How To: Enable SSH

In today’s article, we’ll learn about enabling SSH. SSH is a useful tool for remotely managing your Linux computer. This is a pretty simple, painless, and quick exercise.

Let’s say your computer is in another room, another state, another country, or on another continent entirely. How are you going to manage it? After all, we have servers across the world and it is not even remotely economical to send a person to administer each one of them in person.

On a home-use note, it’s perfectly suitable to manage my own router using SSH. It’s quick, easy, lightweight, effective, doesn’t require an attached monitor, and more. What’s not to like?

Installing SSH on Linux

My homemade router doesn’t have a keyboard attached. It doesn’t have a monitor attached, so it’s not like I can just easily walk over and deal with it.

I just got a new computer, a lovely laptop that I got for a fantastic price. I got it to test Lubuntu. I don’t always want to have to go over to the device and physically use it.

I have a dedicated server in Las Vegas. I live in Maine. It wouldn’t be practical to fly out to Vegas every few days to run updates on the server. It wouldn’t make financial sense to go out there every time the server needed to be rebooted.

These are all great candidates for SSH. SSH stands for “Secure Shell” and it’s a protocol. It’s not an application, and you can use SSH for all those things. A great many applications can communicate over the SSH protocol, including every terminal emulator that I’m familiar with.

The man page defines SSH as:

> *ssh – OpenSSH remote login client*

It has been around since the mid-90s and does nifty things beyond allowing you to control the remote system with commands, it also allows you to transfer files with things like SCP and SFTP. It's right full of nifty features and you might as well become familiar with it. I'll possibly cover both of those in a future article.

It's very, very easy to get going. Simple use your keyboard to open the terminal, by pressing CTRL + ALT + T.

Now, simply type (this one can be easily adjusted to other package managers):

[code]sudo apt install openssh-server[/code]

There… You're done. Well, you are more or less done. You now effectively have SSH running. It starts itself after you install it. That's pretty handy!

Now, make sure you're on the same subnet, and you can connect to the remote computer – the computer where you just installed and enabled SSH. You can do that in a couple of ways. You can do it like this:

[code]ssh remote_user@ip.a.d.d.r[/code]

You can switch the obvious for the obvious, but you will need to know the IP address for the remote computer. That seems a bit tedious, so let's just skip that part. Rather than remembering the IP address (which may change), just remember the name of the device.

So, instead, you'd run:

[code]ssh remote_user@host_name.local[/code]

If that doesn’t make sense, this is how I’d connect to the new laptop:

[code]ssh kgiii@kgiii-msi.local[/code]

Obviously, the hostname is ‘kgiii-msi’ and your hostname will be different. It’ll be the name you gave the computer during the installation process, typically during the same phase where you generated your user account. If you don’t actually know your hostname, you can easily find it. It’s simply:

[code]hostname[/code]

There’s more to this and we’ll likely cover that soon enough, but that’ll get you started. If you have a firewall installed and enabled, you may need to let SSH through if you want to use it. This is such a simple thing that I’d be remiss in my duties to not make folks aware of how simple it is.

And, with that, I thank you my dear reader for taking the time out of your day to humor this old fool. Your feedback is appreciated and keep signing up to that whole newsletter thing. Being old, I tend to forget to submit and share these articles elsewhere. Signing up means you have no excuses for missing an article!

How To: Use ‘find’ to List .iso files in a Directory.

In this article, we’re going to do a real world exercise involving the ‘find’ command, because you really shouldn’t process the output of the ‘ls’ command.

Using The Linux Find Command:

The conventional wisdom is that processing the ‘ls’ command’s output is a Bad Idea®. Don’t take my word for it, read this link (which will save me the time of repeating it). Seriously, read it. It’s well worth the read.

If you absolutely refuse to read the above link, then the reason you shouldn’t parse the output from ‘ls’ is because there are file names that will screw up the output something fierce and you won’t get accurate output. On the other hand, if you take care with how you name your files then it really shouldn’t be much of an issue.

So, how did we get here?

Winter is ebbing and it was not a very spectacular season this year. As winter winds down, it means that it’s time for Spring Cleaning. I highly recommend folks Spring Clean their digital lives as well as their real lives. Come on now, you and I both know that you have files that can easily be deleted and save you some space. In the days of large-capacity storage, we’ve become digital hoarders.

Once in a while you should muck out the stalls. Which is, coincidentally, what I was doing when I decided that this would make a good article. I had close to 300 .iso files, most of which were for varied Linux distros and most of which were no longer current. Hell, some of them were for distros that don’t even exist anymore. (Let’s just say that it had been a while since I cleaned out that directory.)

