

1 mod_perl 1.0 Win32 Installation Instructions

1.1 Description

This document discusses how to install mod_perl 1.0 under Win32, both in building from sources and in installing pre-compiled binary packages.

1.2 Synopsis

Unless you are using an all-in-one package, you should first install Perl and Apache, either from the sources or as binaries. The Perl sources are available from <http://www.cpan.org/src/>, with directions for building contained in *README.win32*. ActiveState also makes the sources available for their binary builds at <ftp://ftp.activestate.com/ActivePerl/src/>, which may contain, in particular, Win32-specific fixes not in the CPAN Perl sources. As a binary, at present, an ActivePerl-compatible Perl, compiled with Visual C++, is the most common one used in the Win32 mod_perl/Apache environment; you can obtain such a prebuilt Perl binary from <http://www.activestate.com/>.

mod_perl 1 builds and tests successfully with either an ActivePerl Perl in the 6xx series, based on perl-5.6.1, or with an ActivePerl Perl in the 8xx series, based on perl-5.8.0 (for the latter, this requires mod_perl-1.29 or later). If you are using perl-5.8, you may want to consider mod_perl 2.0, which although still in a development phase offers several significant performance improvements for Win32 - see modperl-2 in Win32 for details.

The Apache sources and binaries are available at <http://httpd.apache.org/>.

When installing Perl or other related binaries, subtleties may arise in using path names that have spaces in them - you may, for example, have to specify *C:\Program Files* by the DOS 8.3 path name *C:\Progra~1* in certain Apache directives. If you want to avoid this, install, if possible, these packages to locations without spaces in their names (eg, *C:\Perl* for Perl and *C:\Apache* for Apache).

In the following, it may be necessary to invoke certain commands through a DOS prompt. A DOS window may be opened either through a *Command Prompt* option of the *Start* menu, or by choosing to run, from the *Start* menu, *command* or *cmd*, as appropriate.

1.3 Building from sources

You will need

- patience - mod_perl is considered alpha under Win32.
- MSVC++ 5.0+, Apache version 1.3-dev or higher and Perl 5.004_02 or higher.
- As of version 1.24_01, mod_perl will build on Win32 ActivePerls based on Perl-5.6.x (builds 6xx). For ActivePerl builds 8xx, you will need mod_perl-1.29 or later. For binary compatibility you should use the same compiler in building mod_perl that was used to compile your Perl binary; for ActivePerl, this means using VC++ 6.

First obtain the mod_perl 1.0 sources as a tar .gz file - when unpacked, using Winzip or similar tools, a subdirectory *mod_perl-1.xx* will be created.

There are two ways to build mod_perl - with MS Developer Studio, and through command-line arguments to 'perl Makefile.PL'. In both cases Apache should previously have been built and installed - if you are using a binary build of Apache, make sure that you obtain a binary build that includes the Apache libraries and header files. If you're building Apache yourself from sources, make sure to obtain the *win32-src.zip* archive, which has the necessary VC++ makefiles.

1.3.1 Building with MS Developer Studio

- **Setup the Perl side**

Run, from a DOS window in the top-level directory of the mod_perl sources,

```
C:\modperl_src> perl Makefile.PL
C:\modperl_src> nmake
```

This will set up the Perl side of mod_perl for the library build.

- **Build mod_perl.so**

Using MS developer studio,

```
select "File -> Open Workspace ...",
select "Files of type [Projects (*.dsp)]"
open mod_perl-x.xx/src/modules/win32/mod_perl.dsp
```

- **Settings**

```
select "Tools -> Options -> [Directories]"

select "Show directories for: [Include files]", and add

C:\Apache\include
. (should expand to C:\...\mod_perl-x.xx\src\modules\perl)
C:\Perl\lib\Core

select "Project -> Add to Project -> Files", adding:

perl.lib (or perl56.lib) (e.g. C:\perl\lib\Core\perl.lib)
ApacheCore.lib (e.g. C:\Apache\ApacheCore.lib)

select "Build -> Set Active Configuration -> [mod_perl - Win32 Release]"

select "Build -> Build mod_perl.so"
```

