Errors in Go 1.13

Alexandre Thenorio
Einride

A story about an error

Don't just check errors, handle them gracefully

dave.cheney.net/2016/04/27/dont-just-check-errors-handle-them-gracefully (https://dave.cheney.net/2016/04/27/dont-just-check-errors-handle-them-gracefully) 3

Go 2 draft proposal

go.googlesource.com/proposal/+/master/design/go2draft-error-values-overview.md (https://go.googlesource.com/proposal/+/master/design/go2draft-error-values-overview.md)

- Error inspection
- Error formatting

4

Error inspection in 1.13

Unwrap

```
func Unwrap() error
```

Any error implementing this function can be unwrapped

A Cat Error

```
type CatError struct {
    Reason string
    Err    error
}

func (e *CatError) Error() string {
    return e.Reason + ": " + e.Err.Error()
}

func (e *CatError) Unwrap() error {
    return e.Err
}
```

7

errors.Unwrap

```
// 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.
```

```
    errCat := &CatError{
        Err:    os.ErrNotExist,
        Reason: "Cat is undisposed",
    }

    // os.ErrNotExist
    err := errors.Unwrap(errCat)
```

8

CatError

Do I need to make every error a new struct?

fmt.Errorf

New %w verb introduced in `fmt.Errorf`

```
    // returns a new error which implements Unwrap()
    // to retrieve the underlying error
    fmt.Errorf("unable to open database: %w", err)
```

- Only 1 %w verb may exist in the template string

11

errors.Is

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

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

12

errors.Is

Before

```
    if err == io.ErrUnexpectedEOF {
        // do something
    }
```

After

```
    if errors.Is(err, io.ErrUnexpectedEOF) {
        // do something
    }
```

errors.As

```
// 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 responsibl
// for setting target.
```

```
func As(t interface{}) bool
```

14

errors.As

Before

```
    if pathError, ok := err.(*os.PathError); ok {
        // do something
    }
```

After

```
    var pathError *os.PathError
    if errors.As(err, &pathError) {
        // handle pathError
    }
```

15

Error formatting is not included in 1.13

Migrating

```
if err == io.ErrUnexpectedEOF
```

Becomes

```
if errors.Is(err, io.ErrUnexpectedEOF)
```

17

Migrating

```
if e, ok := err.(*os.PathError); ok
```

Becomes

```
var e *os.PathError
if errors.As(err, &e)
```

18

Migrating

- Checks of the form "if err != nil" need not be changed.
- io.EOF need not be changed?

```
Comparisons to io.EOF need not be changed, because io.EOF
should never be wrapped.
```

19

What if I am not able to move to 1.13 immediately

xerrors

xerrors package has the same API as 1.13 error inspection

```
import "golang.org/x/xerrors"
```

21

Thank you

Alexandre Thenorio
Einride
https://github.com/alethenorio (https://github.com/alethenorio)
@alethenorio (http://twitter.com/alethenorio)