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 №6
«Разработка ftp-сервера и ftp-клиента»
по курсу: «Компьютерные сети»

Выполнил:
Студент группы ИУ9-32Б
Гнатенко Т. А.

Проверил:
Посевин Д. П.

Москва, 2022

Цели

Изучить принципы работы протокола ssh

Часть 1. Разработка ftp клиента

Задачи

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

Решение

Исходный код

client.go

```
package main

import (
    "bufio"
    "fmt"
    "io/ioutil"
    "os"
    "strings"
    "time"

    "database/sql"

    _ "github.com/go-sql-driver/mysql"
    log "github.com/mgutz/logxi/v1"
```

```
    "github.com/jlaffaye/ftp"
    "github.com/mmcdole/gofeed"
)

func minHour(i int) string {
    if i > 9 {
        return ""
    } else {
        return "0"
    }
}

func timeStr(currentTime time.Time, str string) string {

    return fmt.Sprintf("%s%d-%d-%d%s%s%d:%s%d:%s%d",
        str,
        currentTime.Day(),
        currentTime.Month(),
        currentTime.Year(),
        str,
        minHour(currentTime.Local().Hour()),
        currentTime.Local().Hour(),
        minHour(currentTime.Local().Minute()),
        currentTime.Local().Minute(),
        minHour(currentTime.Local().Second()),
        currentTime.Local().Second())
}

func main() {
    username := "ftpiu8"
    password := "3Ru7yOTA"
    hostname := "students.yss.su"
    port := "ftp"
    // username := "user"
    // password := "123456"
    // hostname := "localhost"
    // port := "2121"
    // Init client
    c, err := ftp.Dial(hostname + ":" + port)
    if err != nil {
```

```
        log.Error("Can't connect to server", "err",
↪   err.Error())
        return
    }
    log.Info("Connect to server success")

    err = c.Login(username, password)
    if err != nil {
        log.Error("User or password wrong")
        return
    }
    log.Info("Client entry success")

    // send the commands
    for {
        fmt.Print("> ")
        reader := bufio.NewReader(os.Stdin)
        line, _, err := reader.ReadLine()
        if err != nil {
            break
        }
        cmd := strings.Split(string(line), " ")

        switch cmd[0] {
        case "exit":
            {
                break
            }
        case "help":
            {
                for _, v := range []string{"exit", "post",
                ↪  "get", "make", "delete", "news",
                ↪  "tree\n"} {
                    fmt.Print(" ", v)
                }
            }
        case "post":
            {
                file, err := os.Open(cmd[1])
                if err != nil {
```

```
            log.Error(err.Error())
            os.Exit(1)
        }
        defer file.Close()

        err = c.Stor(cmd[1], file)
        if err != nil {
            panic(err)
        }
        log.Info("Posting success")
    }
case "get":
    {
        r, err := c.Retr(cmd[1])
        if err != nil {
            panic(err)
        }
        defer r.Close()
        buf, err := ioutil.ReadAll(r)
        if err != nil {
            log.Error(err.Error())
            os.Exit(1)
        }
        file, err := os.Create(cmd[1])

        if err != nil {
            log.Error(err.Error())
            os.Exit(1)
        }
        defer file.Close()
        file.WriteString(string(buf))
        log.Info("Getting success")
    }
case "make":
    {
        err := c.MakeDir(cmd[1])
        if err != nil {
            log.Error(err.Error())
        }
    }
```

```
        case "delete":
            {
                err := c.Delete(cmd[1])
                if err != nil {
                    log.Error(err.Error())
                }
            }
        case "tree":
            {
                dir, err := c.CurrentDir()
                if err != nil {
                    log.Error(err.Error())
                } else {
                    answer, err := c.NameList(dir)
                    if err != nil {
                        log.Error(err.Error())
                    } else {
                        for _, v := range answer {
                            fmt.Println(v)
                        }
                    }
                }
            }
        case "news":
            {
                db, err := sql.Open("mysql",
↪   "iu9networkslabs:Je2dTYr6@tcp(students.yss.su)/iu9networkslabs")

                if err != nil {
                    log.Error(err.Error())
                    return
                }
                defer db.Close()

                fp := gofeed.NewParser()
                feed, err :=
↪   fp.ParseURL("https://news.mail.ru/rss/90/")
                if err != nil {
                    log.Error(err.Error())
                }
```

```
                // make file
                fileName := "Gnatenko_Tanya" +
↪   timeStr(time.Now(), "_") + ".txt"
                count := 0
                buf := ""
                for _, item := range feed.Items {
                    if count == 10 {
                        break
                    }
                    var isInTable bool
                    db.QueryRow("SELECT EXISTS (select *
↪   from `Typic` where title = ?)",
↪   item.Title).Scan(&isInTable)
                    if isInTable {
                        continue
                    }
                    _, err = db.Exec("insert into `Typic`
↪   (`title`, `category`, `description`, `date`, `time`)
↪   values (?, ?, ?, ?, ?);",
                        item.Title, item.Categories[0],
↪   item.Description, item.PublishedParsed.UTC(),
↪   item.PublishedParsed.Local())

                    if err != nil {
                        log.Error(err.Error())
                    } else {
                        count++
                        for _, v := range
                         ↪  []string{item.Title,
                         ↪  item.Categories[0],
                         ↪  item.Description,
                         ↪  timeStr(item.PublishedParsed.UTC(),
                         ↪  " "),
                         ↪  timeStr(item.PublishedParsed.Local(),
                         ↪  " "), "\n"} {
                            buf += " " + v
                        }

                    }
```

```
                }

                r := strings.NewReader(buf)
                c.Append(fileName, r)
                log.Info("Posting news success")

            }
        }
    }

    // Do something with the FTP conn

    if err := c.Quit(); err != nil {
        log.Error(err.Error())
    }
}
```

