


# What's in the Box?

## Interrogate Your Linux Machine's Hardware

Shell and graphic utilities for  
hardware detection. **FEDERICO KEREKI**



I recently had a problem trying to install the NVIDIA driver for my machine. It seemed the latest driver had stopped supporting my graphics card, and after updating my kernel, I was out of a driver. The question, obviously, was “which card did I have?” But, I didn’t remember. If you have to name the chipset of your motherboard, specify the CPU in your box or get any other kind of hardware-related information, Linux provides several utilities to help you. In my case, I quickly could get the full ID of my graphics card, confirm that it really was getting a bit long in the tooth and decide that a newer one wasn’t such a bad idea.

In this article, I discuss several ways of getting hardware data for your machine. In the most time-honored Linux shell way, I show how to work with several command-line utilities, but if you prefer using a GUI, I also include some graphical programs. And, if you want to get into the nitty-gritty details, I give some pointers on how to get that information by using the `/proc` or `/sys` filesystem.

## The `ls` Command Family

Let’s start the command-line work with a set of several utilities, whose names all start with `ls` (Table 1). Some of these commands provide overlapping

## Glossary

Working with hardware means dealing with several acronyms, and I must admit, I had been using at least a couple of them without remembering precisely what they meant. Here’s a list of definitions you’ll surely need:

- ACPI (Advanced Configuration and Power Interface): related to power aspects.
- AGP (Accelerated Graphics Port): a channel to allow attaching a video graphics card (not typically seen since around 2008).
- APM (Advanced Power Management): older than ACPI, also related to power issues.
- ATA (AT Attachment): “AT”, as in the old IBM AT, a standard to connect storage devices, superseded by SATA in 2003.
- BIOS (Basic Input/Output System): firmware used when booting an Intel-compatible PC.
- DMA (Direct Memory Access): a feature that allows giving hardware access to RAM, independently of the CPU.
- DMI (Desktop Management Interface): a framework for keeping track of devices in a computer.
- IDE (Integrated Drive Electronics): an interface standard that later evolved into ATA.
- IRQ (Interrupt ReQuest): a hardware signal that allows an interrupt handler to process a given event.
- PCI (Peripheral Component Interconnect): a bus standard for attaching varied hardware devices to a computer, created in 1992.
- UEFI (Unified EFI—Extensible Firmware Interface): a 2005 replacement for BIOS, which deprecated the previous 1998 EFI standard.
- USB (Universal Serial Bus): a standard bus defined in 1995 to allow connecting all kinds of peripherals to a computer.
- PATA (Parallel ATA): the new name for ATA, after SATA came out.
- PCIe (PCI Express): a high-speed serial bus that replaced PCI and AGP in 2004.
- RAID (Redundant Array of Independent—originally, “Inexpensive”—Disks): a data storage virtualization technology that combines several drives to work as a single one for performance improvement and/or data redundancy. There are several RAID schemes, including RAID 0 (“striping”), RAID 1 (“mirroring”), RAID 5 (“striping + parity”) and RAID 10 (“striping + mirroring”).
- SATA (Serial ATA): a bus interface to connect storage devices, currently used in practically all PCs.
- SCSI (Small Computer System Interface—pronounced “scuzzy”): a set of standards for connection of devices and transfer of data between computers and peripherals.

Table 1. The ls\* family of commands lets you access all aspects of your hardware.

COMMAND	DESCRIPTION
lsblk	Produces information about all block devices, such as hard disks, DVD readers and more.
lscpu	Shows information like number of CPUs, cores, threads and more.
lsdev	Displays data about all devices of which the system is aware.
lshw	Lists general hardware data—gives information on every detail of your hardware.
lspci	Displays information about PCI buses in your box and devices connected to them, such as graphics cards, network adapters and more.
ls SCSI	Provides information on all SCSI devices or hosts attached to your box, such as hard disk drives or optical drives.
lsusb	Generates information about USB buses in your box and devices connected to them.

information (lsdev or lshw, for instance), but by using all of them, you can get a pretty clear idea of whatever may be inside your Linux box.

Let's start with CPU information. The lscpu command provides data on the CPUs in your box. You can opt to include all CPUs, whether off-line or on-line, with the - .all parameter, or you can select --online and --offline. The --parse option lets you choose what CPU characteristics to list, including number, socket, cache data, maximum and minimum speed (in MHz) and more. In my case, you'll see that my machine has a somewhat old single-socket, four-core, Intel Core 2 Quad CPU, at 2.66GHz:

```
> lscpu
```

```
Architecture:      x86_64
```

```
CPU op-mode(s):    32-bit, 64-bit
Byte Order:        Little Endian
CPU(s):            4
On-line CPU(s) list: 0-3
Thread(s) per core: 1
Core(s) per socket: 4
Socket(s):         1
NUMA node(s):      1
Vendor ID:         GenuineIntel
CPU family:        6
Model:             23
Model name:        Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz
Stepping:          10
CPU MHz:           2003.000
CPU max MHz:       2670.0000
CPU min MHz:       2003.0000
BogoMIPS:          5340.67
Virtualization:    VT-x
L1d cache:         32K
L1i cache:         32K
L2 cache:          2048K
NUMA node0 CPU(s): 0-3
```

(Note: you can get most of this information by examining the `/proc/cpuinfo` file or by browsing the `/sys/bus/cpu/` directories; see the DIY with `/proc` and `/sys` sidebar for more on this.)

Let's move on to block devices, such as hard disks, or CD and DVD units. The `lsblk` command produces information on all available block devices (see Listing 1 for an example). As you can see, I have three hard disks

and a ROM (DVD) device. The three disks are known as `/dev/sda`, `/dev/sdb` and `/dev/sdc`; the ROM device is `/dev/sr0`. The disks are 466GB, 149GB and 2.7TB in size. You can get a little information about partitioning too; for instance, you can see that the first two disks have a swap area enabled, but the third one doesn't. You also can get the mountpoints (`/`, `/disk-laptop` and `/disk-data`) for the three disks.

**Listing 1. The `lsblk` command shows all block (storage) devices. The `--topology` option adds extra details; try `--output-all` for even more.**

```
> lsblk --paths
```

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
/dev/sda	8:0	0	465.8G	0	disk	
__/dev/sda1	8:1	0	4G	0	part [SWAP]	
__/dev/sda2	8:2	0	461.8G	0	part /	
/dev/sdb	8:16	0	149.1G	0	disk	
__/dev/sdb1	8:17	0	4G	0	part [SWAP]	
__/dev/sdb2	8:18	0	145G	0	part /disk-laptop	
/dev/sdc	8:32	0	2.7T	0	disk	
__/dev/sdc1	8:33	0	2.7T	0	part /disk-data	
/dev/sr0	11:0	1	1024M	0	rom	

```
> lsblk --paths --topology
```

NAME	ALIGNMENT	MIN-IO	OPT-IO	PHY-SEC	LOG-SEC	ROTA	SCHED	RQ-SIZE	RA	WSAME
sda	0	512	0	512	512	1	cfq	128 128	0B	
__sda1	0	512	0	512	512	1	cfq	128 128	0B	
__sda2	0	512	0	512	512	1	cfq	128 128	0B	
sdb	0	512	0	512	512	1	cfq	128 128	0B	
__sdb1	0	512	0	512	512	1	cfq	128 128	0B	
__sdb2	0	512	0	512	512	1	cfq	128 128	0B	
sdc	0	4096	0	4096	512	1	cfq	128 128	0B	
__sdc1	0	4096	0	4096	512	1	cfq	128 128	0B	
sr0	0	512	0	512	512	1	cfq	128 128	0B	

There are many possible optional arguments, but the most typically used are `--paths`, which produces full device paths, and `--topology`, if you are interested in internal details, such as physical sector size, I/O scheduler name and so on. You can get owner, group and permissions information with `--perm`, as shown below (and, if you really want detailed information, try `--output-all`, which will list about 50 columns' worth of data):

```
> lsblk --perm
```

NAME	SIZE	OWNER	GROUP	MODE
sda	465.8G	root	disk	brw-rw----
__sda1	4G	root	disk	brw-rw----
__sda2	461.8G	root	disk	brw-rw----
sdb	149.1G	root	disk	brw-rw----
__sdb1	4G	root	disk	brw-rw----
__sdb2	145G	root	disk	brw-rw----
sd	2.7T	root	disk	brw-rw----
__sd1	2.7T	root	disk	brw-rw----
sr0	1024M	root	cdrom	brw-rw----

For SCSI devices, you can add `--scsi` to `lsblk`, but there's also the more specific `lsscsi` command. The basic information it produces is shown below, and it includes all available SCSI devices. In my case, it shows the three hard disks and the optical reader I already found with `lsblk`, plus three card readers. Note

that you also get more information on specific brands and models. For example, I have two Western Digital hard drives (WD5000AAKS and WD30EZR), plus a Maxtor laptop drive (STM316021) and a Sony AD-7200S DVD unit:

```
> lsscsi
```

[2:0:0:0]	disk	ATA	WDC	WD5000AAKS-0	1D05	/dev/sda
[2:0:1:0]	disk	ATA	MAXTOR	STM316021 D		/dev/sdb
[3:0:0:0]	disk	ATA	WDC	WD30EZR-00M	0A80	/dev/sdc
[3:0:1:0]	cd/dvd	SONY	DVD	RW AD-7200S	1.61	/dev/sr0
[4:0:0:0]	disk	Sony	Card_R/W		-CF 1.11	/dev/sdd
[4:0:0:1]	disk	Sony	Card_R/W		-SD 1.11	/dev/sde
[4:0:0:2]	disk	Sony	Card_R/W		-MS 1.11	/dev/sdf

Check out all the possibilities of this command with `lsscsi --help`. You'll see that you really can dig down into SCSI devices with it. And, if you're interested, this command works by scanning the `/sys` filesystem (see Resources, and the DIY with `/proc` and `/sys` sidebar for more information).

Now, let's move on to some other commands. `lsusb` provides information on all USB-connected devices; see Listing 2 for an example. (An alternative is `usb-devices`, but it's somewhat more obscure in its output and has no configuration options.) As in most modern computers, you'll probably have a

lot of such devices. In my case, I have a Bluetooth dongle, Webcam, keyboard, mouse and more. You can get information on a specific bus or device with the `-s` option or select a given vendor with the `-d` option; for the latter, check the USB ID repository (see Resources) for vendor/device numbers. Finally, if you want very detailed information, try the `-v` (verbose) option, but be prepared to read a lot. For my machine, `lsusb -v` produces more than 1,300 lines of output.

Another command that can produce a ton of information is `lspci`, which shows all data on

**Listing 2. The `lsusb` command reports all USB-connected devices, as a list or in tree form.**

```
> lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation
    ↳2.0 root hub
Bus 005 Device 002: ID 054c:01bd Sony Corp. MRW62E
    ↳Multi-Card Reader/Writer
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1
    ↳root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1
    ↳root hub
Bus 003 Device 002: ID 0a12:0001 Cambridge Silicon
    ↳Radio, Ltd Bluetooth Dongle (HCI mode)
Bus 003 Device 006: ID 1e4e:0100 Cubeternet WebCam
Bus 003 Device 005: ID 046d:c317 Logitech, Inc.
    ↳Wave Corded Keyboard
Bus 003 Device 004: ID 04f3:0232 Elan
    ↳Microelectronics Corp. Mouse
Bus 003 Device 003: ID 05e3:0608 Genesys Logic,
    ↳Inc. Hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation
    ↳1.1 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation
    ↳1.1 root hub

> lsusb --tree
/: Bus 05.Port 1: Dev 1, Class=root_hub, Driver=uhci_hcd/2p, 12M
   |__ Port 2: Dev 2, If 0, Class=Mass Storage,
       ↳Driver=usb-storage, 12M
/: Bus 04.Port 1: Dev 1, Class=root_hub, Driver=uhci_hcd/2p, 12M
/: Bus 03.Port 1: Dev 1, Class=root_hub, Driver=uhci_hcd/2p, 12M
   |__ Port 1: Dev 3, If 0, Class=Hub, Driver=hub/4p, 12M
       |__ Port 1: Dev 4, If 0, Class=Human Interface Device,
           ↳Driver=usbhid, 1.5M
       |__ Port 2: Dev 5, If 0, Class=Human Interface Device,
           ↳Driver=usbhid, 1.5M
       |__ Port 2: Dev 5, If 1, Class=Human Interface Device,
           ↳Driver=usbhid, 1.5M
       |__ Port 3: Dev 6, If 0, Class=Video, Driver=uvcdvideo, 12M
       |__ Port 3: Dev 6, If 1, Class=Video, Driver=uvcdvideo, 12M
   |__ Port 2: Dev 2, If 0, Class=Wireless, Driver=btusb, 12M
   |__ Port 2: Dev 2, If 1, Class=Wireless, Driver=btusb, 12M
/: Bus 02.Port 1: Dev 1, Class=root_hub, Driver=uhci_hcd/2p, 12M
/: Bus 01.Port 1: Dev 1, Class=root_hub, Driver=ehci-pci/8p, 480M
```

PCI devices. And, as a matter of fact, this is the actual command I used to remember what kind of graphics card I had:

```
# lspci
00:00.0 Host bridge: Intel Corporation 4 Series
    ↳ Chipset DRAM Controller (rev 03)
00:01.0 PCI bridge: Intel Corporation 4 Series
    ↳ Chipset PCI Express Root Port (rev 03)
00:1b.0 Audio device: Intel Corporation NM10/ICH7
    ↳ Family High Definition Audio Controller (rev 01)
00:1c.0 PCI bridge: Intel Corporation NM10/ICH7
    ↳ Family PCI Express Port 1 (rev 01)
00:1c.1 PCI bridge: Intel Corporation NM10/ICH7
    ↳ Family PCI Express Port 2 (rev 01)
00:1d.0 USB controller: Intel Corporation NM10/ICH7
    ↳ Family USB UHCI Controller #1 (rev 01)
00:1d.1 USB controller: Intel Corporation NM10/ICH7
    ↳ Family USB UHCI Controller #2 (rev 01)
00:1d.2 USB controller: Intel Corporation NM10/ICH7
    ↳ Family USB UHCI Controller #3 (rev 01)
00:1d.3 USB controller: Intel Corporation NM10/ICH7
    ↳ Family USB UHCI Controller #4 (rev 01)
00:1d.7 USB controller: Intel Corporation NM10/ICH7
    ↳ Family USB2 EHCI Controller (rev 01)
00:1e.0 PCI bridge: Intel Corporation 82801 PCI
    ↳ Bridge (rev e1)
00:1f.0 ISA bridge: Intel Corporation 82801GB/GR
    ↳ (ICH7 Family) LPC Interface Bridge (rev 01)
00:1f.1 IDE interface: Intel Corporation 82801G (ICH7
    ↳ Family) IDE Controller (rev 01)
00:1f.2 IDE interface: Intel Corporation NM10/ICH7
    ↳ Family SATA Controller [IDE mode] (rev 01)
00:1f.3 SMBus: Intel Corporation NM10/ICH7 Family
```

```
    ↳ SMBus Controller (rev 01)
```

```
01:00.0 Ethernet controller: Qualcomm Atheros AR8152
```

```
    ↳ v2.0 Fast Ethernet (rev c1)
```

```
04:00.0 VGA compatible controller: NVIDIA Corporation
```

```
    ↳ GK107 [GeForce GT 740] (rev a1)
```

```
04:00.1 Audio device: NVIDIA Corporation GK107 HDMI
```

```
    ↳ Audio Controller (rev a1)
```

Try the `-v` or `-vv` options, for verbose and very verbose listings. To get full information on my (current) graphics card, I proceeded as shown in Listing 3. I now have an NVIDIA GeForce 740, and I'm using the nouveau kernel driver, among other internal details. Of course, to understand the produced information fully, you must have a bit of experience with PCI devices. Try the same command with `-vv`, and you'll see what I'm talking about.

