DEV SHED DOT COM

Introduction to mod_perl (part III): Non-privileged Install

By Stas Bekman

If you like this article, please make a donation to the Perl development grant fund.

Table of Contents

Introduction

As you have seen from my previous articles, mod_perl enabled Apache consists of two main components: Perl modules and Apache itself. While installing Apache without root privileges is a very easy task, one should know how to install Perl modules in non−system wide location. In this article I'll show different ways to tackle this task.

In the examples used in this article I'll use *stas* as a username, and */home/stas* as a home directory of that user.

Installing Perl Modules into a Directory of Choice

Since without superuser permissions you aren't allowed to install modules into system directories like */usr/lib/perl5*, you need to find out how to install the modules under your home directory. It's easy.

First you have to decide where to install the modules. The simplest approach is to simulate the portion of the / file system relevant to Perl under your home directory. Actually we need only two directories:

```
/home/stas/bin
/home/stas/lib
```

We don't have to create them, since that will be done automatically when the first module is installed. 99% of the files will go into the *lib* directory. Occasionally, when some module distribution comes with Perl scripts, these will go into the *bin* directory. This directory will be created if it doesn't exist.

Let's install the *CGI.pm* package, which includes a few other **CGI::*** modules. As usual, download the package from the CPAN repository, unpack it and **chdir** to the newly−created directory.

Now do a standard **perl Makefile.PL** to prepare a *Makefile*, but this time tell **MakeMaker** to use your Perl installation directories instead of the defaults.

```
% perl Makefile.PL PREFIX=/home/stas
```

PREFIX=/home/stas is the only part of the installation process which is different from usual. Note that if you don't like how **MakeMaker** chooses the rest of the directories, or if you are using an older version of it which requires an explicit declaration of all the target directories, you should do this:

```
% perl Makefile.PL PREFIX=/home/stas \
INSTALLPRIVLIB=/home/stas/lib/perl5 \
INSTALLSCRIPT=/home/stas/bin \
INSTALLSITELIB=/home/stas/lib/perl5/site_perl \
INSTALLBIN=/home/stas/bin \
INSTALLMAN1DIR=/home/stas/lib/perl5/man \
INSTALLMAN3DIR=/home/stas/lib/perl5/man3
```

The rest is as usual:

```
% make
% make test
% make install
```

make install installs all the files in the private repository. Note that all the missing directories are created automatically, so there is no need to create them in first place. Here (slightly edited) is what it does :

```
Installing /home/stas/lib/perl5/CGI/Cookie.pm
Installing /home/stas/lib/perl5/CGI.pm
Installing /home/stas/lib/perl5/man3/CGI.3
Installing /home/stas/lib/perl5/man3/CGI::Cookie.3
Writing /home/stas/lib/perl5/auto/CGI/.packlist
Appending installation info to
/home/stas/lib/perl5/perllocal.pod
```

If you have to use the explicit target parameters, instead of a single **PREFIX** parameter, you will find it useful to create a file called for example *~/.perl_dirs* (where ~ is **/home/stas** in our example) containing:

```
PREFIX=/home/stas \
INSTALLPRIVLIB=/home/stas/lib/perl5 \
INSTALLSCRIPT=/home/stas/bin \
INSTALLSITELIB=/home/stas/lib/perl5/site_perl \
INSTALLBIN=/home/stas/bin \
INSTALLMAN1DIR=/home/stas/lib/perl5/man \
INSTALLMAN3DIR=/home/stas/lib/perl5/man3
```

From now on, any time you want to install perl modules locally you simply execute:

```
% perl Makefile.PL `cat ~/.perl_dirs`
% make
% make test
% make install
```

Using this method you can easily maintain several Perl module repositories. For example, you could have one for production Perl and another for development:

```
% perl Makefile.PL `cat ~/.perl_dirs.production`
```

or

```
% perl Makefile.PL `cat ~/.perl_dirs.develop`
```

Making Your Scripts Find the Locally Installed Modules

Perl modules are generally placed in four main directories. To find these directories, execute:

```
% perl -V
```

The output contains important information about your Perl installation. At the end you will see:

```
Characteristics of this binary (from libperl):
Built under linux
Compiled at Apr 6 1999 23:34:07
@INC:
/usr/lib/perl5/5.00503/i386-linux
/usr/lib/perl5/5.00503
/usr/lib/perl5/site_perl/5.005/i386-linux
/usr/lib/perl5/site_perl/5.005
.
```

It shows us the content of the Perl special variable **@INC**, which is used by Perl to look for its modules. It is equivalent to the **PATH** environment variable in Unix shells which is used to find executable programs.

