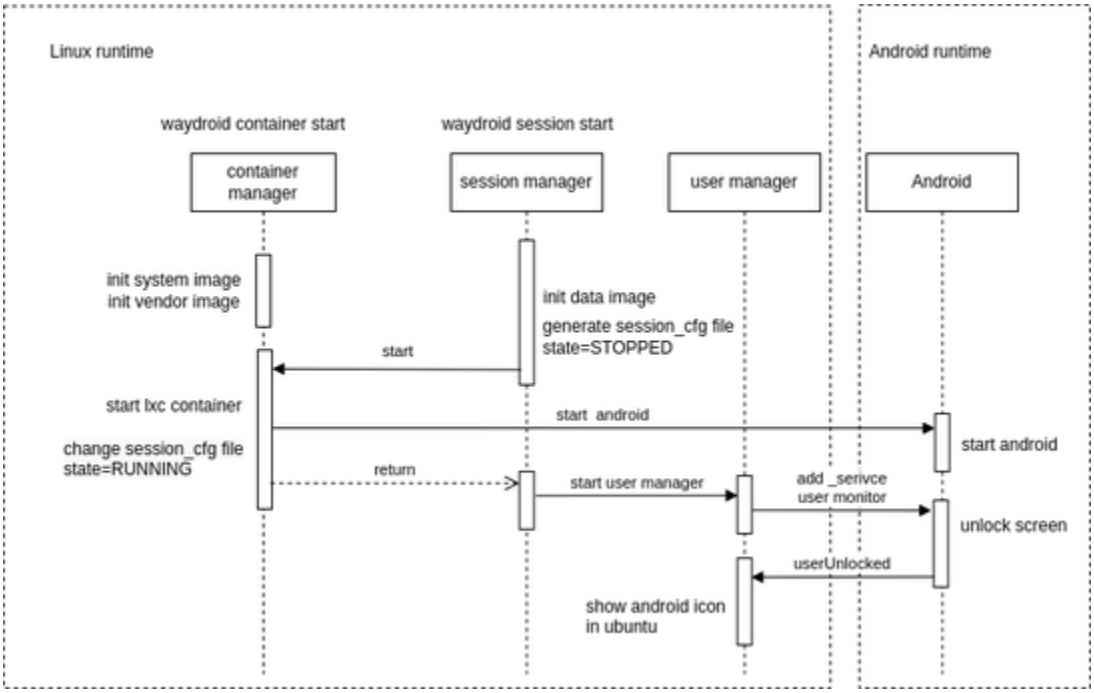
Different users can hold different user sessions.

	Description	**Status**
container	LXC container	RUNNING/STOPPED FROZEN/UNFROZEN
session	Android runtime envrionment one session include: user information userdata information wayland_display socket information pulseaudio socket information session status	RUNNING/STOPPED

All commands except” status” are uni-direction, you need to send “status” to check if you want to guarantee the command.



Relationship between container and session is shown below.



After the above process, the Android application configuration file will be generated in userdata directory(.local/share/applications), which shared between android and Linux.

