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 client]$ go run .
Hello test
Password: 12345678

[test@fedora-laptop server]$ ls
ls
server.go
[test@fedora-laptop server]$ [st@fedora-laptop client]$
```

Сервер

Реализована своя библиотека для работы с псевдотерминалами.

`internal/pty/pty.go`:

```
package pty

import (
    /*
       #define _XOPEN_SOURCE 600
       #include <fcntl.h>
       #include <stdlib.h>
       #include <unistd.h>
    */
```

```
    "C"
)

import (
    "fmt"
    "io"
    "os"
    "runtime"

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

func init() {
    runtime.LockOSThread()
}

func ptyMasterOpen() (C.int, error) {
    fd, err := C.posix_openpt(C.O_RDWR | C.O_NOCTTY)
    if err != nil {
        return 0, err
    }

    res := C.unlockpt(fd)
    if res == -1 {
        return 0, fmt.Errorf("failed")
    }
    return fd, nil
}

func NewPty(program string, args ...string) (io.ReadWriteCloser, error) {
    fd, err := ptyMasterOpen()
    if err != nil {
        return nil, err
    }
    slave := C.ptsname(fd)
    if slave == nil {
        return nil, fmt.Errorf("failed to get ptsname")
    }
    childPid := C.fork()
    if childPid == -1 {
        return nil, fmt.Errorf("failed to fork")
    }
    if childPid != 0 { // parent
        return os.NewFile(uintptr(fd), C.GoString(slave)), nil
    }
    // child
    _, err = unix.Setsid()
    if err != nil {
        os.Exit(1)
    }
    C.close(fd)
    slaveFile, err := os.OpenFile(C.GoString(slave), os.O_RDWR, 075)
    if err != nil {
        return nil, err
    }
    err = unix.Dup2(int(slaveFile.Fd()), int(os.Stdin.Fd()))
    if err != nil {
        return nil, err
    }
    err = unix.Dup2(int(slaveFile.Fd()), int(os.Stdout.Fd()))
    if err != nil {
        return nil, err
```

```
    }
    err = unix.Dup2(int(slaveFile.Fd()), int(os.Stderr.Fd()))
    if err != nil {
        return nil, err
    }
    if slaveFile.Fd() > os.Stderr.Fd() {
        slaveFile.Close()
    }
    unix.Exec(program, args, nil)
    return nil, 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() {
    ssh.Handle(func(s ssh.Session) {
        io.WriteString(s, fmt.Sprintf("Hello %s\n", s.User()))
        rw, err := pty.NewPty("/bin/su", "-", 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))
}
```

Клиент

cmd/client/client.go

```
package main

import (
    "io"
    "os"

    "golang.org/x/crypto/ssh"
)

var remoteConfig = &ssh.ClientConfig{
    User: "test",
    Auth: []ssh.AuthMethod{
        ssh.Password(""),
    },
    HostKeyCallback: ssh.InsecureIgnoreHostKey(),
}
```

```
var localConfig = &ssh.ClientConfig{
    User: "",
    Auth: []ssh.AuthMethod{
        ssh.Password(""),
    },
    HostKeyCallback: ssh.InsecureIgnoreHostKey(),
}

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

func main() {
    client, err := ssh.Dial("tcp", local, localConfig)
    if err != nil {
        panic(err)
    }
    defer client.Close()

    session, err := client.NewSession()
    if err != nil {
        panic(err)
    }
    defer session.Close()

    session.Stdout = os.Stdout
    session.Stderr = os.Stderr
    stdin, err := session.StdinPipe()
    if err != nil {
        panic(err)
    }

    err = session.Shell()
    if err != nil {
        panic(err)
    }

    io.Copy(stdin, os.Stdin)
}
```