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 № 6.1

по курсу «Компьютерные сети»

«Разработка FTP-клиента и FTP-сервера»

Студент группы ИУ9-32Б Волохов А. В.

Преподаватель Посевин Д. П.

Москва 2023

1 Задание

Рассматривается задача разработки FTP-клиента на языке GO на основе пакета

Задача 1: Реализовать FTP-клиент и запустить его .

Задача 2: Протестировать соединение GO FTP-клиента с установленным на сервере students.yss.su FTP-сервером со следующими параметрами доступа: ftp-host: students.yss.su login: ftpiu8 passwd: 3Ru7yOTA

Задача 3: Реализовать следующие функции: загрузку файла GO FTP-клиентом на FTP-сервер; скачивание файла GO FTP-клиентом с FTP-сервера; создание директории go ftp клиентом на FTP-сервере; удаление GO FTP-клиентом файла на FTP-сервере; получение содержимого директории на FTP-сервере с помощью GO FTP- клиента.

Кроме того, рассматривается задача разработки FTP-сервера на языке GO.

Задача 1: Реализовать FTP-сервер на языке GO.

Задача 2: Протестировать соединение FTP-клиента (для тестирования можно использовать консольную версию FTP-клиента используемой операционной системы или FileZilla, WinSCP и д.р.) с реализованным FTP- сервером. Параметры доступа к ftp серверу могут быть заданы в коде программы или во внешнем конфигурационном файле.

Задача 3: FTP-сервер должен обладать следующими функциями: поддерживать авторизацию клиента на FTP-сервере; передавать клиенту список содержимого заданной директории FTP- сервера по запросу; позволять клиенту скачивать файлы из заданной директории FTP сервера по запросу;○ позволять клиенту загружать файлы в заданную директорию FTP сервера по запросу; позволять клиенту создавать директории на FTP-сервере по запросу; позволять клиенту удалять директории на FTP-сервере по запросу.

Исходный код программы представлен в листингах 1– 2– 4– **??**– **??**– **??**.

Листинг 1 — client.go

```
1  package main
2
3  import (
4    "bufio"
5    "fmt"
6    "github.com/jlaffaye/ftp"
7    "io"
8    "os"
9    "path"
10   "strings"
11 )
12
13 const (
14   ftpHost  = "students.yss.su"
15   ftpLogin = "ftpiu8"
16   ftpPass  = "3Ru7yOTA"
17 )
18
19 func main() {
20   client, err := ftp.Dial(ftpHost + ":21")
21   if err != nil {
22     panic(err)
23   }
24   defer client.Quit()
25   err = client.Login(ftpLogin, ftpPass)
26   if err != nil {
27     panic(err)
28   }
29   fmt.Println("FTP client connected successfully!")
30   scanner := bufio.NewScanner(os.Stdin)
31   for {
32     fmt.Print("                                    (upload, download, create,
       delete, list, exit): ")
33     scanner.Scan()
34     command := scanner.Text()
35
36     switch strings.ToLower(command) {
37     case "upload":
38       fmt.Print("
                : ")
39       scanner.Scan()
40       localPath := scanner.Text()
41       fmt.Print("
                                      : ")
42       scanner.Scan()
43       remotePath := scanner.Text()
44       uploadFile(client, localPath, remotePath)
45
46     case "download":
47       fmt.Print("
                : ")
48       scanner.Scan()
49       remotePath := scanner.Text()
50       fmt.Print("
                                      : ")
51       scanner.Scan()
52       localPath := scanner.Text()
53       downloadFile(client, remotePath, localPath)
```

Листинг 2 — client.go - продолжение

```
case "create":
        fmt.Print("                                                    :
    ")
        scanner.Scan()
        directoryName := scanner.Text()
        createDirectory(client, directoryName)

    case "delete":
        fmt.Print("
                                          : ")
        scanner.Scan()
        remotePath := scanner.Text()
        deleteFile(client, remotePath)

    case "list":
        fmt.Print("                                              : ")
        scanner.Scan()
        remotePath := scanner.Text()
        listDirectory(client, remotePath)

    case "exit":
        fmt.Println("                                        .")
        return

    default:
        fmt.Println("                                    .
    ,                                                    .")
    }
  }
}

func uploadFile(client *ftp.ServerConn, localPath, remotePath string) {
  file, err := os.Open(localPath)
  if err != nil {
    panic(err)
  }
  defer file.Close()

  err = client.Stor(remotePath+"/"+getFileName(localPath), file)
  if err != nil {
    panic(err)
  }
  fmt.Printf("File %s uploaded successfully.\n", localPath)
}
```

Листинг 3 — client.go - продолжение

```
1  func downloadFile(client *ftp.ServerConn, remotePath, localPath string)
       {
2    resp, err := client.Retr(remotePath)
3    if err != nil {
4      panic(err)
5    }
6    defer resp.Close()
7
8    file, err := os.Create(localPath + "/" + getFileName(remotePath))
9    if err != nil {
10     panic(err)
11   }
12   defer file.Close()
13
14   _, err = io.Copy(file, resp)
15   if err != nil {
16     panic(err)
17   }
18
19   fmt.Printf("File %s downloaded successfully.\n", remotePath)
20 }
21
22 func createDirectory(client *ftp.ServerConn, directoryName string) {
23   err := client.MakeDir(directoryName)
24   if err != nil {
25     panic(err)
26   }
27   fmt.Printf("Directory %s created successfully.\n", directoryName)
28 }
29
30 func deleteFile(client *ftp.ServerConn, remotePath string) {
31   err := client.Delete(remotePath)
32   if err != nil {
33     panic(err)
34   }
35   fmt.Printf("File %s deleted successfully.\n", remotePath)
36 }
37
38 func listDirectory(client *ftp.ServerConn, directoryPath string) {
39   entries, err := client.List(directoryPath)
40   if err != nil {
41     panic(err)
42   }
43   fmt.Printf("Directory listing for %s:\n", directoryPath)
44   for _, entry := range entries {
45     fmt.Println(entry.Name)
46   }
47 }
48
49 func getFileName(filePath string) string {
50   _, file := path.Split(filePath)
51   return file
52 }
```

Листинг 4 — server.go

```
1  package main
2
3  import (
4    "flag"
5    filedriver "github.com/goftp/file-driver"
6    "github.com/goftp/server"
7    "log"
8  )
9
10 func main() {
11   var (
12     root = flag.String("root", "/home/alexandr/BMSTU_git/IU9-CN-GO/lab6
       .1/server/server_space", "Root directory to serve")
13     user = flag.String("user", "Sasha", "Username for login")
14     pass = flag.String("pass", "123", "Password for login")
15     port = flag.Int("port", 2121, "Port")
16     host = flag.String("host", "", "Host")
17   )
18   flag.Parse()
19   if *root == "" {
20     log.Fatalf("Please set a root to serve with -root")
21   }
22
23   factory := &filedriver.FileDriverFactory{
24     RootPath: *root,
25     Perm:     server.NewSimplePerm("user", "group"),
26   }
27
28   opts := &server.ServerOpts{
29     Factory:  factory,
30     Port:     *port,
31     Hostname: *host,
32     Auth:     &server.SimpleAuth{Name: *user, Password: *pass},
33   }
34
35   log.Printf("Starting ftp server on %v:%v", opts.Hostname, opts.Port)
36   log.Printf("Username %v, Password %v", *user, *pass)
37   server := server.NewServer(opts)
38   err := server.ListenAndServe()
39   if err != nil {
40     log.Fatal("Error starting server:", err)
41   }
42 }
```

```
/home/alexandr/.cache/JetBrains/GoLand2023.2/tmp/GoLand/___go_build_client_go
FTP client connected successfully!
Введите команду (upload, download, create, delete, list, exit): list
Введите удаленный путь: ./
Directory listing for ./:
.
..
huk
upload
local-file.txt
smile
README.md
fazandir
1111
fileToUpload.txt
PekhovaM
remotedirectory
fazan2dir
test456789899
dogdogdog
zhuk
bbb
huktest.txt
upload1
kp.jpg
Volokhov
```

Рис. 1 — Клиент

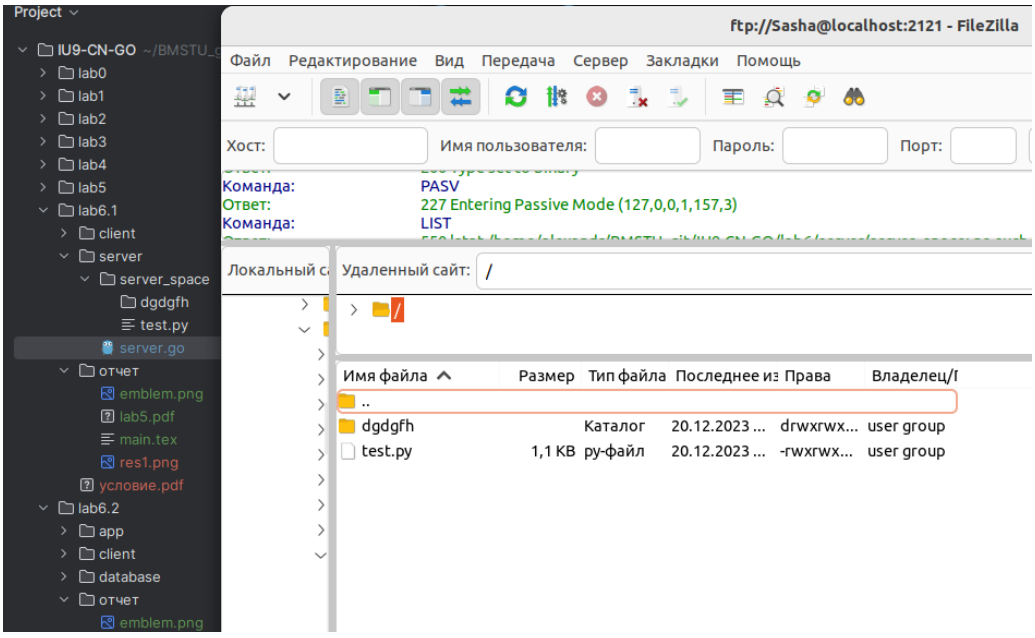

Рис. 2 — Сервер