You may see some harmless warnings, which can be reduced (along with the size of the DLL), by setting:

```
"Project -> Settings -> [C/C++] -> Category: [Code Generation] ->
Use runtime library: [Multithreaded DLL]"
```

As well, if you are using a mod_ssl enabled Apache, you should add *EAPI* to the list of preprocessor definitions under

```
"Project -> Settings -> [C/C++]"
```

- **Testing**

Once mod_perl.so is built you may test mod_perl with:

```
C:\modperl_src> nmake test
```

after which, assuming the tests are OK,

```
C:\modperl_src> nmake install
```

will install the Perl side of mod_perl. The mod_perl.so file built under *mod_perl-1.xx/src/modules/win32/Release* should be copied to your Apache modules directory (eg, *C:\Apache\modules*).

1.3.2 Building with Makefile.PL arguments

Generating the Makefile as, for example,

```
C:\modperl_src> perl Makefile.PL APACHE_SRC=\Apache
```

will build mod_perl (including mod_perl.so) entirely from the command line. The arguments accepted include

- **APACHE_SRC**

This can be one of two values: either the path to the Apache build directory (eg, *..\apache_1.3.xx*), or to the installed Apache location (eg, *\Apache*). This is used to set the locations of ApacheCore.lib and the Apache header files.

- **INSTALL_DLL**

This gives the location of where to install mod_perl.so (eg, *\Apache\modules*). No default is assumed - if this argument is not given, mod_perl.so must be copied manually (in mod_perl-1.29 or later, INSTALL_DLL, if not supplied, will assume a default of *APACHE_SRC/modules*, if this directory exists).

- **INSTALL_LIB**

This option, which is available only in mod_perl-1.29 or later, gives the location of where to install mod_perl.lib (eg, *\Apache\libexec*). This library is needed for building certain 3rd party Apache modules. If this is not supplied, a default of *APACHE_SRC/libexec* will be assumed, if this directory exists.

- **DEBUG**

If true (DEBUG=1), a Debug version will be built (this assumes that a Debug Apache has been built). If false, or not given, a Release version will be built.

- **EAPI**

If true (EAPI=1), EAPI (Extended API) will be defined when compiling. This is useful when building mod_perl against mod_ssl patched Apache sources. If false, or not given, EAPI will not be defined.

After this, running

```
C:\modperl_src> nmake
C:\modperl_src> nmake test
C:\modperl_src> nmake install
```

will complete the installation.

This latter method of building mod_perl will also install the Apache and mod_perl header files, which can then be accessed through the Apache::src module.

If this build fails, you may want to try the sources obtained from svn - see the discussion on the Development Source Distribution for details. Be aware, though, that as well as providing bug fixes, there may be new features being added and tested in the svn versions, so at any given time there are no guarantees that these packages will build and test successfully.

1.4 Binaries

There are two major types of binary packages available for Win32 mod_perl - all-in-one Perl/Apache/mod_perl binaries, and mod_perl ppm (Perl Package Manager) packages.

1.4.1 All-in-one packages

There are a number of binary packages for Win32 that contain the necessary Perl and Apache binaries:

- IndigoPerl from <http://www.indigostar.com/>,
- XAMPP for Windows from <http://www.apachefriends.org/en/xampp-windows.html>
- DeveloperSide.NET for Windows at <http://www.devside.net/web/server/free/software>
- zangweb from <http://www.arbingersys.com/hostsites/zangweb/>

As well, there is a package *perl-win32-bin.exe* from <http://www.apache.org/dyn/closer.cgi/perl/win32-bin/> - see the file *perl-win32-bin.readme* for a description. If you have trouble fetching the whole file at once, the directory <http://www.apache.org/dyn/closer.cgi/perl/win32-bin/perl-win32-bin/> contains this distribution split across multiple files - see *README.join* for instructions on how to join them. Alternatively, if you have Perl already, you can get the script *distinstall*:

