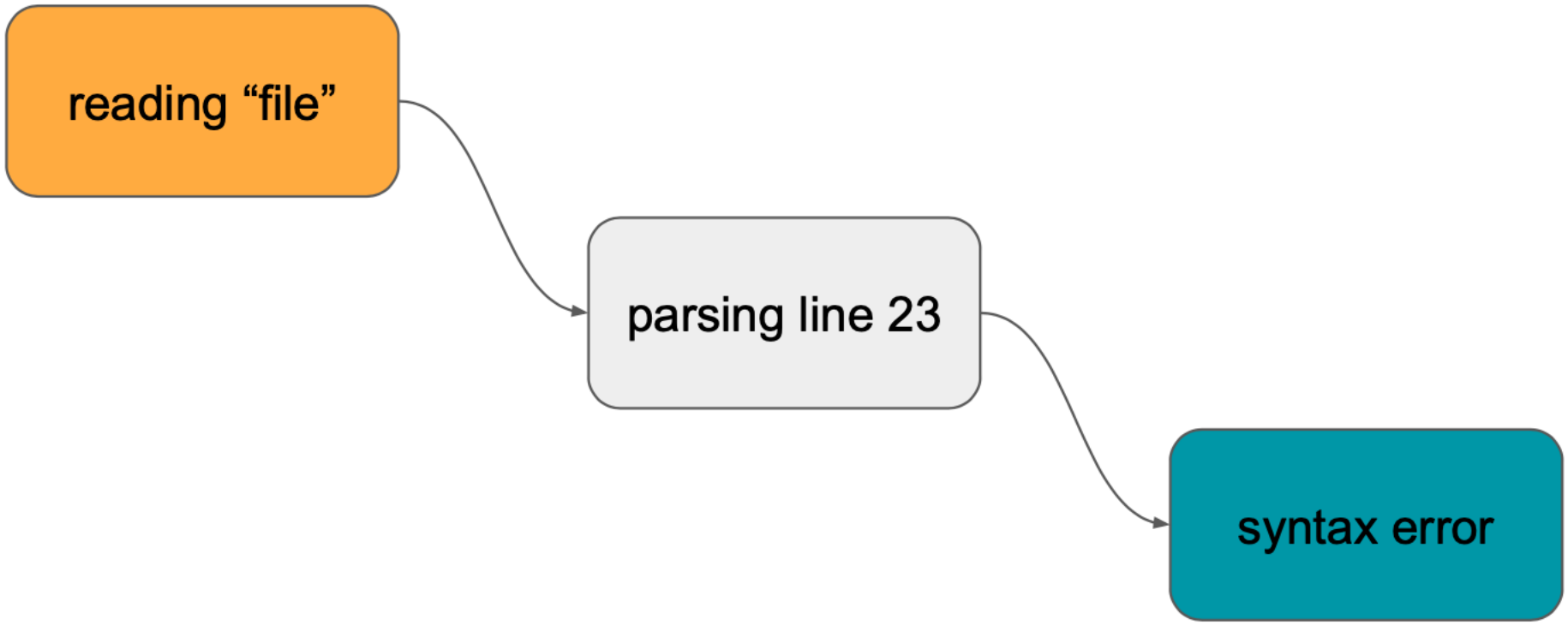
Working with Error Wrapping

Wrapping and the error chain



Error-wrapping support in Go 1.13

errors.Unwrap

errors.Is

errors.As

fmt.Errorf(" ... %w ... ", err)

error 例子

内部细节

如何使用

未能加入的特性

error 使用者

程序员

Debug

程序

错误重试
尝试其他方法
记录日志

返回原始错误

```
func ReadConfig(filename string) (*Config, error) {
    f, err := os.Open(filename)
    if err != nil {
        return nil, err
    }
    defer f.Close()
    var c Config
    if err := json.NewDecoder(f).Decode(&c); err != nil {
        return nil, err
    }
    return &c, nil
}
```

return the error directly

调用方

```
func displayConfigForUser(filename string) {
    c, err := ReadConfig(filename)
    if err != nil {
        fmt.Printf("failed: %v\n", err)
        explainError(err)
        return
    }
    fmt.Printf("%+v\n", c)
}
```