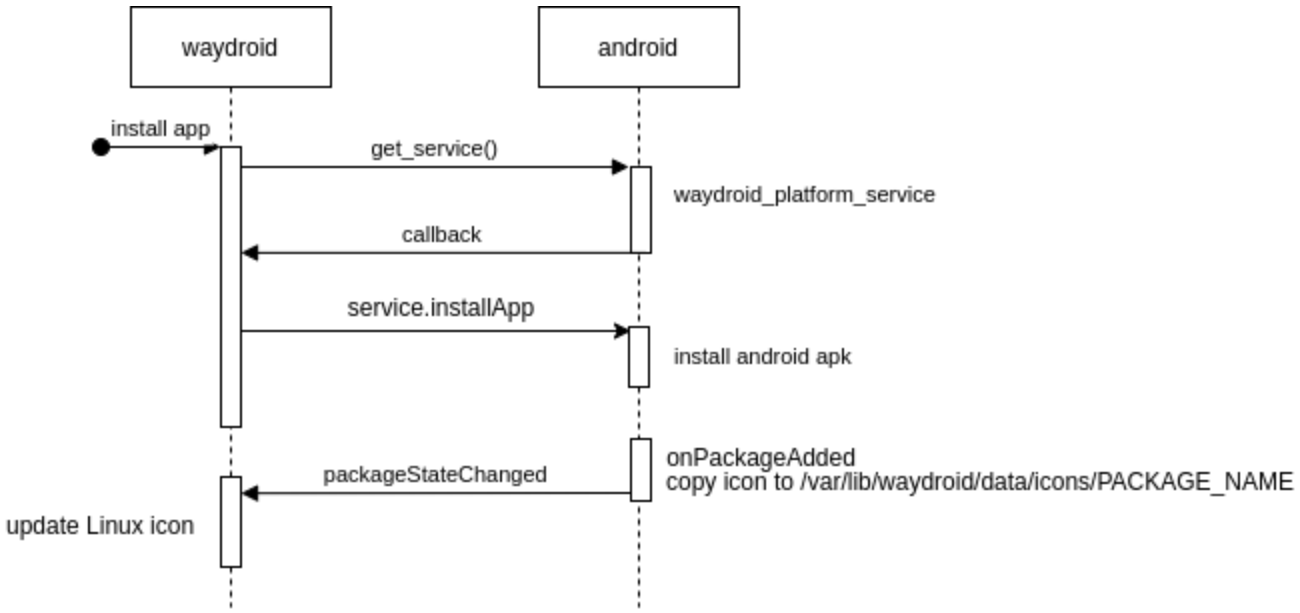
Below shows the android email configuration file.

```
cat ~/.local/share/applications/waydroid.com.android.email.desktop
[Desktop Entry]
Type=Application
Name=Email
Exec=waydroid app launch com.android.email
Icon=/home/gao/.local/share/waydroid/data/icons/com.android.email.png
```

Linux will auto read this file to generate its email icon.

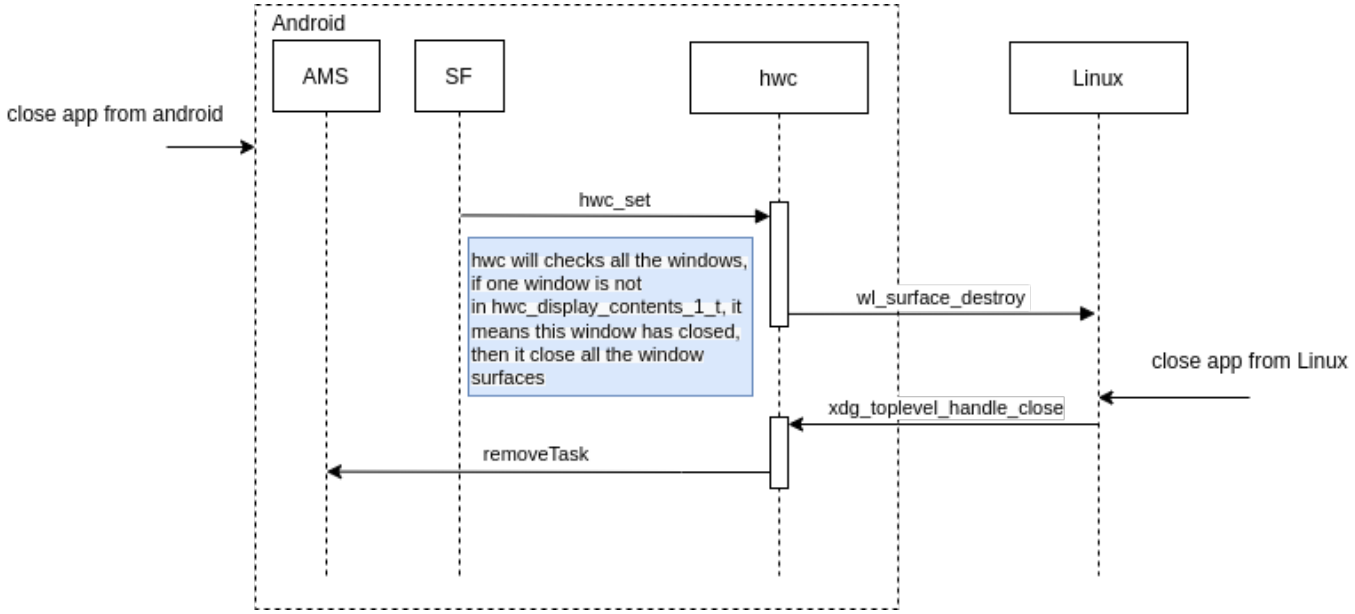
When user press this icon, the **Exec** points out how start the email application in android.

