Tìm hiểu hệ điều hành cho các thiết bị di động Google Android.

[**I.GIỚI THIỀU VỀ HĐH ANDROID, KIẾN TRÚC, CÁC THÀNH PHẦN 3**](#_uew1fp7eymrl)

[1. GIỚI THIỆU VỀ HỆ ĐIỀU HÀNH ANDROID 3](#_349sxe46qbxu)

[1.1. LỊCH SỬ RA ĐỜI 3](#_w0557o2603lv)

[1.2. MỤC ĐÍCH SỬ DỤNG 3](#_30j0zll)

[1.3. THỐNG KÊ SỐ LƯỢNG NGƯỜI DÙNG 4](#_236ozj416s5)

[2. KIẾN TRÚC CỦA HỆ ĐIỀU HÀNH ANDROID 5](#_nvchnz6rq8rs)

[2.1. Applications 6](#_6xtjhga21z4x)

[2.2. Application Framework 7](#_1qonqo7tfpec)

[2.3. Libraries and Android Runtime 8](#_db07q9sdk9td)

[2.4. Linux Kernel 9](#_9yl8rih5htln)

[3. CÁC THÀNH PHẦN 10](#_z6ommjggqlbf)

[3.1. HỆ THỐNG FILE 10](#_8cnz7hdxqn4l)

[3.1.1. Internal storage 11](#_7sud62wdhfnh)

[3.1.2. External storage. 13](#_2s8eyo1)

[3.1.3. Shared Preferences 14](#_lj2ha86lvvc7)

[3.1.4. Databases 15](#_3rdcrjn)

[3.2. QUẢN LÝ BỘ NHỚ 18](#_26in1rg)

[3.2.1. Thu gom rác ( Garbage Collection) 18](#_lnxbz9)

[3.2.2. Share Memory 19](#_35nkun2)

[3.2.3. Phân bổ và thu hồi bộ nhớ ứng dụng 20](#_1ksv4uv)

[3.2.4. Hạn chế bộ nhớ của ứng dụng 21](#_44sinio)

[3.2.5. Phân bổ bộ nhớ giữa các tiến trình 21](#_2jxsxqh)

[3.2.6. Các loại bộ nhớ 22](#_z337ya)

[3.2.7. Kernel swap daemon 23](#_3j2qqm3)

[3.2.8. Low-memory killer 24](#_1y810tw)

[3.3. QUẢN LÝ TIẾN TRÌNH 26](#_4i7ojhp)

[3.3.1. Hệ thống phân cấp 26](#_2xcytpi)

[3.3.2. Tính năng tự động quản lý tiến trình của Android 27](#_1ci93xb)

[3.3.3. Ứng dụng Android có thể tự nhận biết các sự kiện 28](#_3whwml4)

[3.3.4. Recent Apps 28](#_2bn6wsx)

[**II. CÁC TÍNH NĂNG NỔI BẬT CỦA HỆ ĐIỀU HÀNH ANDROID 29**](#_e4dqxn8o4esp)

[**III. MỘT SỐ HẠN CHẾ 30**](#_dch6k16q0x7a)

[**IV. CÁC VẤN ĐỀ AN NINH AN TOÀN CỦA GOOGLE ANDROID 30**](#_kbalqgt8x66r)

[1. CẬP NHẬT BẢO MẬT ĐỊNH KỲ 30](#_bhyvdkkw1zmr)

[2. ỨNG DỤNG ĐỘC HẠI VÀ PHẦN MỀM GIẢ TẠO 31](#_fil7tsnykms8)

[2.1. Ứng dụng Độc Hại 31](#_12j1ce4aig3r)

[2.2. Phần Mềm Giả Mạo 32](#_ag36s8pjdjm8)

[2.3.Biện Pháp Phòng Ngừa 32](#_46rn3bqjh35f)

[3. PHƯƠNG THỨC XÁC THỰC VÀ BẢO MẬT DẤU VÂN TAY 32](#_n47cy51yls4t)

[4. QUYỀN TRUY CẬP ỨNG DỤNG 33](#_eqe1zc8rjym1)

[5. CHẾ ĐỘ KHÓA VÀ MÃ HÓA 33](#_wc33li1h48hm)

[6. CƠ CHẾ SANDBOXING 33](#_f163t1rqbh0b)

[6.1. Nguyên Tắc Hoạt Động 33](#_1sy9d8x289ov)

[6.2. Ưu Điểm và Hạn Chế 33](#_ee2f2m5ocimz)

[6.3. Triển Khai Trong Android 34](#_vv06w0k21xru)

[6.4. Ưu Điểm Cho Người Dùng 34](#_l256lk2r0kke)