Notice that Perl looks for modules in the . directory too, which stands for the current directory. It's the last entry in the above output.

Of course this example is from version *5.00503* of Perl installed on my x86 architecture PC running Linux. That's why you see *i386−linux* and *5.00503*. If your system runs a different version of Perl, operating system, processor or chipset architecture, then some of the directories will have different names.

I also have a perl−5.6.1 installed under **/usr/local/lib/** so when I do:

```
% /usr/local/bin/perl5.6.1 -V
```

I see:

```
@INC:
/usr/local/lib/perl5/5.6.1/i586-linux
/usr/local/lib/perl5/5.6.1
/usr/local/lib/site_perl/5.6.1/i586-linux
/usr/local/lib/site_perl
```

Note that it's still *Linux*, but the newer Perl version uses the version of my Pentium processor (thus the *i586* and not *i386*). This makes use of compiler optimizations for Pentium processors when the binary Perl extensions are created.

All the platform specific files, such as compiled C files glued to Perl with **XS** or C**SWIG**, are supposed to go into the **i386−linux**−like directories.

Important: As we have installed the Perl modules into non−standard directories, we have to let Perl know where to look for the four directories. There are two ways to accomplish this. You can either set the **PERL5LIB** environment variable, or you can modify the **@INC** variable in your scripts.

Assuming that we use perl−5.00503, in our example the directories are:

```
/home/sbekman/lib/perl5/5.00503/i386-linux
/home/sbekman/lib/perl5/5.00503
/home/sbekman/lib/perl5/site_perl/5.005/i386-linux
/home/sbekman/lib/perl5/site_perl/5.005
```

As mentioned before, you find the exact directories by executing **perl −V** and replacing the global Perl installation's base directory with your home directory.

Modifying **@INC** is quite easy. The best approach is to use the **lib** module (pragma), by adding the following snippet at the top of any of your scripts that require the locally installed modules.

```
use lib qw(/home/stas/lib/perl5/5.00503/
/home/stas/lib/perl5/site_perl/5.005);
```

Another way is to write code to modify **@INC** explicitly:

```
BEGIN {
unshift @INC,
qw(/home/stas/lib/perl5/5.00503
/home/stas/lib/perl5/5.00503/i386-linux
/home/stas/lib/perl5/site_perl/5.005
/home/stas/lib/perl5/site_perl/5.005/i386-linux);
}
```

Note that with the **lib** module we don't have to list the corresponding architecture specific directories, since it adds them automatically if they exist (to be exact, when *$dir/$archname/auto* exists).

Also, notice that both approaches *prepend* the directories to be searched to **@INC**. This allows you to install a more recent module into your local repository and Perl will use it instead of the older one installed in the main system repository.

Both approaches modify the value of **@INC** at compilation time. The **lib** module uses the *BEGIN* block as well, but internally.

Now, let's assume the following scenario. I have installed the **LWP** package in my local repository. Now I want to install another module (e.g. mod_perl) and it has **LWP** listed in its prerequisites list. I know that I have **LWP** installed, but when I run **perl Makefile.PL** for the module I'm about to install I'm told that I don't have **LWP** installed.

There is no way for Perl to know that we have some locally installed modules. All it does is search the directories listed in **@INC**, and since the latter contains only the default four directories (plus the . directory), it cannot find the locally installed **LWP** package. We cannot solve this problem by adding code to modify **@INC**, but changing the **PERL5LIB** environment variable will do the trick. If you are using **(t)csh** for interactive work, do this:

```
setenv PERL5LIB /home/stas/lib/perl5/5.00503:
/home/stas/lib/perl5/site_perl/5.005
```

It should be a single line with directories separated by colons (**:**) and no spaces. If you are a **(ba)sh** user, do this:

```
export PERL5LIB=/home/stas/lib/perl5/5.00503:
/home/stas/lib/perl5/site_perl/5.005
```

Again make it a single line. If you use bash you can use multi−line commands by terminating split lines with a backslash (\), like this:

```
export PERL5LIB=/home/stas/lib/perl5/5.00503:\
/home/stas/lib/perl5/site_perl/5.005
```

As with **use lib**, perl automatically prepends the architecture specific directories to **@INC** if those exist.

