**// Command-line arguments**

**// are a common way to parameterize execution of programs.**

**// For example, `go run hello.go` uses `run` and**

**// `hello.go` arguments to the `go` program.**

**package main**

**import (**

**"fmt"**

**"os"**

**)**

**func main() {**

**// `os.Args` provides access to raw command-line**

**// arguments. Note that the first value in this slice**

**// is the path to the program, and `os.Args[1:]`**

**// holds the arguments to the program.**

**fmt.Println("command-line arguments with program: \n", os.Args)**

**fmt.Println("command-line arguments without program: \n", os.Args[1:])**

**// You can get individual args with normal indexing.**

**if len(os.Args) > 3 {**

**arg := os.Args[3]**

**fmt.Println(arg)**

**}**

**fmt.Print("command-line contains ", len(os.Args)-1, " argument(s) ")**

**fmt.Println("/to say nothing of the path to the program/.")**

**for n, arg := range os.Args {**

**fmt.Printf("%3d. %s\n", n, arg)**

**}**

**}**

**/\***

**For such command-line:**

**H:\Work. GO\command\_line\command\_line.exe a1 a2 a3 "a4 a41" a5**

**the program displays:**

**command-line arguments with program:**

**[H:\Work. GO\command\_line\command\_line.exe a1 a2 a3 a4 a41 a5]**

**command-line arguments without program:**

**[a1 a2 a3 a4 a41 a5]**

**a3**

**command-line contains 5 argument(s) /to say nothing of the path to the program/.**

**0. H:\Work. GO\command\_line\command\_line.exe**

**1. a1**

**2. a2**

**3. a3**

**4. a4 a41**

**5. a5**

**\*/**

II\_09.go

**package main**

**import (**

**"fmt"**

**"os"**

**)**

**func main() {**

**stat, err:= os.Stat("test.txt")**

**if err != nil {**

**fmt.Println(err)**

**return**

**}**

**fmt.Println(stat.Size())**

**f, err := os.OpenFile("test.1", os.O\_APPEND, 0666)**

**if err != nil {**

**fmt.Println(err)**

**return**

**}**

**b:= []byte(" 012345")**

**/\* эквивалентно**

**b:= []byte {32, 48, 49, 50, 51, 52}**

**\*/**

**n, err := f.Write(b)**

**fmt.Println(n)**

**if err != nil {**

**fmt.Println(err)**

**return**

**}**

**stat, err = os.Stat("test.1")**

**if err != nil {**

**fmt.Println(err)**

**return**

**}**

**fmt.Println(stat.Size())**

**err = f.Close()**

**if err != nil {**

**fmt.Println(err)**

**return**

**}**

**fmt.Println("file appended successfully")**

**}**

II\_10d\_append.go

**package main**

**import (**

**"fmt"**

**"os"**

**)**

**func main() {**

**file, err := os.Open("morning.txt")**

**if err != nil {**

**fmt.Println(err)**

**return**

**}**

**defer file.Close()**

**fileinfo, err := file.Stat()**

**if err != nil {**

**fmt.Println(err)**

**return**

**}**

**filesize := fileinfo.Size()**

**buffer := make([]byte, filesize)**

**bytesread, err := file.Read(buffer)**

**if err != nil {**

**fmt.Println(err)**

**return**

**}**

**fmt.Println("bytes read: ", bytesread)**

**fmt.Println("bytestream to string: ")**

**fmt.Println(string(buffer))**

**}** II\_10a\_read1.go

**package main**

**import (**

**"fmt"**

**"os"**

**)**

**func main() {**

**f, err := os.Create("test.txt")**

**if err != nil {**

**fmt.Println(err)**

**return**

**}**

**d2 := []byte{104, 101, 108, 108, 111, 32, 119, 111, 114, 108, 100}**

**// d2 := []byte("hello world") - то же самое**

**n2, err := f.Write(d2)**

**if err != nil {**

**fmt.Println(err)**

**f.Close()**

**return**

**}**

**fmt.Println(n2, "bytes written successfully")**

**err = f.Close()**

**if err != nil {**

**fmt.Println(err)**

**return**

**}**

**}** II\_10c\_write.go

**package main**

**import (**

**"fmt"**

**"io"**

**"os"**

**)**

**func main() {**

**const BufferSize = 500**

**file, err := os.Open("morning.txt")**

**if err != nil {**

**fmt.Println(err)**

**return**

**}**

**defer file.Close()**

**buffer := make([]byte, BufferSize)**

**for {**

**bytesread, err := file.Read(buffer)**

**// err value can be io.EOF, which means that we reached the end of**

**// file, and we have to terminate the loop. Note the fmt.Println lines**

**// will get executed for the last chunk because the io.EOF gets**

**// returned from the Read function only on the \*next\* iteration, and**

**// the bytes returned will be 0 on that read.**

**if err != nil {**

**if err != io.EOF {**

**fmt.Println(err)**

**}**

**break**

**}**

**fmt.Println("===> bytes read: ", bytesread)**

**fmt.Println("===> bytestream to string: ")**

**fmt.Println(string(buffer[:bytesread]))**

**}**

**}**

II\_10b\_read2.go