1 Installing mod_perl 2.0

1.1 Description

This chapter provides an in-depth mod_perl 2.0 installation coverage.

1.2 Prerequisites

Before building mod_perl 2.0 you need to have its prerequisites installed. If you don't have them, download and install them first, using the information in the following sections. Otherwise proceed directly to the mod_perl building instructions.

The mod_perl 2.0 prerequisites are:

- **Apache**

  Apache 2.0 is required. mod_perl 2.0 **does not** work with Apache 1.3.

- **Perl**
  - **prefork MPM**

    Requires at least Perl version 5.6.0. But we strongly suggest to use at least version 5.6.1, since 5.6.0 is quite buggy. The only reason we support 5.6.0 is for development reasons (so the build can be tested on systems having only 5.6.0) and those users who want to give it a try, without first having the hassle of updating their perl version.

    You don't need to have threads-support enabled in Perl. If you do have it, it **must** be *ithreads* and not *5005threads*! If you have:

    ```
    % perl5.8.0 -V:use5005threads
    use5005threads='define';
    ```

    you must rebuild Perl without threads enabled or with `-Dusethreads`. Remember that threads-support slows things down and on some platforms it's unstable (e.g., FreeBSD), so don't enable it unless you really need it.

  - **threaded MPMs**

    Require at least Perl version 5.8.0 with ithreads support built-in. That means that it should report:

    ```
    % perl5.8.0 -V:useithreads -V:usemultiplicity
    useithreads='define';
    usemultiplicity='define';
    ```

    If that's not what you see rebuild Perl with `-Dusethreads`.

  - **threads.pm**

If you want to run applications that take benefit of Perl's *threads.pm* Perl version 5.8.1 or higher w/ithreads enabled is required. Perl 5.8.0's *threads.pm* doesn't work with mod_perl 2.0.

- **CPAN Perl Modules**

  The mod_perl 2.0 test suite has several requirements on its own. If you don't satisfy them, the tests depending on these requirements will be skipped, which is OK, but you won't get to run these tests and potential problems, which may exhibit themselves in your own code, could be missed. We don't require them from `Makefile.PL`, which could have been automated the requirements installation, in order to have less dependencies to get mod_perl 2.0 installed.

  Also if your code uses any of these modules, chances are that you will need to use at least the version numbers listed here.

  - **CGI.pm 3.01**
  - **Compress::Zlib 1.09**

1.2.1 Downloading Stable Release Sources

If you are going to install mod_perl on a production site, you want to use the officially released stable components. Since the latest stable versions change all the time you should check for the latest stable version at the listed below URLs:

- **Perl**

  Download from: *http://cpan.org/src/README.html*

  This direct link which symlinks to the latest release should work too: *http://cpan.org/src/stable.tar.gz*.

  For the purpose of examples in this chapter we will use the package named *perl-5.8.x.tar.gz*, where *x* should be replaced with the real version number.

- **Apache**

  Download from: *http://www.apache.org/dist/httpd/*

  For the purpose of examples in this chapter we will use the package named *httpd-2.x.xx.tar.gz*, where *x.xx* should be replaced with the real version number.

1.2.2 Getting Bleeding Edge CVS Sources

If you really know what you are doing you can use the cvs versions of the components. Chances are that you don't want to them on a production site. You have been warned!

- **Perl**

```
# (--delete to ensure a clean state)
% rsync -acvz --delete --force \
  rsync://ftp.linux.activestate.com/perl-current/ perl-current
```

If you are re-building Perl after rsync-ing, make sure to cleanup first:

```
% make distclean
```

before running `./Configure`.

You'll also want to install (at least) LWP if you want to fully test mod_perl. You can install LWP with `CPAN.pm` shell:

```
% perl -MCPAN -e 'install("LWP")'
```

- **Apache**

  To download the cvs version of httpd-2.0 and bring it to the same state of the distribution package, execute the following commands:

```
% cvs -d :pserver:anoncvs@cvs.apache.org:/home/cvspublic login
```

  The password is "anoncvs". Now extract the `APACHE_2_0_BRANCH` branch of httpd-2.0.xx. If you don't use this branch you will get httpd-2.1.xx which at this moment is not supported. Similarly you need `APR_0_9_BRANCH` and `APU_0_9_BRANCH` cvs branches for apr and apr-util projects, respectively.

```
% cvs -d :pserver:anoncvs@cvs.apache.org:/home/cvspublic co \
  -r APACHE_2_0_BRANCH -d httpd-2.0 httpd-2.0
% cd httpd-2.0/srclib
% cvs -d :pserver:anoncvs@cvs.apache.org:/home/cvspublic co \
  -r APR_0_9_BRANCH -d apr apr
% cvs -d :pserver:anoncvs@cvs.apache.org:/home/cvspublic co \
  -r APU_0_9_BRANCH -d apr-util apr-util
% cvs -d :pserver:anoncvs@cvs.apache.org:/home/cvspublic co \
  -r APU_0_9_BRANCH -d apr-iconv apr-iconv
% cd ..
% ./buildconf
```

