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»

«SSH»

Студент группы ИУ9-32Б Волохов А. В.

Преподаватель Посевин Д. П.

Москва 2023

1 Задание

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

Исходный код программы представлен в листингах 1– 2– 3– 4– 5– 6.

Листинг 1 — client.go

```go
package main

import (
  "bufio"
  "fmt"
  "golang.org/x/crypto/ssh"
  "os"
  "strings"
)

func main() {
  reader := bufio.NewReader(os.Stdin)

  fmt.Print("SSH Server Address: ")
  serverAddress, _ := reader.ReadString('\n')
  serverAddress = strings.TrimSpace(serverAddress)

  fmt.Print("Username: ")
  username, _ := reader.ReadString('\n')
  username = strings.TrimSpace(username)

  fmt.Print("Password: ")
  password, _ := reader.ReadString('\n')
  password = strings.TrimSpace(password)

  config := &ssh.ClientConfig{
    User: username,
    Auth: []ssh.AuthMethod{
      ssh.Password(password),
    },
    HostKeyCallback: ssh.InsecureIgnoreHostKey(),
  }

  client, err := ssh.Dial("tcp", serverAddress, config)
  if err != nil {
    fmt.Println("Failed to connect to the SSH server:", err)
    return
  }
  defer func(client *ssh.Client) {
    err := client.Close()
    if err != nil {

    }
  }(client)
  currentDir := "" //

  for {
    fmt.Print("Enter SSH command (or 'exit' to quit): ")
    command, _ := reader.ReadString('\n')
    command = strings.TrimSpace(command)

    if command == "exit" {
      break
    }
    session, err := client.NewSession()
    if err != nil {
      fmt.Println("Failed to create SSH session:", err)
      continue
    }
```

Листинг 2 — client.go - продолжение

```
switch {
    case strings.HasPrefix(command, "ls"):
      output, err := session.CombinedOutput("ls " + currentDir)
      err = session.Close()
      if err != nil {return}
      if err != nil {fmt.Println("Failed to list directory:", err)}
                else {fmt.Println("Directory contents:\n", string(output
    ))}
    case strings.HasPrefix(command, "cd"):
      newDir := strings.TrimSpace(strings.TrimPrefix(command, "cd"))
      Path := newDir
      if currentDir != "" {Path = currentDir + "/" + newDir}
      output, err := session.CombinedOutput("cd " + Path + " && pwd")
      currentDir = Path
      err = session.Close()
      if err != nil {
        return
      }
      if err != nil {
        fmt.Println("Failed to change directory:", err)
      } else {
        currentDir = strings.TrimSpace(string(output))
        fmt.Println("Current directory:", currentDir)
      }
    case strings.HasPrefix(command, "mkdir"):
      dirName := strings.TrimSpace(strings.TrimPrefix(command, "mkdir"))
      _, err := session.CombinedOutput("mkdir " + currentDir + "/" +
    dirName)
      if err != nil {
        fmt.Println("Failed to create directory:", err)
      } else {
        fmt.Println("Directory created:", dirName)
      }
    case strings.HasPrefix(command, "rmdir"):
      dirName := strings.TrimSpace(strings.TrimPrefix(command, "rmdir"))
      Path := dirName
      if currentDir != "" {
        Path = currentDir + "/" + dirName
      }
      _, err := session.CombinedOutput("rmdir " + Path)
      if err != nil {
        fmt.Println("Failed to remove directory:", err)
      } else {
        fmt.Println("Directory removed:", dirName)
      }
    case strings.HasPrefix(command, "mv"):
      args := strings.Split(command, " ")
      if len(args) != 3 {
        fmt.Println("Usage: mv source destination")
      } else {
        sourcePath := args[1]
        destPath := args[2]
        _, err := session.CombinedOutput("mv " + sourcePath + " " +
    destPath)
        if err != nil {
          fmt.Println("Failed to move:", err)
        } else {
          fmt.Println("Moved:", sourcePath, "to", destPath)
        }}
```

Листинг 3 — client.go - продолжение

```
default:
        output, err := session.CombinedOutput(command)

        if err != nil {
          fmt.Println("Failed to execute the command:", err)
        } else {
          fmt.Println("Command output:\n", string(output))
        }
      }
    }

    fmt.Println("SSH session ended.")
}
```

Листинг 4 — server.go

```go
package main

import (
  "fmt"
  "github.com/gliderlabs/ssh"
  "golang.org/x/crypto/ssh/terminal"
  "io"
  "log"
  "os"
  "os/exec"
  "strings"
)

var path = "./lab4/server/server_space/"
var authorizedKeysPath = "./lab4/server/authorized_keys"

func handleSession(session ssh.Session) {
  term := terminal.NewTerminal(session, "enter command >> ")
  io.WriteString(session, fmt.Sprintf("Successful login, %s\n", session.User()))
  for {
    command, err := term.ReadLine()
    if err != nil {
      log.Println(err)
    }
    log.Printf("Received %s: %s", session.User(), command)
    selectCommand(session, command)
  }
}

func selectCommand(session ssh.Session, command string) {
  switch command {
  case "exit":
    return
  case "ping":
    output, err := runCommand("ping", "-c", "4", "google.com")
    if err != nil {
      io.WriteString(session, fmt.Sprintf("Error ping: %s\n", err))
    } else {
      io.WriteString(session, string(output))
    }
  default:
    args := strings.Fields(command)
    if len(args) > 0 {
      switch args[0] {

      case "mkdir":
        if len(args) < 2 {
          io.WriteString(session, "You must specify dir name\n")
        } else {
          err := os.Mkdir(path+args[1], 0755)
          if err != nil {
            io.WriteString(session, fmt.Sprintf("Failed to create dir: %s\n", err))
          } else {
            io.WriteString(session, "Dir created\n")
          }
        }
```