When you have done this, verify the value of the newly configured **@INC** by executing **perl −V** as before. You should see the modified value of **@INC**:

```
% perl −V

Characteristics of this binary (from libperl):
Built under linux
Compiled at Apr 6 1999 23:34:07
```

```
%ENV:
PERL5LIB="/home/stas/lib/perl5/5.00503:
/home/stas/lib/perl5/site_perl/5.005"
@INC:
/home/stas/lib/perl5/5.00503/i386−linux
/home/stas/lib/perl5/5.00503
/home/stas/lib/perl5/site_perl/5.005/i386−linux
/home/stas/lib/perl5/site_perl/5.005
/usr/lib/perl5/5.00503/i386−linux
/usr/lib/perl5/5.00503
/usr/lib/perl5/site_perl/5.005/i386−linux
/usr/lib/perl5/site_perl/5.005
.
```

When everything works as you want it to, add these commands to your *.tcshrc* or *.bashrc* file. The next time you start a shell, the environment will be ready for you to work with the new Perl.

Note that if you have a **PERL5LIB** setting, you don't need to alter the **@INC** value in your scripts. But if for example someone else (who doesn't have this setting in the shell) tries to execute your scripts, Perl will fail to find your locally installed modules. The best example is a crontab script that *might* use a different SHELL environment and therefore the **PERL5LIB** setting won't be available to it.

So the best approach is to have both the **PERL5LIB** environment variable and the explicit **@INC** extension code at the beginning of the scripts as described above.

The CPAN.pm Shell and Locally Installed Modules

The **CPAN.pm** shell saves a great deal of time when you have to deal with Perl modules installation and keeping them up to date. It does the job for us, even detecting the missing modules listed in prerequisites, fetching and installing them. So you might wonder whether you can use **CPAN.pm** to maintain your local repository as well.

When you start the **CPAN** interactive shell, it searches first for the user's private configuration file and then for the system wide one. When I'm logged as user **stas** the two files on my setup are:

```
/home/stas/.cpan/CPAN/MyConfig.pm
/usr/lib/perl5/5.00503/CPAN/Config.pm
```

If there is no **CPAN** shell configured on your system, when you start the shell for the first time it will ask you a dozen configuration questions and then create the *Config.pm* file for you.

If you've got it already system−wide configured, you should have a **/usr/lib/perl5/5.00503/CPAN/Config.pm**. If you have a different Perl version, alter the path to use your Perl's version number, when looking up the file. Create the directory (**mkdir −p** creates the whole path at once) where the local configuration file will go:

```
% mkdir −p /home/stas/.cpan/CPAN
```

Now copy the system wide configuration file to your local one.

```
% cp /usr/lib/perl5/5.00503/CPAN/Config.pm \
/home/stas/.cpan/CPAN/MyConfig.pm
```

The only thing left is to change the base directory of *.cpan* in your local file to the one under your home directory. On my machine I replace **/usr/src/.cpan** (that's where my system's **.cpan** directory resides) with **/home/stas**. I use Perl of course!

```
% perl −pi −e 's|/usr/src|/home/stas|' \
/home/stas/.cpan/CPAN/MyConfig.pm
```

Now you have the local configuration file ready, you have to tell it what special parameters you need to pass when executing the C stage.

Open the file in your favorite editor and replace line:

```
'makepl_arg' => q[],
```

with:

```
'makepl_arg' => q[PREFIX=/home/stas],
```

Now you've finished the configuration. Assuming that you are logged in as the same user you have prepared the local installation for (*stas* in our example), start it like this:

```
% perl −MCPAN −e shell
```

From now on any module you try to install will be installed locally. If you need to install some system modules, just become the superuser and install them in the same way. When you are logged in as the superuser, the system−wide configuration file will be used instead of your local one.

