TCP Servers

package net

```
import "net"
```

Package net provides a portable interface for network I/O, including TCP/IP, UDP, domain name resolution, and Unix domain sockets.

Although the package provides access to low-level networking primitives, most clients will need only the basic interface provided by the Dial, Listen, and Accept functions and the associated Conn and Listener interfaces. The crypto/tls package uses the same interfaces and similar Dial and Listen functions.

The Dial function connects to a server:

```
conn, err := net.Dial("tcp", "google.com:80")
if err != nil {
	// handle error
}
fmt.Fprintf(conn, "GET / HTTP/1.0\r\n\r\n")
status, err := bufio.NewReader(conn).ReadString('\n')
// ...
```

The Listen function creates servers:

```
ln, err := net.Listen("tcp", ":8080")
if err != nil {
	// handle error
}
for {
	conn, err := ln.Accept()
	if err != nil {
		// handle error
	}
	go handleConnection(conn)
}
```

ports

place where things come in and go out

https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

main.go

```go
package main

import (
    "fmt"
    "io"
    "net"
    "time"
)

func main() {
    ln, err := net.Listen("tcp", ":9000")
    if err != nil {
        panic(err)
    }
    defer ln.Close()

    for {
        conn, err := ln.Accept()
        if err != nil {
            panic(err)
        }

        io.WriteString(conn, fmt.Sprint("Hello World\n", time.Now(), "\n"))

        conn.Close()
    }
}
```

telnet is a tcp client

```
14_struct $ telnet localhost 9000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
Hello World
2015-09-15 05:50:21.136671791 -0700 PDT
Connection closed by foreign host.
14_struct $ _
```

main.go

```go
package main

import (
    "fmt"
    "io"
    "net"
    "time"
)

func main() {
    ln, err := net.Listen("tcp", ":9000")
    if err != nil {
        panic(err)
    }
    defer ln.Close()

    for {
        conn, err := ln.Accept()
        if err != nil {
            panic(err)
        }

        io.WriteString(conn, fmt.Sprint("Hello World\n", time.Now(), "\n"))

        conn.Close()
    }
}
```

ctrl+c
to exit program

```
14_struct $ telnet localhost 9000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
Hello World
2015-09-15 05:50:21.136671791 -0700 PDT
Connection closed by foreign host.
14_struct $ _
```

main.go

```go
package main

import (
	"fmt"
	"io/ioutil"
	"net"
)

func main() {
	conn, err := net.Dial("tcp", "localhost:9000")
	if err != nil {
		panic(err)
	}
	defer conn.Close()

	bs, _ := ioutil.ReadAll(conn)
	fmt.Println(string(bs))

}
```

Terminal

```
02_listen $ go run main.go
```

1 - listen

2 - dial

```
03_dial $ go run main.go
Hello World
2015-09-15 06:01:15.219649481 -0700 PDT

03_dial $ _
```

main.go

```go
package main

import (
    "fmt"
    "io/ioutil"
    "net"

    panic(err)
}
defer conn.Close()

bs, _ := ioutil.ReadAll(conn)
fmt.Println(string(bs))

}
```

Terminal

```
02_listen $ go run main.go
```

1 - listen

We have two separate programs running and they're communicating over TCP

2 - dial

```
03_dial $ go run main.go
Hello World
2015-09-15 06:01:15.219649481 -0700 PDT

03_dial $ _
```

godoc.org/net#Listen

func Listen ¶

```
func Listen(net, laddr string) (Listener, error)
```

Listen announces on the local network address laddr. The network net must be a stream-oriented network: "tcp", "tcp4", "tcp6", "unix" or "unixpacket". See Dial for the syntax of laddr.

godoc.org/net#Listener

type Listener

```
type Listener interface {
    // Accept waits for and returns the next connection to the listener.
    Accept() (c Conn, err error)

    // Close closes the listener.
    // Any blocked Accept operations will be unblocked and return errors.
    Close() error

    // Addr returns the listener's network address.
    Addr() Addr
}
```

A Listener is a generic network listener for stream-oriented protocols.

Multiple goroutines may invoke methods on a Listener simultaneously.

type Conn

```go
type Conn interface {
    // Read reads data from the connection.
    // Read can be made to time out and return a Error with Timeout() == true
    // after a fixed time limit; see SetDeadline and SetReadDeadline.
    Read(b []byte) (n int, err error)

    // Write writes data to the connection.
    // Write can be made to time out and return a Error with Timeout() == true
    // after a fixed time limit; see SetDeadline and SetWriteDeadline.
    Write(b []byte) (n int, err error)

    // Close closes the connection.
    // Any blocked Read or Write operations will be unblocked and return errors.
    Close() error
```

godoc.org/io#WriteString

marks PM Hawk J Android GO

func WriteString

```
func WriteString(w Writer, s string) (n int, err error)
```

WriteString writes the contents of the string s to w, which accepts a slice of bytes. If w implements a WriteString method, it is invoked directly.

exercises

tcp echo

- Create an ‘echo’ TCP server. It should accept a connection and write to the connection anything that’s sent to it.