1 Troubleshooting mod_perl problems

1.1 Description

Frequently encountered problems (warnings and fatal errors) and their troubleshooting.

1.2 Building and Installation

1.3 Configuration and Startup

1.3.1 (28)No space left on device

httpd-2.0 is not very helpful at telling which device has run out of precious space. Most of the time when you get an error like:

```
(28)No space left on device:
mod_rewrite: could not create rewrite_log_lock
```

it means that your system have run out of semaphore arrays. Sometimes it's full with legitimate semaphores at other times it's because some application has leaked semaphores and haven't cleaned them up during the shutdown (which is usually the case when an application segfaults).

Use the relevant application to list the ipc facilities usage. On most Unix platforms this is usually an `ipcs(1)` utility. For example linux to list the semaphore arrays you should execute:

```
% ipcs -s
------ Semaphore Arrays --------
key        semid      owner      perms      nsems
0x00000000 2686976    stas      600        1
0x00000000 2719745    stas      600        1
0x00000000 2752514    stas      600        1
```

Next you have to figure out what are the dead ones and remove them. For example to remove the semid 2719745 execute:

```
% ipcrm -s 2719745
```

Instead of manually removing each (and sometimes there can be many of them), and if you know that none of listed the semaphores is really used (all leaked), you can try to remove them all:

```
% ipcs -s | perl -ane ''ipcrm -s $F[1]''
```

httpd-2.0 seems to use the key `0x00000000` for its semaphores on Linux, so to remove only those that match that key you can use:

```
% ipcs -s | perl -ane '/^0x00000000/ && 'ipcrm -s $F[1]''
```

Notice that on other platforms the output of `ipcs -s` might be different, so you may need to apply a different Perl one-liner.

1.3.2 Segmentation Fault when Using DBI

Update DBI to at least version 1.31.

1.3.3 <Perl> directive missing closing '>'

See the Apache::PerlSections manpage.

1.3.4 'Invalid per-unknown PerlOption: ParseHeaders' on HP-UX 11 for PA-RISC

When building mod_perl 2.0 on HP-UX 11 for PA-RISC architecture, using the HP ANSI C compiler, please make sure you have installed patches PHSS_29484 and PHSS_29485. Once installed the issue should go away.

1.4 Shutdown and Restart

1.5 Code Parsing and Compilation

1.6 Runtime

1.6.1 C Libraries Don't See %ENV Entries Set by Perl Code

For example some people have reported problems with `DBD::Oracle` (whose guts are implemented in C), which doesn't see environment variables (like `ORACLE_HOME`, `ORACLE_SID`, etc.) set in the perl script and therefore fails to connect.

The issue is that the C array `environ[]` is not thread-safe. Therefore mod_perl 2.0 unties `%ENV` from the underlying `environ[]` array under the *perl-script* handler.

The `DBD::Oracle` driver or client library uses `getenv()` (which fetches from the `environ[]` array). When `%ENV` is untied from `environ[]`, Perl code will see `%ENV` changes, but C code will not.

The *modperl* handler does not untie `%ENV` from `environ[]`. Still one should avoid setting `%ENV` values whenever possible. And if it is required, should be done at startup time.

In the particular case of the `DBD::` drivers, you can set the variables that don't change (`$ENV{ORACLE_HOME}` and `$ENV{NLS_LANG}` in the startup file, and those that change pass via the `connect()` method, e.g.:

```
my $sid      = 'ynt0';
my $dsn      = 'dbi:Oracle:';
my $user     = 'username/password';
my $password = '';
$dbh = DBI->connect("$dsn$sid", $user, $password)
    or die "Cannot connect: " . $DBI::errstr;
```

Also remember that `DBD::Oracle` requires that *ORACLE_HOME* (and any other stuff like *NSL_LANG* stuff) be in `%ENV` when `DBD::Oracle` is loaded (which might happen indirectly via the `DBI` module. Therefore you need to make sure that wherever that load happens `%ENV` is properly set by that time.

1.6.2 Error about not finding Apache.pm with CGI.pm

You need to install at least version 2.87 of CGI.pm to work under mod_perl 2.0, as earlier CGI.pm versions aren't mod_perl 2.0 aware.

1.6.3 20014:Error string not specified yet

This error is reported when some undefined Apache error happens. The known cases are:

- **when using mod_deflate**

  A bug in mod_deflate was triggering this error, when a response handler would flush the data that was flushed earlier: http://nagoya.apache.org/bugzilla/show_bug.cgi?id=22259 It has been fixed in httpd-2.0.48.

1.6.4 (22)Invalid argument: core_output_filter: writing data to the network

Apache uses the sendfile syscall on platforms where it is available in order to speed sending of responses. Unfortunately, on some systems, Apache will detect the presence of sendfile at compile-time, even when it does not work properly. This happens most frequently when using network or other non-standard file-system.

The whole story and the solutions are documented at:
http://httpd.apache.org/docs-2.0/faq/error.html#error.sendfile

1.6.5 undefined symbol: apr_table_compress

After a successful mod_perl build, sometimes during the startup or a runtime you'd get an "undefined symbol: foo" error. The following is one possible scenario to encounter this problem and possible ways to resolve it.

