Tip 3: Use libraries (even if your sysadmin doesn't want you to)

Ben Klemens

3 October 2011

level: library user
purpose: use the Machine to rage against the Man

You may have noticed the caveats in the last entry about how you have to have root privileges to install to the usual locations on a POSIX system. But you may not have root access if you are using a shared computer at work, or you have an especially controlling significant other.

Then you have to go underground, and make your own private root directory.

The first step is to simply create the directory:

```
mkdir ~/root
```

I already have a `~/tech` directory where I keep all my technical logistics, manuals, and code snippets, so I made a `~/tech/root` directory. The name doesn't matter, but I'll use `~/root` below.

[Your shell replaces the tilde with the full path to your home directory, saving you a lot of typing. But other programs, like `make`, may or may not recognize the tilde as your home directory.]

The second step is to add the right part of your new root system to all the relevant paths. For programs, that's the `PATH` in your `.bashrc`:

```
PATH=~/root/bin:$PATH
```

By putting the `bin` subdirectory of your new directory before the original `PATH`, it will be searched first and your copy of any programs will be found first. Thus, you can substitute in your preferred version of any programs that are already in the standard shared directories of the system.

For libraries you will fold into your C programs, note the new paths to search in the makefile you wrote in Tip #1 and added to in Tip #2:

```
LIBS=-L/home/your_home/root/lib <plus the other flags, like -lgsl -lm ...>
CFLAGS=-I/home/your_home/root/include <plus -g -Wall --std=gnu99...>
```

[Again, Appendix A of `Modeling with Data` goes into detail on dealing with paths and environment variables.]

The last step is to install programs in your new root. If you have the source code and it uses autotools, all you have to do is add `--prefix=~/root` in the right place:

```
./configure --prefix=~/root
make
make install
```

You didn't need `sudo` to do the install step because everything is now in territory you control.

Now that you have a local root, you can use it for other systems as well, such as adding a subdirectory for R packages (e.g., `mkdir ~/root/rlib`) and notifying R about them by adding `R_LIBS=~/root/rlib` to the `~/.Renviron` file.

Because the programs and libraries are in your home directory and have no more permissions than you do, your sysadmin can't complain that they are an imposition on others. If your sysadmin complains anyway, then, as sad as it may be, it may be time to break up.