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";
}

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.

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(_));
```

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