NAME

File::chdir – a more sensible way to change directories

VERSION

version 0.1008

SYNOPSIS

```
use File::chdir;

$CWD = "/foo/bar";      # now in /foo/bar
{
    local $CWD = "/moo/baz";  # now in /moo/baz
    ...
}

# still in /foo/bar!
```

DESCRIPTION

Perl's `chdir()` has the unfortunate problem of being very, very, very global. If any part of your program calls `chdir()` or if any library you use calls `chdir()`, it changes the current working directory for the **whole** program.

This sucks.

File::chdir gives you an alternative, `$CWD` and `@CWD`. These two variables combine all the power of `chdir()`, File::Spec and Cwd.

`$CWD`

Use the `$CWD` variable instead of `chdir()` and Cwd.

```
use File::chdir;
$CWD = $dir;  # just like chdir($dir)!
print $CWD;   # prints the current working directory
```

It can be localized, and it does the right thing.

```
$CWD = "/foo";      # it's /foo out here.
{
    local $CWD = "/bar";  # /bar in here
}
# still /foo out here!
```

`$CWD` always returns the absolute path in the native form for the operating system.

`$CWD` and normal `chdir()` work together just fine.

`@CWD`

`@CWD` represents the current working directory as an array, each directory in the path is an element of the array. This can often make the directory easier to manipulate, and you don't have to fumble with `File::Spec->splitpath` and `File::Spec->catdir` to make portable code.

```
# Similar to chdir("/usr/local/src/perl")
@CWD = qw(usr local src perl);
```

pop, push, shift, unshift and splice all work. pop and push are probably the most useful.

```
pop @CWD;                 # same as chdir(File::Spec->updir)
push @CWD, 'some_dir'     # same as chdir('some_dir')
```

`@CWD` and `$CWD` both work fine together.

NOTE Due to a perl bug you can't localize `@CWD`. See "CAVEATS" for a work around.

EXAMPLES

(We omit the `use File::chdir` from these examples for terseness)

Here's `$CWD` instead of `chdir()`:

```
$CWD = 'foo';           # chdir('foo')
```

and now instead of Cwd.

```
print $CWD;             # use Cwd;  print Cwd::abs_path
```

you can even do zsh style `cd foo bar`

```
$CWD = '/usr/local/foo';
$CWD =~ s/usr/var/;
```

if you want to localize that, make sure you get the parens right

```
{
    (local $CWD) =~ s/usr/var/;
    ...
}
```

It's most useful for writing polite subroutines which don't leave the program in some strange directory:

```
sub foo {
    local $CWD = 'some/other/dir';
    ...do your work...
}
```

which is much simpler than the equivalent:

```
sub foo {
    use Cwd;
    my $orig_dir = Cwd::getcwd;
    chdir('some/other/dir');

    ...do your work...

    chdir($orig_dir);
}
```

`@CWD` comes in handy when you want to start moving up and down the directory hierarchy in a cross-platform manner without having to use File::Spec.

```
pop @CWD;                   # chdir(File::Spec->updir);
push @CWD, 'some', 'dir'    # chdir(File::Spec->catdir(qw(some dir)));
```

You can easily change your parent directory:

```
# chdir from /some/dir/bar/moo to /some/dir/foo/moo
$CWD[-2] = 'foo';
```

CAVEATS

`local @CWD` does not work.

`local @CWD` will not localize `@CWD`. This is a bug in Perl, you can't localize tied arrays. As a work around localizing `$CWD` will effectively localize `@CWD`.

```
{
    local $CWD;
    pop @CWD;
    ...
}
```

Assigning to `@CWD` calls `chdir()` for each element

```
@CWD = qw/a b c d/;
```

Internally, Perl clears `@CWD` and assigns each element in turn. Thus, this code above will do this:

```
chdir 'a';
chdir 'a/b';
chdir 'a/b/c';
chdir 'a/b/c/d';
```

Generally, avoid assigning to `@CWD` and just use push and pop instead.

Volumes not handled

There is currently no way to change the current volume via File::chdir.

NOTES

`$CWD` returns the current directory using native path separators, i.e. \ on Win32. This ensures that `$CWD` will compare correctly with directories created using File::Spec. For example:

```
my $working_dir = File::Spec->catdir( $CWD, "foo" );
$CWD = $working_dir;
doing_stuff_might_chdir();
is( $CWD, $working_dir, "back to original working_dir?" );
```

Deleting the last item of `@CWD` will act like a pop. Deleting from the middle will throw an exception.

```
delete @CWD[-1]; # OK
delete @CWD[-2]; # Dies
```

What should `%CWD` do? Something with volumes?

```
# chdir to C:\Program Files\Sierra\Half Life ?
$CWD{C} = '\\Program Files\\Sierra\\Half Life';
```

DIAGNOSTICS

If an error is encountered when changing `$CWD` or `@CWD`, one of the following exceptions will be thrown:

- *Can't delete except at the end of `@CWD`*
- *Failed to change directory to '$dir'*

HISTORY

Michael wanted `local chdir` to work. p5p didn't. But it wasn't over! Was it over when the Germans bombed Pearl Harbor? Hell, no!

Abigail and/or Bryan Warnock suggested the `$CWD` thing (Michael forgets which). They were right.

The `chdir()` override was eliminated in 0.04.

David became co-maintainer with 0.06_01 to fix some chronic Win32 path bugs.

As of 0.08, if changing `$CWD` or `@CWD` fails to change the directory, an error will be thrown.

SEE ALSO

File::pushd, File::Spec, Cwd, "chdir" in perlfunc, "Animal House" <http://www.imdb.com/title/tt0077975/quotes>

SUPPORT

Bugs / Feature Requests

Please report any bugs or feature requests through the issue tracker at <https://rt.cpan.org/Public/Dist/Display.html?Name=File-chdir>. You will be notified automatically of any progress on your issue.

Source Code

This is open source software. The code repository is available for public review and contribution under the terms of the license.

<https://github.com/dagolden/file-chdir>

```
git clone git://github.com/dagolden/file-chdir.git
```

AUTHORS

- David A Golden <dagolden@cpan.org>
- Michael G Schwern <schwern@pobox.com> (original author)

COPYRIGHT AND LICENSE