Eksamen 2012  
3. semester  
I3MPS - Microprocessor systemer  
Kasper Nissen

**Spørgsmål 8**

**Linux Input Drivers**

**Spørgsmål**

* Hvad dækker begreberne Input Device Driver og Input Event Driver over?
* Hvad er Evdev?
* Eksempler på input event driver typer?
* Eksempel hvilke elementer som en virkelig f.eks. touch-screen driver indeholder og hvordan de hænger sammen

**Pensum:**

Text: Venkateswaran ch 7 (Se fildeling)

Documentation/input/input-programming.txt:

<http://git.kernel.org/?p=linux/kernel/git/stable/linux-stable.git;a=blob;f=Documentation/input/input-programming.txt;h=7f8b9d97bc47d32359150822eaa16b23acfc13be;hb=HEAD>

**Exercise 11**

**Input sub systemet**

Kernens input subsystem er lavet for at kunne håndtere diverse arter af data-input enheder som:

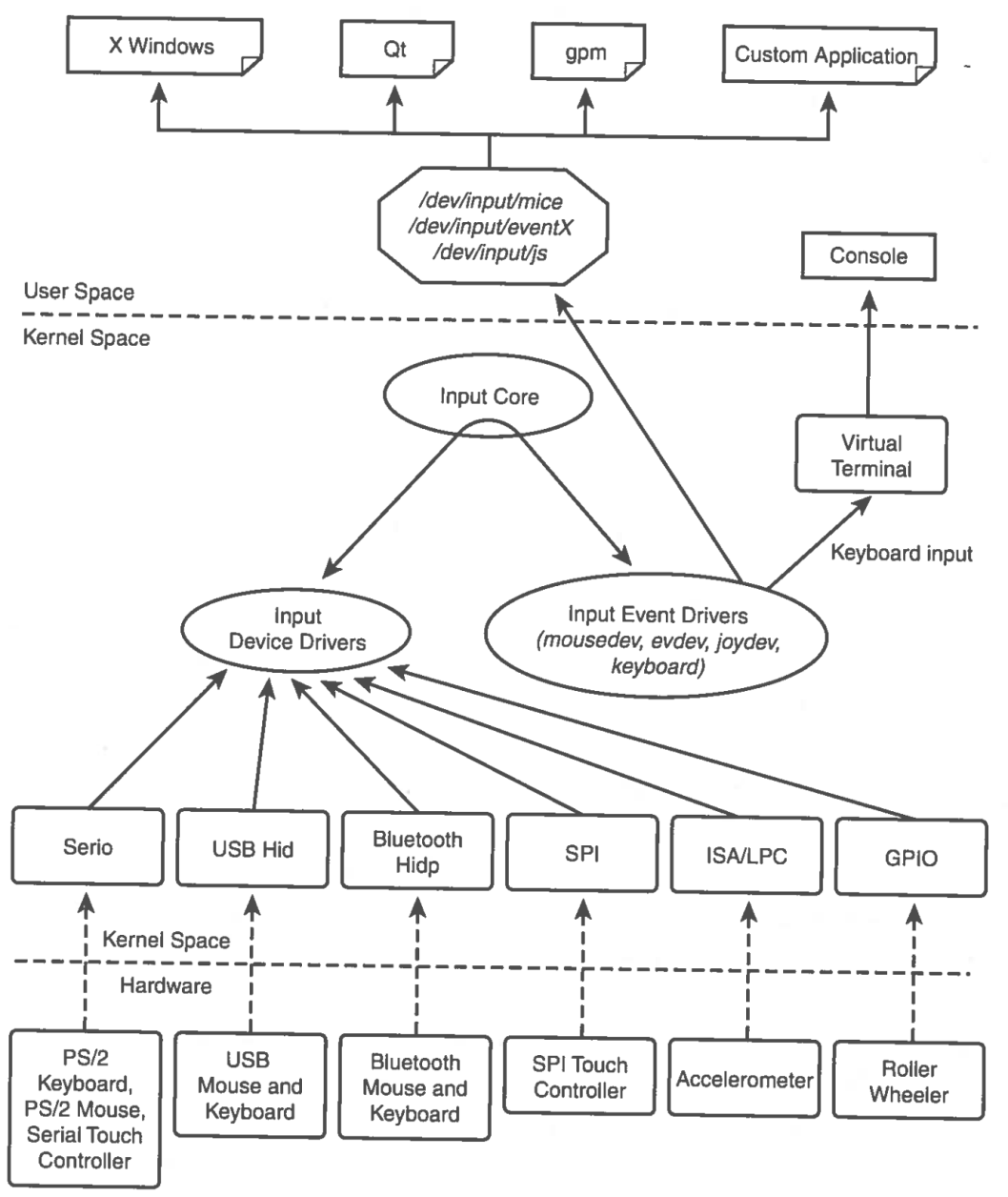
* Keyboard
* Mus
* Trackballs
* Joystick
* Roller wheels
* Touchscreens
* Accelerometers
* Tablets

