NAME

File::stat - by-name interface to Perl's built-in stat() functions

SYNOPSIS

```
use File::stat;
$st = stat($file) or die "No $file: $!";
if ( ($st->mode & 0111) && $st->nlink > 1) ) {
    print "$file is executable with lotsa links\n";
}

if ( -x $st ) {
    print "$file is executable\n";
}

use Fcntl "S_IRUSR";
if ( $st->cando(S_IRUSR, 1) ) {
    print "My effective uid can read $file\n";
}

use File::stat qw(:FIELDS);
stat($file) or die "No $file: $!";
if ( ($st_mode & 0111) && ($st_nlink > 1) ) {
    print "$file is executable with lotsa links\n";
}
```

DESCRIPTION

This module's default exports override the core stat() and lstat() functions, replacing them with versions that return "File::stat" objects. This object has methods that return the similarly named structure field name from the stat(2) function; namely, dev, ino, mode, nlink, uid, gid, rdev, size, atime, mtime, ctime, blksize, and blocks.

As of version 1.02 (provided with perl 5.12) the object provides `"-X"` overloading, so you can call filetest operators (`-f`, `-x`, and so on) on it. It also provides a `->cando` method, called like

```
$st->cando( ACCESS, EFFECTIVE )
```

where *ACCESS* is one of `S_IRUSR`, `S_IWUSR` or `S_IXUSR` from the *Fcntl* module, and *EFFECTIVE* indicates whether to use effective (true) or real (false) ids. The method interprets the `mode`, `uid` and `gid` fields, and returns whether or not the current process would be allowed the specified access.

If you don't want to use the objects, you may import the `->cando` method into your namespace as a regular function called `stat_cando`. This takes an arrayref containing the return values of `stat` or `lstat` as its first argument, and interprets it for you.

You may also import all the structure fields directly into your namespace as regular variables using the :FIELDS import tag. (Note that this still overrides your stat() and lstat() functions.) Access these fields as variables named with a preceding `st_` in front their method names. Thus, `$stat_obj->dev()` corresponds to $st_dev if you import the fields.

To access this functionality without the core overrides, pass the `use` an empty import list, and then access function functions with their full qualified names. On the other hand, the built-ins are still available via the `CORE::` pseudo-package.

BUGS

As of Perl 5.8.0 after using this module you cannot use the implicit $_ or the special filehandle _ with stat() or lstat(), trying to do so leads into strange errors. The workaround is for $_ to be explicit

```
my $stat_obj = stat $_;
```

and for _ to explicitly populate the object using the unexported and undocumented populate() function with CORE::stat():

```
my $stat_obj = File::stat::populate(CORE::stat(_));
```

ERRORS

-%s is not implemented on a File::stat object

The filetest operators `-t`, `-T` and `-B` are not implemented, as they require more information than just a stat buffer.

WARNINGS

These can all be disabled with

```
no warnings "File::stat";
```

File::stat ignores use filetest 'access'

You have tried to use one of the `-rwxRWX` filetests with `use filetest 'access'` in effect. `File::stat` will ignore the pragma, and just use the information in the `mode` member as usual.

File::stat ignores VMS ACLs

VMS systems have a permissions structure that cannot be completely represented in a stat buffer, and unlike on other systems the builtin filetest operators respect this. The `File::stat` overloads, however, do not, since the information required is not available.

NOTE

While this class is currently implemented using the Class::Struct module to build a struct-like class, you shouldn't rely upon this.

AUTHOR

Tom Christiansen