1.4.1 All-in-one packages

```
#####
# A Perl script to retrieve and join split files
# making up a Win32 Perl/Apache binary distribution
#
# Files created by hjsplit (http://www.freebyte.com/hjsplit/)
# with the joining accomplished by hj-join
#
# This script is Copyright 2003, by Randy Kobes,
# and may be distributed under the same terms as Perl itself.
# Please report problems to Randy Kobes #####

use strict;
use warnings;
use Net::FTP;
use Safe;
use Digest::MD5;
use IO::File;
use ExtUtils::MakeMaker;

die 'This is intended for Win32' unless ($^O =~ /Win32/i);

my $theoryx5 = 'theoryx5.uwinnipeg.ca';
my $bsize = 102400;
my $kb = sprintf("%d", $bsize / 1024);
my $cs = 'CHECKSUMS';
my $join = 'join32.exe';

print <<"END";

This script will fetch and then join the files needed for
creating and installing a Perl/Apache Win32 binary distribution from
ftp://$theoryx5/pub/other/.

If the file transfer is interrupted before all the necessary
files are obtained, run the script again in the same directory;
files successfully fetched earlier will not be downloaded again.

A hash mark represents transfer of $kb kB.

Available distributions are:

1. Perl 5.8.7 / Apache 2.0.54 / mod_perl-2.0.1
2. Perl 5.6.1 / Apache 1.3.27 / mod_perl 1.27

It is recommended to install Perl and Apache into fresh locations,
so that current files are not overwritten and that old files do
not linger which may confuse the new installation.

END

my $dist;
my $ans = prompt("Desired distribution (1, 2, or 'q' to quit)?", 1);
CHECK: {
    ($ans =~ /^q/i) and die 'Installation aborted';
    ($ans == 1) and do {
        $dist = 'Perl-5.8-win32-bin';
        last CHECK;
    };
    ($ans == 2) and do {
        $dist = 'perl-win32-bin';
        last CHECK;
    };
    die 'Please answer either 1, 2, or q';
}

my $exe = $dist . '.exe';

my $ftp = Net::FTP->new($theoryx5);
$ftp->login('anonymous', "$dist@perl.apache.org")
    or die "Cannot login to $theoryx5";
$ftp->cwd("pub/other/$dist")
```

```

    or die "Cannot cwd to pub/other/$dist";

my $max;
die "Unable to determine number of files to get" unless ($max = get_max());
my @files = ();

# fetch the CHECKSUMS file
print qq{Fetching "$cs" ...};
$ftp->ascii;
$ftp->get($cs);
print " done!\n";
die qq{Failed to fetch "$cs"} unless (-e $cs);
push @files, $cs;

# evaluate CHECKSUMS
my $cksum;
die qq{Cannot load "$cs" file} unless ($cksum = load_cs($cs));

$ftp->binary;
$ftp->hash(1, $bsize);

# fetch the join program
die qq{Cannot fetch "$join"} unless (fetch($join));
push @files, $join;

# fetch the split files
print "\nFetching $max split files ....\n\n";
for (1 .. $max) {
    my $num = $_ < 10 ? "00$_" : "0$_";
    my $file = $dist . '.exe.' . $num;
    push @files, $file;
    die qq{Cannot fetch "$file"} unless (fetch($file));
}
print "\nFinished fetching split files.\n";
$ftp->quit;

# now join them
if (-e $exe) {
    unlink($exe) or warn qq{Cannot unlink $exe: $!};
}
my @args = ($join);
system(@args);
die qq{Joining files to create "$exe" failed} unless (-e $exe);

# remove the temporary files, if desired
$sans = prompt('Remove temporary files?', 'yes');
if ($sans =~ /^y/i) {
    unlink(@files) or warn "Cannot unlink temporary files: $!\n";
}

# run the exe, if desired
$sans = prompt("Run $exe now?", 'yes');
if ($sans =~ /^y/i) {
    @args = ($exe);
    system(@args);
}
else {
    print "\nDouble click on $exe to install.\n";
}

# fetch a file, unless it exists and the checksum checks
sub fetch {
    my $file = shift;
    local $| = 1;
    if (-e $file) {
        if (verifyMD5($file)) {
            print qq{Skipping "$file" ...}\n;
            return 1;
        }
    }
    else {
        unlink $file or warn qq{Could not unlink "$file"\n};
    }
}

