## Текстовый вариант видеоурока из предыдущего шага

Еще одним инструментом, который пользуется популярностью в языке, является WaitGroup. С помощью WaitGroup мы можем определить группу горутин, которые должны выполняться вместе, как одна группа. Также мы можем ждать окончания их выполнения. WaitGroup — это, на самом деле, тоже структура из того же пакета sync. Мы можем присвоить в переменную wg реализацию WaitGroupы? с помощью создания новой структуры WaitGroup. Как с этим можно работать? посмотрим на примерах кода.

Смотрите, здесь у меня есть метод sleep, в который я передаю время, которое нужно спать, грубо говоря, и нашу структуру WaitGroup. Да? и как видим, что WaitGroup мы передаем тоже по ссылке, потому что нам важно не создавать каждый раз копию, а работать с одним экземпляром. И в этой функции я печатаю, что sleep столько-то секунд, например, потом происходит сам sleep, а потом я делаю wg.Done, к этому чуть позже я вернусь.

Смотрите в main я создаю WaitGroup? то есть это структура, и у нее есть такой метод? как add. На самом деле, что здесь происходит, когда я говорю add1. У WaitGroup есть некий счетчик, который считает, сколько горутин, ну или каких-то процессов в нашем приложении, мы запустили и сколько мы, как долго мы хотим ждать. То есть, когда мы инкрементируем количество процессов, потом вызываем функцию Wait, то функция Wait будет ждать, пока этот counter не уменьшится до 0. Что это значит? Это значит, что? если мы здесь увеличили на 1. И говорим, что вот я запускаю какой-то процесс и регистрирую его в счетчике WaitGroup, то вот этот процесс, который запускается после этого, этот счетчик должен уменьшить после того, как он отработает. Поэтому в методе sleep в конце я вызываю Done, который как раз таки уменьшает на 1 наш счетчик.

То есть получается следующая картина, я добавил в счетчик, что ожидаю, что один процесс запущен, горутина это отрабатывает, и возвращает эту единичку в счетчике, получаем снова 0. И далее опять я увеличиваю наш счетчик на 1, после запуска метода sleep, он завершается, и снова возвращает значение в -1, и так далее. Соответственно, Wait будет ожидать, пока у нас значение счетчика не станет равно 0. Давайте запустим и посмотрим на результат. Смотрите, что здесь произошло. Мы запустили параллельно три горутины и ждем, пока все они три отработают, то есть по факту мы будем ждать три секунды, потому что это время самой долгой горутины. Так как пока эта горутина отрабатывает за три секунды, параллельно вот эти две также отработают, потому что у этой sleep одна секунда, а у этой две. В итоге мы ждем три секунды для завершения всех трех горутин. Посмотрим еще раз, вот программа ожидает, все горутины закончили работать, и мы вышли.

Что было бы без WaitGroup? Ну, как мы помним, нам приходилось ставить sleep-ы и так далее. В данном случае, мы просто завершили main функцию, и горутины не успели отработать. Давайте посмотрим на ситуацию, когда мы, например, добавили в счетчик еще одну операцию, что мы, например, хотим сделать что-то еще, но при этом ничего не делаем. То есть, нету какой-то горутины, которая бы этот счетчик скинула обратно. И мы видим, что мы получили deadlock, потому что Wait начинает ждать, когда счетчик станет равным нулю, а он равным 0 никогда не станет, потому что у нас нет никакой операции, которая бы скинула этот счетчик в 0.