Svetlana Solodilov

May 3, 2022

Android Operating System

The Android operating system is one of the most widely used operating systems for mobile devices, and has been supporting customers across the globe since 2005. Android OS is primarily used inside touchscreen devices such as phones, so users can easily manipulate their devices through common finger motions – swiping, tapping, or even pinching the screen to move from one application to the other. Beyond its user-friendly interface, Android stands out because Android's source code is an open source system, which allows users to use, study, and distribute the code themselves to create custom variants of Android OS. However, when sold, Android products are still packaged with proprietary software, meaning they are owned by an organization and not distributed for free. Being one of the most popular operating system's of our time, Android's operating system is incredibly valuable to study, specifically in the way it manages and schedules threads, and manages memory.

Android OS manages threads in an interesting, systematic way. When an application is launched on Android, the Android system creates a thread of execution for the application known as the 'main thread' (otherwise known as the 'UI thread'). This 'main thread' is incredibly important to the OS, since it interacts with the application's components and manages the Android UI through the Android UI Toolkit, which includes tools to build Android UI. The Android UI Toolkit itself is not thread-safe, so Android has rules that state the Android UI Toolkit should not be accessed outside the main thread. Worker threads (otherwise known as 'background threads'), by contrast, are created separately from the main thread, and are used to run other tasks for the user. They are not allowed to manage the UI. Similar to the Linux OS, a

thread on Android can be in one of six states: new, runnable, blocked, waiting, timed_waiting, or terminated. Each state has a different meaning: NEW indicates that a thread has been created, RUNNABLE is used for when a thread has started, and BLOCKED is for when a thread was competing with another thread for a monitor and failed. The WAITING state is for when a thread is waiting for the results of another thread, while TIMED_WAITING is used when a thread is waiting but only for a certain period of time. The last state, TERMINATED, is when thread execution either completes, or is interrupted.

Threads work closely with application components. An application in Android has four basic components (activity, service, receiver, provider), which allow the user or the Android system to access the application. By default, all components from the same application will run in the same process or thread. However, there are ways to specify which process a certain component belongs to through the manifest file, which is a file that explains important information about the application to the OS.

The way the Android OS handles threads is similar to what we learned in class in that threads under the Android OS have states, just like the ones in Linux do. Similarly, Android applications have various components that can be assigned to different processes/threads, just like a portion of the work can be assigned to a thread with the routine method created in Linux.

The Android Scheduler uses two metrics when determining which thread to schedule: nice values, and cgroups. Nice values are used to measure a thread's priority, and range from 19 (low priority) to -20 (high priority), with 0 being the default priority value. Threads with a higher nice value have a lower priority, and therefore will run less often than threads with a lower nice value (and therefore higher priority). UI/main threads are given a default priority, while background threads are given a lower priority. Android's APIs assign background threads certain

priorities, though users can set priorities in certain situations with the setThreadPriority() function before a thread is run. Android's cgroups (control groups) are used to control threads' use of system resources, such as CPU usage, by partitioning threads into several hierarchical groups. For example, If threads have lower background priorities, they are moved to a cgroup that is only allowed to use a small percentage of the CPU if threads in other groups are busy. This strategy is beneficial, since it would allow background threads to make progress on their code without impacting the processes running in the foreground too much.

Memory management on Android is done through the Android Runtime (ART), Android's runtime environment. ART replaced the previous runtime environment, the Dalvik Virtual Machine, and currently, ART manages memory via paging and mmapping. Paging is a memory management strategy where a system will store or retrieve data from a secondary storage – thus reading and writing data from this secondary storage. In paging, memory is separated into equal sized blocks called pages, while physical memory is divided into frames. When a process requests memory, the pages are then mapped to the frames. If a computer runs out of RAM (primary storage), the Android OS will move pages of memory into the computer's hard disk (secondary storage). This frees up RAM for other processes. However, too much reliance on memory paging might lead to inefficiency, since RAM operates at a much faster level than the disk does, and the OS will be forced to wait for the disk to catch up every time a page is swapped. Memory mapping, otherwise known as mmapping, is when an entire file, or a portion of it, is mapped onto a disk to a range of addresses within the application's address space. The application can then access files on this disk. Mmapping can also allow data to be shared between applications, since each application can map sections of the same file.

Android has a noteworthy method of memory garbage collection. ART keeps track of every piece of memory that is allocated, and if that memory is no longer being used by a program, ART frees it. This is done with zero prompt from the user, and happens automatically. Freed memory goes back on the heap, and is thus reclaimed. Android has a generational memory heap. Whenever an object is allocated, those objects will be classified into generations based on their expected life and the size of the object being allocated. For example, the 'Young generation' is reserved for objects that were most recently allocated. Each generational heap has a limit on the amount of memory that their objects can occupy, and when that limit is reached, garbage collection occurs in that generation. This method of garbage collection is rather efficient: for instance, if an object needs to be allocated, Android can free up the 'Young generation' instead of freeing all generations, so there is enough space to allocate the object being requested.

Android's operating system is incredibly fascinating to study and explore, and gives programmers an insightful look into the way Android manages memory, manages threads, and schedules threads on the CPU. Learning about the way in which various operating systems function can be extraordinarily beneficial since it teaches programmers about an area of computer science that they might otherwise ignore, and allows them to really understand the way in which computer systems function.

Works Cited

"Android Runtime (Art) and Dalvik : Android Open Source Project." *Android Open Source Project*, https://source.android.com/docs/core/runtime.

"Application Fundamentals : Android Developers." *Android Developers*, https://developer.android.com/guide/components/fundamentals.

"Cgroup Abstraction Layer : Android Open Source Project." *Android Open Source Project*, https://source.android.com/docs/core/perf/cgroups#:~:text=Android%20uses%20cgroups%20to%20control,cgroups%20v1%20and%20cgroups%20v2.

Chen, James. "Android Operating System (OS): Definition and How It Works." *Investopedia*, Investopedia, 5 Oct. 2022, https://www.investopedia.com/terms/a/android-operating-system.asp.

Corniel, Esmery. "Memory Management : Paging." *Medium*, Medium, 16 Sept. 2016, https://medium.com/@esmerycornielle/memory-management-paging-43b85abe6d2f.

Herbel, Augusto. "Collecting the Garbage: A Brief History of GC over Android Versions." *Medium*, ProAndroidDev, 29 Dec. 2022, https://proandroiddev.com/collecting-the-garbage-a-brief-history-of-gc-over-android-versions-f7f5583e433c.

Lê, Xuân Lộc. "Thread Handling Android (P1)." *Medium*, Culi Tech Viet, 6 Dec. 2019, https://medium.com/culi-tech-viet/thread-handling-android-p1-28d59d6914af.

Libretexts. “8.1: Memory Paging.” *Engineering LibreTexts*, Libretexts, 3 Apr. 2021, https://eng.libretexts.org/Courses/Delta_College/Operating_System%3A_The_Basics/08%3A_Virtual_Memory/8.1%3A_Memory_Paging.

Liu, Jimmy. “Android Thread 101 (Part II)-Lifecycle of a Thread?” *Medium*, Kuo's Funhouse, 18 Mar. 2022, https://medium.com/kuos-funhouse/android-thread-101-part-ii-lifecycle-of-a-thread-2d9bd35c06d8.

Lockwood, Alex. “Thread Scheduling in Android.” *Android Design Patterns*, 13 Jan. 2014, https://www.androiddesignpatterns.com/2014/01/thread-scheduling-in-android.html.

“Overview of Memory Management ： Android Developers.” *Android Developers*, https://developer.android.com/topic/performance/memory-overview.

Overview of Memory-Mapping - MATLAB & Simulink, https://www.mathworks.com/help/matlab/import_export/overview-of-memory-mapping.html.