Go 1.13前的error处理

```
func explainError(err error) {
    if err == io.ErrUnexpectedEOF {
        fmt.Println("That file ended unexpectedly.")
    } else {
        switch e := err.(type) {
        case *os.PathError: // problem reading file
            if os.IsNotExist(e) {
                fmt.Println("That file doesn't exist.")
            } else {
                fmt.Println("Something about reading that file is bad.")
            }
        case *json.SyntaxError:
            fmt.Println("Are you sure that's a JSON file?")
        }
    }
}
```

compare to “sentinel” error

type switch

predicate function

添加详细信息

```
func ReadConfig(filename string) (*Config, error) {
    f, err := os.Open(filename)
    if err != nil {
        return nil, fmt.Errorf("reading: %v", err)
    }
    defer f.Close()
    var c Config
    if err := json.NewDecoder(f).Decode(&c); err != nil {
        return nil, fmt.Errorf("decoding JSON: %v", err)
    }
    return &c, nil
}
```

add helpful information

error wrapping

目的：为程序员提供更多信息，但不影响程序检查错误

方案：把 error Wrap 到另一个 error 中，可以从 Wrap 后的 error 中获取原始 error

Wrapping errors in Go 1.13

```
func ReadConfig(filename string) (*Config, error) {
    f, err := os.Open(filename)
    if err != nil {
        return nil, fmt.Errorf("reading: %w", err)
    }
    defer f.Close()
    var c Config
    if err := json.NewDecoder(f).Decode(&c); err != nil {
        return nil, fmt.Errorf("decoding JSON: %w", err)
    }
    return &c, nil
}
```

add helpful information and wrap

使用 erorr.Is 替换 ==

```
func explainError(err error) {
    switch {
    case errors.Is(err, io.ErrUnexpectedEOF):   like ==, but unwraps
        fmt.Println("That file ended unexpectedly.")
    case errors.Is(err, os.ErrNotExist):        replaces os predicates
        fmt.Println("That file doesn't exist.")
    default:
    // ...




    }
}
```

使用 errors.As 替换 类型断言

```
func explainError(err error) {
    switch {
    case errors.Is(err, io.ErrUnexpectedEOF):
        fmt.Println("That file ended unexpectedly.")
    case errors.Is(err, os.ErrNotExist):
        fmt.Println("That file doesn't exist.")
    default:
        var perr *os.PathError
        if errors.As(err, &perr) {
            fmt.Printf("Something about %s %q is bad.\n", perr.Op, perr.Path)
        }
        var jerr *json.SyntaxError
        if errors.As(err, &jerr) {
            fmt.Println("Are you sure that's a JSON file?")
        }
    }}
```

like type switch/assertion,
but unwraps

实现细节

```
package os

type PathError struct {
    // ...
    Err error
}

func (p *PathError) Unwrap() error { return p.Err }
```

exported for backward compatibility

Errors.Unwrap

```
func Unwrap(err error) error {
    u, ok := err.(interface { Unwrap() error })
    if !ok {
        return nil
    }
    return u.Unwrap()
}
```

调用 errors.As

```
var perr *os.PathError
if errors.As(err, &perr) {    pass a pointer to the error type
    fmt.Println(perr.Op, perr.Path)
}
```

```
perr, ok := err.(*os.PathError)
```

```
package os

type PathError { ... }

func (e *PathError) Error() string { ... }    the error type is *PathError
```

fmt.Errorf(" … %w … ", err)

```
werr := fmt.Errorf("wrapped: %w", err)

werr.Error() == "wrapped: " + err.Error()

werr.Unwrap() == err
```

开始使用

- 使用 errors.Is/As 处理可能被 Wrap 的error
- 如果文档显式说明了 error 类型，则不必使用

Wrapping returned errors

把 return err 换成 fmt.Errorf(“...%w...”, err) 时，务必注意保持协议

不要 Wrap 假的 error (io.EOF)

errors and codes

```
var (
    NotFound = errors.New("not found")
    InvalidArgument = errors.New("invalid argument")
    Unknown = errors.New("unknown")
)
```

```
fmt.Errorf("retrieving module %s: %w", modulePath, NotFound)
```

```
if errors.Is(err, NotFound) ...
```

未加入的特性

Stack trace

Error format

QA