HID Device and Wine (Update)

Presented 2016 WineConf
St. Paul, Minnesota, USA

By Aric Stewart
Overview

- HID is working
  - Platforms, technologies and devices
  - Why would you care?
    - TL;DR - you likely don't
- Fun Demonstrations
- Looking forward
  - Dinput, winmm, xinput, rawinput, etc...
- Overly Technical bits
HID Device Native Platforms

Working Platforms

- Linux Hidraw
  - Udev device permissions need to be set on the /dev/hidraw<X> device
- OS/X IOHID
  - Working by default

Future Platforms

- Linux Event
- Android
Working HID device technologies

- Hid.dll
  - HidD_ and HidP_ functions
  - Sparse but functioning
- Direct Device Access
  - IOCTLS and FileRead/FileWrite
  - Functional
- Setupapi
  - Device detection and enumeration works
Devices

- Most development focused on Gamepad and Joystick (usages 0x04 and 0x05)
- Keyboards, devices with keys as opposed to buttons, are not working yet
- Mice appear to work
- Oculus HID parts working
- Need more devices tested!
Takeaways

- Programs that directly access HID will now do something
  - Anyone have any?
  - Custom Hardware likely required.
- Programs trying to directly talk to HID devices will now have a chance to success
  - E-cig programs, some dongles
  - Anyone have more?
- Broad range of HID device and program testing is desired
Devices

- Most development focused on Gamepad and Joystick (usages 0x04 and 0x05)
- Mice appear to work
- Oculus HID parts working
- Need more devices tested!
- Keyboards, devices with keys as opposed to buttons, are not working yet.
  - Including the Steam Controller, which reports itself as a keyboard.
Fun Technical Demonstrations.
Looking Forward

- Dinput
  - Native can work! It uses HID
  - ForceFeedback Problems
- Xinput
  - Also client of HID investigation needed
- Winmm
  - Should be able to be fully supported
- RawInput
  - A user level wrapper around HID
Questions?

- No Plug and Play discussion?
  - Complicated enough it warrants its own discussion when required

- Native Drivers?
  - Can theoretically be made to work
  - Lots of ntoskrnl.exe work, setupapi and likely plug and play work

- Questions?
- Comments?
Overly Technical Bits

• Feel free to examine these in the published slides at your own leisure
Wine's HID Device Stack

User Land (CreateFile, hid.dll, etc…)

IOCTLS

Hidclass.sys (class driver)

IRP_MJ_INTERNAL_DEVICE_CONTROL

Winehid.sys (minidriver)

IRP_MJ_INTERNAL_DEVICE_CONTROL

winebus.sys

Platform vtbl

Function calls

Platform Specific Code

Platform Native API calls

Native Platform (udev / hidraw / linux event / iohid)

Bus Device (PDO)

HID Device (FDO)
WINE Plug and Play flow for HID devices

Platform detects a device

Create_hid_device creates a bus bus device

IoInvalidateDeviceRelations is called

Winebus.sys (3rd party driver)

Win32 API

Winehid.sys (3rd party driver)

PnP manager looks for a driver to handle the device by id

Winehid.sys is found, AddDevice is called

Hidclass.sys creates a HID device and add it to the bus device's device stack

Power and PNP irps are sent to the top of the device stack

Winehid.sys and hidclass.sys handles the power and pnp irps, creating interfaces and links
**HID Device Report Flow**

**Winedevice Process**
- Platform generates a device report
- Common hid bus device code receives report and sends it to pending IOCTLs
- Pending IOCTL_HID_READ_REPORT
- IOCTL_HID_READ_REPORT is left pending

**Winebus device report thread**
- IOCTL_HID_READ_REPORT ioctl are signaled
- Report as added to the ring buffer
- Pending ReadFile requests are signaled
- IOCTL_HID_READ_REPORT is sent to the minidriver for the next report.
- IOCTL_HID_READ_REPORT is sent to the forwarded down the driver stack to the bus driver.

**Application Process**
- Application gets the device report
- Application used the HID data
- Application does a read
- If the ring buffer is empty, the read is left pending.

**Win32 API / Native application**
- Winehid.sys (3rd party driver)
- Winebus.sys (3rd party driver)