Input systmet giver følgende fordele:

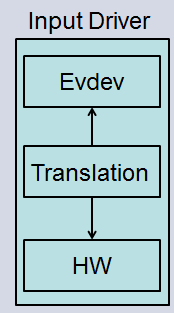
* Ens behandling af enheder med samme funktionalitet, selvom de er fysisk forskellige. F.eks. all mus’ som PS/2, USB, Bluetooth bliver behandlet på samme måde.
* Et nemt event interface, som sender input reporter til programmer i user space. Driverne behøver ikke at oprette, og opretholde /dev noder eller andre adgangsmetoder. I stedet, påvirkes input API’et til bevægelser med mussen, tryk på keyboard eller touchskærm op til user space.
* Enshed, da input systemet tilbyder en række low-level drivers kaldet serio, som giver adgang til input hardware som serial porte eller keyboard controllere.

Som det ses på figuren nedenfor består subsystemet af 2 forskellige klasser af drivere som samarbejder, nemlig:

* Event drivere
  + Ansvaret for interfacing med applikationer
* Device drivere
  + Ansvaret for kommunikationen med input enhederne.(I/O)



**Input Device Driver**



Måden hvorpå man registrer en input driver er via kommandoen:

Input\_allocate\_device();

Set\_bit();

Laver interface til event driveren gennem subsystemet:

Input\_report\_*XYZ*(); // Produce event

Input\_sync(); // Update /dev/input/event

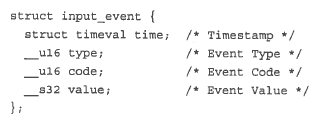
Interface til hardwaren i den anden ende af figuren

**Input Event Driver**

Event interfacet genereret af input subsystemet har udviklet sig til en standard som mange grafiske vindues systemer forstår. Event drivere tilbyder en hardware uafhængig abstraktion til at tale med input enheder.

**Evdev**

Evdev er en generisk input event driver. Hver event pakke produceret af evdev har følgende formater. Defineret i include/linux/input.h

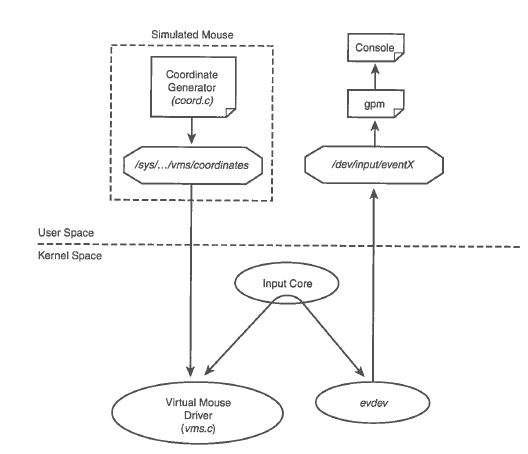


For bedre at forklare hvad en input driver gør, og hvilken rolle evdev spiller vil jeg gennemgå et eksempel:

**Virtual Mouse**

Virtual Mouse applikationen virker ved at vi har en separat applikation(coord.c) som emulere muse bevægelse og sender disse koordinater til en virtual muse driver (vms.c) via en sysfs node, /sys/devices/platform/vms/coordinates. Den virtuelle muse driver kanalisere disse bevægelser up via evdev.

På figuren nedenfor ses hvordan det hele fungere:



Som vi gjorde i øvelsen til denne opgave, kunne man i stedet for coord.c applikationen echo direkte til sysfs noden:

echo 10 10 0 > /sys/device/platform/vms/coordinates

Og som vi så I øvelsen så bevæger cursoren sig 10 hen og 10 op fra det punkt hvor cursoren nu er placeret.

**Eksempler på tilkobling af events**

* + GPM (Mouse that can make the curser move around the screen without X11)
  + X11 (Linux Desktop Manager)
    - Evdev and udev must be enabled in the kernel
    - Evdev uses Udev to configure devices
    - This provides hot-plugging in X11, devices need not to be specified in xorg.conf (X11 configuration file)
  + QT
    - Environment variables must be defined and drivers compiled wtih QT Library

QWS\_KEYBOARD=“LinuxInput:/dev/input/eventXYZ”

QWS\_MOUSE\_PROTO=“IntelliMouse:/dev/input/mouse0”

Gennemgå evt. selve driveren fra opgaven?