Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 4

по курсу «Компьютерные сети»

«Разработка SSH-сервера»

Студент группы ИУ9-31Б Горбунов А. Д.

Преподаватель Посевин Д. П.

Москва 2023

1 Задание

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.

2 Результаты

Исходный код программы представлен в листинге 1, 2, 3, 4

Листинг 1 — ssh client.go

```
1  package main
2
3  import (
4      "bufio"
5    "golang.org/x/crypto/ssh"
6    "fmt"
7      "io"
8      "os"
9      "log"
10 )
11
12
13 func main() {
14     fmt.Print("IP:")
15     ip := ""
16     fmt.Scan(&ip)
17     fmt.Print("login: ")
18     user := ""
19     fmt.Scan(&user)
20     fmt.Print("password: ")
21     password := ""
22     fmt.Scan(&password)
23     fmt.Println(user, password)
24   sshConfig := &ssh.ClientConfig{
25           User: user,
26           HostKeyCallback: ssh.InsecureIgnoreHostKey(),
27           Auth: []ssh.AuthMethod{
28                ssh.Password(password)},
29     }
```

Листинг 2 — ssh client.go(продолжение)

```
    connection, err := ssh.Dial("tcp",  ip + ":22", sshConfig)
    if err != nil {
        log.Println("Failed to dial:", err)
        return
    }
    comand := ""
    for ; comand != "e_exit"; {
    session, err := connection.NewSession()
    if err != nil {
        log.Println("Failed to create session:", err)
      return
    }
    stdin, err := session.StdinPipe()
    if err != nil {
        log.Println("Unable to setup stdin for session:", err)
      return
    }
    go io.Copy(stdin, os.Stdin)

    stdout, err := session.StdoutPipe()
    if err != nil {
        log.Println("Unable to setup stdout for session:", err)
      return
    }
    go io.Copy(os.Stdout, stdout)

    stderr, err := session.StderrPipe()
    if err != nil {
        log.Println("Unable to setup stderr for session:", err)
      return
    }
    go io.Copy(os.Stderr, stderr)

    reader := bufio.NewReader(os.Stdin)
    comand, err := reader.ReadString('\n')
    if err != nil {
     log.Fatal(err)
    }

        err = session.Start(comand)
        if err != nil {
            log.Println(err)
            return
        }
    }
}
```

Листинг 3 — ssh server.go

```
package main

import (
  "fmt"
  "io/ioutil"
  "os"
  "os/exec"
  "github.com/gliderlabs/ssh"
  "time"
)

func main(){
  ssh.Handle(func(s ssh.Session) {
    command := s.Command()
    if len(command) == 0 {
      fmt.Fprintln(s, "No command provided.")
      return
    }

    switch command[0] {
    case "mkdir":
      dirName := command[1]
      err := os.Mkdir(dirName, 0755)
      if err != nil {
        fmt.Fprintln(s, "Failed to create directory:", err)
        break
      }
      fmt.Fprintln(s, "Directory", dirName, "created successfully.")
    case "rmdir":
      dirName := command[1]
      err := os.RemoveAll(dirName)
      if err != nil {
        fmt.Fprintln(s, "Failed to remove directory:", err)
        break
      }
      fmt.Fprintln(s, "Directory", dirName, "removed successfully.")
    case "ls":
      files, err := ioutil.ReadDir(".")
      if err != nil {
        fmt.Fprintln(s, "Failed to list directory:", err)
        break
      }
      for _, file := range files {
        fmt.Fprintln(s, file.Name())
      }
    case "mv":
      src := command[1]
      dest := command[2]
      err := os.Rename(src, dest)
      if err != nil {
        fmt.Fprintln(s, "Failed to move file:", err)
        break
      }
      fmt.Fprintln(s, "File", src, "moved to", dest)
    case "rm":
      fileName := command[1]
      err := os.Remove(fileName)
      if err != nil {
        fmt.Fprintln(s, "Failed to remove file:", err)
        break
      }
      fmt.Fprintln(s, "File", fileName, "removed successfully.")
```

Листинг 4 — ssh server.go(продолжение)

```
    case "ping":
      if len(command) < 2 {
        fmt.Println("Site name not provided.\n")
        break
      }


      cmd := exec.Command("ping", command[1], command[2], command[3])
      stdout, err := cmd.StdoutPipe()
      if err != nil {
        fmt.Sprintf("Failed to create StdoutPipe for ping command: %s\n"
    , err)
      }

      if err := cmd.Start(); err != nil {
        fmt.Sprintf("Failed to start ping command: %s\n", err)
      }

      output, err := ioutil.ReadAll(stdout)
      if err != nil {
        fmt.Sprintf("Failed to read ping output: %s\n", err)
      }

      done := make(chan error, 1)
      go func() { done <- cmd.Wait() }()
      select {
      case <-time.After(10 * time.Second):
        if err := cmd.Process.Kill(); err != nil {
        fmt.Sprintf("Failed to kill ping process: %s\n", err)
        }
        fmt.Println("Ping command timed out.\n")
      case err := <-done:
        if err != nil {
        fmt.Sprintf("Ping command finished with error: %s\n", err)
        }
        fmt.Fprintln(s, string(output))
      }

    default:
      fmt.Fprintln(s, "Unknown command.")
    }
  })

  fmt.Print("IP: ")
  ip := ""
  fmt.Scan(&ip)
  fmt.Print("Port: ")
  port := ""
  fmt.Scan(&port)
  err := ssh.ListenAndServe(ip + ":" + port , nil)
  if err != nil {
    fmt.Println("Failed to start SSH server:", err)
    return
  }
}
```

