

Вау! ИИ готовит к ЕГЭ по информатике Попробовать

Новый конкурс! Нарисуйте <u>комикс</u> или <u>инфографику</u> на любую тему из области информатики. Приём работ до 17 марта.

Урок Спринт 2.1

Горутины

Горутины — это абстракция в языке Go, которая позволяет запускать функции в асинхронном режиме.

Давайте рассмотрим небольшой пример горутины.

```
package main

import (
    "fmt"
)

func doSomething() {
    fmt.Println("hello world")
}

func main() {
    go doSomething()
}
```

Запустите этот код и посмотрите что выведет программа.

Скорее всего — ничего. Почему это так работает рассмотрим чуть позже

```
package main

import (
    "fmt"
    "time"
)

func doSomething() {
```

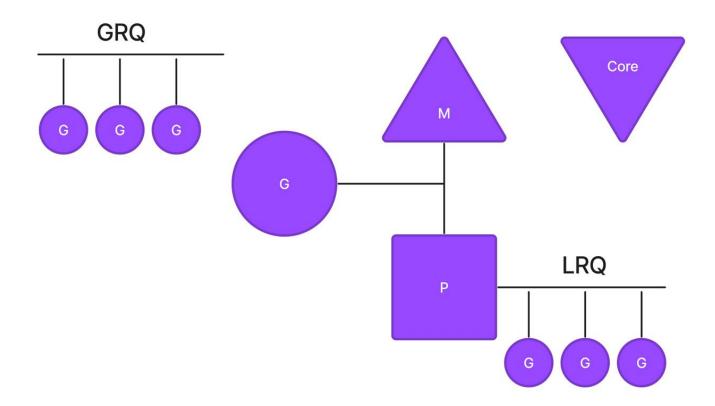
1 of 3 3/15/24, 16:18

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

func main() {
    go doSomething()
    time.Sleep(1 * time.Second)
}
```

В этом примере функция doSomething() запускается в горутине, и затем вызывается time.Sleep(), чтобы основная горутина не завершилась до того, как горутина doSomething() завершит свою работу. Благодаря time.Sleep(), программа успевает вывести "Hello, World!" перед тем как завершится.

Планировщик Go



Планировщик Go работает по такому принципу:

Планировщик Go — это часть языка Go, которая отвечает за управление горутинами. Он решает, какая горутина должна быть запущена, когда и на каком ядре.

Планировщик Go работает на уровне **M:G:P**, где **M** — это количество операционных потоков (0S threads), **G** — это горутины, а **P** — это количество логических процессоров.

Суть этой модели в том, что горутины (G) планируются для выполнения на операционных потоках (M), которые в свою очередь запускаются на логических процессорах (N).

То есть грубо говоря мы берем какой-то поток, на него подставляем Р (некий процессор) и планировщик

2 of 3 3/15/24, 16:18

определяет какая горутина работает в определенный момент времени на этом "процессоре".

В планировщике Go, LRQ (Local Run Queue) и GRQ (Global Run Queue) являются важными структурами данных, которые используются для управления горутинами.

LRQ, или Local Run Queue, — это очередь выполнения для конкретного процессора (P). Каждый Р имеет свою собственную LRQ, которая содержит горутины, готовые к выполнению на этом процессоре.

GRQ, или Global Run Queue, с другой стороны, — это общая очередь выполнения, которая содержит горутины, готовые к выполнению, но которые еще не были назначены конкретному процессору (Р). Когда Р ищет горутину для выполнения и его LRQ пуст, он может взять горутину из GRQ.

Короткая заметка по горутинам:

- Горутина это структура, которая выполняет переданную функцию.
- Самая тяжелый по памяти элемент структуры stack. По умолчанию выделяется 2Кб
- В процессе выполнения стек может увеличиваться, если потребуется.
- У стека есть максимальный размер. 1Гб для 64бит, 250Кб для 32бит

```
package main

import (
    "fmt"
    "runtime"
)

func main() {
    fmt.Println(runtime.NumGoroutine())
}
```

Запустите этот код и посмотрите результат. Вы увидите ответ "1". Но почему? Потому что main — это тоже горутина, она — главная горутина в каждой программе и из нее порождаются другие горутины.

Справка

Исключительное право на учебную программу и все сопутствующие ей учебные материалы, доступные в рамках сервиса, принадлежат АНО ДПО «Образовательные технологии Яндекса». Воспроизведение, копирование, распространение и иное использование программы и материалов допустимо только с предварительного письменного согласия АНО ДПО «Образовательные технологии Яндекса».

Пользовательское соглашение.

© 2018 - 2024 ООО «Яндекс»

3 of 3