Android application installation process shows below.



Android application close process shows below.

The close can be from Android side, or from Linux side.



gbinder

The communication between Android (inside container) and Linux (outside container) can also use binder.

Because the Linux binder is mapped into container, both side can use the same binder to create/provide/get service from each other.

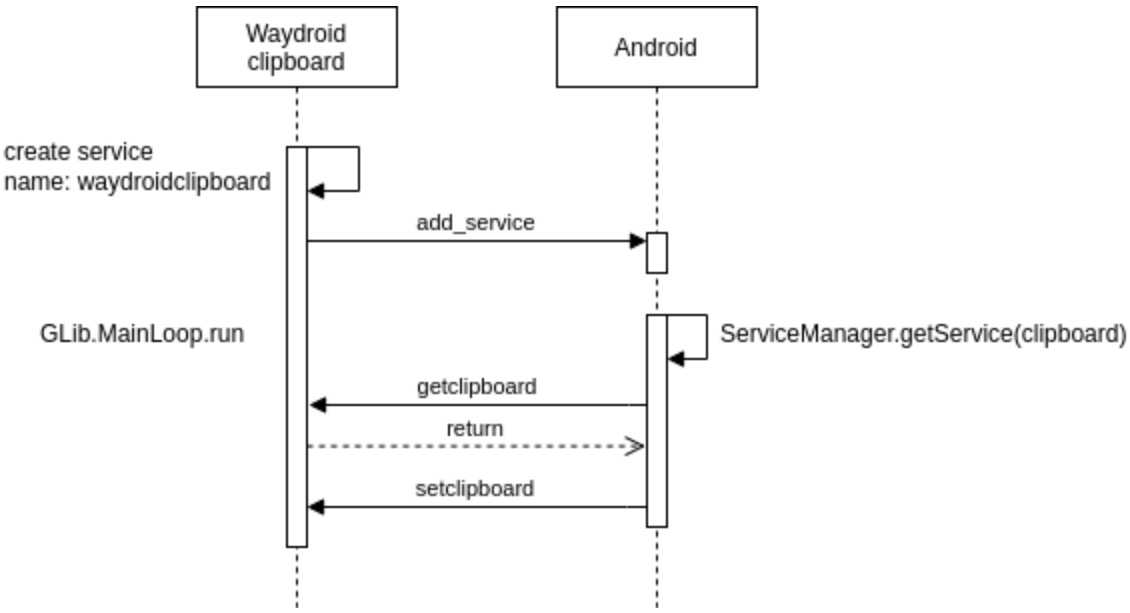
The Linux side c++ binder usage is in this project: https://github.com/mer-hybris/libgbinder.

Its python wrapper is in this project: https://github.com/erfanoabdi/gbinder-python.

Host side Android service

Some service is created at host side, and shared to client side, such as clipboard service.

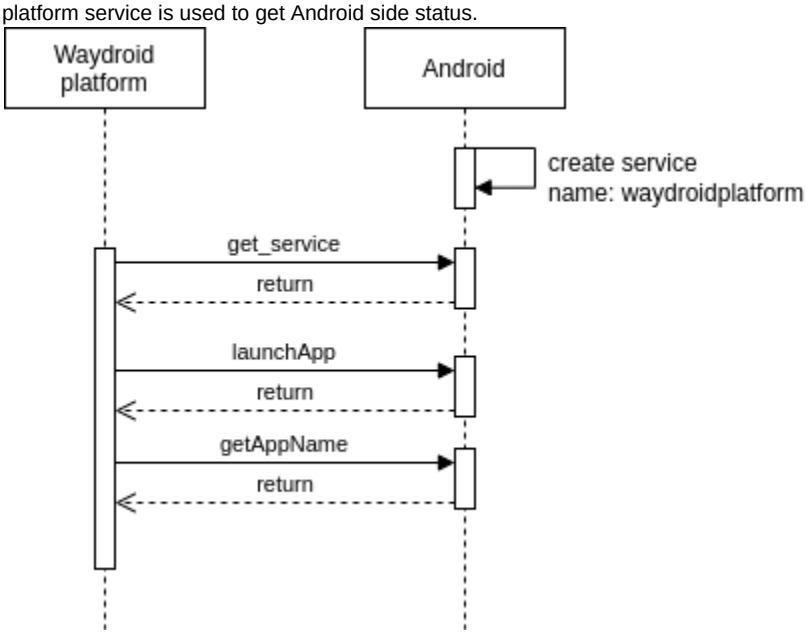
Clipboard service is used to share Linux clipboard data from within container.