# I.**GIỚI THIỀU VỀ HĐH ANDROID, KIẾN TRÚC, CÁC THÀNH PHẦN**

## **GIỚI THIỆU VỀ HỆ ĐIỀU HÀNH ANDROID**

### **1.1. LỊCH SỬ RA ĐỜI**

* Android, Inc. được thành lập tại Palo Alto, California vào tháng 10 năm 2003 bởi Andy Rubin, Rich Miner, Nick Sears và Chris
* Bước đầu, hệ điều hành Android chỉ đơn thuần là phần mềm trên điện thoại di động. Sau đó Android là một nền tảng thiết bị di động được xây dựng trên nhân Linux.
* Từ năm 2008 đến nay, Android đã trải qua 20 lần cập nhật để dần dần cải tiến hệ điều hành. Phiên bản mới nhất hiện nay là Android 15 Vanilla Ice Cream với nội dung khá hoàn chỉnh hơn so với bản cập nhật trước.
* Android được phát triển nhằm cạnh tranh với các hệ điều hành di động khác nhưiOS(Apple),BlackBerryOS,WindowsMobile(Microsoft),Symbian(Nokia), Samsung (Bada),  WebOS  (Palm);
* Biểu tượng của hệ điều hành Android mới là một con robot màu xanh lá cây

A black background with a black square

Description automatically generated with medium confidence

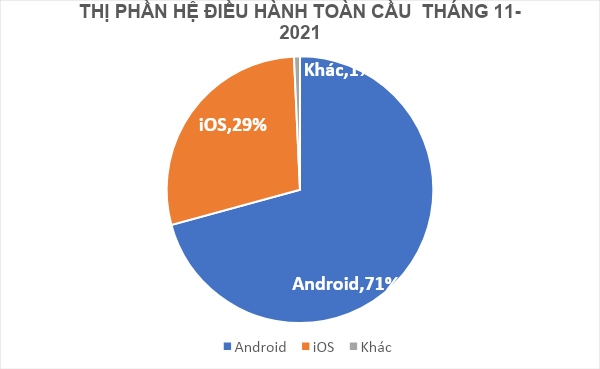
Hình 1.1 : Biểu tượng của HĐH Android

### **1.2. MỤC ĐÍCH SỬ DỤNG**

* Sử dụng làm hệ điều hành cho điện thoại di động, máy tính bảng, tivi, màn hình ô tô, máy chơi game.
* Các lập trình viên Android sử dụng các thư viện lập trình Android để viết các ứng dụng .

### **1.3**. **THỐNG KÊ SỐ LƯỢNG NGƯỜI DÙNG**

* Android chiếm 87,7% thị phần điện thoại thông minh trên toàn thế giới vào thời điểm quý 2 năm 2017, với tổng cộng 2 tỷ thiết bị đã được kích hoạt và 1,3 triệu lượt kích hoạt mỗi ngày
* Tháng 10 năm 2022, đã có khoảng 2.700.000 ứng dụng trên Android, và số lượt tải ứng dụng từ Google Play (cửa hàng ứng dụng chính của Android) năm 2021 ước tính khoảng 113.1 tỷ lượt.
* Hiện nay con số này đã giảm xuống do sự ảnh hưởng lớn của iOS từ Apple và một phần nhỏ của Windows Phone, tuy nhiên Android vẫn dẫn đầu thị phần trong thị trường điện thoại di dộng .



Hình 1.2 : Thị phần hệ điều hành điện thoại di động tháng 11-2021

## **KIẾN TRÚC CỦA HỆ ĐIỀU HÀNH ANDROID**

* Kiến trúc của hệ điều hành Android chứa nhiều thành phần để hỗ trợ bất cứ nhu cầu của thiết bị Android nào. Hệ điều hành android bao gồm các tầng:
* Applications ( Ứng dụng )
* Application FrameWork
* Libraries and Android Runtime
* Linux Kernel (Nhân Linux)

A screenshot of a computer program

Description automatically generated

Hình 2.1 : Sơ đồ các tầng của kiến trúc hệ điều hành Android

### **2.1. Applications**

* Lớp trên cùng của kiến trúc là Application (ứng dụng). Các ứng dụng ta tạo ra sẽ được cài đặt trên tầng này. Ví dụ như: Danh bạ, tin nhắn, trò chơi….
* Đặc điểm của các chương trình:
* Phần mở rộng của ứng dụng là .apk
* Khi mỗi ứng dụng được chạy, nó có một phiên bản Virtual Machine được dựng lên để phục vụ cho nó. Nó có thể là một Active Program: Chương trình có giao diện với người dùng hoặc là một background: chương trình chạy nền hay là dịch vụ.Các ứng dụng được gán số ID của người sử dụng nhằm phân quyền hạn khi sử dụng tài nguyên, cấu hình phần cứng và hệ thống
* Android là một hệ điều hành có tính mở, khác với nhiều hệ điều hành di động khác, Android cho phép một ứng dụng của bên thứ ba được phép chạy nền. Các ứng dụng chạy nền chỉ có một hạn chế nhỏ đó là nó không được phép sử dụng quá 5 ~ 10% công suất CPU, điều đó nhằm để tránh độc quyền trong việc sử dụng CPU.
* Ứng dụng không có điểm vào cố định, không có phương thức main để bắt đầu

