



[Login](#)
[Register](#)
[Submit Article](#)

Try our NEW BETA Search!

Linux Kernel Module Management 101

Wednesday, 12 January 2011 08:01 | Joe 'Zonker' Brockmeier Exclusive
 7 Like 5 likes. [Sign Up](#) to see what your friends like.

The Linux kernel allows drivers and features to be compiled as modules rather than as part of the kernel itself. This means that users can often change features in the kernel or add drivers without recompiling, and that the Linux kernel doesn't have to carry a lot of unnecessary baggage. Want to learn how to manage your modules? It's easy to do, just keep reading.

In this tutorial, we'll walk through the steps of seeing what's already loaded in the running kernel, and adding and removing modules from the kernel.

What's Loaded?

The first utilities that you'll want to get to know are `lsmod` and `modinfo`. Open a terminal and run `lsmod`. Note that you won't need to use `sudo` or log in as root just to probe the modules in the system.

You'll see output like this when you use `lsmod`:

```
Module                Size Used by
parport_pc            18855  0
ppdev                 5030   0
lp                    7462   0
sco                    7209   2
parport               27954  3 parport_pc,ppdev,lp
bridge                39630   0
stp                   1440   1 bridge
bnep                   9427   2
```

This shows the modules that are loaded, their size, and whether they're being used by other modules. Take the `parport` module, for instance. It's being used by several other modules, but what are they? The `modinfo` utility will tell us — maybe.

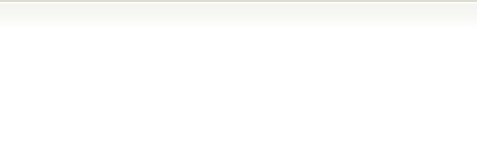
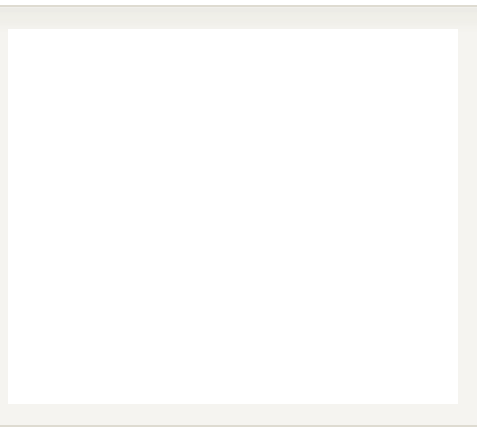
Run `modinfo parport`, and you'll see something like this:

```
filename:      /lib/modules/2.6.32-5-amd64/kernel/drivers/parport/parport.ko
license:      GPL
depends:
vermagic:     2.6.32-5-amd64 SMP mod_unload modversions
```






It tells us where the module is found, and its license, and that it has no dependencies. Unfortunately, we really don't know any more than where we started because this module author has chosen not to provide a description of the module. But many modules will have a description and provide some indication what they are used for. Since we got no joy from the `parport` module, what about trying to find out about one of the modules that depend on it? Let's try `modinfo parport_pc`. This, at least on my system, produces fairly hefty output, but the relevant part here is the description field which provides:

```
description: PC-style parallel port driver
```

So we can surmise that the `parport` driver has something to do with supporting a Parallel Port. In fact, the `parport` module is generic support for parallel ports, and `parport_pc` provides support for parallel ports on `x86/x86_64` systems. This isn't something you'll find with `modinfo`, unfortunately. But when all else fails,



Latest Tutorials

-  [How to Get the Most Out of Your Laptop with Linux](#)
-  [Weekend Project: Use Open Fonts on Your Web Site](#)
-  [Enabling SquirrelMail For Your Web Sites On An ISPConfig 3 Server \(Debian Lenny\)](#)
-  [Linux Kernel Module Management 101](#)
-  [Step by Step on Using Per-User Quotas with Linux](#)

Most Read News

-  [Making wireless work in Ubuntu](#)
-  [All about Linux swap space](#)
-  [Note to new Linux users: No antivirus needed](#)
-  [An introduction to services, runlevels, and rc.d scripts](#)
-  [SSD vs. SATA RAID: A performance benchmark](#)

Ads by Google

- [Linux](#)
- [Install RedHat Linux](#)
- [Linux Server Distribution](#)
- [How to Linux](#)

Latest Software News

check the kernel source under the [Documentation](#) directory.

Removing Modules

Modules can be removed using the `rmmod` utility. The usage is simple, just `rmmod modulename`. However, if we try to remove the `parport` module, we get this error:

```
ERROR: Module parport is in use by parport_pc,ppdev,lp
```

You can force module removal using `rmmod -f`, but that's not a good idea, usually. A better way to do it is to use `modprobe -r` which will automatically look to see what other modules depend on it, and unload those modules as well. If they're in use, then `modprobe` will refuse to remove them as well, unless you use the `-f` option with `modprobe` too.

Installing Modules

What if you have a module you want to load into the kernel? You can do that with `insmod` or `modprobe`.

The preferred method is `modprobe`, because it will also load any modules that the requested module depends on. For instance, if I didn't have the `parport` module loaded and went to load the `lp` or `parport_pc` modules, `modprobe` would go ahead and load `parport` as well.

To load a module using `modprobe` run `modprobe modulename`.

Blacklisting Modules

You may on occasion need to "blacklist" a module. Why would you need this feature? Sometimes a module will cause a conflict with another module, is superseded by another module, or is otherwise undesirable.

To blacklist a module, the easiest way to do it (there's usually more than one way to do things...) is to add the module to `/etc/modprobe.d/blacklist.conf`. For instance, on Debian systems the `evbug` module is automatically blacklisted because it's not something most users will need. To add a module to the blacklist, just add one line to the `blacklist.conf` file:

```
blacklist modulename
```

That's refreshingly straightforward, isn't it?

Summary

Most of the time, you'll only need to mess with kernel modules if your distribution doesn't support hardware out of the box or when you're working with third-party applications like VMware that supply kernel modules of their own.

But it's a good thing to know how to handle kernel modules when and if you need to add or remove them. Even if you don't have a need for them right now, spend some time testing the module tools now — you may find that they come in handy later. Speaking of "later," we've got more to cover. In the next installment, we'll look at compiling modules, module aliases, and much more.



Joe 'Zonker' Brockmeier

GURU

Joe 'Zonker' Brockmeier is a freelance writer and editor with more than 10 years covering IT. Formerly the openSUSE Community Manager for Novell, Brockmeier has written for Linux Magazine, Sys Admin, Linux Pro Magazine, IBM developerWorks, Linux.com, CIO.com, Linux Weekly News, ZDNet, and many other publications. Brockmeier is also a FLOSS advocate and participates in several projects, including GNOME as the PR team lead. You can reach Zonker at jzb@zonker.net and follow him on Twitter.

[Email this](#) [Share This](#) [Set as favorite](#)

Comments (1)

[Subscribe to this comment's feed](#)

[Show/hide comments](#)

Kernel Modules

written by Istimsak, January 12, 2011

Great article, you made working with the kernel less intimidating.



Write comment

[Show/hide comment form](#)

You must be logged in to post a comment. Please register if you do not have an account yet.



[New Features in Amarak 2.4](#)



[Firefox Beta Getting New Database Standard](#)



[Mozilla plots February Firefox 4 Release](#)



[Simple Scan Brings Much-Needed Sanity to SANE](#)



[Here Come the Open Source Rookies!](#)

Linux Jobs

[Sr. Systems Level Software Engineer-L...](#)
Cinci Engineering, Greenville, SC

[Linux System Administrator](#)
Uken Games, Toronto, ON, Canada (Downto...

[Sr. Linux Administrator 15085](#)
Confidential, New York, NY

[Linux System Administrator](#)
Confidential, New York, NY (us)

[Linux System Administrator](#)
Uken Games, Toronto, ON, Canada (Downto...

[More jobs](#) | [Post a job](#)

Powered by JobThread