bytes.Buffer 是 Golang 标准库中的缓冲区，具有读写方法和可变大小的字节存储功能。缓冲区的零值是一个待使用的空缓冲区。

1. 声明

var b bytes.Buffer //直接定义一个Buffer变量，不用初始化，可以直接使用

b := new(bytes.Buffer) //使用New返回Buffer变量

b := bytes.NewBuffer(s []byte) //从一个[]byte切片，构造一个Buffer

b := bytes.NewBufferString(s string) //从一个string变量，构造一个Buffer

1. 写方法

b.Write(d []byte) (n int, err error) //将切片d写入Buffer尾部,返回值为写入长度和错误码

b.WriteString(s string) (n int, err error) //将字符串s写入Buffer尾部

b.WriteByte(c byte) error //将字符c写入Buffer尾部

b.WriteRune(r rune) (n int, err error) //将一个rune类型的数据放到缓冲区的尾部

b.ReadFrom(r io.Reader) (n int64, err error) //从实现了io.Reader接口的可读取对象写入Buffer尾部

1. 读方法

b.Next(n int) []byte //读取 n 个字节数据并返回，如果 buffer 不足 n 字节，则读取全部

b.Read(p []byte) (n int, err error) //一次读取 len(p) 个 byte 到 p 中，每次读取新的内容将覆盖p中原来的内容。成功返回实际读取的字节数，buffer 没有数据返回错误 io.EOF

b.ReadByte() (byte, error) //读取第一个byte并返回，off 向后偏移 n

//读取第一个 UTF8 编码的字符并返回该字符和该字符的字节数，b的第1个rune被拿掉。

//如果buffer为空，返回错误 io.EOF，如果不是UTF8编码的字符，则消费一个字节，返回 (U+FFFD,1,nil)

b.ReadRune() (r rune, size int, err error)

//读取缓冲区第一个分隔符前面的内容以及分隔符并返回，缓冲区会清空读取的内容。

//如果没有发现分隔符，则返回读取的内容并返回错误io.EOF

b.ReadBytes(delimiter byte) (line []byte, err error)

//读取缓冲区第一个分隔符前面的内容以及分隔符并作为字符串返回，缓冲区会清空读取的内容。

//如果没有发现分隔符，则返回读取的内容并返回错误 io.EOF

b.ReadString(delimiter byte) (line string, err error)

//将 Buffer 中的内容输出到实现了 io.Writer 接口的可写入对象中，成功返回写入的字节数，失败返回错误

b.WriteTo(w io.Writer) (n int64, err error)

1. 其他操作方法

b.Bytes() []byte //返回字节切片

b.Cap() int //返回 buffer 内部字节切片的容量

b.Grow(n int) //为 buffer 内部字节切片的容量增加 n 字节

b.Len() int //返回缓冲区数据长度，等于 len(b.Bytes())

b.Reset() //清空数据

b.String() string //字符串化

b.Truncate(n int) //丢弃缓冲区中除前n个未读字节以外的所有字节。如果 n 为负数或大于缓冲区长度，则引发 panic

b.UnreadByte() error //将最后一次读取操作中被成功读取的字节设为未被读取的状态，即将已读取的偏移offset 减 1

b.UnreadRune() error //将最后一次 ReadRune() 读取操作返回的 UTF8 字符 rune设为未被读取的状态，即将已读取的偏移 offset 减去 字符 rune 的字节数