1 mod_perl 2.0 Win32 Installation Instructions

1.1 Description

This document deals with installation specifics on Win32 platforms.

1.2 Synopsis

As described in the discussion of issues in multithreaded win32, a mod_perl 1.0 enabled server based on Apache 1.3 on Win32 is limited to a single thread serving a request at a time. This effectively prevents concurrent processing, which can have serious implications for busy sites. This problem is addressed in the multi-thread/multi-process approach of mod_perl 2.0/Apache 2.x, which consequently requires a Perl built with ithreads enabled.

There are some threading issues in perl-5.6 (upon which ActivePerl builds 6xx are based) which cause problems with mod_perl 2.0 on Win32. Consequently, the minimum required perl version is 5.8 (upon which ActivePerl builds 8xx are based) for use with mod_perl 2.0 on Win32.

1.3 Installing

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. If you are building Perl from source then you must enable both USE_MULTI and USE_ITHREADS in the Makefile to enable ithreads, as required for mod_perl 2.0 on Win32. ActivePerl builds also enable USE_IMP_SYS, providing the implicit "host" layer which gives a `fork()` emulation, but this is at the cost of disabling PERL_MALLOC which may have significant performance implications since Win32's system `malloc()` is notably slower than Perl's in some situations. Thus, unless you require the `fork()` emulation or specifically want an ActivePerl- compatible build then you may want to disable USE_IMP_SYS and enable PERL_MALLOC. (Note that you cannot currently enable PERL_MALLOC with USE_IMP_SYS enabled as well.) 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/.

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

As of this writing, mod_perl 2.0 is known to compile and work with both an ActivePerl-compatible perl-5.8 (ActivePerl build 8xx) and with an otherwise similar Perl having USE_IMP_SYS disabled and PERL_MALLOC enabled. See the section on Apache/mod_perl binaries below for details on a suitable repository containing mod_perl ppm packages, and also how to obtain other Win32 binary packages.

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:\Apache2* for Apache 2.0).

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.

1.3.1 Building from sources

If you are building mod_perl 2.0 from sources, it is probably also best to do the same for Apache 2.0. The Apache 2.0 sources can be obtained from http://httpd.apache.org/, which when unpacked will contain at the top-level a Visual Studio project file (make sure to obtain the *win32-src.zip* archive). Choose the `InstallBin - Win32 Release` target to build and install Apache 2.0, which by default will be placed in */Apache2*. At the present time you must have version 2.0.47 or greater of Apache2 in order to build mod_perl.

Having built and installed Apache 2.0, next obtain the mod_perl 2.0 sources. First obtain the mod_perl 2.0 sources as a `tar.gz` file - when unpacked, using Winzip or similar tools, a subdirectory *mod_perl-x.xx* will be created. Next, run the command

```
C:\modperl_src> perl Makefile.PL MP_AP_PREFIX=\Path\to\Apache2
```

Then

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

will build and test mod_perl 2.0. mod_perl 2.0 on Win32 is considered at an alpha stage of development, so not all the tests may pass.

The final command,

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

will install the necessary mod_perl 2.0 files into your Perl directory tree, and also copy *src/modules/perl/mod_perl.so* into your */Path/to/Apache2/modules/* directory.

If this build fails, or you want features not present in the official releases, you may want to try the sources obtained from svn - see the discussion on the 2.0 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.

1.3.2 PPM Packages

The following assumes you already have ActivePerl 8xx (*not* 6xx) from http://www.activestate.com/ and a Win32 Apache 2.x binary from http://httpd.apache.org/. In installing this, you might avoid some future problems by choosing installation directories that do not have spaces in their names (eg, *C:/Apache2*). At this time you must have version 2.0.47 or greater of Apache2 in order to install the mod_perl 2 ppm package.

After installing Perl and Apache 2.x, 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();

# 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';
            $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 /;
    $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/ppms/mod_perl.ppd
```

for Apache/2.2, and

```
C:\> ppm install
     http://theoryx5.uwinnipeg.ca/ppms/mod_perl-2.0.ppd
```

for Apache/2.0. Another way, which will be useful if you plan on installing additional Apache modules, is to add the following repository within `ppm`:

```
http://theoryx5.uwinnipeg.ca/ppms/
```

for ActivePerl 819 or higher, or

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

for ActivePerl 818 or lower; see the help utility within `ppm` for details on how to do this. mod_perl 2.0 can then be installed, within the ppm shell, as

```
ppm> install mod_perl
```

for Apache/2.2, and as

```
ppm> install mod_perl-2.0
```

for Apache/2.0. This will install the necessary modules under an *Apache2* subdirectory in your Perl tree, so as not to disturb a possible existing *Apache* directory from mod_perl 1.0. See the section below on configuring mod_perl to add this directory to the `@INC` path for searching for modules.

The preceding http://theoryx5.uwinnipeg.ca/ppms/ repository is appropriate for ActivePerl 8xx builds, based on perl-5.8. If you're using an ActivePerl 10xx build, based on perl-5.10, you can install mod_perl via

```
C:\> ppm install
     http://cpan.uwinnipeg.ca/PPMPackages/10xx/mod_perl.ppd
```

for Apache/2.2. The corresponding repository that can be added to ppm is

```
http://cpan.uwinnipeg.ca/PPMPackages/10xx/
```

after which mod_perl can be installed as

```
ppm> install mod_perl
```

Note that ActivePerl 8xx and ActivePerl 10xx are not binary compatible, which means ppm packages compiled, for example, for an 8xx build are not compatible with ActivePerl 10xx.

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 Apache2 modules directory (eg, *C:/Apache2/modules/*). If this fails, you can get *mod_perl.so* from http://theoryx5.uwinnipeg.ca/ppms/x86/ and install it to your Apache2 modules directory by hand.

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. Also note that modules compiled under Apache/2.0 are not compatible with modules compiled under Apache/2.2, so be sure to install the mod_perl ppm package appropriate for your version of Apache/2.x.

The mod_perl package available from this site will always use the latest mod_perl sources available from CPAN 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 want to try a later development version of mod_perl 2, get the *mod_perl-dev.ppd* ppm package instead; the development version may contain bug fixes that were found since the last CPAN release, but it may also contain experimental features that have not been fully tested.

If you encounter problems loading *mod_perl.so*, ensure that the mod_perl version you are using matches that of Apache, make sure you are using at least Apache/2.0.47, and also make certain `Perl` is in your `PATH` environment variable or try adding the Apache directive

```
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.

1.3.3 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, at http://www.apache.org/dyn/closer.cgi/perl/win32-bin/ there is a package *Perl-5.8-win32-bin.exe* containing a binary version of perl-5.8 (compatible with ActivePerl 8xx), together with Apache 2.0 and mod_perl 2.0. See the file *Perl-5.8-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-5.8-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*:

```
######################################################################
# 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 neccessary
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
$ans = prompt('Remove temporary files?', 'yes');
if ($ans =~ /^y/i) {
  unlink(@files) or warn "Cannot unlink temporary files: $!\n";
```

```
}

# run the exe, if desired
$ans = prompt("Run $exe now?", 'yes');
if ($ans =~ /^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};
    }
  }
  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.

1.4 See Also

The directions for configuring mod_perl 2.0 on Win32, the mod_perl documentation, http://httpd.apache.org/, http://www.activestate.com/, and the FAQs for mod_perl on Win32. Help is also available through the archives of and subscribing to the mod_perl mailing list.

1.5 Maintainers

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

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

1.6 Authors

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

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

Table of Contents: