Table of Contents

1 buffer
1.1 Basic Usage
1.2 Writing Numbers
1.3 Writing Byte Sequences
1.4 Writing Structured Data
1.5 Size Hints
1.6 Buffer as Logger
1.7 Converting to String/Bytes
1.8 Binary Viewing

buffer

The buffer package provides a flexible byte buffer implementation for efficient binary data handling and serialization.

Basic Usage

Create a new buffer and write basic data:

```
1
2   test "basic buffer operations" {
3     let buf = @buffer new()
4
5
6     buf..write_byte(b'H')..write_byte(b'i')
7
8
9     inspect(buf is_empty(), content="false")
10    inspect(buf length(), content="2")
11
12
13    let bytes = buf contents()
14    inspect(
15      bytes,
16      content=(
17        #|b"\x48\x69"
18      ),
19    )
20
21
22    buf reset()
23    inspect(buf is_empty(), content="true")
24  }
```

Writing Numbers

Write numbers in different encodings:

```

  test "number serialization" {
    inspect(
      @buffer new()

      ..write_int_be(42)
      ..write_int_le(42)
       to_bytes(),
      content=(
        #|b"\x00\x00\x00\x2a\x2a\x00\x00\x00"
      ),
    )
    inspect(
      @buffer new()

      ..write_float_be(3.14)
      ..write_float_le(3.14)
       to_bytes(),
      content=(
        #|b"\x40\x48\xf5\xc3\xc3\xf5\x48\x40"
      ),
    )
    inspect(
      @buffer new()

      ..write_int64_be(0xAABBCCDDEEL)
      ..write_int64_le(0xAABBCCDDEEL)
       to_bytes(),
      content=(
        #|b"\x00\x00\x00\xaa\xbb\xcc\xdd\xee\xee\xdd\xcc\xbb\xaa\x00\x00\x
      ),
    )
    inspect(
      @buffer new()

      ..write_uint_be(0x2077U)
      ..write_uint_le(0x2077U)
       to_bytes(),
      content=(
        #|b"\x00\x00\x20\x77\x77\x20\x00\x00"
      ),
    )
  }
```

Writing Byte Sequences

Write sequences of bytes:

```

  test "byte sequence writing" {
    let buf = @buffer new()


    let bytes = b"Hello"
    buf write_bytes(bytes)


    buf write_iter(bytes iter())
    let contents = buf to_bytes()
    inspect(
      contents,
      content=(
        #|b"\x48\x65\x6c\x6c\x6f\x48\x65\x6c\x6c\x6f"
      ),
    )
  }
```

Writing Structured Data

Write structured data that implements Show:

```

  test "object writing" {
    let buf = @buffer new()


    buf write_object(42)


    let contents = buf contents()
    inspect(
      contents,
      content=(
        #|b"\x34\x00\x32\x00"
      ),
    )
  }
```

Size Hints

Provide size hints for better performance:

```
1
2   test "buffer with size hint" {
3
4     let buf = @buffer new(size_hint=1024)
5
6
7     for i in 0..<100 {
8       buf write_int_le(i)
9     }
10
11
12    inspect(buf length(), content="400")
13  }
```

Buffer as Logger

The buffer implements the Logger trait for Show:

```
1
2   test "buffer as logger" {
3     let buf = @buffer new()
4     let array = [1, 2, 3]
5
6
7     array output(buf)
8     let contents = buf contents()
9     inspect(
10      contents,
11      content=(
12        #|b"\x5b\x00\x31\x00\x2c\x00\x20\x00\x32\x00\x2c\x00\x20\x00\x33\x
13      ),
14    )
15  }
```

Converting to String/Bytes

Methods for converting buffer contents:

```
1
2   test "buffer conversion" {
3     let buf = @buffer new()
4     buf write_byte(b'a')
5     buf write_byte(b'b')
6     buf write_byte(b'c')
7     let bytes = buf to_bytes()
8     inspect(
9       bytes,
10      content=(
11        #|b"\x61\x62\x63"
12      ),
13    )
14  }
```

Binary Viewing

Support for viewing subsets of bytes:

```

  test "byte view writing" {
    let buf = @buffer new()
    let bytes = b"Hello World"


    buf write_bytesview(bytes[0:5])
    let contents = buf to_bytes()
    inspect(
      contents,
      content=(
        #|b"\x48\x65\x6c\x6c\x6f"
      ),
    )
  }
```