Guest side Android service

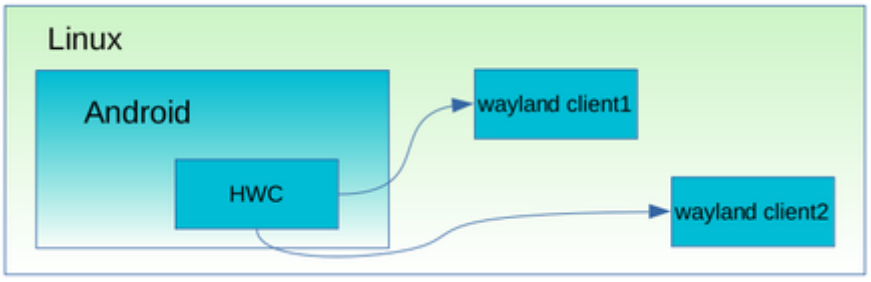
Some service is created at client side, and shared to host side, such as platform service.

platform service is used to get Android side status.



Android display sharing

Android modified HWC1 to send graphic buffer to Linux side by wayland buffer sharing protocol.



HWC1 interface.

```
.module_api_version = HWC_MODULE_API_VERSION_0_1,
```

API version.

```
pdev->base.common.version = HWC_DEVICE_API_VERSION_1_1;
```

Wayland buffer type

There are 2 kinds of buffer scenarios.

When configured as gbm mode, it will use dmabuf.

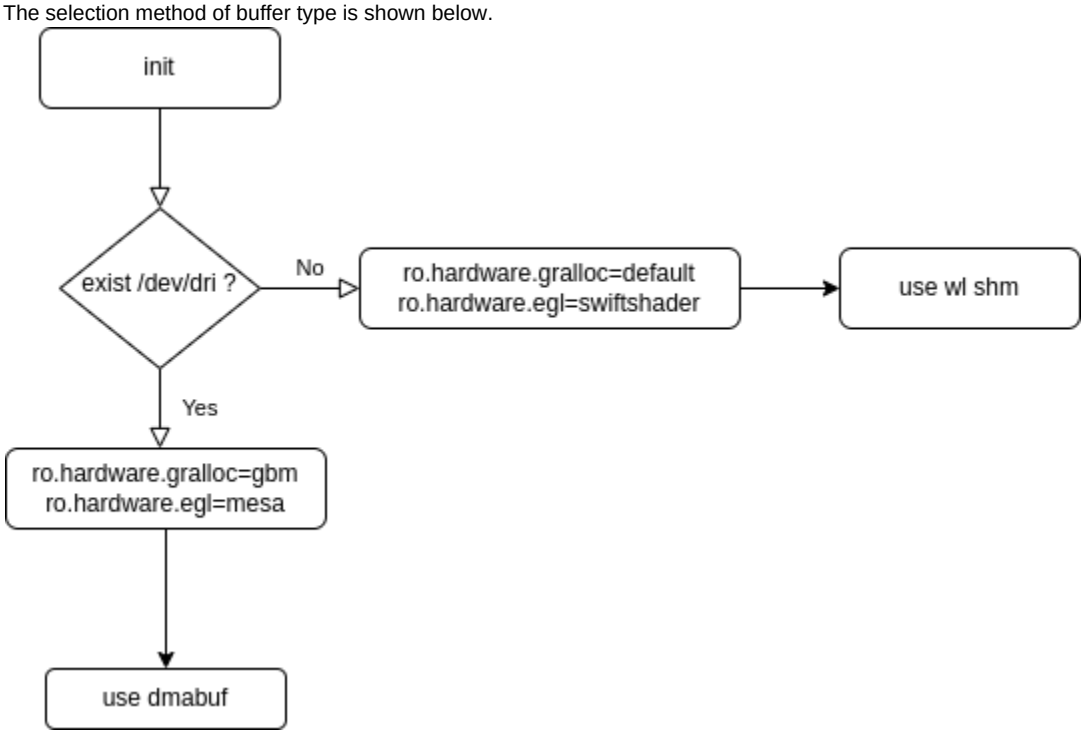
```
ro.hardware.gralloc=gbm
```

When configured as default mode, it will switch to shm.

When we run waydroid in virtual machine, we need to configure to use shm.

```
ro.hardware.gralloc=default
```

The selection method of buffer type is shown below.



Window mode

There are 2 kinds of window mode, single window mode and multi window mode, it can be configured as below.

```
waydroid prop set persist.waydroid.multi_windows true
```

In single window mode, all windows are composed to one window, whereas multi window mode uses separate windows.

The hwcomposer.cpp workflow is analyzed below.

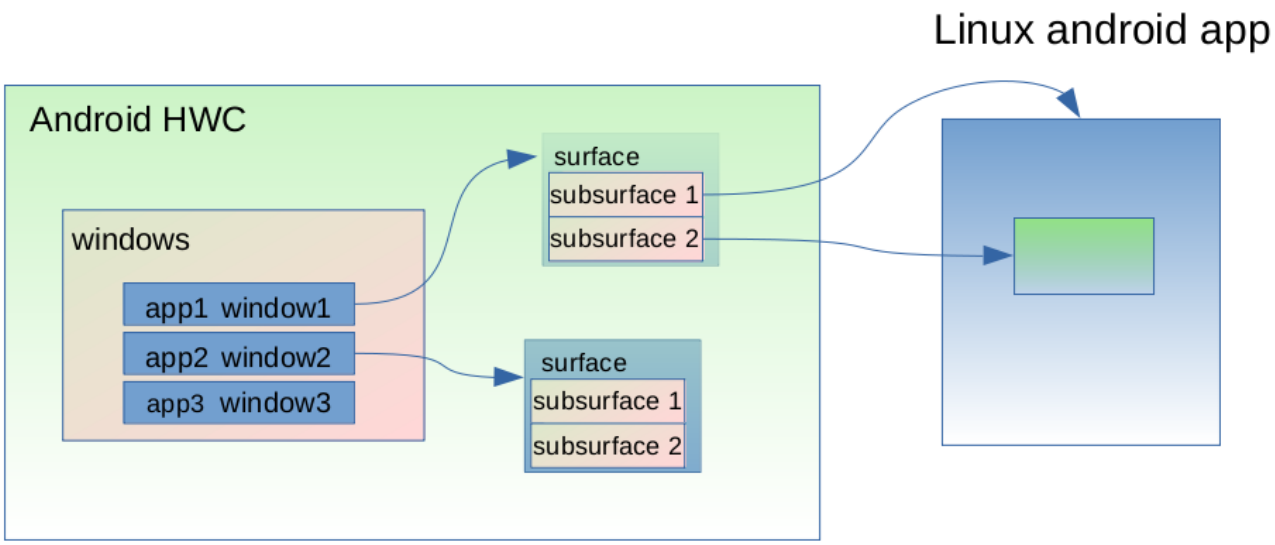
Single window mode Process:

1. in prepare() function, all layers' compositionType will be changed to HWC_FRAMEBUFFER, which means all layers will be composed as one layer by GPU, and the compositionType of this layer is HWC_FRAMEBUFFER_TARGET
2. in set() function, create a wl_buffer, copy the buffer from HWC_FRAMEBUFFER_TARGET to wl_buffer, commit wl_buffer to Linux wayland compositor.