If you have used more than just the **PREFIX** variable, modify *MyConfig.pm* to use them. For example if you have used these variables:

```
perl Makefile.PL PREFIX=/home/stas \
INSTALLPRIVLIB=/home/stas/lib/perl5 \
INSTALLSCRIPT=/home/stas/bin \
INSTALLSITELIB=/home/stas/lib/perl5/site_perl \
INSTALLBIN=/home/stas/bin \
INSTALLMAN1DIR=/home/stas/lib/perl5/man \
INSTALLMAN3DIR=/home/stas/lib/perl5/man3
```

then replace **PREFIX=/home/stas** in the line:

```
'makepl_arg' => q[PREFIX=/home/stas],
```

with all the variables from above, so that the line becomes:

```
'makepl_arg' => q[PREFIX=/home/stas \
INSTALLPRIVLIB=/home/stas/lib/perl5 \
INSTALLSCRIPT=/home/stas/bin \
INSTALLSITELIB=/home/stas/lib/perl5/site_perl \
INSTALLBIN=/home/stas/bin \
INSTALLMAN1DIR=/home/stas/lib/perl5/man \
INSTALLMAN3DIR=/home/stas/lib/perl5/man3],
```

If you arrange all the above parameters in one line, you can remove the backslashes (\).

Making a Local Apache Installation

Just like with Perl modules, if you don't have permissions to install files into the system area you have to install them locally under your home directory. It's almost the same as a plain installation, but you have to run the server listening to a port number greater than 1024 since only root processes can listen to lower numbered ports.

Another important issue you have to resolve is how to add startup and shutdown scripts to the directories used by the rest of the system services. You will have to ask your system administrator to assist you with this issue.

To install Apache locally, all you have to do is to tell **.configure** in the Apache source directory what target directories to use. If you are following the convention that I use, which makes your home directory look like the / (base) directory, the invocation parameters would be:

```
./configure --prefix=/home/stas
```

Apache will use the prefix for the rest of its target directories instead of the default **/usr/local/apache**. If you want to see what they are, before you proceed add the *−−show−layout* option:

```
./configure --prefix=/home/stas --show-layout
```

You might want to put all the Apache files under **/home/stas/apache** following Apache's convention:

```
./configure --prefix=/home/stas/apache
```

If you want to modify some or all of the names of the automatically created directories:

```
./configure --prefix=/home/stas/apache \
--sbindir=/home/stas/apache/sbin \
--sysconfdir=/home/stas/apache/etc \
--localstatedir=/home/stas/apache/var \
--runtimedir=/home/stas/apache/var/run \
--logfiledir=/home/stas/apache/var/logs \
--proxycachedir=/home/stas/apache/var/proxy
```

That's all!

Also remember that you can start the script only under a user and group you belong to. You must set the **User** and **Group** directives in *httpd.conf* to appropriate values.

Manual Local mod_perl Enabled Apache Installation

Now when we have learned how to install local Apache and Perl modules separately, let's see how to install mod_perl enabled Apache in our home directory. It's almost as simple as doing each one separately, but there is one wrinkle you need to know about which I'll mention at the end of this section.

Let's say you have unpacked the Apache and mod_perl sources under */home/stas/src* and they look like this:

```
% ls /home/stas/src
/home/stas/src/apache_x.x.x
/home/stas/src/mod_perl-x.xx
```

where *x.xx* are the version numbers as usual. You want the Perl modules from the mod_perl package to be installed under */home/stas/lib/perl5* and the Apache files to go under */home/stas/apache*. The following commands will do that for you:

```
% perl Makefile.PL \
PREFIX=/home/stas \
APACHE_PREFIX=/home/stas/apache \
APACHE_SRC=../apache_x.x.x/src \
DO_HTTPD=1 \
USE_APACI=1 \
EVERYTHING=1
% make &make test &make install
% cd ../apache_x.x.x
% make install
```

If you need some parameters to be passed to the **.configure** script, as we saw in the previous section use **APACI_ARGS**. For example:

```
APACI_ARGS='--sbindir=/home/stas/apache/sbin, \
--sysconfdir=/home/stas/apache/etc, \
--localstatedir=/home/stas/apache/var, \
--runtimedir=/home/stas/apache/var/run, \
--logfiledir=/home/stas/apache/var/logs, \
--proxycachedir=/home/stas/apache/var/proxy'
```

Note that the above multiline splitting will work only with **bash**, **tcsh** users will have to list all the parameters on a single line.