```

1.4.1 All-in-one packages

```
    }
  }
  my $size = sprintf("%d", $ftp->size($file) / 1024);
  print "\nFetching $file ($size kB) ... \n";
  $ftp->get($file);
  print "Done! \n";
  unless (-e $file) {
    warn qq{Unable to fetch "$file" \n};
    return;
  }
  unless (verifyMD5($file)) {
    print qq{CHECKSUM check for "$file" failed. \n};
    unlink $file or warn qq{Cannot unlink "$file": $! \n};
    return;
  }
  return 1;
}

# routines to verify the CHECKSUMS for a file
# adapted from the MD5 check of CPAN.pm

# load the CHECKSUMS file into $cksum
sub load_cs {
  my $cs = shift;
  my $fh = IO::File->new;
  unless ($fh->open($cs)) {
    warn qq{Could not open "$cs": $! \n};
    return;
  }
  local($/);
  my $eval = <$fh>;
  $fh->close;
  $eval =~ s/\015?\012/\n/g;
  my $comp = Safe->new();
  my $cksum = $comp->reval($eval);
  if ($@) {
    warn qq{eval of "$cs" failed: $@ \n};
    return;
  }
  return $cksum;
}

# verify a CHECKSUM for a file
sub verifyMD5 {
  my $file = shift;
  my ($is, $should);
  my $fh = IO::File->new;
  unless ($fh->open($file)) {
    warn qq{Cannot open "$file": $!};
    return;
  }
  binmode($fh);
  unless ($is = Digest::MD5->new->addfile($fh->hexdigest) {
    warn qq{Could not compute checksum for "$file": $!};
    $fh->close;
    return;
  }
  $fh->close;
  if ($should = $cksum->{$file}->{md5}) {
    my $test = ($is eq $should);
    printf qq{ Checksum for "$file" is %s \n},
      ($test) ? 'OK.' : 'NOT OK.';
    return $test;
  }
  else {
    warn qq{Checksum data for "$file" not present in $cs. \n};
    return;
  }
}

# get number of split files
```

```

sub get_max {
    my $dir = $ftp->ls();
    my $count = 0;
    foreach (@$dir) {
        $count++ if m!$dist.exe.\d+$!;
    }
    return $count;
}

```

which, when invoked as `perl distinstall`, will fetch and join the files for you.

As well as including a number of non-core modules, both of these packages contain `mod_perl`. See the documentation on the web sites and that included with the packages for installation instructions. Both of these also include an ActiveState-compatible `ppm` (Perl Package Manager) utility for adding and upgrading modules.

For the adventuresome who want a taste of things to come, or for those who want to avoid the multithreading limitations of `mod_perl 1.0`, a `mod_perl-2.0/Apache-2.0` binary distribution is available - see the discussion of `modperl-2` on Win32 for details. Be aware though that `mod_perl 2.0` is still in a development phase, and that a minimum Perl version of 5.8 (ActivePerl 8xx) is required.

1.4.2 PPM Packages

For ActivePerl users (or compatible), there are also PPM `mod_perl` packages available. For this, if you don't already have it, get and install the latest Win32 Apache binary from <http://httpd.apache.org/>.

Both ActivePerl and Apache binaries are available as MSI files for use by the Microsoft Installer - as discussed on the ActiveState site, users of Windows 95 and 98 may need to obtain this. In installing these packages, you may find it convenient when transcribing any Unix-oriented documentation to choose installation directories that do not have spaces in their names (eg, `C:\Perl` and `C:\Apache`).

After installing Perl and Apache, you can then install `mod_perl` via the PPM utility. ActiveState does not maintain `mod_perl` in their `ppm` repository, so you must get it from a different location other than ActiveState's site. A quick way to do this is to download the script `mpinstall`:

```

#!C:/Perl/bin
#####
# A Perl script to fetch and install via ppm mod_perl on Win32
# Copyright 2002, by Randy Kobes.
# This script may be distributed under the same terms as Perl itself.
# Please report problems to Randy Kobes #####

use strict;
use warnings;
use ExtUtils::MakeMaker;
use LWP::Simple;
use File::Copy;
use Config;
use Safe;
use Digest::MD5;
require Win32;
require File::Spec;

die "This only works for Win32" unless $^O =~ /Win32/i;
die "No mod_perl ppm package available for this Perl" if ($] < 5.006001);

my ($apache2, $apache, $apache22);
my @drives = drives();

