
## Roll your own Linux with Slax

# NO SLACKER

With its novel package manager, Slax makes it simple to install new software and easy to build your own distributions. **BY MARTIN STREICHER**

Because the Linux kernel and all its attendant utilities are open source, you can combine the software any way you'd like. Indeed, many Linux distributions, Live CDs, and virtual appliances are available, each with its own bent or specialty.

Ideally, building a custom Linux distribution would be as simple as burning a CD. In practice, though, the process is more complex, typically requiring the expertise of a developer or system administrator.

Slax is a notable exception. Akin to shopping online, you can assemble and generate a custom Linux distribution with a few clicks of your mouse. Slax is so incredibly friendly and easy to maintain, you won't believe it.

First, let's look at Slax and its package manager, then let's handcraft a Linux distribution. The test machine is a virtual computer running in Parallels Desktop on Mac OS X. Nonetheless, you can achieve the same results with physical media and hardware.

## Introducing Slax

The standard Slax distribution comprises a core and a large collection of modules that can be added to the core. In this respect, Slax is similar to other Linux kits. For instance, Ubuntu Linux contains everything you need to get started, but it can be customized extensively with the Aptitude package manager. However, the similarities between Slax and other Linux distros end there because adding a Slax module is as simple as clicking an icon on a web page.

Figure 1 captures a portion of the Slax games catalog. Each of the four modules shown has an icon, a name and version number, a description, and three buttons: *download*, *add to build*, and *activate*. Clicking on *activate* enables a module in an instant.

To try Slax, go to the Slax homepage [1], download the ISO image, then burn a CD or load the ISO onto a virtual machine and boot. In a moment, you will see the Slax desktop.

To launch a terminal window, click on Konsole (the screen icon) in the status bar at the bottom. Next, click on the K Menu at the bottom left, launch the Konqueror browser, and point it at the Slax homepage. Now click on *modules* then *develop* in the next page, then switch to Konsole and type *python --version*.

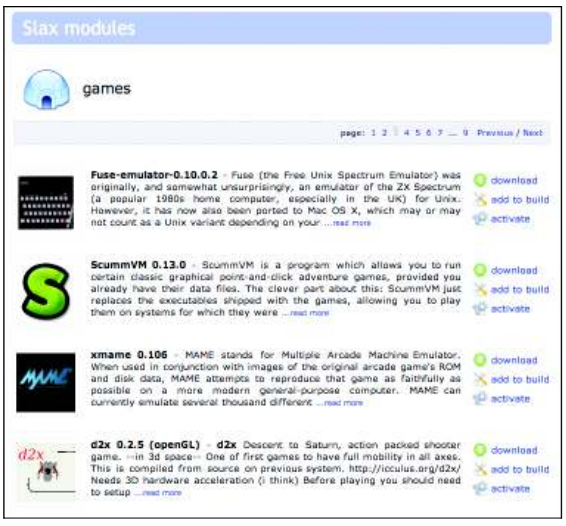


**Figure 1: A few of the games available in Slax modules.**

Because Python is not part of the Slax core, you get an error message. To fix that problem, go to your browser again, navigate the list of developer modules until you find Python, and click *activate*. A series of message balloons should pop up at the top left. After the balloons indicate success, return to the terminal window and retype *python --version*. This time the Python version number should be output to your screen.

Disabling a module is just as easy. To remove Python, click on the K Menu and choose *System | Slax Module Manager*. From there, select the Python module and click *Remove Selected Module*. If you try the *python* command as before to confirm the uninstall, the response is *No such file or directory*.

Interestingly, activating a module does not physically copy software to the local hard drive. Instead, it uses two file systems – httpfs and AUFS – to make it appear as if the module's files are local.

Httpfs mounts a remote ISO image on the local system, making its contents available immediately. Meanwhile, AUFS (short for Another Union FileSystem) unions one or more directories into a single, virtual directory. AUFS treats each directory as an overlay.

For instance, if directory */usr/bin* contains *bash* (the Bash Shell), directory */tmp/a* contains *usr/bin/zsh* (the Z Shell), and directory */tmp/b* contains *usr/bin/fish* (the Fish shell), the union of these three directories makes all three shells available from */usr/bin*. Thus, activation mounts a remote ISO image to a local directory and unions the contents of the remote directory with the contents of a local directory to make megabytes of software available in an instant.

To see the list of activated modules, turn again to the Slax Module Manager, which you can also use to browse the local filesystem and activate local modules. With web activation, whenever you shut down Slax, all activated modules

are deactivated and must be re-activated via the Internet at the next boot. Luckily, you can store a module locally and re-activate it every time or just some of the time, as you'll see next.

## Installing a Module

In addition to the web browser interface, Slax provides alternative techniques to activate a module. For example, recall that each module in the Slax catalog has a *download* button. Clicking this button downloads the module to the local filesystem (a destination you choose) and leaves it in its bundled form. Once a module is local, you can activate/deactivate manually, activate at every boot, or activate only when you need it.

To activate a local module manually, simply double-click its file from Konqueror or use the *activate* command.

For example, if you download the game Pix Frogger to *~/pixfrogger.lzm*, you can activate the module with a double-click via Konqueror or with the *activate* command:

```
# activate ~/pixfrogger.lzm
module file is stored inside the ➥
  union, moving to ➥
  /mnt/live/memory/modules first...
```

Now you can run *pixfrogger*, which you'll find (virtually) in */usr/bin*.

To deactivate the module, double-click the file */mnt/live/memory/modules/pixfrogger.lzm*, use the Module Manager, or use *deactivate*:

```
# deactivate /mnt/live/memory/➥
  modules/pixfrogger.lzm
```

Only the superuser, root, can activate and deactivate modules.

If you run Slax from writable media and want to activate a module at every boot, copy the module to directory */slax/modules*. Then, to keep pixfrogger available at all times, run:

```
# cp /mnt/live/memory/modules/➥
  pixfrogger.lzm /slax/modules
```

However, if want to activate a module selectively at boot time, copy it to */slax/optional* and name the module in the *boot* command. For instance, if *pixfrogger.lzm* is in */slax/optional/pixfrogger.lzm*, the command *boot slax load=pixfrogger* will activate Pix Frogger during boot.

Optionally, you can name entire subdirectories in the boot command. Assuming all of your games are in */slax/optional/games*, the command *boot slax load=games* would activate every module in the games subdirectory. A request for a nonexistent module or directory is silently ignored.

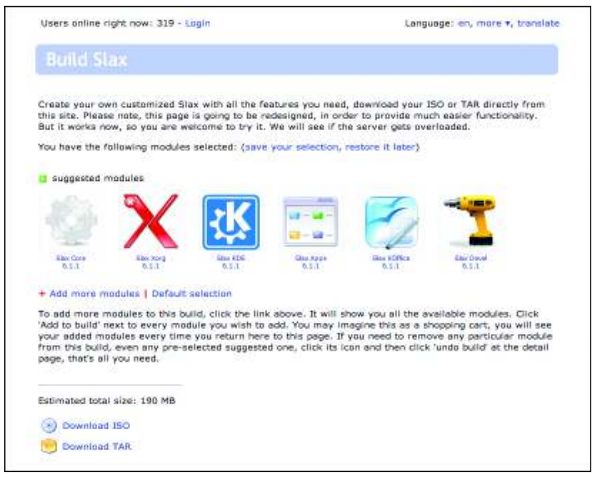


**Figure 2: Of the six default modules, only Core is mandatory.**

If you download a module and activate it, it is copied automatically to */mnt/live/memory/modules* – a virtual path for */slax/modules* – and activated after reboot.

## Building a Distribution

To build your own Slax-based distribution, you combine the core with all of the modules you want, then let Build Slax aggregate them and package it all into a single downloadable ISO.

To start, go to the Slax homepage and click *build slax*. By default, Build Slax starts each distribution with six modules, as shown in Figure 2. The first module, Core, is mandatory; the others – Xorg, KDE, Apps, KOffice, and Devel – can be removed at your discretion by mousing over each icon and clicking the red X that appears.

To add more modules, click on the *Add more modules* link and browse the catalog of all available modules. To add a module, click *Add to build*. To remove a module, click *Undo build*.

When finished, or to see your progress, simply return to the Build Slax page. Now the Python module appears under *verified modules*. The estimated size has increased from 190 to 195MB. Again, you can mouse over a module and click the red X to delete it.

When you are finished choosing modules, simply click the *Download ISO* button at the bottom of the window and save the file. Then, either burn a CD or load the ISO into a virtual machine and boot the image. When the desktop appears, run the *python --version* command to see that it is indeed present. Now you can disseminate your handiwork to Python coders everywhere.

Build Slax can save and restore your module selections, too. These features are useful if you want to bootstrap a new distribution from an existing one. Just above the list of modules are two links: one to *save your selection*, which keeps the file on your local hard drive, and one to *restore it later*, which produces a list in simple text that it uploads and interprets.

## Slax Not Slacking

Slax is the brainchild of Tomas Matejicek and is based on Slackware Linux. The current version of Slax is 6.1.1 (as of July 1, 2009), and maintenance releases are offered frequently to squash bugs and add features. Slax is available in 28 languages.

Slax can boot off a USB thumb drive (USB 2.0 provides superior performance) and can be installed on a hard drive. And because Slax is generally very small, it's usually easy to carve out a small partition on an existing Windows system, for example, and copy Slax manually to the new partition (a "frugal install"). Instructions for this dual-boot configuration can be found in the Slax forums.

The initial install of Slax provides a capable suite of desktop applications, including *Kopete* for instant messaging, *K3b* for burning media, *JuK* for music, and KPlayer for playing other media.

Of course, if you need different or additional software, you now know how to build your very own flavor of Linux. ■

### INFO

[1] Slax: *http://www.slax.org*