Часть 2. Разработка ftp сервера

Задачи

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.

Решение

Исходный код

server.go

```
package main

import (
    "flag"
    "log"

    filedriver "github.com/goftp/file-driver"
    "github.com/goftp/server"
)

func main() {
    var (
        root = flag.String("root", "dir", "Root directory to
 ↪  serve")
        user = flag.String("user", "user", "Username for
 ↪  login")
        pass = flag.String("pass", "123456", "Password for
 ↪  login")
        port = flag.Int("port", 2121, "Port")
        host = flag.String("host", "localhost", "Host")
    )
    flag.Parse()
    if *root == "" {
        log.Fatalf("Please set a root to serve with -root")
    }

    factory := &filedriver.FileDriverFactory{
        RootPath: *root,
        Perm:     server.NewSimplePerm("user", "group"),
    }

    opts := &server.ServerOpts{
        Factory:  factory,
        Port:     *port,
        Hostname: *host,
```

```
        Auth:     &server.SimpleAuth{Name: *user, Password:
    ↪  *pass},
    }

    log.Printf("Starting ftp server on %v:%v",
    ↪  opts.Hostname, opts.Port)
    log.Printf("Username %v, Password %v", *user, *pass)
    server := server.NewServer(opts)
    err := server.ListenAndServe()
    if err != nil {
        log.Fatal("Error starting server:", err)
    }
}
```

Вывод

Часть 2. Совместная работа ftp сервера и ftp клиента

Задачи

Задача: Используя свои результаты Лабораторной работы №3 «Импорт новостей в базу данных из RSS-потока» реализовать приложение, которое получает актуальные новости (дублировать нельзя), сохраняет их в файл, а файл загружает на ftp сервер. Имя файла должно иметь формат _дата_время.txt. В итоге на ftp-сервере сервере будут располагаться файлы с уникальными текстами новостей. Если хоть одна новость будет задублена — задача не засчитывается.

Вывод