errors

Package errors go1.15.2 Latest

Published: **Sep 9, 2020** | License: BSD-3-Clause | Standard library

Overview

Package errors implements functions to manipulate errors.

The New function creates errors whose only content is a text message.

The Unwrap, Is and As functions work on errors that may wrap other errors. An error wraps another error if its type has the method

```
Unwrap() error
```

If e.Unwrap() returns a non-nil error w, then we say that e wraps w.

Unwrap unpacks wrapped errors. If its argument's type has an Unwrap method, it calls the method once. Otherwise, it returns nil.

A simple way to create wrapped errors is to call fmt.Errorf and apply the %w verb to the error argument:

```
errors.Unwrap(fmt.Errorf("... %w ...", ..., err, ...))
```

returns err.

Is unwraps its first argument sequentially looking for an error that matches the second. It reports whether it finds a match. It should be used in preference to simple equality checks:

```
if errors.Is(err, os.ErrExist)
```

is preferable to

```
if err == os.ErrExist
```

because the former will succeed if err wraps os.ErrExist.

As unwraps its first argument sequentially looking for an error that can be assigned to its second argument, which must be a pointer. If it succeeds, it performs the assignment and returns true. Otherwise, it returns false. The form

```
var perr *os.PathError
if errors.As(err, &perr) {
    fmt.Println(perr.Path)
}
```

is preferable to

```
if perr, ok := err.(*os.PathError); ok {
    fmt.Println(perr.Path)
}
```

because the former will succeed if err wraps an *os.PathError.

func As

```
func As(err error, target interface{}) bool
```

As finds the first error in err's chain that matches target, and if so, sets target to that error value and returns true. Otherwise, it returns false.

The chain consists of err itself followed by the sequence of errors obtained by repeatedly calling Unwrap.

An error matches target if the error's concrete value is assignable to the value pointed to by target, or if the error has a method As(interface{}) bool such that As(target) returns true. In the latter case, the As method is responsible for setting target.

An error type might provide an As method so it can be treated as if it were a different error type.

As panics if target is not a non-nil pointer to either a type that implements error, or to any interface type.

func Is

```
func Is(err, target error) bool
```

Is reports whether any error in err's chain matches target.

The chain consists of err itself followed by the sequence of errors obtained by repeatedly calling Unwrap.

An error is considered to match a target if it is equal to that target or if it implements a method Is(error) bool such that Is(target) returns true.

An error type might provide an Is method so it can be treated as equivalent to an existing error. For example, if MyError defines

```
func (m MyError) Is(target error) bool { return target == os.ErrExist }
```

then Is(MyError{}, os.ErrExist) returns true. See syscall.Errno.Is for an example in the standard library.

func New

```
func New(text string) error
```

New returns an error that formats as the given text. Each call to New returns a distinct error value even if the text is identical.

func Unwrap

```
func Unwrap(err error) error
```

Unwrap returns the result of calling the Unwrap method on err, if err's type contains an Unwrap method returning error. Otherwise, Unwrap returns nil.