

Readers & Writers

01

Reader

02

bufio

03

Writer



Reader & Writer

- | Reader & Writer are interfaces that allow reading from and writing to I/O sources
- | Network sockets, files, arbitrary arrays
- | Multiple implementations in standard library
- | Reader is a low-level implementation
 - | Usually want to work with bufio package instead of Reader directly



Interfaces

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

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

Reader

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

- | Each call to Read() will fill the provided p buffer
- | The number of bytes read will be returned as n
- | When all bytes have been read, err will be io.EOF



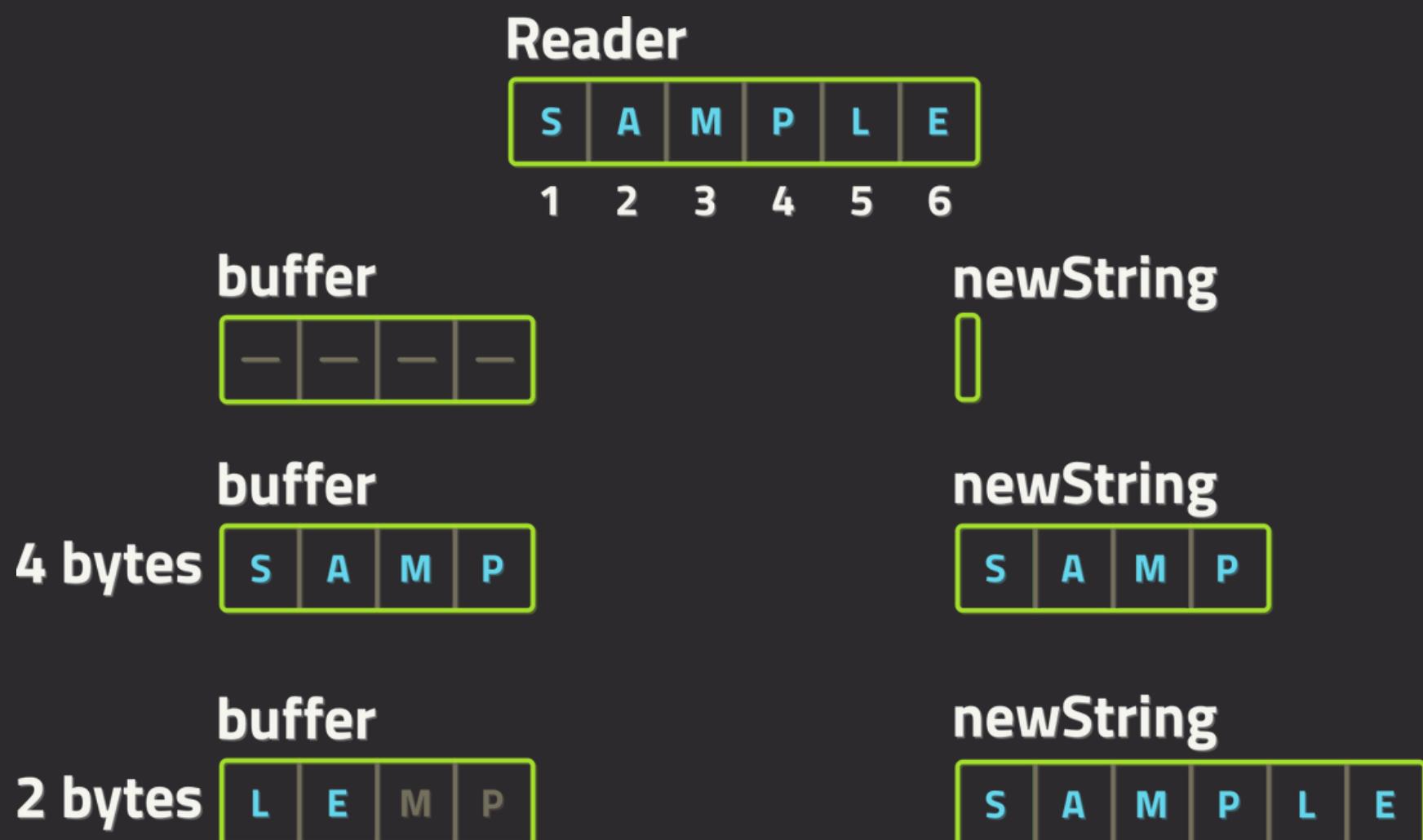
Example

```
reader := strings.NewReader("SAMPLE")

var newString strings.Builder
buffer := make([]byte, 4)
for {
    numBytes, err := reader.Read(buffer)
    chunk := buffer[:numBytes]
    newString.Write(chunk)
    fmt.Printf("Read %v bytes: %c\n", numBytes, chunk)
    if err == io.EOF {
        break
    }
}
fmt.Printf("%v\n", newString.String())
```

```
Read 4 bytes: [S A M P]
Read 2 bytes: [L E]
Read 0 bytes: []
SAMPLE
```

Walkthrough



```
reader := strings.NewReader("SAMPLE")  
  
var newString strings.Builder  
buffer := make([]byte, 4)  
for {  
    numBytes, err := reader.Read(buffer)  
    chunk := buffer[:numBytes]  
    newString.Write(chunk)  
    fmt.Printf("Read %v bytes: %c\n", numBytes, chunk)  
    if err == io.EOF {  
        break  
    }  
}  
fmt.Printf("%v\n", newString.String())
```

bufio

- | bufio package provides Reader & Writer buffering
- | No need to manually manage buffers or construct data

```
source := strings.NewReader("SAMPLE")
buffered := bufio.NewReader(source)
// Can also use buffered.ReadBytes here.
newString, err := buffered.ReadString('\n')
if err == io.EOF {
    fmt.Println(newString)
} else {
    fmt.Println("something went wrong...")
}
```

bufio.Scanner

| bufio.Scanner provides convenience functions

```
// Read lines from standard input
scanner := bufio.NewScanner(os.Stdin)
lines := make([]string, 0, 5)
for scanner.Scan() {
    lines = append(lines, scanner.Text())
}
if scanner.Err() != nil {
    fmt.Println(scanner.Err())
}
fmt.Printf("Line count: %v\n", len(lines))
for _, line := range lines {
    fmt.Printf("Line: %v\n", line)
}
```

```
> printf "these\nare\nsome\nwords" \
> | go run ./lecture.go
Line count: 4
Line: these
Line: are
Line: some
Line: words
```

Writer

| Writer is nearly symmetrical with Reader

```
buffer := bytes.NewBufferString("")
numBytes, err := buffer.WriteString("SAMPLE")
if err != nil {
    fmt.Println(err)
} else {
    fmt.Printf("Wrote %v bytes: %c\n", numBytes, buffer)
}
```

Recap

- | Reader & Writer are interfaces that allow reading from and writing to I/O sources
- | Using Reader directly requires manually populating a buffer
- | The `bufio` stdlib package provides auto-buffered reads
- | The `bufio.Scanner` type can automatically read and delimit inputs