Edging closer to the hardware for kernel CI on input devices

Benjamin Tissoires
Edging closer to the hardware for kernel CI on input devices

This talk is about input...

... But also about CI

totally exciting!
Who's that bearded guy?

**Why me?**

Kernel developer
input & HID stacks
"the one guy who breaks your keyboard, mouse, or touchscreen"

Now
"still that guy, but also the one who delays your HID patches upstream"
HID

- Human Interface Device
- plug and play (end of the 90's)
- at first USB only, now with Bluetooth, I2C, BLE

Easy for mice

Report descriptor
X is 8 bits longs
Y is 8 bits longs
buttons 1, 2, 3 are 1 bits long
5 bits of constant to fill in up to 8 bits

events
00 02 02 X: 0 Y: 2 B2 ⟷
00 00 02 X: 0 Y: 0 B2 ↠
00 00 00 X: 0 Y: 0 B2 ⟷
The suspect

**HID**

- Human Interface Device

- plug and play (end of the 90's)

- at first USB only, now with Bluetooth, I2C, BLE

Easy for mice
Not so much for multitouch panels

**Report descriptor**

- X is 8 bits longs
- Y is 8 bits longs
- tip switch 1 bits long
- 7 bits of constant to fill in up to 8 bits

**events**

- 05 02 01 00 00 00
- 00 00 00 02 05 01
- 00 00 00 00 00 00

X: 5 Y: 0
X: 0 Y: 2
X: 0 Y: 2
X: 5
????
The Problems

How to ensure there is no regressions?

Do I have an army of multitouch panels?

Yes, but...

Do I have a navy of mice?

Yes, but...

Note to self: do not break Linus' touchpad
The Problems

Drawbacks of manual testing

- more generic code means more manual tests
- time consuming
- full testing?
- not very compatible with more responsibility

is it really worth it?
The Past

First idea (2012-2014)

hid-multitouch only

1. get hid-record logs
2. replay them locally (hid-replay)
3. see the differences
4. cry

need to update the DB for each kernel
imprecise
takes too much time
doesn't scale

uhid.ko (June 2012)
The Past (well still today)

Userspace / libinput (2013 +)

- for every commit, write a test
- tests are simple (who said unit?)
- saves a ton of time

evdev to userspace

barely no downside
tedious

uinput.ko (well before git, before 2.6.12, before 2005)
Now what?

hid-tools (end of 2017 +)

hid protocol only

True kernel unit tests for HID
- rewrite hid-replay / hid-record in python
- library to parse report descriptors
- library to generate reports from the descriptors
- full test suite with unit tests
- currently keyboards, some mice, multitouch panels

barely no downside
tedious

uhid.ko

XDC 2019 - Montreal
What about CI?

thanks for the attendance BTW
Gitlab

- for every commit or MR, run a test
- runners are owned by Freedesktop, but you can add your private one
- saves a ton of time

But lots of duplicated pipelines across projects

https://gitlab.freedesktop.org
The suspects

Duplicated pipeline (classic)

freedesktop's gitlab

- run on some random upstream image
- update
- install dependencies
- run the tests
The usual suspects

**Duplicated pipeline (enhanced)**

freedesktop's gitlab

*setup job:*
- create a base test image if it does not exist
- update it
- install dependencies
- publish the image

*in a separate job:*
- use the test image container
- run the tests
Wayland's CI templates (1/3)

freedesktop's gitlab

Users of ci-templates objectives:
- save time
- have a reproducible environment to be able to fix the $*?! bug raised during CI
- make sure your project build continuously on $DISTRO

https://gitlab.freedesktop.org/wayland/ci-templates
- libinput
- mesa
- xorg/xserver
- weston

Yes you can

XDC 2019 - Montreal
Wayland's CI templates (2/3)

include:
- project: 'wayland/ci-templates'
  ref: 955e61e67...
  file: '/templates/arch.yml'

variables:
  ARCH_PKGS: 'git gcc pkgconfig meson ...'
  ARCH_EXEC: 'bash .gitlab-ci/arch_install.sh foo'
  ARCH_TAG: '2019-08-07.0'
  UPSTREAM_REPO: libinput/libinput
  ARCH_CONTAINER_IMAGE: $CI_REGISTRY_IMAGE/archlinux/rolling:$ARCH_TAG

arch@container-build:
  extends: .arch@container-ifnot-exists
  stage: arch_container_build
  variables:
    GIT_STRATEGY: none
    ARCH_VERSION: rolling

arch@check:
  image: $ARCH_CONTAINER_IMAGE
  script:
    - wget https://gitlab.freedesktop.org

DISTRO_PKGS: tells which RPMs, debs, packages are required in your tests
DISTRO_EXEC: optional script to run
DISTRO_TAG: tag of the image ready for testing (won't be rebuilt if exists)

Forks can also update the image, or pull yours to save power

ready to use
free to re-use (MIT)
quite useful
Wayland's CI templates (3/3)

freedesktop's gitlab

- include the template, and have benefits from it
- all the images are based on upstream and built locally
- CI of the CI
- has support for creating Ubuntu, Fedora, Arch and Debian images
- has support for creating arm64 images
- will have support for running VMs in the containers (wip branch for now)

https://gitlab.freedesktop.org/wayland/ci-templates
You won't believe it

Kernel HID CI

Objective: run CI suite for every commit I push, and every submitted patch I care for

- Use of ci-templates with qemu
- `.gitlab-ci.yml` not in the tested tree

Got to patch gitlab to not have `.gitlab-ci.yml` in the sources (https://gitlab.com/gitlab-org/gitlab-ce/issues/15041)

Got to play with `docker`, `podman`, `qemu` and `bash`

Got to play with kubernetes

https://gitlab.banquise.eu/bentiss/linux

I'm still the "the one guy by who breaks your keyboard, mouse, or touchscreen, but also the one who delays your HID patches upstream", and I am so hype
You won't believe it

Kernel HID CI

<table>
<thead>
<tr>
<th>#</th>
<th>Status</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>passed</td>
<td>mfd: mc13xxx: Add m...</td>
</tr>
<tr>
<td>52</td>
<td>passed</td>
<td>input: touchscreen mc...</td>
</tr>
<tr>
<td>48</td>
<td>passed</td>
<td>Bluetooth: hidp: Fix a...</td>
</tr>
<tr>
<td>47</td>
<td>passed</td>
<td>Input - elan_i2c: remo...</td>
</tr>
<tr>
<td>002942</td>
<td></td>
<td>Import - touch screen mi...</td>
</tr>
</tbody>
</table>
Thanks for coming in on a Friday morning

https://gitlab.freedesktop.org/wayland/ci-templates