Multi window mode Process:

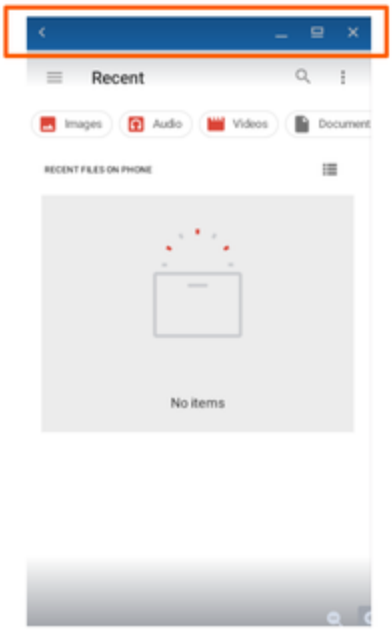
1. in prepare() function, all layers' compositionType will be changed to HWC_OVERLAY, which means all layers will be composed by HWC separately.
2. in set() function, create wl_buffer for each layer, and commit wl_buffer one by one.

In multi window mode, the HWC data struct is shown below.



Android Freeform mode

In multi window mode, waydroid uses android freeform mode to show the status bar of each application, this can be the same as a normal status bar of a Linux desktop application.



Fence and vsync in HWC

Each graphic buffer has one acquireFence and one releaseFence.

The fence is used to guarantee the consistency of graphic buffer, because graphic buffer will be shared between different processes.

Only layer with compositionType HWC_OVERYLAY and HWC_FRAMEBUFFER_TARGET will handle these 2 fence. The HWC_FRAMEBUFFER layer doesn't need to handle these fence.

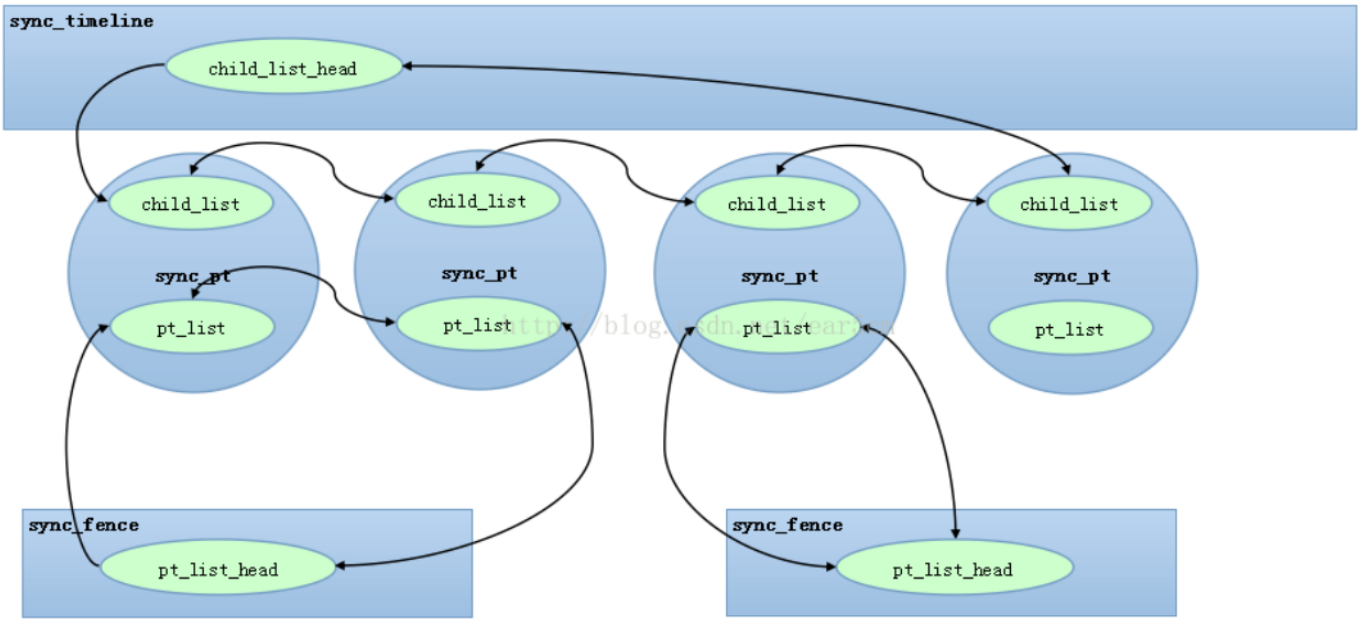
https://android.googlesource.com/platform/hardware/libhardware/+/master/include/hardware/hwcomposer.h

For the case of acquireFence, layer should wait for the signal of acquireFence before it's buffer can be copied or shared(in dmabuf case) to wl_buffer. It means the application finishes rendering this graphic buffer.

