Текстовый вариант видеоурока из предыдущего шага

Итак, многозадачность – это своего рода, комплекс методов и средств, которые позволяют нам переключаться между нашими задачами. Тем самым, это может происходить параллельно или псевдопараллельно.

Важной деталью в этом процессе является переключение контекста. Что это такое? – Это процесс прекращения выполнения одной задачи с сохранением информации и состояния. То есть каждый раз, перед тем как нам переключиться на другую задачу, нам необходимо сохранить некоторые данные, которые мы накопили и создали в процессе выполнения этой самой задачи.

Для чего это нужно? Для того чтобы во время возврата к этой задаче и продолжения её выполнения мы загрузили те же данные, на которых остановились перед тем, как переключились на следующую задачу.

То есть, как это происходит? Как приведено, например, на слайде: у нас запускается задача task1. В процессе её выполнения и на момент запуска тоже выделяется некоторая память для сохранения служебной информации и возможно, информации по этой задаче. В процессе выполнения эта информация может меняться, дальше поступает сигнал, что необходимо переключиться на следующую задачу, и в этот момент происходит переключение Context-а. То есть мы сохраняем Context задачи 1, потом восстанавливаем Context задачи 2, если он был. Если не было, то создается новый, и переключаемся на задачу 2. И как вы можете понимать, каждый раз при переключении на новые задачи происходит процесс переключения Context -а, то есть мы сохраняем какие-то данные и какие-то данные восстанавливаем из памяти. Сама по себе эта операция достаточно объёмная и ресурсоёмкая.

И перед тем, как приступить к рассмотрению параллельности в Go, давайте продолжим нашу небольшую вводную часть и вспомним, что такое «процессы» и «потоки» в операционной системе. Чем же они отличаются? Процесс, как правило, означает запуск нашей программы, то есть наша программа при старте запускается в новом процессе.

В свою очередь, поток – это запуск части нашего процесса, то есть наш поток работает внутри процесса. При этом память процессов изолирована, то есть, у нас нет доступа к памяти одного процесса из другого процесса. В свою очередь, потоки в одном процессе разделяют между собой память. Потоки, соответственно, легковеснее, чем процессы, потому что процесс – это более глобальная сущность, чем поток.

И, в связи с этим, потоки завершаются гораздо быстрее, чем процессы. Но и вспомнив про переключения контекста, конечно же, переключить контекст на процессе существенно дороже, чем переключать его на потоке, потому что, как мы сказали выше, процессы не такие легковесные, как потоки.