



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

```

1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "golang.org/x/crypto/ssh"
7     "os"
8     "strings"
9 )
10
11 func main() {
12     reader := bufio.NewReader(os.Stdin)
13
14     fmt.Print("SSH Server Address: ")
15     serverAddress, _ := reader.ReadString('\n')
16     serverAddress = strings.TrimSpace(serverAddress)
17
18     fmt.Print("Username: ")
19     username, _ := reader.ReadString('\n')
20     username = strings.TrimSpace(username)
21
22     fmt.Print("Password: ")
23     password, _ := reader.ReadString('\n')
24     password = strings.TrimSpace(password)
25
26     config := &ssh.ClientConfig{
27         User: username,
28         Auth: []ssh.AuthMethod{
29             ssh.Password(password),
30         },
31         HostKeyCallback: ssh.InsecureIgnoreHostKey(),
32     }
33
34     client, err := ssh.Dial("tcp", serverAddress, config)
35     if err != nil {
36         fmt.Println("Failed to connect to the SSH server:", err)
37         return
38     }
39     defer func(client *ssh.Client) {
40         err := client.Close()
41         if err != nil {
42
43         }
44     }(client)
45     currentDir := "" //
46
47     for {
48         fmt.Print("Enter SSH command (or 'exit' to quit): ")
49         command, _ := reader.ReadString('\n')
50         command = strings.TrimSpace(command)
51
52         if command == "exit" {
53             break
54         }
55         session, err := client.NewSession()
56         if err != nil {
57             fmt.Println("Failed to create SSH session:", err)
58             continue
59         }

```

Листинг 2 — client.go - продолжение

```

1 switch {
2     case strings.HasPrefix(command, "ls"):
3         output, err := session.CombinedOutput("ls " + currentDir)
4         err = session.Close()
5         if err != nil {return}
6         if err != nil {fmt.Println("Failed to list directory:", err)}
7         else {fmt.Println("Directory contents:\n", string(output
8     case strings.HasPrefix(command, "cd"):
9         newDir := strings.TrimSpace(strings.TrimPrefix(command, "cd"))
10        Path := newDir
11        if currentDir != "" {Path = currentDir + "/" + newDir}
12        output, err := session.CombinedOutput("cd " + Path + " && pwd")
13        currentDir = Path
14        err = session.Close()
15        if err != nil {
16            return
17        }
18        if err != nil {
19            fmt.Println("Failed to change directory:", err)
20        } else {
21            currentDir = strings.TrimSpace(string(output))
22            fmt.Println("Current directory:", currentDir)
23        }
24        case strings.HasPrefix(command, "mkdir"):
25            dirName := strings.TrimSpace(strings.TrimPrefix(command, "mkdir"))
26            _, err := session.CombinedOutput("mkdir " + currentDir + "/" +
27            dirName)
28            if err != nil {
29                fmt.Println("Failed to create directory:", err)
30            } else {
31                fmt.Println("Directory created:", dirName)
32            }
33            case strings.HasPrefix(command, "rmdir"):
34                dirName := strings.TrimSpace(strings.TrimPrefix(command, "rmdir"))
35                Path := dirName
36                if currentDir != "" {
37                    Path = currentDir + "/" + dirName
38                }
39                _, err := session.CombinedOutput("rmdir " + Path)
40                if err != nil {
41                    fmt.Println("Failed to remove directory:", err)
42                } else {
43                    fmt.Println("Directory removed:", dirName)
44                }
45                case strings.HasPrefix(command, "mv"):
46                    args := strings.Split(command, " ")
47                    if len(args) != 3 {
48                        fmt.Println("Usage: mv source destination")
49                    } else {
50                        sourcePath := args[1]
51                        destPath := args[2]
52                        _, err := session.CombinedOutput("mv " + sourcePath + " " +
53                        destPath)
54                        if err != nil {
55                            fmt.Println("Failed to move:", err)
56                        } else {
57                            fmt.Println("Moved:", sourcePath, "to", destPath)
58                        }
59                    }
60                }

```

Листинг 3 — client.go - продолжение

```
1 default :  
2     output, err := session.CombinedOutput(command)  
3  
4     if err != nil {  
5         fmt.Println("Failed to execute the command:", err)  
6     } else {  
7         fmt.Println("Command output:\n", string(output))  
8     }  
9 }  
10 }  
11  
12     fmt.Println("SSH session ended.")  
13 }
```