```

1.4.2 PPM Packages

```
# find a possible Apache2 directory
APACHE2: {
    for my $drive (@drives) {
        for my $p ('Apache2', 'Program files/Apache2',
                  'Program Files/Apache Group/Apache2') {
            my $candidate = File::Spec->catpath($drive, $p);
            if (-d $candidate) {
                $apache2 = $candidate;
                last APACHE2;
            }
        }
    }
}
if ($apache2) {
    $apache2 = fix_path($apache2);
    my $ans = prompt(qq{Install mod_perl-2 for "$apache2"?}, 'yes');
    $apache2 = undef unless ($ans =~ /^y/i);
}

# if no Apache2, try to find Apache1
unless ($apache2) {
    APACHE: {
        for my $drive (@drives) {
            for my $p ('Apache', 'Program Files/Apache',
                      'Program Files/Apache Group/Apache') {
                my $candidate = File::Spec->catpath($drive, $p);
                if (-d $candidate) {
                    $apache = $candidate;
                    last APACHE;
                }
            }
        }
    }
}
if ($apache) {
    $apache = fix_path($apache);
    my $ans = prompt(qq{Install mod_perl 1 for "$apache"?}, 'yes');
    $apache = undef unless ($ans =~ /^y/i);
}

# check Apache versions
if ($apache or $apache2) {
    my $vers;
    if ($apache) {
        $vers = qx{"$apache/apache.exe" -v};
        die qq{"$apache" does not appear to be version 1.3}
            unless $vers =~ m!Apache/1.3!;
    }
    else {
        my $vers;
        for my $binary(qw(Apache.exe httpd.exe)) {
            my $b = File::Spec->catfile($apache2, 'bin', $binary);
            next unless -x $b;
            $vers = qx{"$b" -v};
            last;
        }
        die qq{Cannot determine the Apache version} unless $vers;
        die qq{"$apache2" does not appear to be version 2.x}
            unless $vers =~ m!Apache/2.!;
        $apache22 = 1 if $vers =~ m!Apache/2.2!;
    }
}
# prompt to get an Apache installation directory
else {
    my $dir = prompt("Where is your apache installation directory?", '');
    die 'Need to specify the Apache installation directory' unless $dir;
    $dir = fix_path($dir);
    die qq{"$dir" does not exist} unless (-d $dir);
    if ($dir =~ /Apache2/) {
        my $ans = prompt(qq{Install mod_perl 2 for "$dir"?}, 'yes');
    }
}
```

```

    $apache2 = $dir if ($ans =~ /^y/i);
}
else {
    my $ans = prompt(qq{Install mod_perl 1 for "$dir?"}, 'yes');
    $apache = $dir if ($ans =~ /^y/i);
}
unless ($apache or $apache2) {
    my $mpv = prompt('Which mod_perl version would you like [1 or 2]?', 2);
    if ($mpv == 1) {
        $apache = $dir;
    }
    elsif ($mpv == 2) {
        $apache2 = $dir;
    }
    else {
        die 'Please specify either "1" or "2"';
    }
}
}

die 'Please specify an Apache directory' unless ($apache or $apache2);
my $theoryx5 = 'http://theoryx5.uwinnipeg.ca';
my $ppms = $theoryx5 . '/ppms/';
my $ppmsx86 = $ppms . 'x86/';
my $ppmpackages = $theoryx5 . '/ppmpackages/';
my $ppmpackagesx86 = $ppmpackages . 'x86/';
my ($ppd, $tgz, $ppdfile, $tgzfile, $checksums, $so_fetch, $so_fake);
my $so = 'mod_perl.so';
my $cs = 'CHECKSUMS';

# set appropriate ppd and tar.gz files
if ($] < 5.008) {
    $checksums = $ppmpackagesx86 . $cs;
    if ($apache2) {
        die 'No mod_perl 2 package available for this perl version';
    }
    else {
        my $ans = prompt('Do you need EAPI support for mod_ssl?', 'no');
        if ($ans =~ /^n/i) {
            $ppdfile = 'mod_perl.ppd';
            $tgzfile = 'mod_perl.tar.gz';
            $so_fake = 'mod_perl.so';
        }
        else {
            $ppdfile = 'mod_perl-eapi.ppd';
            $tgzfile = 'mod_perl-eapi.tar.gz';
            $so_fake = 'mod_perl-eapi.so';
        }
        $ppd = $ppmpackages . $ppdfile;
        $tgz = $ppmpackagesx86 . $tgzfile;
        $so_fetch = $ppmpackagesx86 . $so_fake;
    }
}
else {
    $checksums = $ppmsx86 . $cs;
    if ($apache2) {
        my $ans = prompt('Do you want the latest mod_perl 2 development version?', 'no');
        if ($ans =~ /^n/i) {
            if ($apache22) {
                $ppdfile = 'mod_perl.ppd';
                $tgzfile = 'mod_perl.tar.gz';
                $so_fake = 'mod_perl.so';
            }
            else {
                $ppdfile = 'mod_perl-2.0.ppd';
                $tgzfile = 'mod_perl-2.0.tar.gz';
                $so_fake = 'mod_perl-2.0.so';
            }
        }
    }
    else {
        $ppdfile = 'mod_perl-dev.ppd';
    }
}