So, I went to town and cleaned out about 200 files. Some of those files had outlasted entire hard drives, being backed up, backed up again, and backed up some more – migrating like herds of reindeer across the steppes just above the taiga. Unlike the reindeer, these files had no significant value to me and it pained me exactly none to delete them.

Now, all the files are more or less sensibly named. I could easily have just used the ‘ls’ command, but it’s just bad form to do so. For example, I could have used:

[code]ls -la | grep iso[/code]

I could have output them to a handy text file for storage with:

[code]ls -la | grep iso >> linux_images.txt[/code]

I did not. No. No, I did not. Instead, I behaved myself and used the ‘find‘ command. Using ‘find’ is a bit more complex, but it’s worth learning how to use it.

The find command is a holdover from Unix. It first showed up in 1976 with Version 5. It was designed along with, and meant to be used in unison with ‘cpio‘, which oddly didn’t appear in Unix until Version 7. The cpio was all about archiving and actually still exists though you’ve probably never used it.

Anyhow, ‘find’ is defined as thus:

> *find – search for files in a directory hierarchy*

That bit about a directory hierarchy? That’s actually a bit of a sticking point with the ‘find’ command. It insists you include the directory in the command. You can’t just open the terminal, navigate to the desired directory, and run the command. You might just as well stay right there in your home directory and run the command right there! Going anywhere ain’t gonna help!

I wanted to list the .iso files in the directory and so I ended up with this command:

[code]find /media/kgiii/elements4/Distros -iname “*.iso”[/code]

Yes, yes I do have 4 WD Elements external drives connected to

the PC where I do a bunch of my work. Don't you judge me! It's 2021! I'll do what I want!

Anyhow, of all the modifiers to that command, the important one for this exercise is the '-iname'. That is pretty handy and it means you don't have to worry about case sensitivity.

> *-iname*
> *Like -name, but the match is case insensitive. For example, the patterns `fo*' and `F??' match the file names `Foo', `FOO', `foo', `fOo', etc. The pattern `*foo*` will also match a file called '.foobar'*

The rest is self-explanatory. I obviously want anything that ends with .iso. The wildcard '*' indicates that I want any characters in front of the .iso to be included for the purposes of making my list.

NOTE: As you can see, I used quotes around the text I wanted to be found. That's mandatory. You have to quote 'em, else 'find' doesn't do its thing.

So, what can you do with this? You can count them, if you'd like. You would have learned that back when we were talking about 'awk' and 'putting it all together'. You just need to use a pipe and then 'wc -l'. So, that command would look like:

[code]find /media/kgiii/elements4/Distros -iname "*.iso" | wc -l[/code]

(If you're curious, I'm down to just 98 .iso files in that directory. That's a pretty impressive cleaning job, if I do say so myself!)

And, what else can you do with it? Anything you want! You tell me what else you can do with it. How else can you apply this? How do you use the 'find' command? How am I supposed to know what else you do with it?!? You get to decide how you use this information. Seriously, leave a comment letting me know how

you can use the ‘find’ command in your daily computing.

Like always, thanks for reading. This article is a bit longer than it really needs to be. Still, you can sign up for the newsletter and get silly articles like this at least every other day. I write ’em ahead of time and use the scheduling feature that WordPress has. I’m not sure why, but this seems to make writing articles on a schedule easier. Maybe it takes the pressure off because it gives me at least two days to write an article? Dunno, I ain’t that kinda doctor!

Let’s Play With ‘apt-cache’ Some More!

People use apt, and apt-get, all the time. We use it for the most basic things. We use apt to install, remove, and purge software. However, there’s apt-cache and it’s pretty handy.

A couple of days ago, I published an article about using apt-cache to find the official project homepages of your installed software. Today, I’m going to quickly cover some of the other things you can do with apt-cache. Obviously, if you don’t have apt and apt-cache then this article will do you exactly no good.

So, where to begin? Let’s just assume you’ve already installed inxi and you know how to open the terminal by pressing CTRL + ALT + T on your keyboard.

Done? Good, let’s get started!

First, if you want to display a bunch of generic information, you can use the following:

[code]apt-cache show inxi[/code]

That will show you a bunch of information about a package. You don’t even have to have it installed. For this one, you will have to have the complete package name for it to be successful. In the next command, that’s not really required.

[code]apt-cache search inxi[/code]

For example, you could type in ‘inx’ and it will find inxi, among other things. You can use that command with the ‘–full’ switch, and get a ton of information, like so:

[code]apt-cache search –full inxi[/code]

Anyhow, you don’t even have to use an application name with the search. You can search for keywords and find applications that way. If you wanted to see what text editors you might have available (you’ll need to weed through them carefully) then you’d use this command:

[code]apt-cache search text editor[/code]

Go ahead and give it a try. You might be surprised at the vast number of results you’ll get with that command. Seriously, it’s a lot of results. There’s probably some text editors you’ve never heard of before hidden among those results!

Next on the list is checking the policy. This way you can see what version is installed, what version is available, and you can even see what repository was the source for the application. It’s just as easy as the rest.

[code]apt-cache policy inxi[/code]

Among this giant, perhaps overwhelming, source of data are a couple of other neat things you can do. You can easily see both the dependencies and the reverse dependencies.

For clarity sake, the dependencies are the extra software that

needs to be installed for the package in question to function. The reverse dependencies are what packages require the installation of the package in question in order to be fully functional.

To find the dependencies:

[code]apt-cache depends inxi[/code]

And the reverse dependencies:

[code]apt-cache rdepends inxi[/code]

And, there you have it. Those are the most common ways you're going to use apt-cache. There are other ways and there is more information available, but those are pretty much all the ways you can expect to use it in the normal course of activities. If you want to know more, you can always check the man page. To do that, it's just:

[code]man apt-cache[/code]

And, there's one bonus round! There's pretty much no good reason to run this, other than curiosity, but you can actually get some pretty cool stats about how many packages are available, how many are real packages, how many are virtual packages, and things like that. It's a pretty simple command.

[code]apt-cache stats[/code]

See? Another lovely way to use the terminal to gather information. I use the terminal *nearly exclusively* to manage my installed software.

How To: Use ‘apt-cache’ to Find Homepage for Your Installed Apps

It can come in pretty handy to know for certain the homepage for the applications you have installed. You can do this with ‘apt-cache’. I’ll show you how. This is a pretty easy article to follow and just another tool to add to your toolbox.

NOTE: This is only valid for systems that use apt. As the title indicates, it requires ‘apt-cache’. Without apt-cache, this page will do you no good. None good. That’s how much it will do you. None!

Why would you want to know the homepage – and, more so, the preferred homepage? For starters, in the days of GitHub and everyone forking, and awkward application names that aren’t easily searched for, it’s hard to know which site is the correct one.

Maybe you want to report a bug? Maybe you want to request a feature? Maybe you want to make a donation? Maybe you just want to thank the author for writing such awesome software? Maybe you want to know where the homepage is because you need support and you’re not sure where to turn to?

There are all sorts of reasons why you might want to know the homepage of a piece of software. It’s actually something that’s important. It’s also something your system already knows and will happily show you if you know the proper magical incantation.

Like many other articles, you’re gonna want the terminal for this. Let’s go ahead and get that opened by using your keyboard and pressing CTRL + ALT + T.

Got your terminal emulator open? Good! Let’s start with the command.

[code]apt-cache show inxi[/code]

If you do not have ‘inxi’ installed, feel free to use a different application. Note that you do not need to use sudo for this. Not all apt commands require sudo. You only need sudo when you’re actually doing administrative tasks. See? I saved you some typing!

Anyhow, in the text output from the above command you’ll see a line that starts with “Homepage:”. If you hadn’t already guessed it, that’s the line that tells you the authors homepage. This, of course, only works on installed applications. For sanity and space sake, it’s not like your system has all that information downloaded for all the possible packages. Thus, it works on naught but the apps you have already installed.

So, let’s go ahead and make the command a little more precise. We’ll pipe the output through grep and get rid of the cruft we don’t actually need. In that same terminal, go ahead and enter:

[code]apt-cache show inxi | grep Homepage[/code]

NOTE: The command contains a capitalized letter H because Linux is often case-sensitive and is certainly case-sensitive in this case. If you don’t believe me, try it with a lowercase h!

But wait, there’s more!

Not only is there homepage information in there, there’s sometimes some useful nuggets of information in there. If you have LibreOffice installed, go ahead and check (skip the pipe and grepping) to see what the output is. Inside, it has a ton of additional information, including listing ways that you can

extend LibreOffice by installing more software.

And there you have it. You can now easily find the homepage for the applications you have installed. Should you need to contact the author, check for information, or just see if they did anything else, you now know how to do that. It’s a little hidden nugget that most folks don’t seem to know. Well, now they do…

Yay! You made it all the way to the bottom. You deserve a treat. Seeing as you’ve already got the terminal open, and seeing as we’re dealing with apt-cache, let’s just get some pretty neat stats with:

[code]apt-cache stats[/code]

That’s it and thanks for reading. I appreciate the audience and am happy that I finally am putting some effort into this project. I’ve been meaning to do this for years, but something always got in the way. If you want to get notified when new articles are posted, just scroll up and sign up for the newsletter. I promise not to send you any commercial emails and I won’t give any of your private data away.

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!