How To: List PCI Information From The Terminal (lspci)

Continuing with a theme, this article will explain how to use the terminal to view what's attached via PCI (Peripheral Component Interconnect) to your computer. PCI devices are the ones in the add-on slots in your computer. They're typically internal devices, though you can get USB powered external devices with PCI-e slots, they kind of defeat the purpose. For this article, we'll be using lscpi.

You may also be interested in reading up about lshw and lscpu. I didn't intend for this to be a 'series' but that's what it's starting to look like. That's nice, however. They're small bites that let you sample the buffet that is the Linux terminal.

This one is just a little more complicated than the last. For starters, you may not have lscpi immediately available. You may have it installed by default, but you might not. It really depends on your distro. Either way, it's in your default repositories.

If you don't have it installed, it's in the package 'pciutils'. So, use your package manager to install it. For example, with apt you'd do this:

[code]sudo apt install pciutils[/code]

Once you have it installed, you'll see the manual defines 'lspci' as:

> *lspci – list all PCI devices*

If you don't know, PCI devices are a class of devices that are add-ons to your motherboard. They're mostly the devices that go in the slots, such as graphics cards or sound cards. They

can be used for all sorts of things, these days even being used for M.2 devices that hold SSD drives for rapid storage. They're great because they have a fast bus speed, meaning that your computer can interact with them faster because data moves faster in both directions than it does for, for example, a USB device.

Except PCI devices aren't just limited to the things that go in slots. Your motherboard probably uses that same spec to interact with other devices. For example, your Ethernet and sound card may be listed – even though they're 'on-board' and not actually add-on cards. That's your motherboard using the same sort of bus spec but not actually using a physical port. Additionally, there are different types of PCI specifications, but we don't need to get into that today.

And, well, once again the 'lspci' name tells you what it does, now that you know that's what it does. It lists PCI devices, just as the name implies. There are a couple of ways that you'd realistically want to use it.

NOTE: The results from 'lspci' are drawn from the The PCI ID Repository and may actually not be accurate. Yup. You could get inaccurate results from this command, but we throw it around daily as though it's infallible. And now you know…

Where were we? Oh, yeah… We were going to use 'lscpi' for something useful. So, let's crack open that terminal by using your keyboard to press CTRL + ALT + T and we'll first enter:

[code]lspci[/code]

That will list all your PCI devices in a quick and easy to read list. You may also want to get the verbose output and that's done like:

[code]lspci -v[/code]

The output of that command should be fairly obvious. After

all, it does what it says it does on the tin. That's also perhaps the more useful of the commands. It's definitely the one that I use most frequently.

If you take the numbers from the start of each line from the output of the above command and use the -t switch you'll actually get an understandable 'tree' output that will help you further understand what's going on inside your case without opening it up. If one PCI device has multiple entries (as many do) it'll make that easier to understand. It's simply:

[code]lspci -t[/code]

However you can easily put the two of those together and simply get a great verbose tree output with:

[code]lspci -vt[/code]

That's plenty easy to understand but some folks may find it a bit overwhelming. I don't usually need that much information, so I tend to run the command without any switches. As I bounce between devices, it's enough to just check and make sure I know what I'm working with.

NOTE: Older versions required *-vvv* for verbose and *-tree* were needed to perform those operations. The current versions simply use the *-v* and *-t* switches.

And there you have it. Yet another way to view hardware information from within the terminal. You may have noticed a trend and probably can narrow down your guess as to what the next article is gonna be about! If lscpi's parent package is not installed, it's really easy to get and you can then run the command. If you don't have access to many tools, you almost certainly have access to this one.

Like always, thanks for reading. Be sure to sign up for the newsletter. I'll only send you site-related material and won't

sell your email address to anyone. I promise, I won't ever send you any spam – just site stuff!

How To: List CPU Information From The Terminal (lscpu)

In the last article I wrote, I explained how to use 'lshw' to get information about your hardware from the terminal. This article will show you how to gather CPU information from the terminal.

This will be yet another quick article, but again this is a valuable tool to do hardware diagnostics when you don't have other tools available.

Just like last time, the command is self-evident once you really look at it. The command is 'lscpu' and it does what you'd expect – 'list CPU information'. It is defined as thus in the manual:

> *lscpu – display information about the CPU architecture*

So, let's give it a shot. Crack open your default terminal emulator by pressing CTRL + ALT + T. And, let's just start by entering:

[code]lscpu[/code]

And, there you have some fairly easy to understand. It's often used as a quick way to tell if your CPU supports 64 bit instructions. You'll see something like this if it does:

[code]Architecture: x86_64[/code]

NOTE: Pretty much all modern computer hardware supports the 64 bit instruction set, but there are still some 32 bit machines out there and being used.

Unlike the ‘lshw’ command, this one doesn’t need to be run as an administrator. It runs as a regular user just fine and running it as an admin doesn’t get you any more information.

Also unlike the ‘lshw’ command, there isn’t any other way to run it that’s all that interesting. You can read the man page, but you’ll seldom need to use this in any other way other than the way described here. Just type the command, get the information, and move on with whatever it was you were doing!

See? I told you this one would be another quick and easy article. Thanks for reading and don’t be scared of signing up to the newsletter. It’s not like you’ll be inundated with piles of unwanted email! You’ll just get notified when there’s a new article and I promise to not sell your email address to anyone.

How To: List Hardware From The Terminal (lshw)

You may not always have inxi available. The person trying to help you may want more specific info. You may need more information about your hardware. Who knows? There’s all sorts of reasons to use ‘lshw’ in your day-to-day computing.

Today is another short and easy article (you’re welcome). It’s just a very brief command, one of several, that we’re going to

learn how to use. The command in question is ‘lshw‘. The manual helpfully defines it as:

> *lshw – list hardware*

Which, now that you look at it, makes perfectly good sense. This is one of those Linux commands that’s not at all cryptic when you think about it. It looks like what it does, it does what you’d intuitively think it does. (It’s actually a subset of the info you get with ‘hwinfo’, but that’s not important right now!)

What is important is that there are two realistic ways to use it – and both of them should be run as an administrator. The first is just as you’d expect:

[code]sudo lshw[/code]

That will output a ton of information about the hardware you have in (or connected to) your computer. It’s rather overwhelming and it’s not often that you’ll be asked to post all of it. You may be asked to (or want to) run it with the -C option, such as:

[code]sudo lshw -C cpu[/code]

You may even be asked to use grep with it, such as:

[code]sudo lshw | grep wireless[/code]

Then, there’s one more way to run the program (under normal circumstances) and that’s to run the whole thing and to get the whole output in a shorter format. The command is probably just like you’d expect.

[code]sudo lshw -short[/code]

Lo and behold! Would you look at that! Ha! Isn’t that fantastic? You actually get readable output that’s suitable for your normal user who just wants to know what sorta

hardware they’re working with while getting a few extra bits of info! It even includes juicy nuggets like:

[code]/1 wlxe4beed0e5f5c network Wireless interface[/code]

Now, if you want to refine it even further, why not try this:

[code]sudo lshw -short | grep network[/code]

And, there you have it… You have another way to see your hardware in the terminal and it should be consistent across any major distro you touch. You shouldn’t have any trouble using the command and it’s easy to remember when you want to ‘list the hardware’. There are other uses, but those two are the most common. For more information about the lshw command try:

[code]man lshw[/code]

or

[code]info lshw[/code]

Like always, thanks for reading. Feel free to subscribe to the newsletter. I promise to not give away your private information – and you never know what sorta incentives I may stick in there. See? I told you this one would be short and easy!

How To: Properly Install Proprietary Drivers in Ubuntu

There is some confusion about installing the proprietary drivers in Ubuntu. This article hopes to clear that up by

telling you how to properly install drivers in Ubuntu.

First, this only works for the drivers that Ubuntu has access to. In this case, it's usually things like graphics cards, sound cards, some networking gear, and things like that.

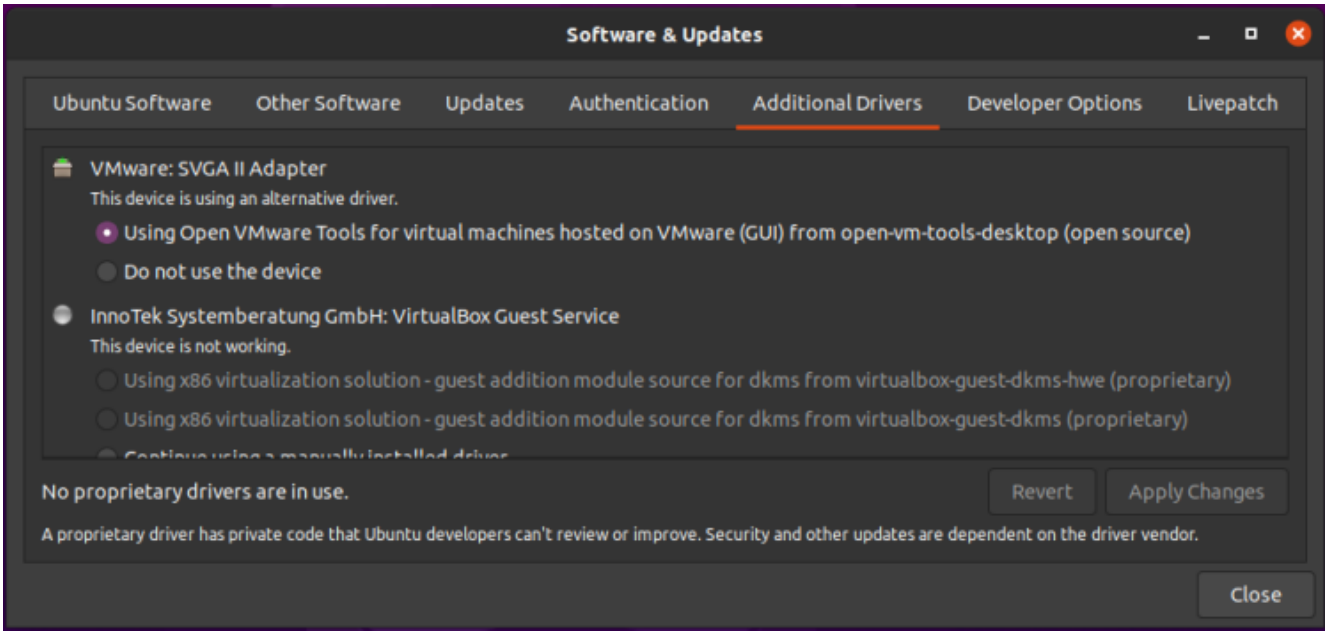
Ubuntu does not have all the possible drivers. If you have to go get them from GitHub and compile them yourself, this obviously isn't the article for you.

One of the most confusing is the Nvidia video card drivers. If you use Ubuntu, an official flavor of Ubuntu, or a derivative of Ubuntu, **DO NOT** download the .run file from Nvidia's site. While it may work, it will quite possibly not work with dkms and you will have to spend significant time fixing it every time the kernel is updated. It quite likely lead to breakage.

Yes, this means having some patience. But, have some patience because the drivers *will* make it down to the repos and will have then been tested. The drivers you get from the official repos will not only update, they'll update with the rest of the system **AND** they'll work properly with the kernel updates. When the kernel updates, the system itself will insert the appropriate drivers calls.

Doing this any other way will quite likely lead to hardship – and it's a hardship that you don't need to have. It's a hardship that's easily avoided. If you read the forums and question/answer sites, doing this the wrong way results in at least one question *almost every single day*.