  Once extracted, whenever you want to sync with the latest httpd-2.0 version and rebuild, run:

```
% cd httpd-2.0
% cvs up -dP
% make distclean && ./buildconf
```

1.2.3 Configuring and Installing Prerequisites

If you don't have the prerequisites installed yet, install them now.

- **Perl**

```
% cd perl-5.8.x
% ./Configure -des
```

If you need the threads support, run:

```
% ./Configure -des -Dusethreads
```

If you want to debug mod_perl segmentation faults, add the following *./Configure* options:

```
-Doptimize='-g' -Dusedevel
```

Now build it:

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

- **Apache**

```
% cd httpd-2.x.xx
% ./configure --prefix=$HOME/httpd/prefork --with-mpm=prefork
% make && make install
```

1.3 Installing mod_perl from Binary Packages

As of this writing only the binaries for the Win32 platform are available, kindly prepared and maintained by Randy Kobes. See the documentation on Win32 binaries for details.

Some RPM packages can be found using rpmfind services, e.g.:

http://www.rpmfind.net/linux/rpm2html/search.php?query=mod_perl&submit=Search+... However if you have problems using them, you have to contact those who have created them.

1.4 Installing mod_perl from Source

Building from source is the best option, because it ensures a binary compatibility with Apache and Perl. However it's possible that your distribution provides a solid binary mod_perl 2.0 package.

For Win32 specific details, see the documentation on Win32 installation.

1.4.1 Downloading the mod_perl Source

First download the mod_perl source.

- **Stable Release**

  Download from: *http://perl.apache.org/download/*

  This direct link which symlinks to the latest release should work too: *http://perl.apache.org/dist/mod_perl-2.0-current.tar.gz.*

For the purpose of examples in this chapter we will use the package named *mod_perl-2.x.xx.tar.gz*, where *x.xx* should be replaced with the real version number.

Open the package with:

```
% tar -xvzf mod_perl-2.x.xx.tar.gz
```

or an equivalent command.

- **CVS Bleeding-Edge Version**

  To download the cvs version of modperl-2.0 execute the following commands:

  ```
  % cvs -d :pserver:anoncvs@cvs.apache.org:/home/cvspublic login
  ```

  The password is "anoncvs".

  ```
  % cvs -d :pserver:anoncvs@cvs.apache.org:/home/cvspublic co modperl-2.0
  ```

  You can also try the latest CVS snapshot:

  http://cvs.apache.org/snapshots/modperl-2.0/

1.4.2 Configuring mod_perl

Before you proceed make sure that Apache 2.0 has been built and installed. mod_perl **cannot** be built before that.

Like any other Perl module, mod_perl is configured via the *Makefile.PL* file, but requires one or more configuration options:

```
% cd modperl-1.99_xx
% perl Makefile.PL <options>
```

where *options* is an optional list of (key,value) pairs.

The following sections give the details about all the available options, but let’s mention first the most important ones.

If you want to have mod_perl 1.0 and 2.0 installed under the same perl tree you need to enable `MP_INST_APACHE2`:

```
% perl Makefile.PL MP_INST_APACHE2=1 <other options>
```

It seems that most users use pre-packaged Apache installation, most of which tend to spread the Apache files across many directories (i.e. not using --enable-layout=Apache, which puts all the files under the same directory). If Apache 2.0 files are spread under different directories, you need to use at least the `MP_APXS` option, which should be set to a full path to the `apxs` executable. For example:

```
% perl Makefile.PL MP_INST_APACHE2=1 MP_APXS=/path/to/apxs
```

For example RedHat Linux system installs the `httpd` binary, the `apxs` and `apr-config` scripts (the latter two are needed to build mod_perl) all in different locations, therefore they configure mod_perl 2.0 as:

```
% perl Makefile.PL MP_INST_APACHE2=1 MP_APXS=/path/to/apxs \
  MP_APR_CONFIG=/another/path/to/apr-config <other options>
```

However a correctly built Apache shouldn't require the `MP_APR_CONFIG` option, since `MP_APXS` should provide the location of this script.

If however all Apache 2.0 files were installed under the same directory, mod_perl 2.0's build only needs to know the path to that directory, passed via the `MP_AP_PREFIX` option:

```
% perl Makefile.PL MP_INST_APACHE2=1 MP_AP_PREFIX=$HOME/httpd/prefork
```

These and other options are discussed in the following sections.

1.4.2.1 Boolean Build Options

The following options are boolean and can be set with `MP_XXX=1` or unset with `MP_XXX=0`, where XXX is the name of the option.

1.4.2.1.1 MP_PROMPT_DEFAULT

Accept default values for all would-be prompts.

1.4.2.1.2 MP_GENERATE_XS

Generate XS code from parsed source headers in *xs/tables/$httpd_version*. Default is 1, set to 0 to disable.

1.4.2.1.3 MP_USE_DSO

Build mod_perl as a DSO (*mod_perl.so*). This is the default. It'll be turned off if `MP_USE_STATIC=1` is used.

1.4.2.1.4 MP_USE_STATIC

Build static mod_perl (*mod_perl.a*). This is the default. It'll be turned off if `MP_USE_DSO=1` is used.

`MP_USE_DSO` and `MP_USE_STATIC` are both enabled by default. So mod_perl is built once as *mod_perl.a* and *mod_perl.so*, but afterwards you can choose which of the two to use.

META: The following is not implemented yet.

```
mod_perl and ends up with a src/modules/perl/mod_perl.{so,a} and
src/modules/perl/ldopts.  to link modperl static with httpd, we just
need some config.m4 magic to add 'ldopts' and mod_perl.a to the build.
so one could then build httpd like so:
```

```
ln -s ~/apache/modperl-2.0/src/modules/perl $PWD/src/modules
./configure --with-mpm=prefork --enable-perl=static ...

we not be configuring/building httpd for the user as 1.x attempted.

downside is one will need to have configured httpd first, so that
headers generated.  so it will probably be more like:

./configure --with-mpm=prefork ...
(go build modperl)
./config.nice --enable-perl=static && make

we could of course provide a wrapper script todo this, but don't want
to have this stuff buried and tangled like it is in 1.x
```

1.4.2.1.5 MP_STATIC_EXTS

Build `Apache::*.xs` as static extensions.

1.4.2.1.6 MP_USE_GTOP

Link with *libgtop* and enable *libgtop* reporting.

1.4.2.1.7 MP_COMPAT_1X

`MP_COMPAT_1X=1` or a lack of it enables several mod_perl 1.0 back-compatibility features, which are deprecated in mod_perl 2.0. It's enabled by default, but can be disabled with `MP_COMPAT_1X=0` during the build process.

When this option is disabled, the following things will happen:

- Environment variable `GATEWAY_INTERFACE` will be enabled only if `PerlOptions +SetupEnv` is enabled and its value would be the default:

  ```
  CGI/1.1
  ```

  and not:

  ```
  CGI-Perl/1.1
  ```

  The use of `$ENV{GATEWAY_INTERFACE}` is deprecated and the existance of `$ENV{MOD_PERL}` should be checked instead.

- Deprecated special variable, `$Apache::__T` won't be available. Use `${^TAINT}` instead.

- *$ServerRoot* and *$ServerRoot/lib/perl* won't be appended to `@INC`. Instead use:

  ```
  PerlSwitches -I/path/to/server -I/path/to/server/lib/perl
  ```

in *httpd.conf* or:

```
use Apache::Server ();
use Apache::ServerUtil ();
use Apache::Process ();
my $pool = Apache->server->process->pool;
push @INC, Apache::Server::server_root_relative($pool, "");
push @INC, Apache::Server::server_root_relative($pool, "lib/perl");
```

in *startup.pl*.

- The following deprecated configuration directives won't be recognized by Apache:

  ```
  PerlSendHeader
  PerlSetupEnv
  PerlHandler
  PerlTaintCheck
  PerlWarn
  ```

  Use their 2.0 equivalents instead.

1.4.2.1.8 MP_DEBUG

Turn on debugging (`-g -lperld`) and tracing.

1.4.2.1.9 MP_MAINTAINER

Enable maintainer compile mode, which sets `MP_DEBUG=1` and adds the following `gcc` flags:

```
-DAP_DEBUG -Wall -Wmissing-prototypes -Wstrict-prototypes \
-Wmissing-declarations \
-DAP_DEBUG -DAP_HAVE_DESIGNATED_INITIALIZER
```

To use this mode Apache must be build with `--enable-maintainer-mode`.

1.4.2.1.10 MP_TRACE

Enable tracing

1.4.2.1.11 MP_INST_APACHE2

Install all the **.pm* modules relative to the *Apache2/* directory.