For the case of releaseFence, HWC will create releaseFence for each Layer. The timeline will be increased by 1 after wl_buffer release. Because the timeline is increased, all releaseFence whose sync point before this timeline will send signal, then the graphic buffer which hold this releaseFence can be reused in SurfaceFlinger. It means the display finishes displaying this graphic buffer.

> The timing of releaseFence signal is defined as after the HWC has finished reading from the buffer.
>
> In wl shm buffer case, the releaseFence should be signaled after the graphic buffer copies to the shm.
>
> In wl dmabuf case, it should be signaled after the display finished rendering. (In the callback of wl_buffer_release).

The Fence data structure in driver shows below.



https://asg-ivi-part.atlassian.net/wiki/spaces/IVIPART2/pages/edit-v2/2293797

Vsync is used to notify the finish of displaying one frame.

When SurfaceFlinger receives vsync signal, it will start a new composition.

Audio

Inside Android, waydroid added alsa-lib to support pulseaudio.

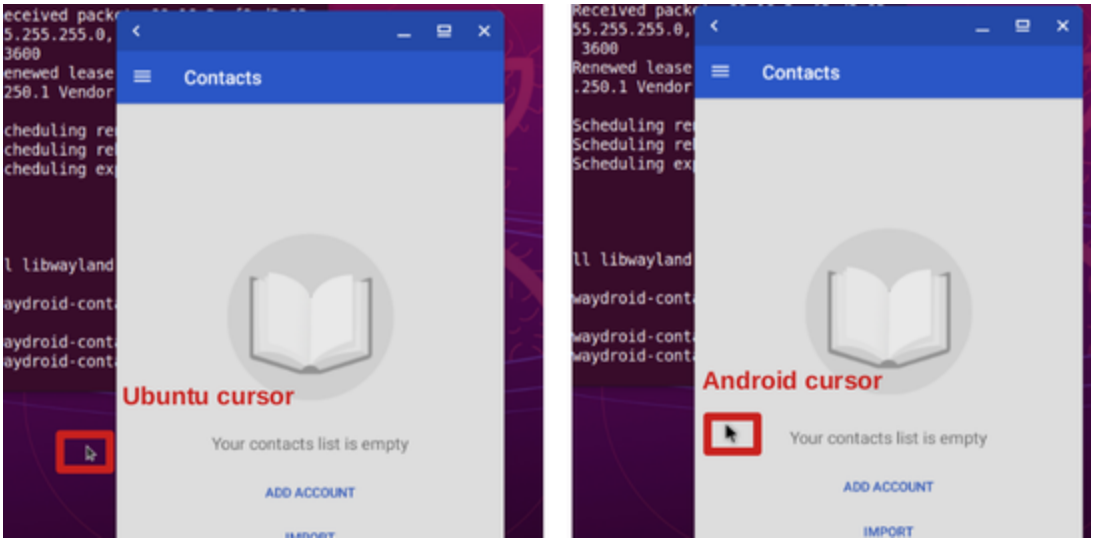
By using pulseaudio, the audio stream inside android can be routed to Linux outside the container.

Input support

In Android, when insert a mouse, there will be a new layer called “Sprite”, which shows the cursor layer.