Let's say you ran mod_perl's test suite:

```
% make test
```

and got errors, and you looked in the *error_log* file (*t/logs/error_log*) and saw one or more "undefined symbol" errors, e.g.

```
% undefined symbol: apr_table_compress
```

- **Step 1**

  From the source directory (same place you ran "make test") run:

  ```
  % ldd blib/arch/auto/APR/APR.so | grep apr-
  ```

  META: ldd is not available on all platforms, e.g. not on Darwin/OS X

  You you should get a full path, for example:

  ```
  libapr-0.so.0 => /usr/local/apache2/lib/libapr-0.so.0 (0x40003000)
  ```

  or

  ```
  libapr-0.so.0 => /some/path/to/apache/lib/libapr-0.so.0 (0x40003000)
  ```

  or something like that. It's that full path to libapr-0.so.0 that you want.

- **Step 2**

  Do:

  ```
  % nm /path/to/your/libapr-0.so.0 | grep table_compress
  ```

  for example:

  ```
  % nm /usr/local/apache2/lib/libapr-0.so.0 | grep table_compress
  ```

  You should get something like this:

  ```
  0000d010 T apr_table_compress
  ```

  Note that the "grep table_compress" is only an example, the exact string you are looking for is the name of the "undefined symbol" from the error_log. So, if you got "undefined symbol: apr_holy_grail" then you would do

  ```
  % nm /usr/local/apache2/lib/libapr-0.so.0 | grep holy_grail
  ```

- **Step 3**

  Now, let's see what shared libraries your apache binary has. So, if in step 1 you got */usr/local/apache2/lib/libapr-0.so.0* then you will do:

```
% ldd /usr/local/apache2/bin/httpd
```

if in step 1 you got */foo/bar/apache/lib/libapr-0.so.0* then you do:

```
% ldd /foo/bar/apache/bin/httpd
```

The output should look something like this:

```
libssl.so.2 => /lib/libssl.so.2 (0x40023000)
libcrypto.so.2 => /lib/libcrypto.so.2 (0x40054000)
libaprutil-0.so.0 => /usr/local/apache2/lib/libaprutil-0.so.0 (0x40128000)
libgdbm.so.2 => /usr/lib/libgdbm.so.2 (0x4013c000)
libdb-4.0.so => /lib/libdb-4.0.so (0x40143000)
libexpat.so.0 => /usr/lib/libexpat.so.0 (0x401eb000)
libapr-0.so.0 => /usr/local/apache2/lib/libapr-0.so.0 (0x4020b000)
librt.so.1 => /lib/librt.so.1 (0x40228000)
libm.so.6 => /lib/i686/libm.so.6 (0x4023a000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x4025c000)
libnsl.so.1 => /lib/libnsl.so.1 (0x40289000)
libdl.so.2 => /lib/libdl.so.2 (0x4029f000)
libpthread.so.0 => /lib/i686/libpthread.so.0 (0x402a2000)
libc.so.6 => /lib/i686/libc.so.6 (0x42000000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Those are name => value pairs showing the shared libraries used by the `httpd` binary.

Take note of the value for *libapr-0.so.0* and compare it to what you got in step 1. They should be the same, if not, then mod_perl was compiled pointing to the wrong Apache installation. You should run "make clean" and then

```
% perl Makefile.pl MP_APACHE_CONFIG=/path/to/apache/bin/apr-config
```

using the correct path for the Apache installation.

- **Step 4**

  You should also search for extra copies of *libapr-0.so.0*. If you find one in */usr/lib* or */usr/local/lib* that will explain the problem. Most likely you have an old pre-installed apr package which gets loaded before the copy you found in step 1.

  On most Linux (and Mac OS X) machines you can do a fast search with:

  ```
  % locate libapr-0.so.0
  ```

  which searches a database of files on your machine. The "locate" database isn't always up-to-date so a slower, more comprehensive search can be run (as root if possible):

  ```
  % find / -name "libapr-0.so.0*"
  ```

  or

```
% find /usr/local -name "libapr-0.so.0*"
```

You might get output like this:

```
/usr/local/apache2.0.47/lib/libapr-0.so.0.9.4
/usr/local/apache2.0.47/lib/libapr-0.so.0
/usr/local/apache2.0.45/lib/libapr-0.so.0.9.3
/usr/local/apache2.0.45/lib/libapr-0.so.0
```

in which case you would want to make sure that you are configuring and compiling mod_perl with the latest version of apache, for example using the above output, you would do:

```
% perl Makefile.PL MP_AP_CONFIG=/usr/local/apache2.0.47
% make
% make test
```

There could be other causes, but this example shows you how to act when you encounter this problem.

1.7 Issues with APR Used Outside of mod_perl

It doesn't strictly belong to this document, since it's talking about APR usages outside of mod_perl, so this may move to its own dedicated page, some time later.

Whenever using an `APR::` package outside of mod_perl, you need to:

```
use APR;
```

in order to load the XS subroutines. For example:

```
% perl -MApache2 -MAPR -MAPR::UUID -le 'print APR::UUID->new->format'
```

1.8 Maintainers

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

- **Stas Bekman**

1.9 Authors

- **Stas Bekman**

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

Table of Contents: