Модель Android потоков

В рамках Android процессов, может быть множество потоков исполнения. Каждый поток(thread) является отдельным последовательным потоком(flow) управления в пределах всей программы – инструкции выполняются последовательно, одна за другой, и … планировщиком задач операционной системы.

Пока процесс приложения запущен системой и предотвращает прямое вмешательство в данные в адресном пространстве памяти других процессов, потоки могут быть запущены кодом приложения и могут обмениваться данными с другими потоками в этом же процессе. Помимо общих данных, которые совместно используют все потоки в одном и том же процессе, поток может использовать свой собственный кэш памяти для хранения своих данных в своём собственном пространстве памяти.

Основной поток

Когда процесс приложения запускается, исключая служебные потоки из DVM, система создаёт потоки выполнения, называемые основными. Эти потоки, как гласит название, играют большую роль во времени жизни приложения, так как этот поток взаимодействует с компонентами Android UI, обновляет состояние и их вид на экране устройства.

Более того, обычно, все компоненты Android приложений (Activity, Service, ContentProvider, и BroadcastsReceiver) выполняются выше по приоритету, над исполняемым главным потоком. На изображении показаны списки потоков, запущенных внутри процесса приложения с главным потоком во главе списка и уникальным ID потока (thread ID, TID), присвоенным системой:

Основной поток, также известный как UI Thread, также является потоком, в котором происходит обработка UI, чтобы ваше приложение было максимально ответственно выполнено, вы должны:

избегать любое длительное выполнение задач, таких как ввод/вывод (input/output, I/O) что может повлечь за собой блокировку обработки данных на неопределённое количество времени

избегать нагружающих процессор задач, что могут надолго занять поток

На представленной диаграмме показаны все основные взаимодействия и компоненты, участвующие в выполнении потоком строки Looper'а:

UI/Главный поток, к которому подключён Looper, содержит очередь сообщений (MessageQueue) с некоторой единицей работы, которая должна выполняться последовательно.

Когда сообщение готово к обработке в очереди, Looper Thread извлекает сообщение из очереди и далее синхронно передаёт его целевому обработчику, определённому в сообщении.

Когда цель Handler'а заканчивает свою работу с текущим сообщением, Looperthread будет готов к обработке следующего сообщения, доступного в очереди. Следовательно, если Handler потратил значительное количество времени на обработку сообщения, это помешает Looper'у с обработкой других отложенных сообщений.

Например, когда мы прописываем в коде метод onCreate() в Activityclass, то он будет запущен в главном потоке. Более того, когда мы добавляем слушателя (Listener в Java – уведомляемый о некотором событий объект) к компонентам пользовательского интерфейса для обработки нажатий и жестов пользовательского ввода, обратная связь со слушателем выполняется в главном потоке.

Для приложений, которые выполняют мало операций ввода-вывода или обработки, таких как приложения, которые не выполняют комплексные математические вычисления, не используют сеть для реализации функций, или не используют ресурсы файловой системы, это хорошая однопоточная модель. Тем не менее, если нам нужно выполнить, нагружающие процессор, вычисления, читать или записывать файлы из постоянной памяти, или связаться с веб-сервисом, любые события, поступающие дальше, пока выполняется данная задача будут заблокированы, пока мы не закончим.

Начиная с версии 5.0 (Lollipop), был введён новый важный поток именуемый как RenderThread, который добавляет плавную анимацию для пользовательского интерфейса, даже когда основной поток полностью загружен.

Внимание! Этот перевод, возможно, ещё не готов.

Его статус: перевод редактируется

http://tl.rulate.ru/book/12504/619199