If you are even more electronically/digitally minded, `lsdev` produces information about your installed hardware, including interrupts, ports, addresses and all such internal details. This command provides no options, and it's not likely you'll use it unless you are dealing very closely with hardware. Listing 4 shows an abbreviated example of the output. This command scans `/proc/interrupts`, `/proc/ioports` and `/proc/dma`, as described in the DIY with `/proc`

**Listing 3. The -v option provides more detailed information; -vv goes even deeper.**

```
# lspci -v -s 4:00.0
04:00.0 VGA compatible controller: NVIDIA Corporation
    ↳GK107 [GeForce GT 740] (rev a1)
    ↳(prog-if 00 [VGA controller])
        Subsystem: eVga.com. Corp. Device 2742
        Flags: bus master, fast devsel, latency 0, IRQ 27
        Memory at fd000000 (32-bit, non-prefetchable) [size=16M]
        Memory at e0000000 (64-bit, prefetchable) [size=256M]
        Memory at de000000 (64-bit, prefetchable) [size=32M]
        I/O ports at ec00 [size=128]
        [virtual] Expansion ROM at fe000000 [disabled] [size=512K]
        Capabilities: [60] Power Management version 3
        Capabilities: [68] MSI: Enable+ Count=1/1 Maskable- 64bit+
        Capabilities: [78] Express Endpoint, MSI 00
        Capabilities: [b4] Vendor Specific Information: Len=14
        Capabilities: [100] Virtual Channel
        Capabilities: [128] Power Budgeting
        Capabilities: [600] Vendor Specific Information: ID=0001
            ↳Rev=1 Len=024
        Capabilities: [900] #19
        Kernel driver in use: nouveau
        Kernel modules: nouveau
```

and /sys sidebar.

Finally, if you've made it this far, the `lshw` command is a sort of catch-all that can produce lots of information on all of your installed hardware. The `-short` option provides a (somewhat) abbreviated listing of everything in your box (see Listing 5, and note some interesting lines, "To Be Filled By

O.E.M.", which show that someone was careless when setting up my motherboard). With this command, you get information on the system, buses, memory, processor, display, network and everything else.

Notice the "class" column in Listing 5. You can get a hint of the full information that `lshw` can provide by using the `-class` parameter to limit



**Listing 4.** The `lsdev` command provides information on interrupts, ports and direct memory access.

```
> lsdev
Device          DMA   IRQ  I/O Ports
-----
                                7
0000:00:1d.0          c480-c49f
0000:00:1d.1          c800-c81f
0000:00:1d.2          c880-c89f
...
... (several lines snipped out)
...
eth0                  29
fpu                   00f0-00ff
gpio_ich              0480-04bf 04b0-04bf
i801_smbus            19 0400-041f
i8042                 1 12
iTCO_wdt              0830-0833 0830-0833 0860-087f 0860-087f
keyboard              0060-0060 0064-0064
...
... (several lines snipped out)
...
timer                 0
timer0                0040-0043
timer1                0050-0053
uhci_hcd              c480-c49f c800-c81f c880-c89f cc00-cc1f
uhci_hcd:usb2         23
uhci_hcd:usb3         19
uhci_hcd:usb4         18
uhci_hcd:usb5         16
vesafb                03c0-03df
```

output. For example, see below the detailed specs on my network card; it shows the vendor, model and plenty of other details (warning: this is the

kind of output you get if you don't restrict the command with `-short`; for my machine, `lshw` with no extra options produces a listing more than

## Listing 5. The `lshw` command includes information on all your hardware."

```
# lshw -short
H/W path          Device          Class          Description
=====
    ↳By O.E.M.
/0                  bus             G41M-VS3.
/0/0               memory          64KiB BIOS
/0/4               processor       Core 2 Quad (To Be
    ↳Filled By O.E.M.)
/0/4/5             memory          128KiB L1 cache
/0/4/6             memory          4MiB L2 cache
/0/d               memory          4GiB System Memory
/0/d/0             memory          4GiB DIMM SDRAM
    ↳Synchronous
/0/d/1             memory          DIMM [empty]
/0/100             bridge          4 Series Chipset
    ↳DRAM Controller
/0/100/1           bridge          4 Series Chipset
    ↳PCI Express Root Port
/0/100/1/0         display         GK107 [GeForce
    ↳GT 740]
/0/100/1/0.1       multimedia      GK107 HDMI Audio
    ↳Controller
/0/100/1b          multimedia      NM10/ICH7 Family
    ↳High Definition Audio Controller
/0/100/1c          bridge          NM10/ICH7 Family
    ↳PCI Express Port 1
/0/100/1c.1        bridge          NM10/ICH7 Family
    ↳PCI Express Port 2
/0/100/1c.1/0      eth0            network        AR8152 v2.0 Fast
    ↳Ethernet
/0/100/1d          bus             NM10/ICH7 Family
    ↳USB UHCI Controller #1
/0/100/1d/1        usb2            bus             UHCI Host Controller
/0/100/1d.1        bus             NM10/ICH7 Family
    ↳USB UHCI Controller #2
/0/100/1d.1/1      usb3            bus             UHCI Host Controller
/0/100/1d.1/1/1    bus             USB2.0 Hub
/0/100/1d.1/1/1/1  input           OM
/0/100/1d.1/1/1/2  input           USB Multimedia
    ↳Keyboard
/0/100/1d.1/1/1/3  multimedia      USB2.0 Camera
/0/100/1d.1/1/2    communication    Bluetooth Dongle
    ↳(HCI mode)

...several lines snipped out...

/0/1               scsi2            storage
/0/1/0.0.0         /dev/sda         disk           500GB WDC
    ↳WD5000AAKS-0
/0/1/0.0.0/1       /dev/sda1        volume         4102MiB Linux
    ↳swap volume
/0/1/0.0.0/2       /dev/sda2        volume         461GiB EXT4 volume
/0/1/0.1.0         /dev/sdb         disk           160GB MAXTOR
    ↳STM316021
/0/1/0.1.0/1       /dev/sdb1        volume         4094MiB Linux
    ↳swap volume
/0/1/0.1.0/2       /dev/sdb2        volume         145GiB EXT3 volume
/0/2               scsi3            storage
/0/2/0.0.0         /dev/sdc         disk           3TB WDC
    ↳WD30EZRX-00M
/0/2/0.0.0/1       /dev/sdc1        volume         2794GiB EXT4 volume
/0/2/0.1.0         /dev/cdrom       disk           DVD RW AD-7200S
```

500 lines long):

```
# lshw -class network
```

```
*-network
```

```
description: Ethernet interface
product: AR8152 v2.0 Fast Ethernet
vendor: Qualcomm Atheros
physical id: 0
bus info: pci@0000:01:00.0
logical name: eth0
version: c1
serial: bc:5f:f4:12:e0:f1
size: 100Mbit/s
capacity: 100Mbit/s
```

```
width: 64 bits
```

```
clock: 33MHz
```

```
capabilities: pm msi pciexpress vpd bus_master
```

```
↳cap_list ethernet physical tp 10bt 10bt-fd
```

```
↳100bt 100bt-fd autonegotiation
```

```
configuration: autonegotiation=on broadcast=yes
```

```
↳driver=atl1c driverversion=1.0.1.1-NAPI
```

```
↳duplex=full latency=0 link=yes multicast=yes
```

```
↳port=twisted pair speed=100Mbit/s
```

```
resources: irq:29 memory:fcfc0000-fcffffff
```

```
↳ioport:dc00(size=128)
```

The `lshw` command has several other interesting options. For example,

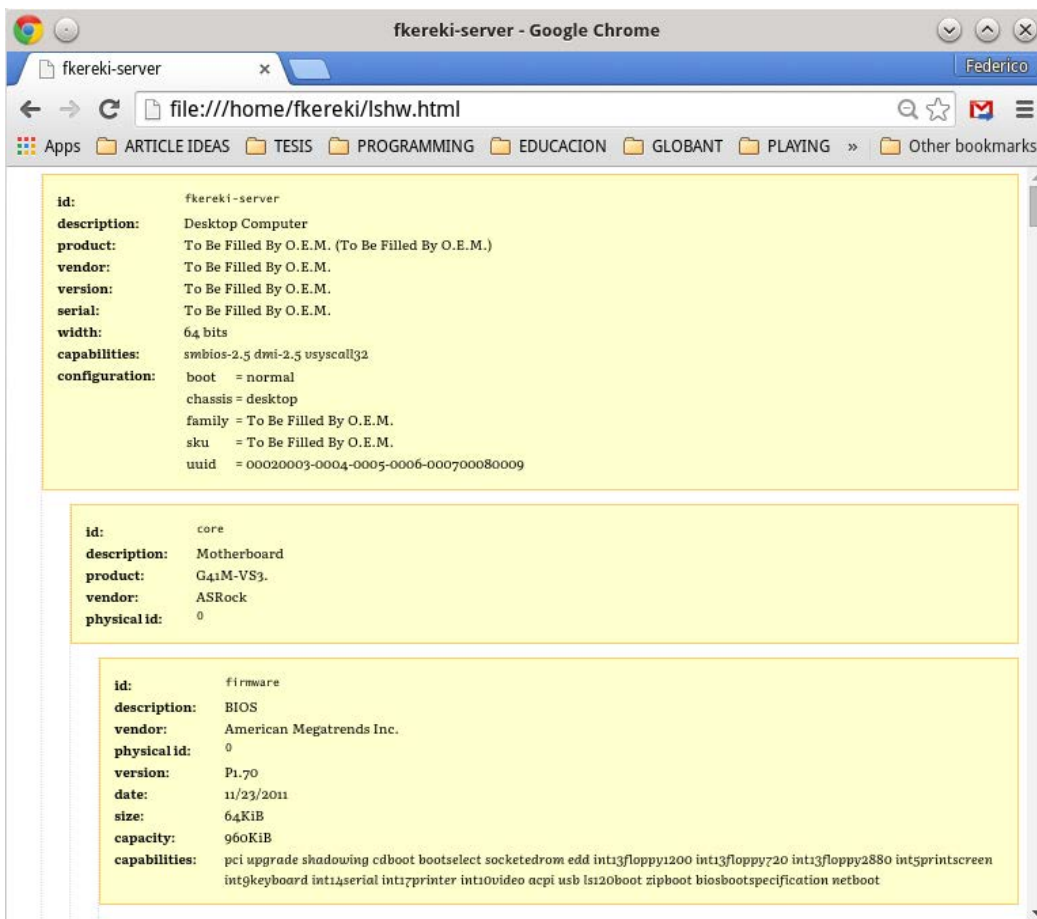


Figure 1. The `lshw` command also can produce HTML or XML output; the former is shown here.

it can produce either HTML or XML output (add the `-html` or `-xml` options); the former is appropriate for showing in a browser, while the latter is useful if you want to store or process your hardware information. See Figure 1 for just a small part of the full hardware description of my box. For security purposes, the `-sanitize` option removes sensitive information, such as serial numbers. There's even an `-X` option to use a graphical interface (I'll get to that later).

So far, I've discussed several `ls*` commands, and even if they are not actually a "family", they are my favorite tools. It's easy to remember them by typing `ls` and letting type-ahead suggest the rest. However, there are more command-line possibilities, so let's take a look.

## More Command-Line Options

Let's start with some general commands. The first, `dmidecode`, allows you to dump the computer's DMI (or SMBIOS; see the What's SMBIOS? sidebar) in a more readable format. If the table is found, its contents are dumped record by record, similar to this:

```
# dmidecode -t 6
# dmidecode 2.12
SMBIOS 2.5 present.

Handle 0x0009, DMI type 6, 12 bytes
Memory Module Information

    Socket Designation: DIMM0
    Bank Connections: 0 1
    Current Speed: Unknown
    Type: DIMM SDRAM
    Installed Size: 4096 MB (Double-bank Connection)
    Enabled Size: 4096 MB (Double-bank Connection)
    Error Status: OK
```

## What's SMBIOS?

How does Linux recognize what devices are installed? Since 1995, the SMBIOS (System Management BIOS) specification has provided this kind of information, doing away with the need for potentially worrisome operations like hardware probing. This standard (used by DMI) is geared to the Intel 32- and 64-bit processor architecture systems. Basically, it defines a structure with appropriate data for each kind of device, such as CPU, RAM, system slots and more. On principle, you could parse and decode this table by yourself, but several of the commands shown in this article already do that job. If you are curious about the specifics of the standard, see the Resources section.

Table 2. SMBIOS has several record types that you can select with `dmidecode`.

TYPE	DESCRIPTION
0	BIOS
1	System
2	Baseboard
3	Chassis
4	Processor
5	Memory Controller
6	Memory Module
7	Cache
8	Port Connector
9	System Slots
10	On-board Devices
11	OEM Strings
12	System Configuration Options
13	BIOS Language
14	Group Associations
15	System Event Log
16	Physical Memory Array
17	Memory Device
18	32-bit Memory Error
19	Memory Array Mapped Address
20	Memory Device Mapped Address
21	Built-in Pointing Device
22	Portable Battery
23	System Reset
24	Hardware Security
25	System Power Controls
26	Voltage Probe
27	Cooling Device
28	Temperature Probe
29	Electrical Current Probe
30	Out-of-band Remote Access
31	Boot Integrity Services
32	System Boot
33	64-bit Memory Error
34	Management Device
35	Management Device Component
36	Management Device Threshold Data
37	Memory Channel
38	IPMI Device
39	Power Supply
40	Additional Information
41	Onboard Devices Extended Information
42	Management Controller Host Interface
126	Disabled Entry
127	"End-of-Table" Special Marker
128-255	OEM-specific Data

If I were to give an award for "Most Talkative Command", surely it would go to `hwinfo`, another command that can dump all the hardware information on your computer.

Handle 0x000A, DMI type 6, 12 bytes

Memory Module Information

Socket Designation: DIMM1

Bank Connections: 4 5

Current Speed: Unknown

Type: DIMM SDRAM

Installed Size: Not Installed

Enabled Size: Not Installed

Error Status: OK

If you don't want to list the entire table (several hundred lines in my computer), you can restrict the output to a specific type of entry, according to SMBIOS definitions (Table 2).

You also can use specific keywords to restrict the output to a few types (Table 3).

If I were to give an award for "Most Talkative Command", surely

it would go to `hwinfo`, another command that can dump all the hardware information on your computer. On my machine, running `hwinfo` without any parameters produces more than 12,000 lines, including several memory dumps of the SMBIOS table. You can produce a much more compact version with the `--short` option (Listing 6).

**Table 3.** You also can use special keywords to get related information from SMBIOS.

SMBIOS Keyword	SMBIOS Types
bios	0,13
system	1,12,15,23,32
baseboard	2,10,41
chassis	3
processor	4
memory	5,6,16,17
cache	7
connector	8
slot	9

**Listing 6.** The `hwinfo` command can be quite talkative; using the `--short` option makes it more manageable.

```
# hwinfo --short
cpu:
    Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz, 2670 MHz
    Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz, 2336 MHz
    Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz, 2670 MHz
    Intel(R) Core(TM)2 Quad CPU Q8400 @ 2.66GHz, 2670 MHz
keyboard:
    Logitech USB Multimedia Keyboard
mouse:
    Elan Microelectronics OM
monitor:
    SAMSUNG SA300/SA350
    SAMSUNG S20B300
graphics card:
    nVidia VGA compatible controller
sound:
    Intel NM10/ICH7 Family High Definition Audio Controller
    nVidia GK107 HDMI Audio Controller
storage:
    Intel 82801G (ICH7 Family) IDE Controller
    Intel NM10/ICH7 Family SATA Controller [IDE mode]
network:
    eth0    Atheros AR8152 v2.0 Fast Ethernet
network interface:
    lo      Loopback network interface
    eth0    Ethernet network interface
disk:
    /dev/sda    WDC WD5000AAKS-0
    /dev/sdb    MAXTOR STM316021
    /dev/sdc    WDC WD30EZRX-00M

...etc (rest of the listing, snipped out)
```

You can restrict `hwinfo` to a specific type of hardware by adding an option, such as `--monitor` or `--printer`. Get the whole list of options with `hwinfo --help`. For instance, I can dump the optical unit data with `hwinfo --cdrom` (Listing 7). The `--listmd` option lets you include

RAID devices, which usually aren't included in the standard output.

Of the command-line programs I'm covering in this article, `inxi` is more colorful, even if only moderately (Figure 2).

If invoked with no parameters, it will just produce a line like the

**Listing 7. The `hwinfo` command can restrict its output to specific hardware, as the `cdrom` device, for example.**

```
# hwinfo --cdrom
25: SCSI 301.0: 10602 CD-ROM (DVD)
    [Created at block.249]
    Unique ID: KD9E.SGHalmfn+h9
    Parent ID: w7Y8.xyd+qedQTr5
    SysFS ID: /class/block/sr0
    SysFS BusID: 3:0:1:0
    SysFS Device Link: /devices/pci0000:00/0000:00:1f.2/
    ata4/host3/target3:0:1/3:0:1:0
    Hardware Class: cdrom
    Model: "SONY DVD RW AD-7200S"
    Vendor: "SONY"
    Device: "DVD RW AD-7200S"
    Revision: "1.61"
    Driver: "ata_piix", "sr"
    Driver Modules: "ata_piix", "sr_mod"
    Device File: /dev/sr0 (/dev/sg3)
    Device Files: /dev/sr0, /dev/cdrom, /dev/cdrw,
    ➔ /dev/disk/by-id/ata-Optiarc_DVD_RW_AD-7200S,
    ➔ /dev/disk/by-path/pci-0000:00:1f.2-ata-2.1,
    ➔ /dev/dvd, /dev/dvdrw
    Device Number: block 11:0 (char 21:3)
    Features: CD-R, CD-RW, DVD, DVD-R, DVD-RW, DVD-R DL,
    ➔ DVD+R, DVD+RW, DVD+R DL, DVD-RAM, MRW, MRW-W
    Drive status: no medium
    Config Status: cfg=no, avail=yes, need=no, active=unknown
    Attached to: #14 (IDE interface)
    Drive Speed: 48
```



```

fkereki : bash
File Edit View Bookmarks Settings Help
# inxi -v7 -%
System:   Host: fkereki-server Kernel: 4.1.5-1-desktop x86_64 (64 bit, gcc: 5.1.1)
Desktop: KDE 5 Distro: openSUSE 20150819 (x86_64) VERSION = 20150819 CODENAME = Tumbleweed # /etc/SuSE-release is deprecated
and will be removed in the future, use /etc/os-release instead
Machine:  Mobo: ASRock model: G41M-VS3 Bios: American Megatrends version: P1.70 date: 11/23/2011
CPU:       Quad core Intel Core2 Quad CPU Q8400 (-MCP-) cache: 2048 KB flags: (lm nx sse sse2 sse3 sse4_1 ssse3 vmx) bmips: 21362.7
Clock Speeds: 1: 2336.00 MHz 2: 2670.00 MHz 3: 2003.00 MHz 4: 2336.00 MHz
Graphics:  Card: NVIDIA GK107 [GeForce GT 740] bus-ID: 04:00.0
X.org: 1.17.2 drivers: (unloaded: fbdev,vboxvideo,vesa) tty size: 135x34 Advanced Data: N/A for root
Audio:      Card-1: NVIDIA GK107 HDMI Audio Controller driver: snd_hda_intel bus-ID: 04:00.1 Sound: ALSA ver: k4.1.5-1-desktop
Card-2: Intel NM10/ICH7 Family High Definition Audio Controller driver: snd_hda_intel bus-ID: 00:1b.0
Network:    Card: Qualcomm Atheros AR8152 v2.0 Fast Ethernet driver: atl1c ver: 1.0.1.1-NAPI port: dc00 bus-ID: 01:00.0
IF: eth0 state: up speed: 100 Mbps duplex: full mac: bc:5f:f4:12:e0:f1
WAN IP: 167.57.52.84 IF: eth0 ip: 192.168.1.200 ip-v6: N/A
Drives:     HDD Total Size: 3660.7GB (67.9% used) 1: /dev/sda WDC_WD5000AAKS 500.1GB
2: /dev/sdc WDC_WD30EZRX 3000.6GB 3: /dev/sdb MAXTOR_STM316021 160.0GB
Optical: /dev/sr0 model: N/A rev: N/A dev-links: cdrom,cdwr,dvd,dvdrw
Features: speed: 48x multisession: yes audio: yes dvd: yes rw: cd-r,cd-rw,dvd-r,dvd-ram state: N/A
Partition:  ID: / size: 455G used: 294G (65%) fs: ext4 dev: /dev/sda2
label: data_fk uuid: ca9ef4e1-cb19-4a11-91d3-fb1c8d02691d
ID: /disk-data size: 2.7T used: 1.9T (74%) fs: ext4 dev: /dev/sdc1
label: disk_3TB uuid: 61905806-b525-4a01-9af6-9239781a60aa
ID: /disk-laptop size: 143G used: 105G (77%) fs: ext3 dev: /dev/sdb2
label: laptop_150GB uuid: 2a95ecdb-3832-4eab-8ac5-4df88bae49ac
ID: swap-1 size: 4.30GB used: 1.26GB (29%) fs: swap dev: /dev/sda1
label: swap_fk uuid: 31be6b72-2c1e-41da-ab9d-d1c291333d40
ID: swap-2 size: 4.29GB used: 0.00GB (0%) fs: swap dev: /dev/sdb1
label: N/A uuid: b47096da-0a16-4a56-9b95-ef562847c101
Unmounted: ID: /dev/sr0 size: 1.07G label: N/A uuid: N/A
Sensors:    System Temperatures: cpu: 86.0C mobo: N/A gpu: 36.0
Fan Speeds (in rpm): cpu: N/A
Info:       Processes: 214 Uptime: 2 days 23:23 Memory: 2886.8/3949.4MB Runlevel: 5 Gcc sys: 5.2.1 alt: 4.8
Client: Shell inxi: 1.7.24
# █

```

Figure 2. inxi, even if only a command-line tool, at least tries to use some colors.

following, showing CPU, kernel, uptime and a few more details:

```

CPU~Quad core Intel Core2 Quad CPU Q8400 (-MCP-)
  ↳clocked at 2003.000 Mhz Kernel~4.1.5-1-desktop
  ↳x86_64 Up~2 days 23:24 Mem~2377.4/3949.4MB
  ↳HDD~3660.7GB(67.9% used) Procs~202 Client~Shell
  ↳inxi~1.7.24

```

However, you can use lots of options to get specific data. For example, you can set the verbosity level with options `-v0` (minimum) through `-v7` (maximum verbosity). The `-x` option allows including extra

information for some hardware. Check out `inxi -h` to get all possible options. For instance, you can get audio information with `inxi -A` or graphics card data with `inxi -G` and so on:

```

# inxi -A

Audio:      Card-1: NVIDIA GK107 HDMI Audio Controller
  ↳driver: snd_hda_intel Sound: ALSA ver: k4.1.5-1-desktop
          Card-2: Intel NM10/ICH7 Family High
  ↳Definition Audio Controller driver: snd_hda_intel

```

Now, let's finish with some GUI options.

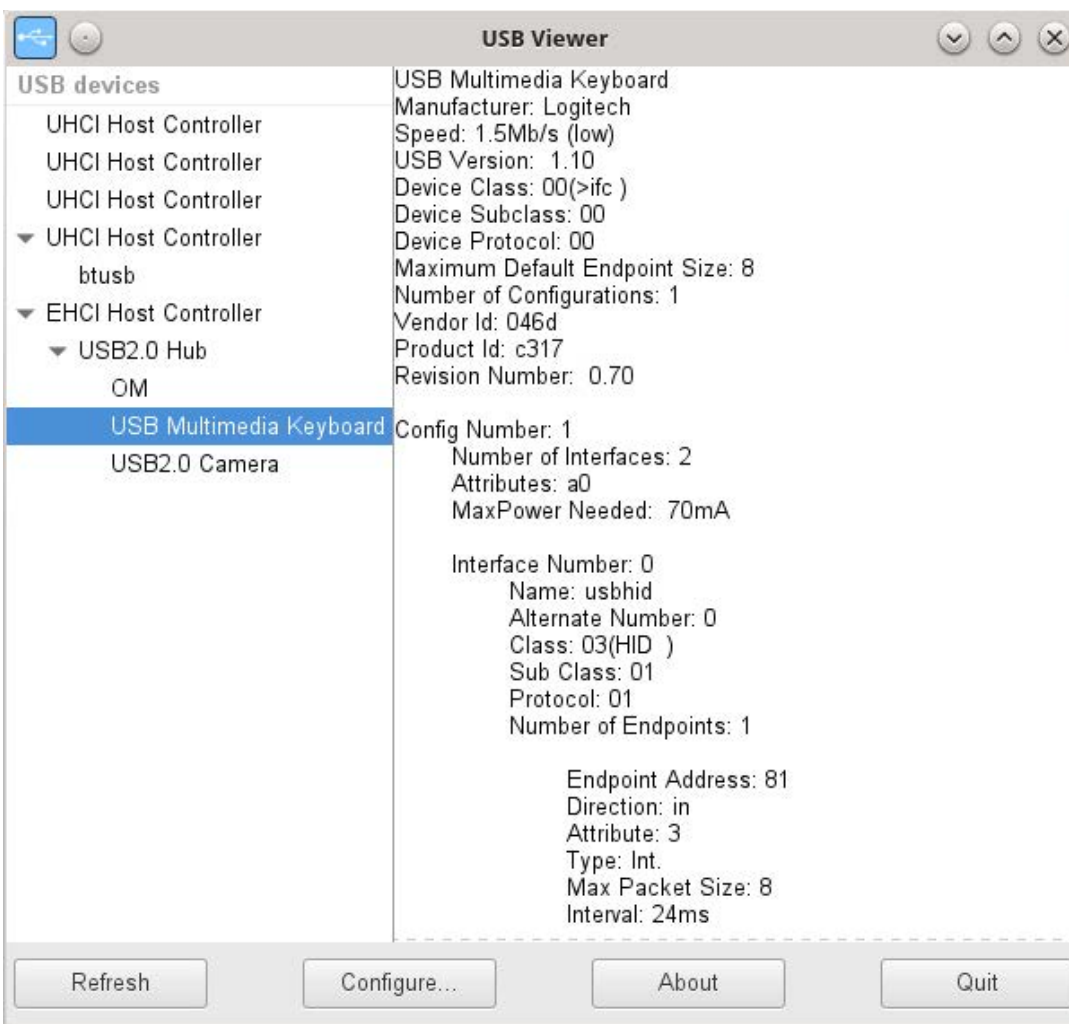


## The GUI Way

To start with, `usbview` is a rough graphic equivalent of `lsusb` or `usb-devices`, which I discussed earlier. It's quite simple to use, with no options or parameters. It shows two columns: the left one is a tree of all available USB devices, and the right one gives the full details. Figure 3 shows details on my USB keyboard.

Let's move on to a command I already discussed, which shares

the display style: `lsusb -X`. Instead of producing a listing (as shown previously), the `-X` option produces a graphic interface with several columns on the left to let you choose what hardware to inspect. An area to the right shows the full hardware details for the chosen device. Figure 4 shows the result of analyzing my optical DVD reader/writer unit; the provided information includes other details, such as the logical unit name, its



**Figure 3.** The `usbview` command shows the details of all USB devices in tree form.

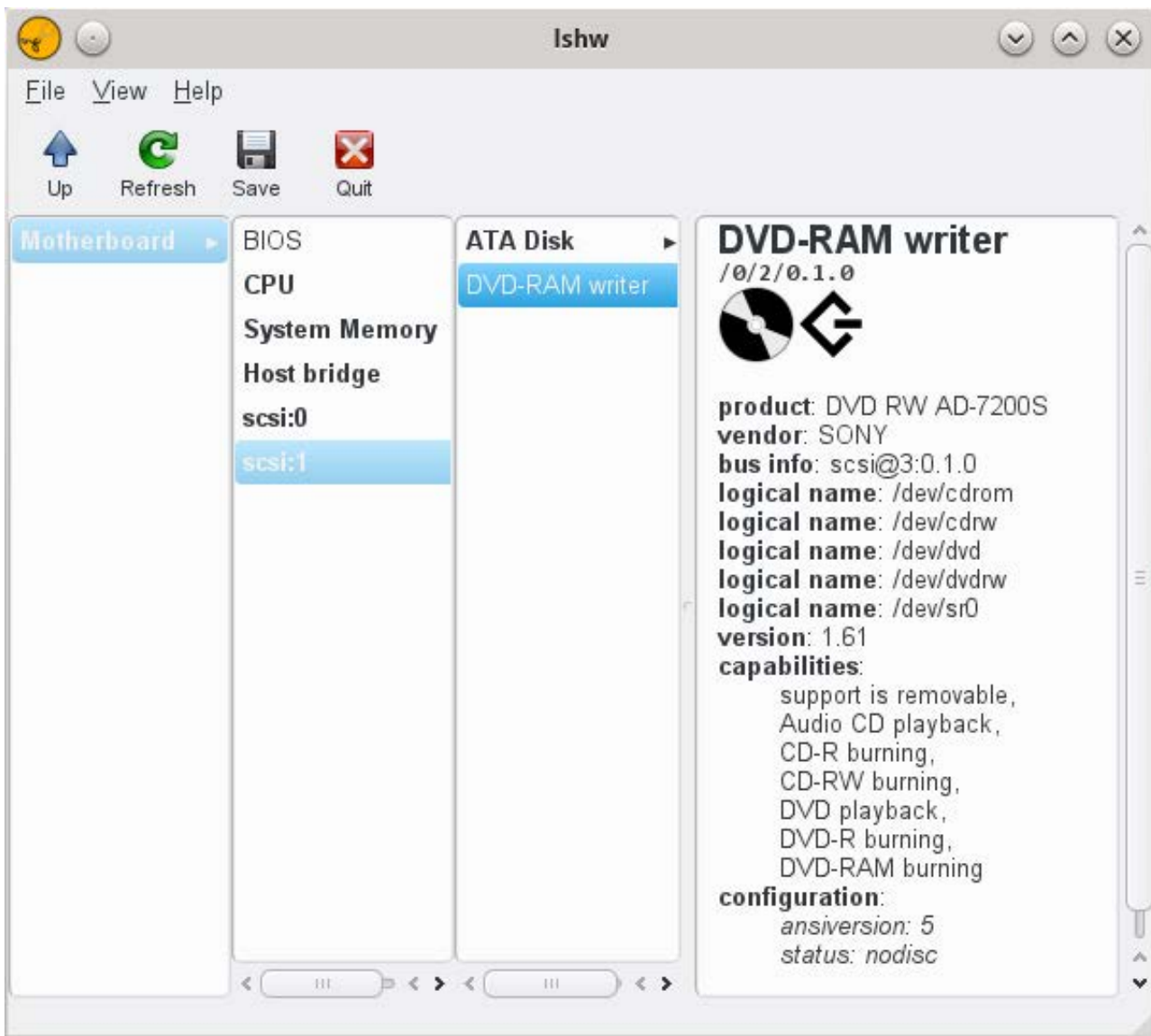
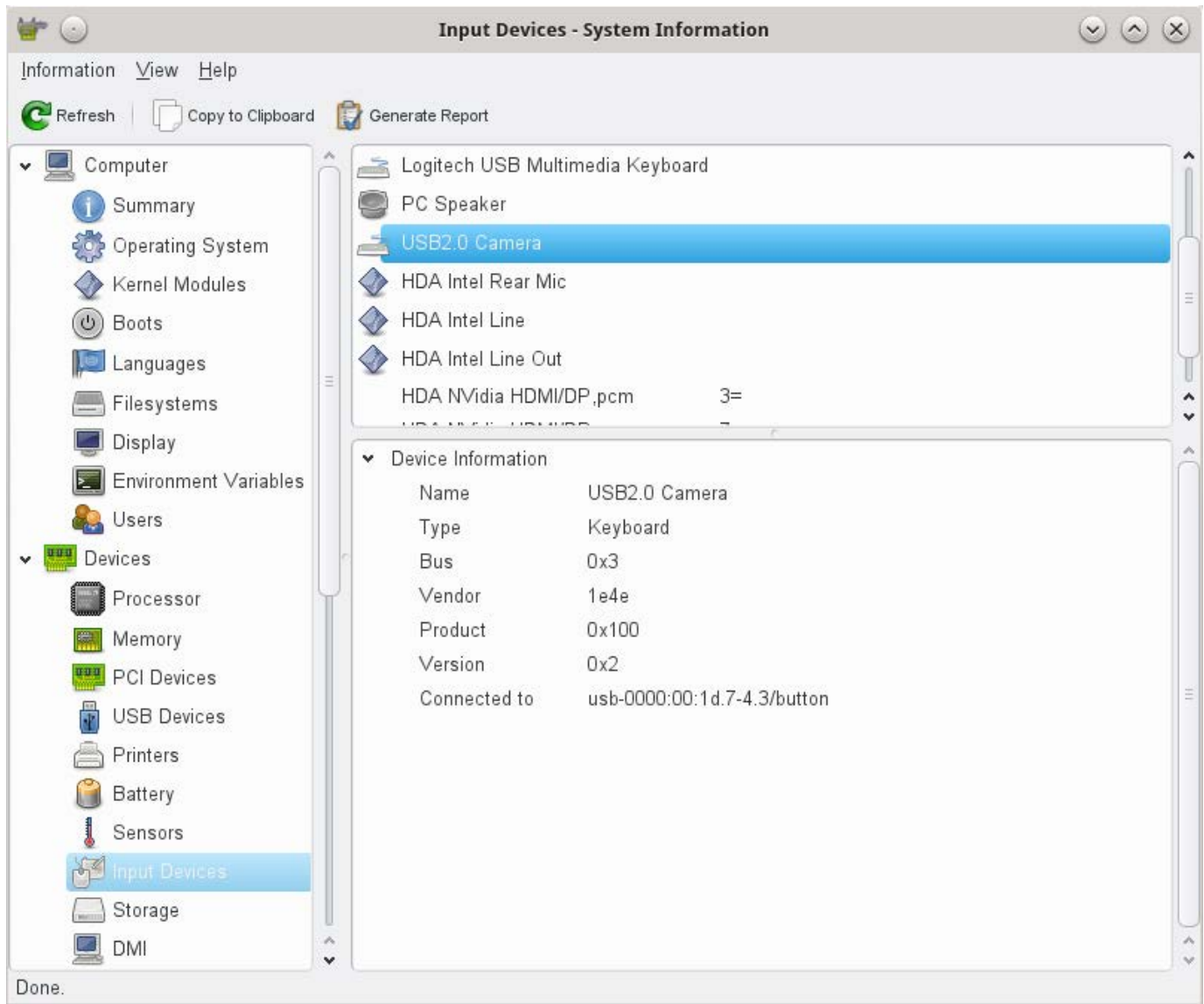


Figure 4. The `lshw -X` command produces a graphic interface that lets you browse all hardware devices.

capabilities and more.

Another interesting program is `hardinfo`, which “is not dead, but needs a maintainer”, according to its GitHub page (see Resources.) This program shows a tree structure to the left with four main branches:

- Computer shows lots of details about your machine: some are related to software and not to hardware.
- Devices includes all devices in your box, grouped by category.



**Figure 5.** The `hardinfo` command includes several extra pieces of data, not limited only to hardware.

- Network not only shows network card details, but also some other aspects, such as DNS servers or routing.
- Benchmarks lets you see how your machine fares against other

computers, but because of the lack of updates, the comparisons are against old CPUs.

Figure 5 shows sample output. There are two more options. The "Information" menu entry allows

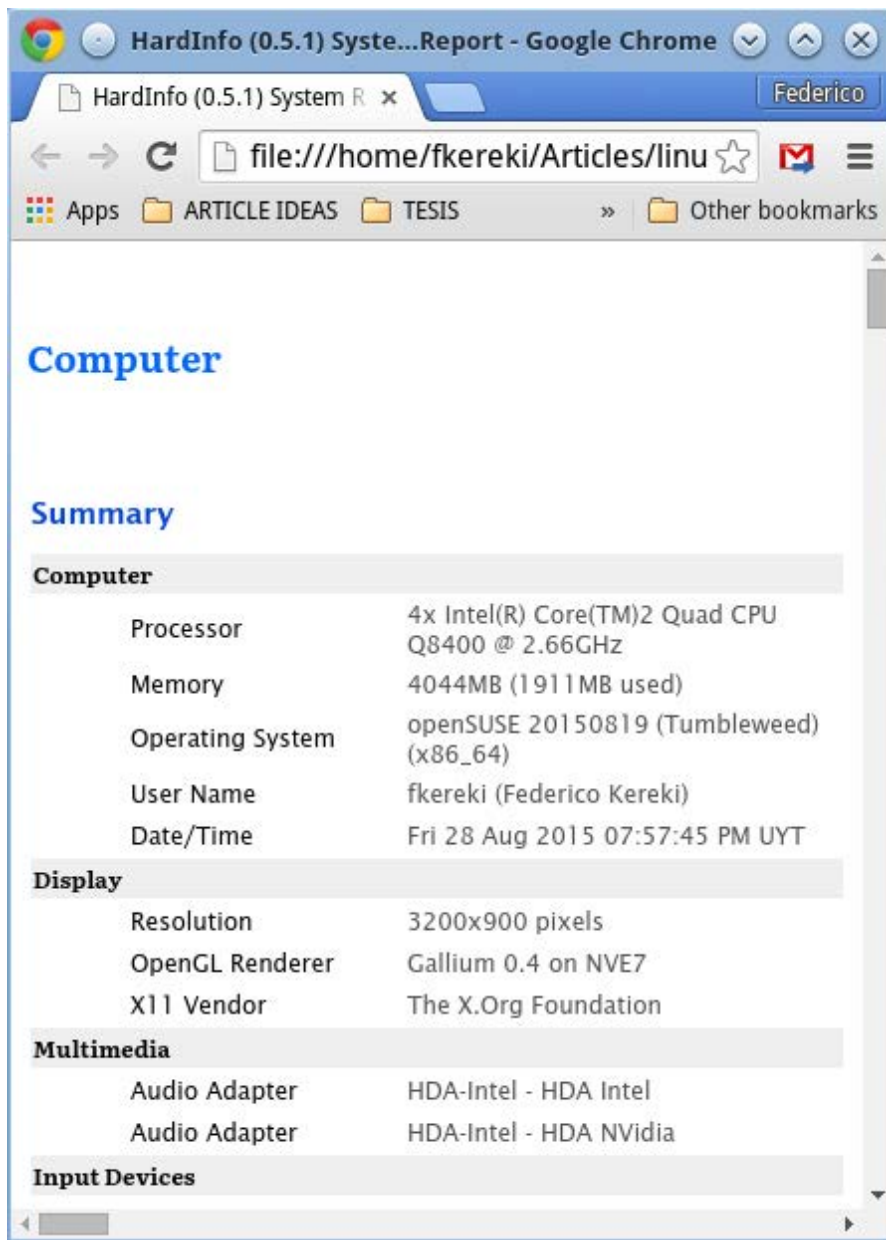
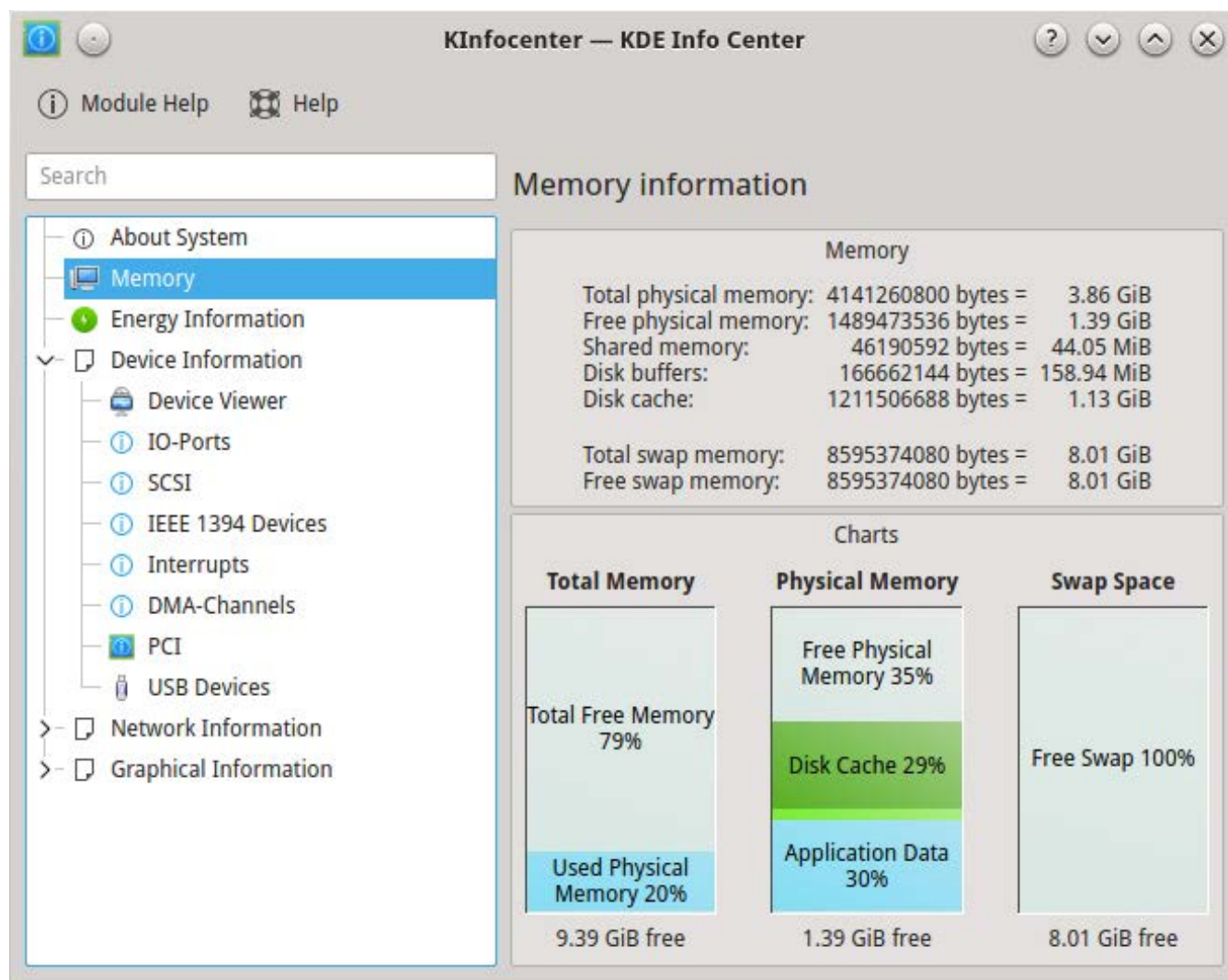


Figure 6. The `hardinfo` command can produce an HTML or text report describing your complete system.

you to produce a report, in either HTML or plain-text format, choosing whichever parts interest you. The “Network Updater” should let you update the internal program data, including more recent benchmark results, but when I tried to run it, I got a “Contacting HardInfo

Central Database (failed)” message. See Figure 6 for a example of the produced HTML report.

Let’s end with KDE’s own `kinfocenter`. This utility (see Figure 7, which shows RAM details for my machine) is similar to the previous tools I’ve been describing, and it



**Figure 7.** KDE's own kinfocenter shows not only hardware details, but plenty of other system data as well.

offers a left pane with a tree with all available options and a right pane with more details on the chosen option at the left.

The program doesn't restrict itself to hardware details, but shows all kinds of other information, such as "Samba Status", "Energy Information" or "X-Server", just to mention a few.

## Conclusion

I've covered a lot of commands that let you query your Linux machine and learn, in more or less detail, what's exactly in it. And if you need to, you even can get at the base data by yourself and whip up your own hardware-inspection tool. ■



## DIY with /proc and /sys

Linux is full of directories and files, but the /proc and /sys directories are really strange. They don't actually exist, but they allow you to browse them. They are full of zero-length empty files, but you can open and view them. The /proc directory preceded /sys, and it has basically all details about running processes (hence, the /proc name). Over time, more files were added to it, mostly "virtual" ones, which don't actually exist, but are created on the fly if you try to access them. (Most virtual files sport a current timestamp, which shows that they constantly are kept up to date and their contents are the latest possible.) The /sys directory is more modern. It appeared around the time of the 2.6 kernel to introduce more order and a better structure than provided by the older /proc, which had just grown in a sort of haphazard way. Many of the files (but not all) in /proc are duplicated in /sys, and whenever possible, you should pick the files in the latter directory. The /sys directory has several subdirectories:

- block/ has an entry pointing to each block device.
- bus/ has directories for each bus type, and within each, two subdirectories: devices/ and drivers/. The former has a directory for each device, pointing to the device's directory in /root, and the latter has a directory for each driver that was loaded for devices on the given bus.
- class/ has directories for each type of object; some examples are block/, graphics/, net/, sound/ and so on.
- dev/ provides directories for each type of device (for example, dev/block/ or

dev/char/), each with directories for each appropriate device.

- devices/ contains the global device hierarchy, with every physical device in your system.
- firmware/ includes directories for firmware-specific objects; for example, acpi/ or memmap/, but the particular directories in your own machine depend on the firmware drivers in your kernel.
- fs/ has a directory for each filesystem type in your machine, each with further directories for each specific device; for example, I have /sys/fs/ext4/sda2, because the disk mounted as /dev/sda2 uses ext4.
- kernel/ has several files related to the currently loaded kernel.
- module/ has a subdirectory for each and every module loaded into the kernel.
- power/ represents the power subsystem.

When you get to the deepest levels of any branch, you may find any number of individual files, which you can read to get attributes of the given object. What files? That's a hard question to answer, because it depends on which specific branch you are visiting, so you'll have to do a bit of work before you get to extract information from the /sys directory. (See Resources for some pointers about this.) Also, be aware that you can write to some of the files, and that will imply modifying the corresponding parameter—be warned: do this with care! However, if you keep at it, you'll be able to duplicate the functionality of most of the tools shown in this article, which often work the same way.

## Resources

Read about the SMBIOS standard at <http://www.dmtf.org/standards/smbios>. At the time of this writing, the latest version is 3.0.0, dated 2/15/2015.

You can find information on sysfs at <https://www.kernel.org/doc/Documentation/filesystems/sysfs.txt> and more specific documentation at <https://www.kernel.org/doc/Documentation/ABI/stable>.

Regarding the older proc fs, check <https://www.kernel.org/doc/Documentation/filesystems/proc.txt>.

The USB ID repository at <http://www.linux-usb.org/usb-ids.html> has the full list of all known IDs used in USB devices.

The PCI ID repository at <http://www.pcidatabase.com> provides a centralized list of PCI device IDs.

The `lscpu` and `lsblk` commands are part of the `util-linux` package, available at <https://www.kernel.org/pub/linux/utils/util-linux>. For documentation, check out <http://linux.die.net/man/1/lscpu> and <http://linux.die.net/man/8/lsblk>, respectively.

Read about `lsscsi` options at <http://sg.danny.cz/scsi/lsscsi.html> and find a manual page at <http://linux.die.net/man/8/lsscsi>.

For the `lsdev` man page, see <http://linux.die.net/man/8/lsdev>.

The lshw home page is at <http://www.ezix.org/project/wiki/HardwareLiStEr>, and its manual page is at <http://linux.die.net/man/1/lshw>.

See `lsusb` in the “usbutils” page at <https://github.com/gregkh/usbutils>, and get more information at <http://linux.die.net/man/8/lsusb>.

You can find `lspci` at <http://mj.ucw.cz/sw/pciutils> (home of the “PCI Utilities”) and the man page at <http://linux.die.net/man/8/lspci>.

Check out `usbview` at <http://www.kroah.com/linux/usb> and its man page at <http://linux.die.net/man/8/usbview>.

The hardinfo source repository is at <https://github.com/lpereira/hardinfo>, but first check your distribution's repositories; it's likely to already be there. Note that the program's last update was more than two years ago, and no further maintenance has been done.

You can find KInfoCenter at <https://www.kde.org/applications/system/kinfocenter>.

**Federico Kereki is a Uruguayan systems engineer with more than 25 years of experience doing consulting work, developing systems and teaching at universities. He is currently working as a UI Architect at Globant, using a good mixture of development frameworks, programming tools and operating systems—and FLOSS, whenever possible! He has written several articles on security, software development and other subjects for**

*Linux Journal*, IBM developerWorks and other Web sites and publications. He also wrote the *Essential GWT* book. You can reach Federico at [fkereki@gmail.com](mailto:fkereki@gmail.com).

|||||

**Send comments or feedback via**  
**<http://www.linuxjournal.com/contact>**  
**or to [ljeditor@linuxjournal.com](mailto:ljeditor@linuxjournal.com).**