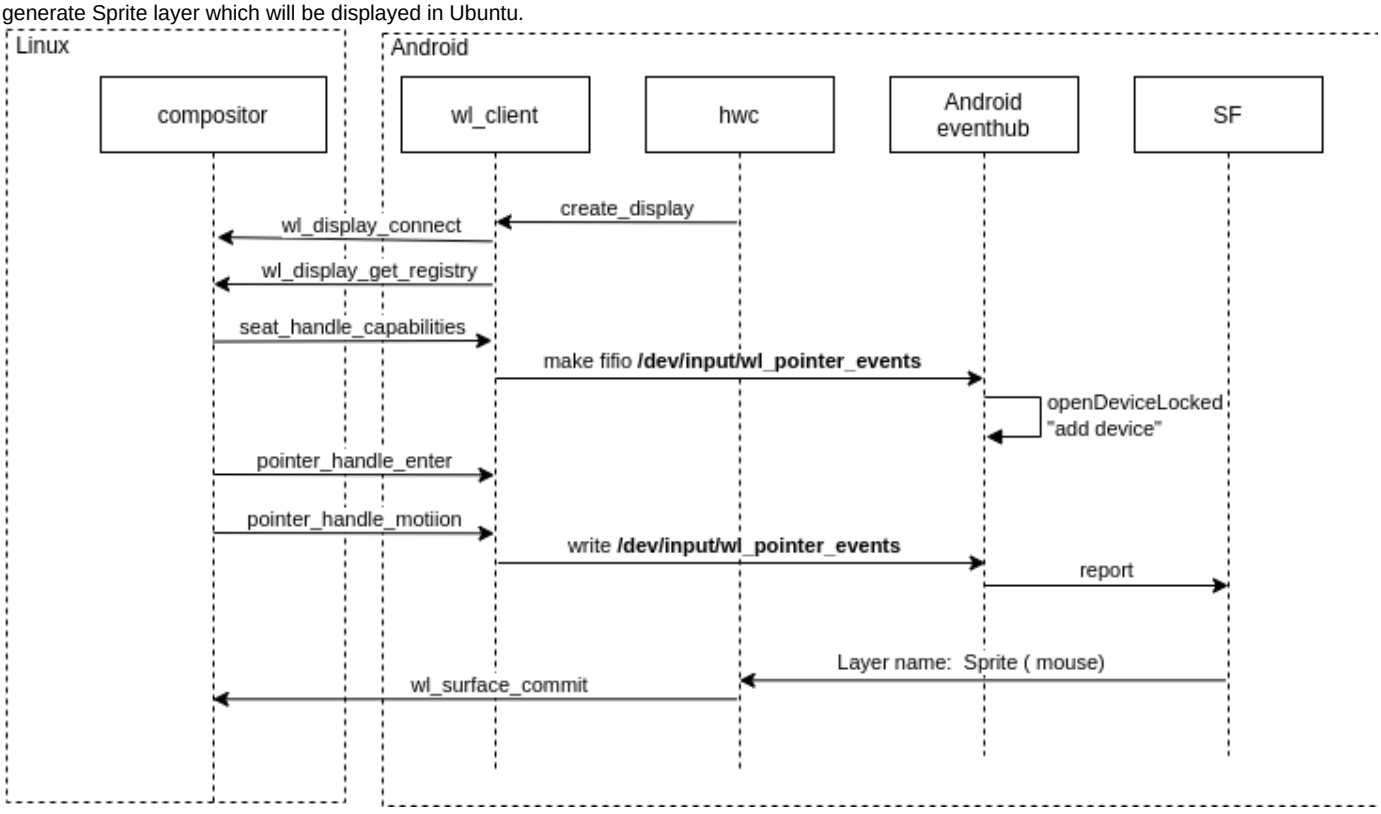
```
Layer name      Z |  Window Type |  Comp Type |  Transform |   Disp
Frame (LTRB) |          Source Crop (LTRB)

- - - - - - - - - - - - - - - - -- - - - - - - - - - - - - - - - -- - - - -
- - - - - - - - - - -
 Sprite#0  331000 |            0 |     DEVICE |          0 | 1493  246
1518  278 |    0.0    0.0   25.0   32.0
- - - - - - - - - - - - - - - - -- - - - - - - - - - - - - - - - -- - - - -
- - - - - - - - - - -
```

The cursor in Ubuntu and Android can be smoothly switched, which means when you move into the surface of Android applications, the cursor will change to Android style, and when you move out of the Android surface, it will switch back to Ubuntu style.



The flowchart of handling the cursor is shown below.

The key point is that when the cursor in Ubuntu hovers on the Android surface, Ubuntu will send input event to Android. Then Android will generate Sprite layer which will be displayed in Ubuntu.



How to use Waydroid in our appbridge solution

Change container control code to python

- easy to control the container status
- easy with log print
- use one python binary to control all the work flow

Gbinder

- use gbinder to create Android service outside the container.
- control Android by gbinder
- get Android status and sync to Linux.

Android mutli window mode

- currently we use WOD as Android display solution, so wayland solution will not be suitable
- If we use Ubuntu in future, this would be a good candidate for desktop solution.