Beautiful (and strange) I/O

Lightning Talk, Go and Cloud Native Leipzig

@BasislagerCo, golangleipzig.space, 2019-03-15, 19:00

Go Proverb

- The bigger the interface, the weaker the abstraction

Exemplified in package io

Generic I/O with io.Reader and io.Writer. A few other interfaces:

	R	W	C	S
io.Reader	x			
io.Writer		x		
io.Closer			x	
io.Seeker				x
io.ReadWriter	x	x		
io.ReadCloser	x		x	
io.ReadSeeker	x			x
io.WriteCloser		x	x	
io.WriteSeeker		x		x
io.ReadWriteCloser	x	x	x	
io.ReadWriteSeeker	x	x		x

Missing things

Libraries might implement missing pieces, e.g.

- ReadSeekCloser, ReaderAtCloser

From: github.com/go4org/go4.

IO interface list

- io.ReaderAt (offset)
- io.ReaderFrom
- io.WriterAt (offset)
- io.WriterTo

Use cases

- io.ReaderAt, io.WriterAt -- (parallel writes) with offset

Sidenote: For filesystems, there is a pread(2) system call in Linux

> read from or write to a file descriptor at a given offset ...
> The pread() and pwrite() system calls are especially useful in **multithreaded applications**. They allow multiple threads to perform I/O on the **same file descriptor** without being affected by changes to the file offset by other threads.

Use cases

- io.ReaderFrom -- a data structure, that know how to deserialize itself

Example, different JSON API structs, but each of them implements io.ReaderFrom, so the data fetch can be separated -- fetchLocation(location string, r io.ReaderFrom)

Readers for types

Rune

- io.RuneReader
- io.RuneScanner (support for rewind)

Byte

- io.ByteReader
- io.ByteScanner (support for rewind)
- io.ByteWriter

String

- io.StringWriter (new in 1.12)

Who implements these interfaces?

- files, atomic files
- buffered io
- response bodies
- compression algorithms
- hash sums
- image, JSON, xml encoders, decoders
- utilities like counters, test data generators, stream splitters, mutli-readers
- and much more

A simple interface

```
type Reader interface {
    func Read(p []byte) (n int, err error)
}

type Writer interface {
    func Write(p []byte) (n int, err error)
}
```