Листинг 5 — server.go - продолжение

```
case "rmdir":
        if len(args) < 2 {
          io.WriteString(session, "You need to specify dir to delete\n")
        } else {
          err := os.Remove(path + args[1])
          if err != nil {
            io.WriteString(session, fmt.Sprintf("Failed to delete: %s\n", err))
          } else {
            io.WriteString(session, "Dir successfully deleted\n")
          }
        }

      case "ls":
        files, err := os.ReadDir(path)
        if err != nil {
          io.WriteString(session, fmt.Sprintf("Error reading dir: %s\n", err))
        } else {
          var output strings.Builder
          for _, file := range files {
            output.WriteString(file.Name())
            output.WriteString("\n")
          }
          io.WriteString(session, output.String())
        }

      case "mv":
        if len(args) < 3 {
          io.WriteString(session, "You need to specify src and dst paths\n")
        } else {
          err := os.Rename(path+args[1], path+args[2]+"/"+args[1])
          if err != nil {
            io.WriteString(session, fmt.Sprintf("Error while moving: %s\n", err))
          } else {
            io.WriteString(session, "File successfully moved\n")
          }
        }

      case "rm":
        if len(args) < 2 {
          io.WriteString(session, "You need to specify file to delete\n")
        } else {
          err := os.Remove(path + args[1])
          if err != nil {
            io.WriteString(session, fmt.Sprintf("Error while deleting: %s\n", err))
          } else {
            io.WriteString(session, "Successfully deleted\n")
          }
        }
      case "cd":
        if len(args) < 2 {
          io.WriteString(session, "You need to specify dir to change\n")
        } else {
          _, err := os.ReadDir(path + args[1])
```

Листинг 6 — server.go - продолжение

```
if err != nil {
            io.WriteString(session, fmt.Sprintf("Error - no such dir: %s\n", err))
          } else {
            io.WriteString(session, "Dir changed successfully\n")
            path += args[1] + "/"
          }
        }

      default:
        io.WriteString(session, "Wrong command\n")
      }
    } else {
      io.WriteString(session, "Wrong command\n")
    }
  }
}

func runCommand(name string, args ...string) ([]byte, error) {
  cmd := exec.Command(name, args...)
  output, err := cmd.CombinedOutput()

  if err != nil {
    return nil, fmt.Errorf("command execution failed with: %s", err)
  }
  return output, nil
}

func main() {
  server := ssh.Server{
    Addr: "localhost:2222",
    Handler: func(s ssh.Session) {
      handleSession(s)
    },
    PasswordHandler: func(ctx ssh.Context, password string) bool {
      log.Printf("User authentication %s", ctx.User())
      return true
    },
  }
  log.Println("SSH started at port 2222")

  err := server.ListenAndServe()
  if err != nil {
    log.Fatalf("Error starting server: %v", err)
  }
}
```

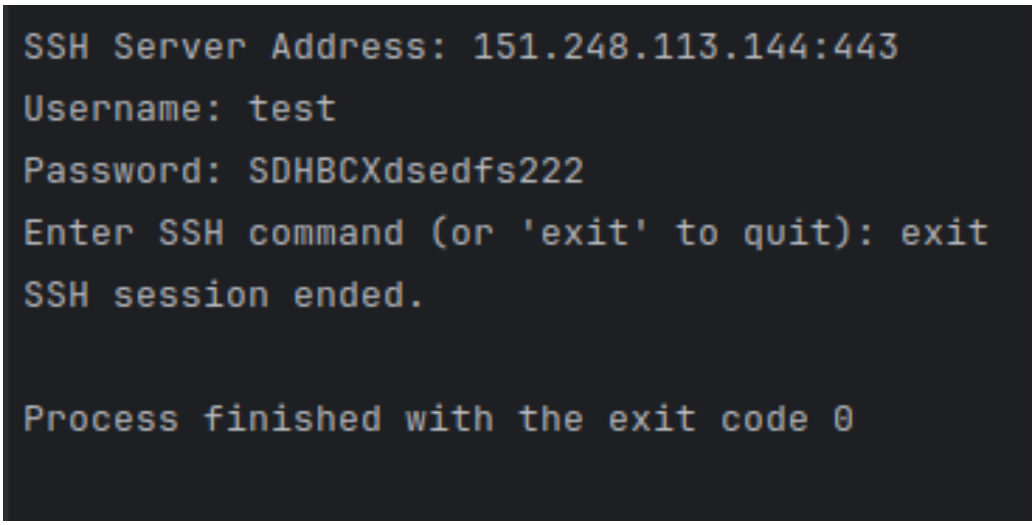

Рис. 1 — Клиент

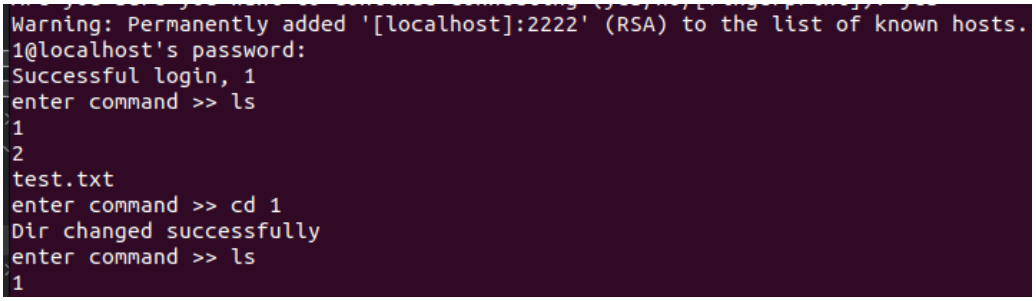

Рис. 2 — Сервер