```

1.4.2 PPM Packages

```
        $tgzfile = 'mod_perl-dev.tar.gz';
        $so_fake = 'mod_perl-dev.so';
    }
    $ppd = $ppms . $ppdfile;
    $tgz = $ppmsx86 . $tgzfile;
    $so_fetch = $ppmsx86 . $so_fake;
}
else {
    my $ans = prompt('Do you need EAPI support for mod_ssl?', 'no');
    if ($ans =~ /^n/i) {
        $ppdfile = 'mod_perl-1.ppd';
        $tgzfile = 'mod_perl-1.tar.gz';
        $so_fake = 'mod_perl-1.so';
    }
    else {
        $ppdfile = 'mod_perl-eapi-1.ppd';
        $tgzfile = 'mod_perl-eapi-1.tar.gz';
        $so_fake = 'mod_perl-eapi-1.so';
    }
    $ppd = $ppms . $ppdfile;
    $tgz = $ppmsx86 . $tgzfile;
    $so_fetch = $ppmsx86 . $so_fake;
}
}

my $tmp = $ENV{TEMP} || $ENV{TMP} || '.';
chdir $tmp or die "Cannot chdir to $tmp: $!";

# fetch the ppd and tar.gz files
print "Fetching $ppd ...";
getstore($ppd, $ppdfile);
print " done!\n";
die "Failed to fetch $ppd" unless -e $ppdfile;
print "Fetching $tgz ...";
getstore($tgz, $tgzfile);
print " done!\n";
die "Failed to fetch $tgz" unless -e $tgzfile;
print "Fetching $so_fetch ...";
getstore($so_fetch, $so_fake);
print " done!\n";
die "Failed to fetch $so_fetch" unless -e $so_fake;
print "Fetching $checksums ...";
getstore($checksums, $cs);
print " done!\n";
die "Failed to fetch $checksums" unless -e $cs;

# check CHECKSUMS for the tar.gz and so files
my $cksum = load_cs($cs);
die "Could not load $cs: $!" unless $cksum;
die qq{CHECKSUM check for "$tgzfile" failed.\n}
unless (verifyMD5($cksum, $tgzfile));
die qq{CHECKSUM check for "$so_fake" failed.\n}
unless (verifyMD5($cksum, $so_fake));
unless ($so_fake eq $so) {
    rename($so_fake, $so) or die "Rename of $so_fake to $so failed: $!";
}

# edit the ppd file to reflect a local installation
my $old = $ppdfile . '.old';
rename ($ppdfile, $old)
    or die "renaming $ppdfile to $old failed: $!";
open(my $oldfh, $old) or die "Cannot open $old: $!";
open(my $newfh, ">$ppdfile") or die "Cannot open $ppdfile: $!";
while (<$oldfh>) {
    next if /;
    s/$tgz/$tgzfile/;
    print $newfh $_;
}
close $oldfh;
close $newfh;
```

```

# install mod_perl via ppm
my $ppm = $Config{bin} . '\ppm';
my @args = ($ppm, 'install', $ppdfile);
print "\n@args\n";
system(@args) == 0 or die "system @args failed: $?";

# figure out where to place mod_perl.so
my $modules = $apache ? "$apache/modules" : "$apache2/modules";
$modules = prompt("Where should $so be placed?", $modules);
die "Please install $so to your Apache modules directory manually"
    unless $modules;
$modules = fix_path($modules);

unless (-d $modules) {
    my $ans = prompt(qq{"$modules" does not exist. Create it?}, 'yes');
    if ($ans =~ /^y/i) {
        mkdir $modules or die "Cannot create $modules: $!";
    }
    else {
        $modules = undef;
    }
}
# move mod_perl.so to the Apache modules directory
if ($modules) {
    print "\nMoving $so to $modules ...";
    move($so, qq{$modules})
        or die "Moving $so to $modules failed: $!";
    print " done!\n";
}
else {
    die "Please install $so to your Apache modules directory manually"
}

# clean up, if desired
my $ans = prompt("Remove temporary installation files from $tmp?", 'yes');
if ($ans =~ /^y/i) {
    unlink ($ppdfile, $old, $tgzfile, $cs, $so)
        or warn "Cannot unlink files from $tmp: $!";
}

# get the name and location of the perlxx.dll
(my $dll = $Config{libperl}) =~ s!\.lib$.dll!;
$dll = $Config{bin} . '/' . $dll;
$dll =~ s!\\!\/!g;

# suggest a minimal httpd.conf configuration
my $ap = $apache || $apache2;
print <<"END";

mod_perl was successfully installed.
To try it out, put the following directives in your
Apache httpd.conf file (under $ap/conf):

LoadFile "$dll"
LoadModule perl_module modules/$so

in the section where other apache modules are loaded.
You may also have to add $Config{bin}
to your PATH environment variable.

For more information, visit http://perl.apache.org/.

END

# routine to verify the CHECKSUMS for a file
# adapted from the MD5 check of CPAN.pm

sub load_cs {
    my $cs = shift;
    my ($cksum, $fh);
    unless (open $fh, $cs) {

