How To: Enable Password-less SUDO.

This isn’t the smartest thing you can do. In fact, you probably shouldn’t do this. But, if you are comfortable with your physical security, you can use sudo without a password.

In my case, there’s not a whole lot folks are going to do with sudo on my computer. Anyone with physical access to my device is someone that I trust. I also run a ton of commands when hanging out in the support sites and I am frankly just tired of typing my password when I use sudo.

So, let’s get rid of it. Start by pressing CTRL + ALT + T, and then enter:

[code]sudo nano /etc/sudoers[/code]

Scroll down to the bottom and add this line:

[code]<your_username> ALL=(ALL) NOPASSWD:ALL[/code]

Where “<your_username>” substitute it with your actual username on your computer. Now save it with:

CTRL + X

Y

ENTER

See that? You also may have just learned how use ‘nano’ to edit and save a text file while in the terminal. Pretty neat, huh? Anyhow, scroll up a little and look to the right. There’s a spot where you can enter a name and email address. If you do that (and confirm the email address) then you’ll get handy notices in the email when there’s a new article. I promise, I won’t send you a single non-site related email – ever.

Fun With The ‘AWK’ Command

Today, I’ll show you a real-world use of the venerable AWK command. I use the AWK command fairly regularly.

First, a little about AWK. The AWK command has been around since 1977 and is named after the three people who wrote the original version: Alfred **A**ho, Peter **W**einberger, and Brian **K**ernighan. It was first officially released in Version 7 Unix (V7) and has been a valuable text processing program ever since.

This is just one instance of how I used it to process some text.

I was going through the statistics at the Music For Us site. Among the statistics is a list of countries from which MFU has had visitors. Given the age of the site, it’s a fairly long list.

The format of the text was as follows:

[code]1 United States United States 8,645
2 Germany Germany 1,044
3 China China 846
4 France France 672
5 Canada Canada 538[/code]

I only cared about the name of the countries, but it was a very long list (more than 100 entries) and I didn’t want to edit them out manually. So, it was a pretty easy decision to use AWK to process it.

First, I copied the text into a document called ‘countries.txt’. I then opened my terminal and headed to that

directory. This was the command I used:

[code]awk ‘{ print $2 }’ countries.txt > finished.txt[/code]

So, I basically told it to use AWK to process countries.txt, to extract the second column, and to write that data to a file called finished.txt. In this case, I could have easily extracted any of the columns without much difficulty. You simply need to change the $2 to whichever column you want.

If you’d like a full explanation of the command, you can actually check it out on one of my favorite sites, explainshell.com. (That link will take you directly to the explanation of this specific command.)

Anyhow, the output was as expected. It looks like:

[code]United
Germany
China
France
Canada[/code]

(If you want to see the full list, check out the paste here. By the way, you can feel free to use that site for your own paste needs. We have our own free pastebin service.)

I guess the final verdict is that Linux really lets me be lazy! Instead of processing that text manually, I used the AWK command and processed it all in less than a second. I didn’t have to sit there and copy/paste. I didn’t have to spend the time verifying it. I just had to run one simple command and I had the data I needed, nice and neat.

The thing is, I really only need the basics so I have giant holes in my knowledge – and yet, I know enough to do things like that, making my life easier. There’s always so much more to learn, and that’s awesome.

How To: Change Your Default Terminal

It’s actually pretty easy to change your default terminal emulator. This post will tell you how to change your terminal to the terminal emulator your prefer.

Your Linux distro came with a default terminal emulator. It may have even come with a couple of terminals installed. That doesn’t mean it’s the best choice for you, it just means that the developers chose to include it.

For example, you might like XFCE-terminal, or you may prefer Terminator. The choices are endless, and Wikipedia has a ton of them listed.

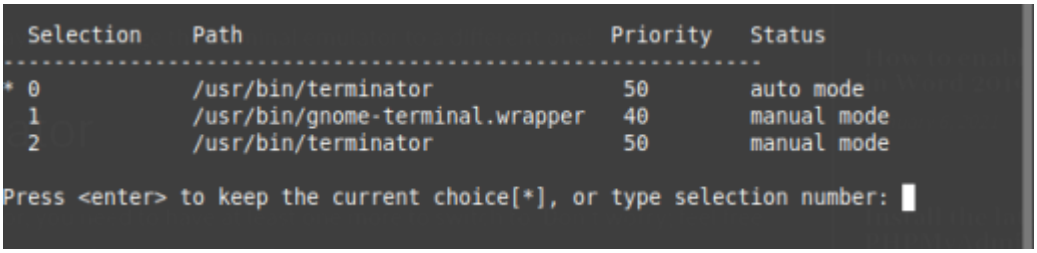
Generally, you can press CTRL ALT T and open your default terminal. If you’d like to change the behavior, first you need to install a new terminal. For example, if you use APT and you want to install Terminator:

[code]sudo apt install terminator[/code]

Now, let’s make it the default. Either open a terminal or use the existing open terminal and enter the following:

[code]sudo update-alternatives –config x-terminal-emulator[/code]

It should look a little like this:



Just pick the number of the terminal emulator you'd like to be the new default and press enter.

To test this, finish up the tasks above and then press CTRL ALT T (or the keyboard shortcuts your distro has set up) and it should now open to your new default.

When Did I Last Reboot My Linux Box?

For any number of reasons, including bragging rights, you may want to know when you last rebooted your Linux computers. It's actually pretty easy and I'll show you how.

Crack open your terminal emulator, CTRL + ALT + T will often do it, and enter the following:

[code]last reboot[/code]

What will tell you when you last rebooted, who booted (system), the kernel used, date and time, and how long the system was up for.

For example:

[code]reboot system boot 5.4.0-52-generic Fri Nov 6 19:22 – 18:25 (9+23:02)
reboot system boot 5.4.0-52-generic Mon Nov 2 12:24 – 19:22

(4+06:57)
reboot system boot 5.4.0-51-generic Wed Oct 21 17:23 – 19:22 (16+02:58)[/code]

If you want to see the three most recent boots, you could just use the ‘head’ command.

[code]$ last reboot | head -3
reboot system boot 5.4.0-58-generic Thu Dec 31 14:56 still running
reboot system boot 5.4.0-58-generic Thu Dec 31 09:32 – 14:55 (05:22)
reboot system boot 5.4.0-58-generic Sat Dec 26 01:33 – 09:32 (5+07:59)[/code]

You can even use grep to see how many times you rebooted in a month, for example:

[code]$ last reboot | grep Nov
reboot system boot 5.4.0-54-generic Mon Nov 30 14:28 – 14:20 (3+23:51)
reboot system boot 5.4.0-54-generic Sat Nov 28 15:24 – 14:13 (1+22:48)
reboot system boot 5.4.0-54-generic Fri Nov 27 15:41 – 15:10 (23:29)
reboot system boot 5.4.0-54-generic Wed Nov 25 16:44 – 15:14 (1+22:29)
reboot system boot 5.4.0-53-generic Wed Nov 18 17:03 – 16:19 (6+23:15)
reboot system boot 5.4.0-53-generic Tue Nov 17 18:26 – 16:55 (22:29)
reboot system boot 5.4.0-53-generic Mon Nov 16 19:09 – 17:11 (22:01)
reboot system boot 5.4.0-53-generic Mon Nov 16 18:25 – 18:25 (00:00)
reboot system boot 5.4.0-52-generic Fri Nov 6 19:22 – 18:25 (9+23:02)
reboot system boot 5.4.0-52-generic Mon Nov 2 12:24 – 19:22

(4+06:57)[/code]

And there you have it. It’s much easier than you might think.

Prevent Duplicates in Bash History

As you may have noticed, bash keeps a history. If you open your terminal and push the up arrow, you’ll see the last command you used. This fills up with duplicates. This is how you ensure it doesn’t keep duplicates.

It’s pretty easy. Just open your terminal and enter:

[code]echo ‘export HISTCONTROL=ignoreboth:erasedups’ >> ~/.bashrc[/code]

After that, bash should no longer keep duplicates of previously entered commands.

See? Linux is pretty simple.