Basically the installation is complete. The only remaining problem is the **@INC** variable. This won't be correctly set if you rely on the **PERL5LIB** environment variable unless you set it explicitly in a startup file which is **require**'d before loading any other module that resides in your local repository. A much nicer approach is to use the **lib** pragma as we saw before, but in a slightly different way−−we use it in the startup file and it affects all the code that will be executed under mod_perl handlers. For example:

```
PerlRequire /home/stas/apache/perl/startup.pl
```

where **startup.pl** starts with:

```
use lib qw(/home/stas/lib/perl5/5.00503/
/home/stas/lib/perl5/site_perl/5.005);
```

Note that you can still use the hard−coded **@INC** modifications in the scripts themselves, but be aware that scripts modify **@INC** in **BEGIN** blocks and mod_perl executes the **BEGIN** blocks only when it performs script compilation. As a result, **@INC** will be reset to its original value after the scripts are compiled and the hard−coded settings will be forgotten.

The only place you can alter the "original" value is during the server configuration stage either in the startup file or by putting

```
PerlSetEnv Perl5LIB \
/home/stas/lib/perl5/5.00503/:/home/stas/lib/perl5/site_perl/5.005
```

in *httpd.conf*, but the latter setting will be ignored if you use the **PerlTaintcheck** setting, and I hope you do use it.

The rest of the mod_perl configuration and use is just the same as if you were installing mod_perl as superuser.

Local mod_perl Enabled Apache Installation with CPAN.pm

Assuming that you have configured **CPAN.pm** to install Perl modules locally as explained earlier in this article, the installation is very simple. Start the **CPAN.pm** shell, set the arguments to be passed to **perl Makefile.PL** (modify the example setting to suit your needs), and tell **CPAN.pm** to do the rest for you:

```
% perl -MCPAN -eshell
cpan> o conf makepl_arg 'DO_HTTPD=1 USE_APACI=1 EVERYTHING=1 \
PREFIX=/home/stas APACHE_PREFIX=/home/stas/apache'
cpan> install mod_perl
```

When you use **CPAN.pm** for local installations, after the mod_perl installation is complete you must make sure that the value of **makepl_arg** is restored to its original value.

The simplest way to do this is to quit the interactive shell by typing *quit* and reenter it. But if you insist here is how to make it work without quitting the shell. You really want to skip this :)

If you want to continue working with **CPAN** *without* quitting the shell, you must:

1. remember the value of **makepl_arg**
2. change it to suit your new installation
3. build and install mod_perl
4. restore it after completing mod_perl installation

this is quite a cumbersome task as of this writing, but I believe that **CPAN.pm** will eventually be improved to handle this more easily.

So if you are still with me, start the shell as usual:

```
% perl -MCPAN -eshell
```

First, read the value of the **makepl_arg**:

```
cpan> o conf makepl_arg

PREFIX=/home/stas
```

It will be something like **PREFIX=/home/stas** if you configured **CPAN.pm** to install modules locally. Save this value:

```
cpan> o conf makepl_arg.save PREFIX=/home/stas
```

Second, set a new value, to be used by the mod_perl installation process. (You can add parameters to this line, or remove them, according to your needs.)

```
cpan> o conf makepl_arg 'DO_HTTPD=1 USE_APACI=1 EVERYTHING=1 \
PREFIX=/home/stas APACHE_PREFIX=/home/stas/apache'
```

Third, let **CPAN.pm** build and install mod_perl for you:

```
cpan> install mod_perl
```

Fourth, reset the original value to **makepl_arg**. We do this by printing the value of the saved variable and assigning it to **makepl_arg**.

```
cpan> o conf makepl_arg.save

PREFIX=/home/stas

cpan> o conf makepl_arg PREFIX=/home/stas
```

Not so neat, but a working solution. You could have written the value on a piece of paper instead of saving it to **makepl_arg.save**, but you are more likely to make a mistake that way.

References

The Apache site's URL: http://perl.apache.org

CPAN is the Comprehensive Perl Archive Network. The Master site's URL is http://cpan.org/. CPAN is mirrored at more than one hundred sites around the world. (http://cpan.org/SITES.html)