October 2020

Detecting suboptimal USB connections and informing a user about it

Armijn Hemel

Detecting suboptimal USB connections and informing a user about it

Abstract

There are different hardware standards that share the same physical connectors and are backwards compatible. Examples are USB 1.1, USB 2.0 and USB 3.0. The different hardware standards specify different speeds per profile and different maximum speeds. Plugging a fast device into a port that only supports older standards will mean that the slower speed of the older standard will be used which could lead to a decrease in hardware performance and a suboptimal user experience.

By informing the user if he/she makes a suboptimal choice in connecting devices to a computer, the performance of the devices, as well as the user experience, can be vastly improved.

Keywords

USB, hardware, Linux

Background

USB 1.1, USB 2.0 and USB 3.0 all have the same hardware plug for backwards compatibility, but they are different standards, that specify different speeds per profile and different maximum speeds. Plugging a fast device into a port that only supports older standards will mean that the slower speed of the older standard will be used which could lead to a decrease in performance. Copying a file from a USB 3.0 harddisk plugged into a USB 1.1 port will take much longer than if it would have been plugged into a USB 3.0 port.

Modern operating systems support sending events about hardware changes to programs that are interested in it. Examples are udev and systemd on Linux.

By knowing what hardware is inside a computer and matching the characteristics of a plugged in device with the port it is plugged into as soon as it is plugged into this port, a user can be informed in case there is a suboptimal arrangement for hardware.

Example

A computer has three USB ports:

1. Port A: USB 1.1
2. Port B: USB 2.0
3. Port C: USB 3.0

The user as three devices:

1. Device K: USB 2.0 camera
2. Device L: USB 3.0 harddisk
3. Device M: USB 1.1 optical mouse

The optimal configuration would be:

- Port A: Device M
- Port B: Device K
- Port C: device L

An example of a suboptimal configuration would be:

- Port A: device L
- Port B: device K
- Port C: device M

Solution

1. The operating system queries the hardware and stores characteristics about the various extension ports.
2. When a user plugs in a piece of hardware the operating system queries the hardware that was plugged in.
3. the operating system checks in which port the hardware was plugged in and looks up the characteristics of the port
4. the operating system checks if there are free ports on the system that better suit the characteristics of the hardware and issues a warning if there is a better configuration possible.
5. the operating system sends an event containing information that there is a suboptimal configuration
6. a program on the desktop listening to these events displays a pop up to inform the user that a suboptimal hardware configuration was chosen.

Proposing better hardware configuration layouts

Using the information queried from the hardware and the ports it is possible for the operating system to suggest the user a better arrangement of the hardware on the system. The following steps follow after finding a suboptimal configuration. Based on the stored configuration of the characteristics of the hardware ports and the hardware characteristics of the plugged in devices a better solution can be computed. Finding a better configuration could be initiated by the user, or done automatically by the system without direct intervention by the user.

7. for each device the maximum needed speed is looked up. The devices are sorted from high speed to low speed. All ports on the system are sorted by maximum possible speed, from high speed to low speed.
8. for each port (fastest port first) a device is assigned
9. a better configuration is reported to the user.