Текстовый вариант видеоурока из предыдущего шага

Помимо цикла, мы можем работать с каналами, с помощью инструкции select. Особенностью этой инструкции является то, что здесь мы можем работать и опрашивать несколько каналов.

Как мы видели ранее, если мы будем последовательно читать из разных каналов, то это у нас не очень получится, потому что мы будем постоянно блокироваться. Ну, и вы видели, что цикл тоже блокируется на канале, пока канал не закроется. А инструкция select как раз позволяет нам читать из разных каналов, и при этом избегать блокировок. Соответственно, мы можем запустить select в горутине и ожидать какую-то коммуникационную операцию для выполнения. Но есть нюансы. В приведенном виде select будет блокироваться, поэтому на примерах мы разберем, как с этим можно работать более корректно.

Вы можете заметить, что чем-то select похож на switch, только используется ключевое слово «select», а в кейсе у нас не булевые выражения, а работа с каналами. И, конечно же, посмотрим на код. В функции main у нас теперь есть 2 канала: ch и quit. Как мы видим, они оба не буферизированные. То есть, любая операция чтения или записи в этот канал будет приводить к блокировке.

Смотрите, теперь у нас есть select. И в одном из кейсов этого select-а, мы читаем с канала ch. И просто выводим результат. Во втором, здесь тоже немножко новое для вас может быть. Мы здесь просто читаем значение с quit, но никуда его не присваиваем.

Для чего это сделано? Можно просто вычитывать значение, то есть блокироваться на канале, дожидаясь, пока там что-то появится, и как только там что-то появилось, нам нужно что-то сделать. Таким образом обычно делают, когда нам не важно, что лежит в канале: нам важна сама суть того, что там что-то появилось.

Но в этом случае, если мы запустим нашу программу, то мы получим deadlock. Почему он срабатывает? Ну, все банально и просто: мы просто блокируемся на каком-то из каналов, и ничего не можем сделать, потому что в каналы никто не пишет.

Для того, чтобы избежать такой ошибки, мы можем сделать следующее: как и у switch, у select-а есть default. Если мы напишем default и выведем, например, что это default, в этом случае мы провалимся просто в дефолтное состояние, выведем «default», и никакой ошибки уже не произойдет.

Как это можно использовать? Давайте напишем маленькую функцию, которая будет просто писать в наш канал. Давайте укажем, что это канал int-ов, и канал тоже для записи. Ну, и в функции мы просто запишем в канал значение 1.

Смотрите, теперь мы объявили наши два канала, и для того, чтобы не блокироваться, ну, то есть, если мы попытаемся сделать вот таким образом, то мы заблокируемся. Смотрите, нам опять выдаст: «ошибка». «Deadlock». Потому что канал ch не буферизированный, мы начинаем туда писать, и блокируемся. Поэтому давайте запишем значение в этот канал в горутине. Для этого нам нужен метод, который мы только, что написали, и запустим его в горутине. Сначала запишем в ch, а потом запишем в канал quit, и посмотрим, что произойдет.

Смотрите, в этом случае мы снова столкнулись с ситуацией, когда горутина не успевает отработать. Давайте уже воспользуемся функцией, которая уже нам знакома: Gosched, про которую я рассказывал. Мы обращаемся к планировщику, и говорим ему, чтобы он переключился на следующую горутину.

Посмотрим, что изменилось. Смотрите, здесь у нас сработал quit, давайте запустим еще раз. И вас может удивить, что сначала у нас срабатывал кейс quit, теперь - ch, и снова - quit. На самом деле, так происходит, потому что порядок

отработки вот этих кейсов не определен заранее, и этим занимается планировщик, поэтому заранее мы сказать не можем, в каком порядке они будут работать. И от запуска к запуску этот порядок может меняться.

И смотрите, какой здесь есть интересный момент: то есть в данном случае мы пишем в наши оба канала, потом переключаемся на горутины, чтобы в каналы записать, и следом select уже отрабатывает и выводит нам значение.

Что, если мы запишем всего в один канал, а во второй канал уже запишем после того, как вызовется select? Давайте я здесь еще добавлю sleep, и выполним нашу программу. Смотрите, здесь мы, хоть и записали значение в канал quit, но мы ничего не вывели, потому что, после того, как мы записали все значения в ch, сработал select, мы вывели это значение, и все. Select отработал, программа пошла дальше. Дальше мы записали в канал quit, подождали 1 секунду, и завершились. Поэтому частой практикой является запуск select в цикле. Давайте попробуем это сделать.

Напишем какую-нибудь функцию, которая читает, передадим туда наши каналы, и скажем о том, что эти каналы - для чтения. Теперь давайте перенесем select в нашу функцию, но запустим это в бесконечном цикле. То есть просто напишем for, и в бесконечном цикле будем выполнять, но здесь нам нужен какой-то сигнал, чтобы завершить нашу функцию, чтобы не повесить ее и бесконечно не крутить в цикле select. Давайте условимся так: если мы в quit получили какое-то значение, то мы просто выходим из функции. Таким образом мы можем сделать следующее: при старте функции main, мы просто в горутине запускаем нашу функцию read, передаем туда наши каналы, и дальше мы можем, на самом деле, в любой последовательности отправить значения в наши каналы. Они должны обработаться. Давайте посмотри на результат. Да, у нас в данном случае произошел выход. Мы убрали здесь вывод информации, давайте сделаем вот так.

Что теперь должно произойти? Мы запустили горутину read, которая в фоне в бесконечном цикле проверяет наши каналы. Как только в кана приходят какие-то данные, горутина сразу отрабатывает, и выводит их нам на экран. То есть я сначала пишу в горутину ch, потом пишу в горутину quit. Сначала сработает вот эта строчка, а потом вот эта строчка.

Давайте еще раз запустим. Да, и смотрите: пока цикл отрабатывал, и в канале ничего не было, мы попадали в default. После того, как мы попали в quit, напечатали quit, и вышли из функции.

Таким образом мы можем запустить горутину, которая в фоне постоянно слушает наши каналы, при этом не блокируется и, как только там появляются какие-то значения, мы выполняем некоторые действия.