persist.txt
USB device persistence during system suspend

Alan Stern <stern@rowland.harvard.edu>

September 2, 2006 (Updated February 25, 2008)

What is the problem?

According to the USB specification, when a USB bus is suspended the bus must continue to supply suspend current (around 1-5 mA). This is so that devices can maintain their internal state and hubs can detect connect-change events (devices being plugged in or unplugged). The technical term is "power session".

If a USB device's power session is interrupted then the system is required to behave as though the device has been unplugged. It's a conservative approach; in the absence of suspend current the computer has no way to know what has actually happened. Perhaps the same device is still attached or perhaps it was removed and a different device plugged into the port. The system must assume the worst.

By default, Linux behaves according to the spec. If a USB host controller loses power during a system suspend, then when the system wakes up all the devices attached to that controller are treated as though they had disconnected. This is always safe and it is the "officially correct" thing to do.

For many sorts of devices this behavior doesn't matter in the least. If the kernel wants to believe that your USB keyboard was unplugged while the system was asleep and a new keyboard was plugged in when the system woke up, who cares? It'll still work the same when you type on it.

Unfortunately problems _can_ arise, particularly with mass-storage devices. The effect is exactly the same as if the device really had been unplugged while the system was suspended. If you had a mounted filesystem on the device, you're out of luck -- everything in that filesystem is now inaccessible. This is especially annoying if your root filesystem was located on the device, since your system will instantly crash.

Loss of power isn't the only mechanism to worry about. Anything that interrupts a power session will have the same effect. For example, even though suspend current may have been maintained while the system was asleep, on many systems during the initial stages of wakeup the firmware (i.e., the BIOS) resets the motherboard's USB host controllers. Result: all the power sessions are destroyed and again it's as though you had unplugged all the USB devices. Yes, it's entirely the BIOS's fault, but that doesn't do _you_ any good unless you can convince the BIOS supplier to fix the problem (lots of luck!).

On many systems the USB host controllers will get reset after a suspend-to-RAM. On almost all systems, no suspend current is available during hibernation (also known as swsusp or suspend-to-disk). You can check the kernel log after resuming to see if either of these

has happened; look for lines saying "root hub lost power or was reset".

In practice, people are forced to unmount any filesystems on a USB device before suspending. If the root filesystem is on a USB device, the system can't be suspended at all. (All right, it _can_ be suspended -- but it will crash as soon as it wakes up, which isn't much better.)

What is the solution?

The kernel includes a feature called USB-persist. It tries to work around these issues by allowing the core USB device data structures to persist across a power-session disruption.

It works like this. If the kernel sees that a USB host controller is not in the expected state during resume (i.e., if the controller was reset or otherwise had lost power) then it applies a persistence check to each of the USB devices below that controller for which the "persist" attribute is set. It doesn't try to resume the device; that can't work once the power session is gone. Instead it issues a USB port reset and then re-enumerates the device. (This is exactly the same thing that happens whenever a USB device is reset.) If the re-enumeration shows that the device now attached to that port has the same descriptors as before, including the Vendor and Product IDs, then the kernel continues to use the same device structure. In effect, the kernel treats the device as though it had merely been reset instead of unplugged.

The same thing happens if the host controller is in the expected state but a USB device was unplugged and then replugged, or if a USB device fails to carry out a normal resume.

If no device is now attached to the port, or if the descriptors are different from what the kernel remembers, then the treatment is what you would expect. The kernel destroys the old device structure and behaves as though the old device had been unplugged and a new device plugged in.

The end result is that the USB device remains available and usable. Filesystem mounts and memory mappings are unaffected, and the world is now a good and happy place.

Note that the "USB-persist" feature will be applied only to those devices for which it is enabled. You can enable the feature by doing (as root):

echo 1 >/sys/bus/usb/devices/.../power/persist

where the "..." should be filled in the with the device's ID. Disable the feature by writing 0 instead of 1. For hubs the feature is automatically and permanently enabled and the power/persist file doesn't even exist, so you only have to worry about setting it for devices where it really matters.

Is this the best solution?

Perhaps not. Arguably, keeping track of mounted filesystems and memory mappings across device disconnects should be handled by a centralized Logical Volume Manager. Such a solution would allow you to plug in a USB flash device, create a persistent volume associated with it, unplug the flash device, plug it back in later, and still have the same persistent volume associated with the device. As such it would be more far-reaching than USB-persist.

On the other hand, writing a persistent volume manager would be a big job and using it would require significant input from the user. This solution is much quicker and easier -- and it exists now, a giant point in its favor!

Furthermore, the USB-persist feature applies to _all_ USB devices, not just mass-storage devices. It might turn out to be equally useful for other device types, such as network interfaces.

WARNING: USB-persist can be dangerous!!

When recovering an interrupted power session the kernel does its best to make sure the USB device hasn't been changed; that is, the same device is still plugged into the port as before. But the checks aren't guaranteed to be 100% accurate.

If you replace one USB device with another of the same type (same manufacturer, same IDs, and so on) there's an excellent chance the kernel won't detect the change. The serial number string and other descriptors are compared with the kernel's stored values, but this might not help since manufacturers frequently omit serial numbers entirely in their devices.

Furthermore it's quite possible to leave a USB device exactly the same while changing its media. If you replace the flash memory card in a USB card reader while the system is asleep, the kernel will have no way to know you did it. The kernel will assume that nothing has happened and will continue to use the partition tables, inodes, and memory mappings for the old card.

If the kernel gets fooled in this way, it's almost certain to cause data corruption and to crash your system. You'll have no one to blame but yourself.

YOU HAVE BEEN WARNED! USE AT YOUR OWN RISK!

That having been said, most of the time there shouldn't be any trouble at all. The USB-persist feature can be extremely useful. Make the most of it.