1.4.2.2 Non-Boolean Build Options

set the non-boolen options with MP_XXX=value.

1.4.2.2.1 MP_APXS

Path to `apxs`. For example if you've installed Apache 2.0 under */home/httpd/httpd-2.0* as DSO, the default location would be */home/httpd/httpd-2.0/bin/apxs*.

1.4.2.2.2 MP_AP_PREFIX

Apache installation prefix, under which the *include/* directory with Apache C header files can be found. For example if you've have installed Apache 2.0 in directory *\Apache2* on Win32, you should use:

```
MP_AP_PREFIX=\Apache2
```

If Apache is not installed yet, you can point to the Apache 2.0 source directory, but only after you've built or configured Apache in it. For example:

```
MP_AP_PREFIX=/home/stas/apache.org/httpd-2.0
```

Though in this case `make test` won't automatically find `httpd`, therefore you should run `t/TEST` instead and pass the location of `apxs` or `httpd`, e.g.:

```
% t/TEST -apxs /home/stas/httpd/prefork/bin/apxs
```

or

```
% t/TEST -httpd /home/stas/httpd/prefork/bin/httpd
```

1.4.2.2.3 MP_APR_CONFIG

If APR wasn't installed under the same file tree as httpd, you may need to tell the build process where it can find the executable `apr-config`, which can then be used to figure out where the apr and aprutil *include/* and *lib/* directories can be found.

1.4.2.2.4 MP_CCOPTS

Add to compiler flags, e.g.:

```
MP_CCOPTS=-Werror
```

(Notice that `-Werror` will work only with the Perl version 5.7 and higher.)

1.4.2.2.5 MP_OPTIONS_FILE

Read build options from given file. e.g.:

```
MP_OPTIONS_FILE=~/.my_mod_perl2_opts
```

1.4.2.3 mod_perl-specific Compiler Options

1.4.2.3.1 -DMP_IOBUFSIZE

Change the default mod_perl's 8K IO buffer size, e.g. to 16K:

```
MP_CCOPTS=-DMP_IOBUFSIZE=16384
```

1.4.2.4 mod_perl Options File

Options can also be specified in the file *makepl_args.mod_perl2* or *.makepl_args.mod_perl2*. The file can be placed under `$ENV{HOME}`, the root of the source package or its parent directory. So if you unpack the mod_perl source into */tmp/mod_perl-2.x/* and your home is */home/foo/*, the file will be searched in:

```
/tmp/mod_perl-2.x/makepl_args.mod_perl2
/tmp/makepl_args.mod_perl2
/home/foo/makepl_args.mod_perl2
/tmp/mod_perl-2.x/.makepl_args.mod_perl2
/tmp/.makepl_args.mod_perl2
/home/foo/.makepl_args.mod_perl2
```

If the file specified in `MP_OPTIONS_FILE` is found the *makepl_args.mod_perl2* will be ignored.

Options specified on the command line override those from *makepl_args.mod_perl2* and those from `MP_OPTIONS_FILE`.

If your terminal supports colored text you may want to set the environment variable `APACHE_TEST_COLOR` to 1 to enable the colored tracing which makes it easier to tell the reported errors and warnings, from the rest of the notifications.

1.4.3 Re-using Configure Options

Since mod_perl remembers what build options were used to build it if first place, you can use this knowledge to rebuild itself using the same options. Simply `chdir(1)` to the mod_perl source directory and run:

```
% cd modperl-2.x.xx
% perl -MApache::Build -e rebuild
```

1.4.4 Compiling mod_perl

Next stage is to build mod_perl:

```
% make
```

1.4.5 Testing mod_perl

When mod_perl has been built, it's very important to test that everything works on your machine:

```
% make test
```

If something goes wrong with the test phase and want to figure out how to run individual tests and pass various options to the test suite, see the corresponding sections of the bug reporting guidelines or the Apache::Test Framework tutorial.

1.4.6 Installing mod_perl

Once the test suite has passed, it's a time to install mod_perl.

```
% make install
```

If you install mod_perl system wide, you probably need to become *root* prior to doing the installation:

```
% su
# make install
```

1.5 If Something Goes Wrong

If something goes wrong during the installation, try to repeat the installation process from scratch, while verifying all the steps with this document.

If the problem persists report the problem.

1.6 Maintainers

Maintainer is the person(s) you should contact with updates, corrections and patches.

- Stas Bekman <stas (at) stason.org>

1.7 Authors

- Stas Bekman <stas (at) stason.org>
- Doug MacEachern <dougm (at) covalent.net>

Only the major authors are listed above. For contributors see the Changes file.

Table of Contents: