NAME

Fatal - replace functions with equivalents which succeed or die

SYNOPSIS

```
    use Fatal qw(open close);

    sub juggle { . . . }
    import Fatal 'juggle';
```

DESCRIPTION

`Fatal` provides a way to conveniently replace functions which normally return a false value when they fail with equivalents which raise exceptions if they are not successful. This lets you use these functions without having to test their return values explicitly on each call. Exceptions can be caught using `eval{}`. See *perlfunc* and *perlvar* for details.

The do-or-die equivalents are set up simply by calling Fatal's `import` routine, passing it the names of the functions to be replaced. You may wrap both user-defined functions and overridable CORE operators (except `exec`, `system` which cannot be expressed via prototypes) in this way.

If the symbol `:void` appears in the import list, then functions named later in that import list raise an exception only when these are called in void context--that is, when their return values are ignored. For example

```
use Fatal qw/:void open close/;

# properly checked, so no exception raised on error
if(open(FH, "< /bogotic") {
 warn "bogo file, dude: $!";
}

# not checked, so error raises an exception
close FH;
```

BUGS

You should not fatalize functions that are called in list context, because this module tests whether a function has failed by testing the boolean truth of its return value in scalar context.

AUTHOR

Lionel Cons (CERN).

Prototype updates by Ilya Zakharevich <ilya@math.ohio-state.edu>.