Урок Спринт 1.13

Логирование в Go

Представьте ситуацию: вы пишете программу, программа выдаёт ошибку, а вы не знаете, откуда она взялась. Обидно, правда? Можно убить на расследование много часов, но так и не найти причину сбоя.

В этой непростой ситуации нам помогут **логи**.

В логах программы записывают события или сообщения.

Вот где ведутся логи:

1. **Файлы логов** — программа создаёт текстовые файлы и записывает в них события, сообщения об ошибках, информацию о работе и т. д. Файлы логов хранятся в определённой директории на диске. Это самый распространённый тип хранения логов.
2. **Консоль** — программа выводит логи в консольный интерфейс, который может прочесть её оператор или разработчик.
3. **Центральные системы мониторинга** — логи отправляются в центральные системы мониторинга (Elasticsearch и Kibana, Splunk или Prometheus с Grafana), которые анализируют эти данные.
4. **Базы данных** — логи вносятся в базы данных для анализа и отчётности.

Рассмотрим файлы и консоль подробнее.

Для логирования в Go есть встроенный пакет `log`. Он помогает записывать сообщения логов в различные уровни серьёзности: `Info`, `Warning` и `Error`.

Вот как можно задействовать `log` для записи сообщения `Hello, World!` в лог уровня `Info`:

```
package main

import (
    "log"
)

func main() {
    log.Println("Hello, World!")
}
```

При запуске этой программы в консоли появится сообщение:

```
2021/10/20 12:00:00 Hello, World!
```

С помощью `log` также можно задавать формат вывода сообщений и выбирать место для записи логов. Вот какой код нужен, чтобы записать логи в файл `log.txt`:

```
package main

import (
    "log"
    "os"
)

func main() {
    // Открываем файл
    file, err := os.OpenFile("log.txt", os.O_CREATE|os.O_WRONLY|os.O_APPEND, 0644)
    if err != nil {
        return
    }
    // Закрываем файл после выхода из main
```

```
    defer file.Close()
    // Конфигурируем логгер, чтобы он выводил лог в файл
    log.SetOutput(file)

    log.Println("Hello, World!")
}
```

Разберёмся в коде подробнее:

1. `"log.txt"`: строка с именем файла, который вы хотите открыть или создать.
2. `os.O_CREATE|os.O_WRONLY|os.O_APPEND`: комбинация флагов, которые определяют, как будет открыт файл:

- `os.O_CREATE`: указывает, что файл должен быть создан, если его нет. Если файл существует, этот флаг ничего не делает.
- `os.O_WRONLY`: файл будет открыт только для записи (write-only). Вы не сможете читать из этого файла внутри программы.
- `os.O_APPEND`: данные будут добавляться в конец файла, а не перезаписывать его содержимое.

3. `0644`: восьмеричное число, которое даёт права доступа к файлу. `0644` означает, что файл будет доступен для чтения и записи владельцу файла, а остальным — только для чтения.

Что произойдёт, когда эта строка кода будет выполнена и если файл `"log.txt"` при этом будет успешно открыт или создан с указанными параметрами? Переменная `file` будет содержать указатель на этот файл, а переменная `err` будет равна `nil` — то есть указывать, что ошибок при открытии файла не было.

Вы можете использовать переменную `file`, чтобы записывать данные в файл `"log.txt"`. Если у вас возникнут ошибки при открытии файла, информацию о них можно будет найти в переменной `err`.