




Golang Developer. Professional

otus.ru

 Проверить, идет ли запись

Меня хорошо видно && слышно?

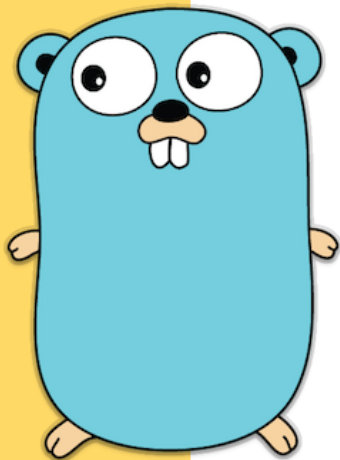
 Ставим “+”, если все хорошо
“-”, если есть проблемы

Тема вебинара

CLI

Романовский Алексей

Разработчик в Resolver Inc.



Правила вебинара



Активно
участвуем



Off-topic обсуждаем
в учебной группе

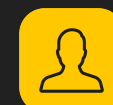


Задаем вопрос
в чат или голосом



Вопросы вижу в чате,
могу ответить не сразу

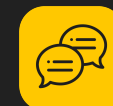
Условные обозначения



Индивидуально



Время, необходимое
на активность



Пишем в чат



Говорим голосом



Документ



Ответьте себе или
задайте вопрос

цели занятия

после занятия вы сможете: работать с операционной системой из программы на Go.

Краткое содержание

- обработка аргументов командной строки: flags, pflag, cobra;
- работа с переменными окружения;
- запуск внешних программ;
- создание временных файлов;
- обработка сигналов.

Соглашения и стандартны на CLI

- POSIX: https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap12.html

```
utility_name[-a][-b][-c option_argument]
            [-d|-e][-f[option_argument]][operand...]
```

- GNU: https://www.gnu.org/prep/standards/standards.html#Command_002dLine-Interfaces

```
utility_name -v
utility_name --verbose
```

flag/pflag

- <https://golang.org/pkg/flag/>
- <https://github.com/spf13/pflag>

(пример на сл слайде)


```
func main() {  
    var msg, cfg string  
  
    //verbose := flag.Bool("verbose", false, "verbose output")  
    verbose := flag.Bool("verbose", "v", false, "verbose output")  
  
    //flag.StringVar(&cfg, "cfg", "config.yaml", "config file")  
    pflag.StringVar(&cfg, "cfg", "config.yaml", "config file")  
  
    pflag.Parse() // flag.Parse()  
  
    if *verbose {  
        fmt.Println("you say:", msg)  
    } else {  
        fmt.Println(msg, cfg)  
    }  
}
```

pflag: флаги без значений

```
pflag.StringVar(&flagvar, "port", "80", "message to print")  
pflag.Lookup("port").NoOptDefVal = "8080"
```

Флаг	Значение
--port=9999	flagvar=9999
--port	flagvar=8080
[nothing]	flagvar=80

Сложные CLI приложения

```
git commit -m 123  
  
docker pull  
  
aws s3 ls s3://bucket-name
```

- [Философия и best practices](#)
- <https://github.com/spf13/cobra/>
- <https://github.com/urfave/cli>

Демонстрация кода (cobra, cobra-cli)

Переменные окружения (дополнительно)

(изучим это в IDE)

```
env := os.Environ() // слайс строк "key=value"
fmt.Println(env[0]) // USER=rob

user, ok := os.LookupEnv("USER")
fmt.Println(user) // rob

os.Setenv("PASSWORD", "qwe123") // установить
os.Unsetenv("PASSWORD")         // удалить (для новых процессов)
fmt.Println(os.ExpandEnv("$USER lives in ${CITY}")) // "шаблонизация"
```

Запуск внешних программ

os/exec , (изучим это в IDE)

```
cmd := exec.Command("git", "commit", "-am", "fix")
```

```
type Cmd struct {  
    Path string // Путь к запускаемой программе  
    Args []string // Аргументы командной строки  
    Env  []string // Переменные окружения ("key=value")  
    Dir  string  // Рабочая директория  
  
    // Поток ввода, вывода и ошибок для программы (/dev/null если nil!)  
    Stdin io.Reader  
    Stdout io.Writer  
    Stderr io.Writer  
    ...  
}
```

```
cmd.CombinedOutput() // ждёт, отдаёт весь вывод как массив, обрабатывает код результата  
cmd.Run() // ждёт ответа, но ничего не делает с потоками  
cmd.Start() // не ждёт. Должны вызвать Wait()
```

Сигналы

Сигналы - механизм OS, позволяющий посылать уведомления программе в особых ситуациях.

Сигнал	Поведение	Применение
SIGINT	Завершить	<code>Ctrl+C</code> в консоли
SIGKILL	Завершить	<code>kill -9</code> , остановка зависших программ
SIGHUP	Завершить	Сигнал для переоткрытия логов и перечитывания конфига
SIGUSR1		На усмотрение пользователя
SIGUSR2		На усмотрение пользователя
SIGPIPE	Завершить	Отправляется при записи в закрытый файловый дескриптор
SIGSTOP	Остановить	При использовании отладчика
SIGCONT	Продолжить	При использовании отладчика

Некоторые сигналы, например `SIGINT` , `SIGUSR1` , `SIGHUP` , можно игнорировать или установить обработчик.

Некоторые, например `SIGKILL` , обработать нельзя.

Обработка сигналов

```
func main() {  
    c := make(chan os.Signal, 1)  
    signal.Notify(c, syscall.SIGINT, syscall.SIGKILL)  
    signal.Ignore(syscall.SIGTERM)  
  
    for s := range c {  
        fmt.Println("Got signal:", s)  
    }  
}
```

Работа с файловой системой

В пакете `os` содержится большое количество функций для работы с файловой системой.

```
// изменить права доступа к файлу
func Chmod(name string, mode FileMode) error

// изменить владельца
func Chown(name string, uid, gid int) error

// создать директорию
func Mkdir(name string, perm FileMode) error

// создать директорию (вместе с родительскими)
func MkdirAll(path string, perm FileMode) error

// переименовать файл/директорию
func Rename(oldpath, newpath string) error

// удалить файл (пустую директорию)
func Remove(name string) error

// удалить рекурсивно rm -rf
func RemoveAll(path string) error

// создать временный файл со случайным именем
func CreateTemp("dir", "prefix") (*os.File, error)
```

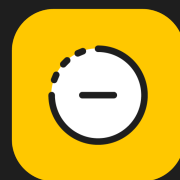
Временные файлы: safefile

<https://github.com/dchest/safefile>

Вопросы?



Ставим “+”,
если вопросы есть



Ставим “-”,
если вопросов нет

**Заполните, пожалуйста,
опрос о занятии
по ссылке в чате**

Следующий вебинар

12 апреля

Рефлексия



Ссылка на вебинар будет в ЛК за 15 минут



Материалы к занятию в ЛК — можно изучать



Обязательный материал обозначен красной лентой



Спасибо за внимание!

Приходите на следующие вебинары

Романовский Алексей

Разработчик в Resolver Inc.