Листинг 4 — server.go

```

1 package main
2
3 import (
4     "fmt"
5     "github.com/gliderlabs/ssh"
6     "golang.org/x/crypto/ssh/terminal"
7     "io"
8     "log"
9     "os"
10    "os/exec"
11    "strings"
12 )
13
14 var path = "./lab4/server/server_space/"
15 var authorizedKeysPath = "./lab4/server/authorized_keys"
16
17 func handleSession(session ssh.Session) {
18     term := terminal.NewTerminal(session, "enter command >> ")
19     io.WriteString(session, fmt.Sprintf("Successful login, %s\n", session.
20         User()))
21     for {
22         command, err := term.ReadLine()
23         if err != nil {
24             log.Println(err)
25         }
26         log.Printf("Received %s: %s", session.User(), command)
27         selectCommand(session, command)
28     }
29 }
30
31 func selectCommand(session ssh.Session, command string) {
32     switch command {
33     case "exit":
34         return
35     case "ping":
36         output, err := runCommand("ping", "-c", "4", "google.com")
37         if err != nil {
38             io.WriteString(session, fmt.Sprintf("Error ping: %s\n", err))
39         } else {
40             io.WriteString(session, string(output))
41         }
42     default:
43         args := strings.Fields(command)
44         if len(args) > 0 {
45             switch args[0] {
46             case "mkdir":
47                 if len(args) < 2 {
48                     io.WriteString(session, "You must specify dir name\n")
49                 } else {
50                     err := os.Mkdir(path+args[1], 0755)
51                     if err != nil {
52                         io.WriteString(session, fmt.Sprintf("Failed to create dir: %
53 s\n", err))
54                     } else {
55                         io.WriteString(session, "Dir created\n")
56                     }
57                 }
58             }
59         }
60     }
61 }

```

Листинг 5 — server.go - продолжение

```

1 case "rmdir":
2     if len(args) < 2 {
3         io.WriteString(session, "You need to specify dir to delete\n")
4     } else {
5         err := os.Remove(path + args[1])
6         if err != nil {
7             io.WriteString(session, fmt.Sprintf("Failed to delete: %s\n",
8 , err))
9         } else {
10            io.WriteString(session, "Dir successfully deleted\n")
11        }
12    }
13    case "ls":
14        files, err := os.ReadDir(path)
15        if err != nil {
16            io.WriteString(session, fmt.Sprintf("Error reading dir: %s\n",
17 err))
18        } else {
19            var output strings.Builder
20            for _, file := range files {
21                output.WriteString(file.Name())
22                output.WriteString("\n")
23            }
24            io.WriteString(session, output.String())
25        }
26    case "mv":
27        if len(args) < 3 {
28            io.WriteString(session, "You need to specify src and dst paths
29 \n")
30        } else {
31            err := os.Rename(path+args[1], path+args[2]+"/"+args[1])
32            if err != nil {
33                io.WriteString(session, fmt.Sprintf("Error while moving: %s\
34 n", err))
35            } else {
36                io.WriteString(session, "File successfully moved\n")
37            }
38        }
39    case "rm":
40        if len(args) < 2 {
41            io.WriteString(session, "You need to specify file to delete\n"
42 )
43        } else {
44            err := os.Remove(path + args[1])
45            if err != nil {
46                io.WriteString(session, fmt.Sprintf("Error while deleting: %
47 s\n", err))
48            } else {
49                io.WriteString(session, "Successfully deleted\n")
50            }
51        }
52    case "cd":
53        if len(args) < 2 {
54            io.WriteString(session, "You need to specify dir to change\n")
55        } else {
56            _, err := os.ReadDir(path + args[1])

```

Листинг 6 — server.go - продолжение

```

1 if err != nil {
2     io.WriteString(session, fmt.Sprintf("Error - no such dir: %s\n", err))
3     } else {
4         io.WriteString(session, "Dir changed successfully\n")
5         path += args[1] + "/"
6     }
7 }
8
9     default:
10        io.WriteString(session, "Wrong command\n")
11    }
12    } else {
13        io.WriteString(session, "Wrong command\n")
14    }
15 }
16 }
17
18 func runCommand(name string, args ...string) ([]byte, error) {
19     cmd := exec.Command(name, args...)
20     output, err := cmd.CombinedOutput()
21
22     if err != nil {
23         return nil, fmt.Errorf("command execution failed with: %s", err)
24     }
25     return output, nil
26 }
27
28 func main() {
29     server := ssh.Server{
30         Addr: "localhost:2222",
31         Handler: func(s ssh.Session) {
32             handleSession(s)
33         },
34         PasswordHandler: func(ctx ssh.Context, password string) bool {
35             log.Printf("User authentication %s", ctx.User())
36             return true
37         },
38     }
39     log.Println("SSH started at port 2222")
40
41     err := server.ListenAndServe()
42     if err != nil {
43         log.Fatalf("Error starting server: %v", err)
44     }
45 }

```



```
SSH Server Address: 151.248.113.144:443
Username: test
Password: SDHBCXdsedfs222
Enter SSH command (or 'exit' to quit): exit
SSH session ended.

Process finished with the exit code 0
```

Рис. 1 — Клиент

```
Warning: Permanently added '[localhost]:2222' (RSA) to the list of known hosts.
1@localhost's password:
Successful login, 1
enter command >> ls
1
2
test.txt
enter command >> cd 1
Dir changed successfully
enter command >> ls
1
```

Рис. 2 — Сервер