Результат запуска представлен на рисунке 1, 2

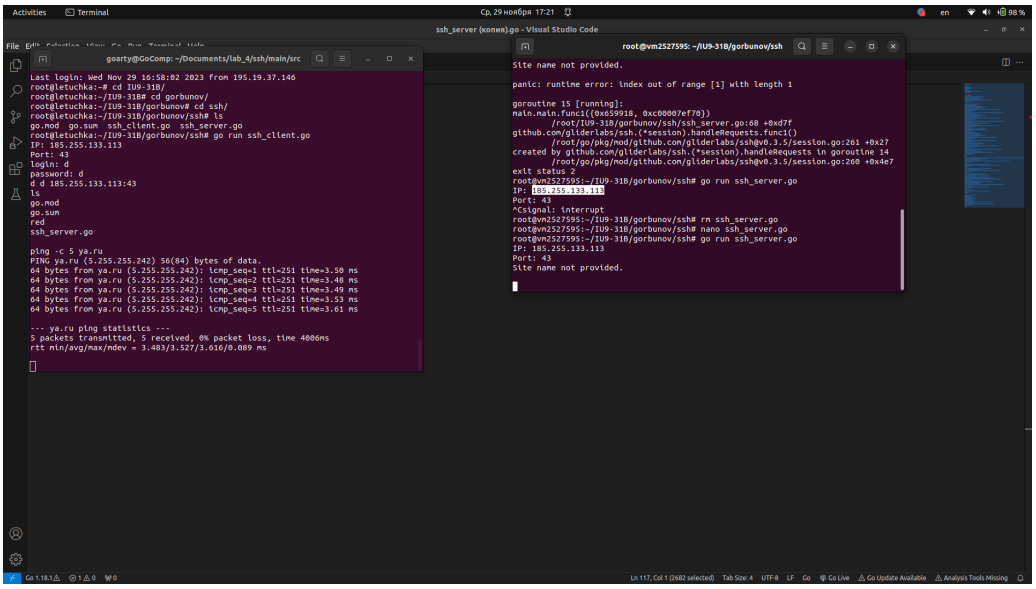

Рис. 1 — Пример ping

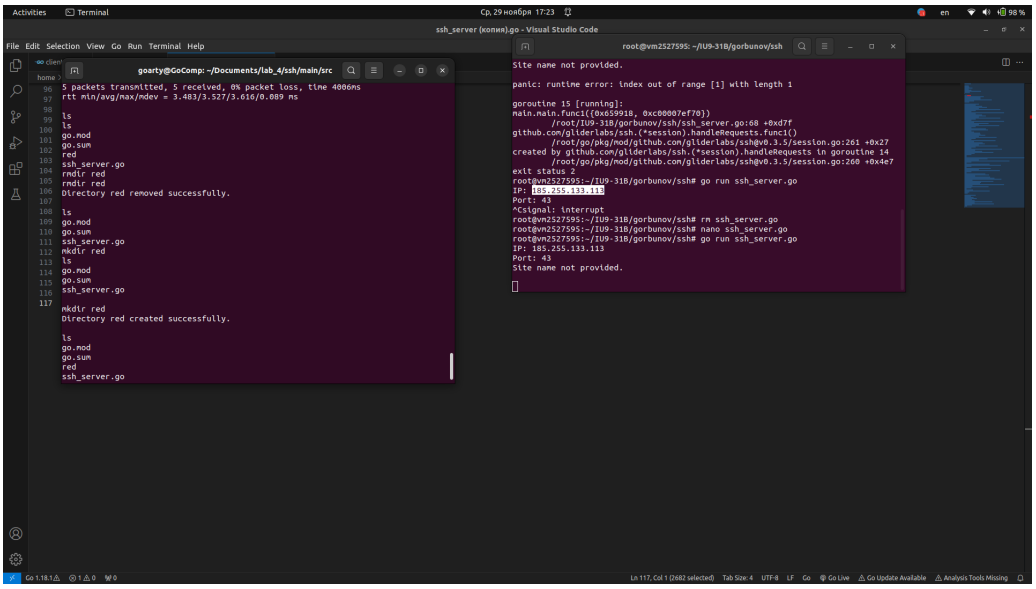

Рис. 2 — Создание директория