```

1.4.2 PPM Packages

```
        warn "Could not open $cs: $!";
        return;
    }
    local($/);
    my $eval = <$fh>;
    $eval =~ s/\015?\012/\n/g;
    close $fh;
    my $comp = Safe->new();
    $cksum = $comp->reval($eval);
    if ($@) {
        warn $@;
        return;
    }
    return $cksum;
}

sub verifyMD5 {
    my ($cksum, $file) = @_;
    my ($fh, $is, $should);
    unless (open($fh, $file)) {
        warn "Cannot open $file: $!";
        return;
    }
    binmode($fh);
    unless ($is = Digest::MD5->new->addfile($fh->hexdigest) {
        warn "Could not compute checksum for $file: $!";
        close($fh);
        return;
    }
    close($fh);
    if ($should = $cksum->{$file}->{md5}) {
        my $test = $is eq $should ? 1 : 0;
        printf qq{Checksum for "$file" is %s\n},
            ($test == 1) ? 'OK.' : 'NOT OK.';
        return $test;
    }
    else {
        warn "Checksum data for $file not present in CHECKSUMS.\n";
        return;
    }
}

sub fix_path {
    my $file = shift;
    $file = Win32::GetShortPathName($file);
    $file =~ s!\\!/\!g;
    return $file;
}

sub drives {
    my @drives = ();
    eval{require Win32API::File;};
    return map {"$_:\""} ('C' .. 'Z') if $@;
    my @r = Win32API::File::getLogicalDrives();
    return unless @r > 0;
    for (@r) {
        my $t = Win32API::File::GetDriveType($_);
        push @drives, $_ if ($t == 3 or $t == 4);
    }
    return @drives > 0 ? @drives : undef;
}
```

and save it as, for example, *mpinstall*. Invoking this as `perl mpinstall` on a command line will take you through a dialogue, based on your configuration, which will determine and install, via ppm, the desired `mod_perl` ppm package.

The direct way to install mod_perl via ppm is simply as (broken over two lines for readability)

```
C:\> ppm install
      http://theoryx5.uwinnipeg.ca/ppmpackages/mod_perl.ppd
```

for Activeperl 6xx builds, and as

```
C:\> ppm install
      http://theoryx5.uwinnipeg.ca/ppms/mod_perl-1.ppd
```

for 8xx builds. Another way, which will be useful if you plan on installing additional Apache modules, is to add the repository where the mod_perl package is kept to the ppm shell utility. For ppm2 this may be done with the `set repository alias location` command, while for ppm3 (the default with ActivePerl 8xx) the appropriate command is `repository add alias location`; see the help utility within the ppm shell for details. For 6xx builds, the appropriate location is

```
http://theoryx5.uwinnipeg.ca/cgi-bin/ppmserver?urn:/PPMServer
```

while for for 8xx builds it is

```
http://theoryx5.uwinnipeg.ca/cgi-bin/ppmserver?urn:/PPMServer58
```

After this, you can, within the ppm shell, use the `install` command to either install `mod_perl`, for 6xx, or `mod_perl-1`, for 8xx. For ppm2, use the `set save` command to save the theoryx5 repository to your PPM configuration file, so that future PPM sessions will search this repository, as well as ActiveState's, for requested packages. If you are running mod_ssl under Apache, then you should obtain the `mod_perl-eapi` package for 6xx or the `mod_perl-eapi-1` package for 8xx instead.

Note that, because of binary incompatibilities, one should *not* install packages for ActivePerl 8xx from a repository containing packages for ActivePerl 6xx, and vice-versa, particularly if these packages contain XS-based modules.

The mod_perl PPM package also includes the necessary Apache DLL `mod_perl.so`; a post-installation script should be run which will offer to copy this file to your Apache modules directory (eg, `C:\Apache\modules`). If this fails, you can grab the appropriate dll and install it manually. For 6xx builds, this is at <http://theoryx5.uwinnipeg.ca/ppmpackages/x86/>, for which the relevant file is either `mod_perl.so` or, for EAPI support, `mod_perl-eapi.so`. For 8xx builds, the location is <http://theoryx5.uwinnipeg.ca/ppms/x86/>, for which the relevant file is either `mod_perl-1.so` or, for EAPI support, `mod_perl-eapi-1.so`. You should then copy this file to your Apache modules directory and rename it as `mod_perl.so`, if necessary.

The mod_perl package available from this site will always use the latest mod_perl sources compiled against the latest official Apache release; depending on changes made in Apache, you may or may not be able to use an earlier Apache binary. However, in the Apache Win32 world it is particularly a good idea to use the latest version, for bug and security fixes. If you encounter problems in loading `mod_perl.so`, ensure that the mod_perl version you are using matches that of Apache, make certain Perl is in your PATH environment variable, or try adding the Apache directive

1.5 See Also

```
LoadFile "C:/Path/to/your/Perl/bin/perlxx.dll"
```

before loading *mod_perl.so*. If all else fails, a reboot may help.

If the *theoryx5.uwinnipeg.ca* repository is down, you can access these packages at <http://www.apache.org/dyn/closer.cgi/perl/win32-bin/ppms/>, for builds 8xx, and <http://www.apache.org/dyn/closer.cgi/perl/win32-bin/ppmpackages/>, for builds 6xx.

1.5 See Also

The directions for configuring *mod_perl* 1.0 on Win32, the *mod_perl* documentation, and the FAQs for *mod_perl* on Win32. Help is also available through the archives of and subscribing to the *mod_perl* mailing list.

1.6 Maintainers

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

- Randy Kobes <randy@theoryx5.uwinnipeg.ca>

1.7 Authors

- Randy Kobes <randy@theoryx5.uwinnipeg.ca>

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

Table of Contents:

1	mod_perl 1.0 Win32 Installation Instructions	1
1.1	Description	2
1.2	Synopsis	2
1.3	Building from sources	2
1.3.1	Building with MS Developer Studio	3
1.3.2	Building with Makefile.PL arguments	4
1.4	Binaries	5
1.4.1	All-in-one packages	5
1.4.2	PPM Packages	9
1.5	See Also	16
1.6	Maintainers	16
1.7	Authors	16