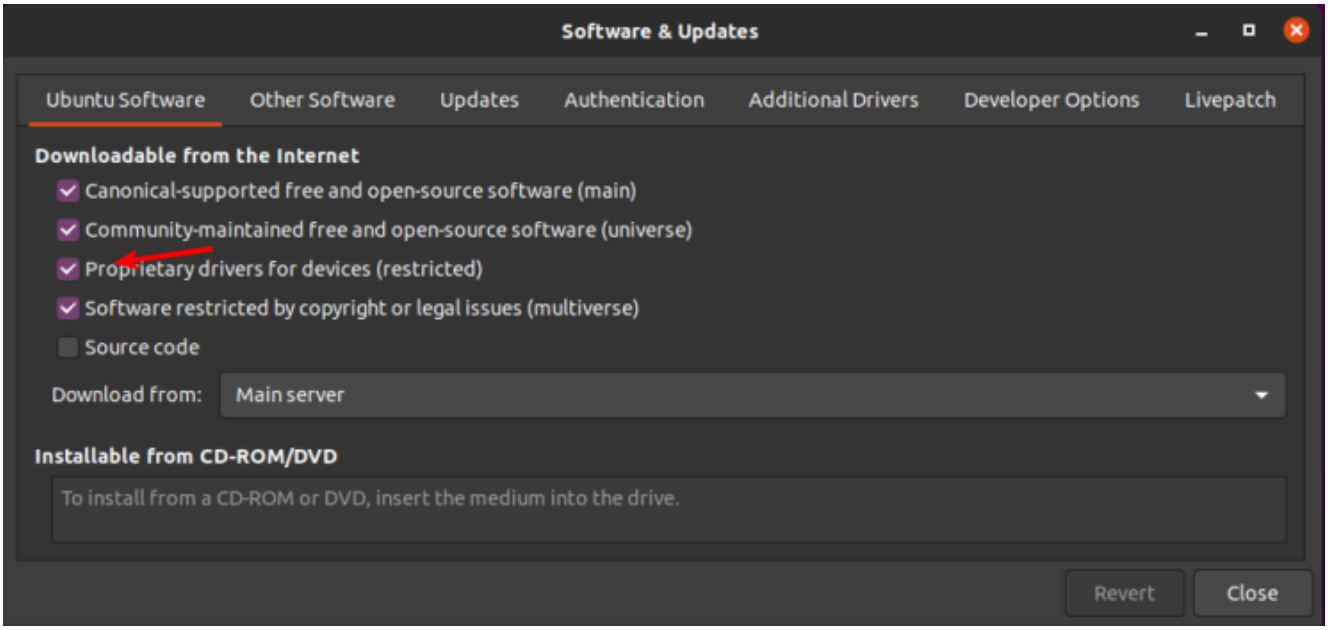
The first way is easy. It's even in the GUI. Search your menu for 'Additional Drivers' or similar (it may be only listed as "Software & Updates" depending on your Ubuntu flavor). It looks like this:



This should be self-explanatory, so I will just leave this here.

See? Pretty easy. Just pick what you want, apply the changes, and reboot.

NOTE: You will need to have the 'restricted' repository enabled in order to do this. That should be assumed, but some of you may not know this. So, a quick screenshot should make this even easier to figure out – it's in the same app as the above screenshot, but it's in the first tab. It looks a little like this:



The arrow should make it clear. That repo needs to be enabled

for this.

The second way is to use the terminal. Let’s go ahead and get the terminal opened by pressing CTRL + ALT + T on your keyboard.

Now, let’s check and see what drivers we can automatically install from the terminal:

[code]ubuntu-drivers devices[/code]

The output should let you know what drivers are available for your devices. Again, this is pretty self-explanatory. You really don’t even need to enter that command, you can do it all automatically and get the recommended drivers automatically installed. It’s easy. Just run this:

[code]sudo ubuntu-drivers autoinstall[/code]

That will go through and install the recommended drivers for your devices automatically. That will install the proprietary drivers, the ones with binary blobs and decidedly not opensource drivers. If those are the drivers you want, that’s the easiest way to install them.

Simply run the command, reboot, and you’re done. Not only are you done, you shouldn’t have to mess with them again – ever again. They will update automatically, they will automatically be applied when you update the kernel, and they will *generally* just work.

For the time being, we’re ignoring the idea of using the opensource drivers. I’ll simply say that I do quite well without needing proprietary drivers. I hardly ever bother installing them – unless I absolutely need a feature that’s not offered with the opensource drivers. I find that it’s still a working operating system and I can still easily meet my goals. You do you and you make the decision, but at least do it the right way after making that decision.

As always, thanks for reading. There’s a newsletter that will email you when a new article is published. That’s all it does. If you’re wanting to keep up with the site, that’s exactly how you do it! Well, there are push notifications available for those that prefer that. So, you do have choices! Either way, I won’t send you any spam. I promise!