Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления
Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа
«Разработка SSH-сервера и SSH-клиента»
по курсу: «Компьютерные сети»

Выполнил:
Студент группы ИУ9-31Б
Старовойтов А. И.

Проверил:
Посевин Д. П.

Москва, 2022

Цели

Рассматривается задача разработки SSH-сервера и SSH-клиента на языке GO.

Задачи

Реализовать ssh сервер на языке GO с применением указанных пакетов и запустить его на localhost. Проверка работы должна проводиться путем использования программы ssh в ОС Linux/Unix или PuTTY в ОС Windows. Должны работать следующие функции:

- ☒ авторизация клиента на ssh сервере;
- ☒ создание директории на удаленном сервере;
- ☒ удаление директории на удаленном сервере;
- ☒ вывод содержимого директории;
- ☒ перемещение файлов из одной директории в другую;
- ☒ удаление файла по имени;
- ☒ вызов внешних приложений, например ping.

Протестировать соединение Go SSH-клиента к серверу реализованному в предыдущей задаче, а также к произвольному ssh серверу. Требования: SSH-клиент должен поддерживать следующие функции:

- ☒ авторизация клиента на SSH-сервере;
- ☒ создание директории на удаленном SSH-сервере;
- ☒ удаление директории на удаленном SSH-сервере;
- ☒ вывод содержимого директории;
- ☒ перемещение файлов из одной директории в другую;
- ☒ удаление файла по имени;
- ☒ вызов внешних приложений, например ping.

Решение

Сервер:

```
[st@fedora-laptop server]$ go run .
2022/10/22 22:14:42 starting ssh server on port 2222...
```

Клиент:

```
[st@fedora-laptop lab4]$ go run cmd/client/client.go
Hello test
[test@fedora-laptop ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=60 time=17.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=60 time=17.8 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1090ms
rtt min/avg/max/mdev = 17.294/17.539/17.784/0.245 ms
[test@fedora-laptop ~]$ exit
```

```
logout
[st@fedora-laptop lab4]$
```

Сервер

Реализована своя библиотека для работы с псевдотерминалами.

internal/pty/pty.go:

```
package pty

import (
    "fmt"
    "io"
    "os"
    "os/exec"
    "syscall"
    "unsafe"

    "golang.org/x/sys/unix"
)

func NewPty() (*os.File, string, error) {
    master, err := os.OpenFile("/dev/ptmx",
 ↪  syscall.O_RDWR|syscall.O_NOCTTY|syscall.O_CLOEXEC, 0)
    if err != nil {
        return nil, "", err
    }

    err = unlockpt(master)
    if err != nil {
        return nil, "", err
    }

    slave, err := ptsname(master)
    if err != nil {
        return nil, "", err
    }

    return master, slave, nil
}

func ExecWithPty(command string, args ...string) (io.ReadWriteCloser,
 ↪  error) {
    master, slaveName, err := NewPty()
    if err != nil {
        return nil, err
    }

    slave, err := os.OpenFile(slaveName, syscall.O_RDWR, 0)
    if err != nil {
```

```
        return nil, err
    }
    defer slave.Close()

    cmd := exec.Command(command, args...)
    cmd.Stdin = slave
    cmd.Stdout = slave
    cmd.Stderr = slave
    cmd.SysProcAttr = &syscall.SysProcAttr{
        Setctty: true,
        Setsid:  true,
    }
    err = cmd.Start()
    if err != nil {
        return nil, err
    }

    return master, nil
}

func ioctl(fd, flag, data uintptr) error {
    if _, _, err := syscall.Syscall(syscall.SYS_IOCTL, fd, flag, data);
     ↪  err != 0 {
        return err
    }
    return nil
}

func unlockpt(f *os.File) error {
    var data int32
    return ioctl(f.Fd(), syscall.TIOCSPTLCK,
     ↪  uintptr(unsafe.Pointer(&data)))
}

func ptsname(f *os.File) (string, error) {
    var pty int32
    err := ioctl(f.Fd(), syscall.TIOCGPTN,
 ↪  uintptr(unsafe.Pointer(&pty)))
    if err != nil {
        return "", err
    }
    return fmt.Sprintf("/dev/pts/%v", pty), nil
}

func ttySetRaw(f *os.File) error {
    termios, err := unix.IoctlGetTermios(int(f.Fd()), unix.TCGETS)
    if err != nil {
        return err
    }
```

```
    termios.Iflag &^= unix.IGNBRK | unix.BRKINT | unix.PARMRK |
↪   unix.ISTRIP | unix.INLCR | unix.IGNCR | unix.ICRNL | unix.IXON
    termios.Oflag &^= unix.OPOST
    termios.Lflag &^= unix.ECHO | unix.ECHONL | unix.ICANON | unix.ISIG
↪   | unix.IEXTEN
    termios.Cflag &^= unix.CSIZE | unix.PARENB
    termios.Cflag |= unix.CS8
    termios.Cc[unix.VMIN] = 1
    termios.Cc[unix.VTIME] = 0

    err = unix.IoctlSetTermios(int(f.Fd()), unix.TCSETS, termios)
    if err != nil {
        return err
    }

    return nil
}
```

cmd/server/server.go

```
package main

import (
    "fmt"
    "io"
    "log"

    "github.com/gliderlabs/ssh"
    "github.com/stewkk/iu9-networks/lab4/internal/pty"
)

func main() {
    authHandler := ssh.PasswordAuth(func(ctx ssh.Context, password
↪   string) bool {
        return ctx.User() == "test" && password == "12345678"
    })
    ssh.Handle(func(s ssh.Session) {
        io.WriteString(s, fmt.Sprintf("Hello %s\n", s.User()))
        rw, err := pty.ExecWithPty("/bin/sudo", "--login", "--user",
↪   s.User())
        if err != nil {
            panic(err)
        }
        defer rw.Close()
        go func() {
            io.Copy(rw, s)
        }()
        io.Copy(s, rw)
    })

    log.Println("starting ssh server on port 2222...")
```

```
    log.Fatal(ssh.ListenAndServe(":2222", nil, authHandler))
}
```

Клиент

cmd/client/client.go

```
package main

import (
    "os"

    "github.com/helloyi/go-sshclient"
    "golang.org/x/crypto/ssh"
    "golang.org/x/term"
)

var (
    remote = ""
    local  = "localhost:2222"
)

func main() {
    client, err := sshclient.DialWithPasswd(local, "test", "12345678")
    if err != nil {
        panic(err)
    }

    // with a terminal config
    config := &sshclient.TerminalConfig{
        Term:   "xterm",
        Height: 40,
        Weight: 80,
        Modes: ssh.TerminalModes{
            ssh.TTY_OP_ISPEED: 14400, // input speed = 14.4kbaud
            ssh.TTY_OP_OSPEED: 14400, // output speed = 14.4kbaud
        },
    }

    // Set stdin in raw mode.
    oldState, err := term.MakeRaw(int(os.Stdin.Fd()))
    if err != nil {
        panic(err)
    }
    defer func() { _ = term.Restore(int(os.Stdin.Fd()), oldState) }()
     ↪  // Best effort.

    err = client.Terminal(config).Start()
    if err != nil {
        panic(err)
    }
```

```
}
```