### **2.2. Application Framework**

* Đây là tầng cung cấp cho các nhà phát triển,lập trình viên ứng dụng sử dụng các dịch vụ của Android trong ứng dụng của họ. Nó cung cấp sự truy cập phần cứng và cũng giúp quản lý giao diện người dùng với tài nguyên ứng dụng.
* Application FrameWork bao gồm một số dịch vụ chính sau:
* Activity Manager - Điều khiển tất cả khía cạnh của vòng đời và cung cấp ngăn xếp điều hướng chung.
* Content Provides – Cho phép ứng dụng truy cập hoặc chia sẻ dữ liệu với ứng dụng khác.
* Resource Manager –  Cung cấp quyền truy cập vào các tài nguyên nhúng không mã hóa như chuỗi, cài đặt màu và bố cục giao diện người dùng.
* Notifications Manager – Cho phép ứng dụng hiển thị cảnh báo và thông báo đến người dùng.
* View System - đa dạng và có thể mở rộng, ta có thể sử dụng để xây dựng giao diện người dùng (UI) của ứng dụng. Bao gồm lists, grids, text boxes, buttons,… và thậm chí cả trình duyệt web có thể nhúng (embeddable web browser)
* Các Developer có toàn quyền truy cập vào các framework API giống nhau mà các ứng dụng hệ thống Android sử dụng.

### **2.3. Libraries and Android Runtime**

* Đây là phần thứ 3 của kiến trúc và nằm ở lớp thứ 2 từ dưới lên. Phần này cung cấp 1 bộ phận quan trọng là Dalvik Virtual Machine là 1 loại máy ảo Java được thiết kế đặc biệt để tối ưu cho Android.
* Dalvik VM sử dụng các đặc trưng của nhân Linux như quản lý bộ nhớ và đa luồng, những thứ mà đã có sẵn trong Java. Dalvik VM giúp mọi ứng dụng Android chạy trong tiến trình riêng của nó, với các thể hiện (instance) riêng của Dalvik virtual Machine.
* Android Runtime cũng cung cấp 1 tập các thư viện chính giúp các nhà phát triển ứng dụng Android có thể viết ứng dụng Android bằng Java
* Đây là các thư viện dựa trên Java phục vụ cho việc phát triển Android. Ví dụ của các thư viện này bao gồm các thư viện ứng dụng dùng để xây dựng giao diện người dùng, vẽ đồ họa hay truy cập cơ sở dữ liệu. 1 số thư viện chính của Android:
* android.app - Cung cấp quyền truy cập vào ứng dụng và là nền tảng của tất cả ứng dụng Android.
* android.content - Cung cấp quyền truy cập nội dung (content), truyền tải thông điệp giữa các ứng dụng hay các thành phần của ứng dụng.
* android.database - Được sử dụng để truy cập dữ liệu của content provider và cơ sở dữ liệu SQLite
* android.opengl - giao diện các phương thức Java để sử dụng OpenGL ES
* android.os - Cung cấp các ứng dụng với quyền truy cập vào các dịch vụ của hệ điều hành bao gồm thông điệp, các dịch vụ hệ thống và các giao tiếp nội tại
* android.text - Được sử dụng để hiển thị và điều chỉnh chữ trên màn hình thiết bị
* android.view - Các thành phần cơ bản trong việc xây dựng giao diện người dùng của ứng dụng.

android.widget - Tập các thành phần giao diện người dùng đã được xây dựng sẵn như các nút, các nhãn (label), list view,....

* android.webkit - Tập các lớp cho phép xây dựng khả năng duyệt web.
* Ở trên lớp nhân Linux là tập các thư viện bao gồm WebKit - trình duyệt Web mã nguồn mở, được biết đến như thư viện libc, cơ sở dữ liệu SQLite hữu dụng cho việc lưu trữ và chia sẻ dữ liệu ứng dụng, các thư viện chơi và ghi âm audio, video, hay các thư viện SSL chịu trách nhiệm bảo mật Internet...

### **2.4. Linux Kernel**

* Android được phát triển từ nhân Linux nên có thể tận dụng các tính năng bảo mật chính và cho phép các nhà sản xuất thiết bị phát triển trình điều khiển phần cứng.
* Tất cả hoạt động của điện thoại muốn thi hành được thì đều được thực hiện ở mức cấp thấp ở tầng này bao gồm quản lý bộ nhớ (memory management), giao tiếp với phần cứng (driver model), thực hiện bảo mật (security), quản lý tiến trình (tiến trình).
* Tuy được phát triển dựa vào nhân Linux nhưng thực ra nhân Linux đã được nâng cấp và sửa đổi rất nhiều để phù hợp với tính chất của những thiết bị điện thoại di động.
* Tầng này có các phần chủ yếu:
* **Display Driver:** Điều khiển việc hiển thị lên màn hình cũng như thu nhận nhứng điều khiển của người dùng lên màn hình (di chuyển, cảm ứng…)
* **Camera Driver:** Điều khiển hoạt động của camera, nhận luồng dữ liệu từ camera trả về
* **Bluetooth Driver:** Điều khiển thiết bị phát và thu sóng Bluetooth
* **USB Driver:** Quản lý hoạt động của các cổng giao tiếp USB
* **Keypad Driver:** Điều khiển bàn phím
* **Wifi Driver:** Chịu trách nhiệm về việc thu phát sóng wifi
* **Audio Driver:** Điều khiển các bộ thu phát âm thanh, giải mã các tính hiệu dạng audio thành tín hiệu và ngược lại
* **Binder IPC Driver:** Chịu trách nhiệm về việc kết nối và liên lạc với mạng vô tuyến như CDMA, GSM, 3G, 4G, E để đảm bảo những chức năng truyền thông được thực hiện
* **M-System Driver:** Quản lý việc đọc ghi lên các thiết bị nhớ như thẻ SD, Flash
* **Power Management:** Giám sát việc tiêu thụ điện năng

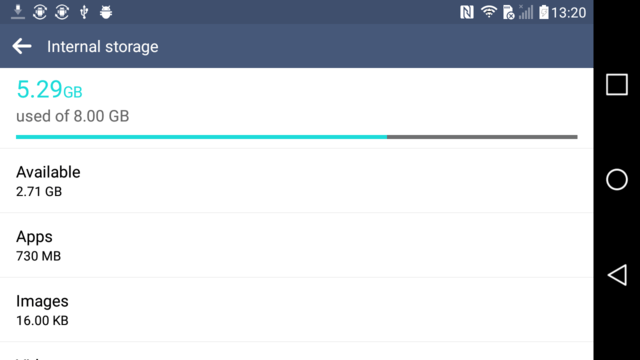
## **CÁC THÀNH PHẦN**

### **3.1. HỆ THỐNG FILE**

* Những cách lưu trữ dữ liệu có sẵn mà Android cung cấp:
* *Internal file storage*: Lưu trữ các file riêng tư của ứng dụng trong hệ thống file của thiết bị.
* *External file storage*: Lưu trữ file trên hệ thống file chia sẻ ra được bên ngoài. Thường dùng cho các file mà người dùng được chia sẽ, chẳng hạn như hình ảnh, video.
* *Shared Preferences*: Lưu trữ dữ liệu nguyên thủy bởi các cặp key-value.
* *Database*: Lưu trữ các dữ liệu có cấu trúc trong cơ sở dữ liệu riêng tư.

#### **3.1.1. Internal storage**

* Theo mặc định thì các files được lưu vào Internal Storage (bộ nhớ trong) sẽ là riêng tư đối với ứng dụng của ta, và các ứng dụng khác sẽ không kết nối được (người dùng cũng không được trừ khi chúng có quyền truy cập root). Điều này làm cho Internal Storage là một nơi lưu trữ dữ liệu tốt mà người dùng không cần truy cập trực tiếp. Hệ thống cung cấp một thư mục riêng tư trong hệ thống file cho mỗi ứng dụng, nơi ta có thể sắp xếp bất kì tệp nào mà ứng dụng của ta cần.



Hình 3.1: Internal storage

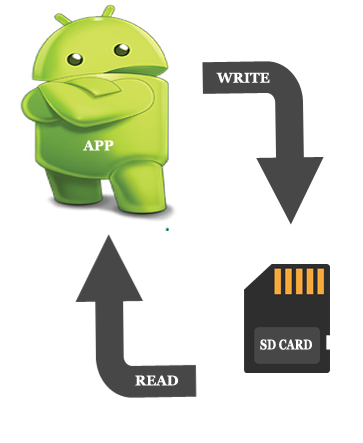
* Khi người dùng gỡ cài đặt ứng dụng của ta, các file được lưu trữ trong Internal Storage được xóa bỏ. Bởi do hành vi này mà ta không nên dùng Internal Storage để lưu trữ dữ liệu mà người dùng dự kiến muốn sử dụng độc lập với ứng dụng của ta. Ví dụ ứng dụng của ta cho phép chụp ảnh,người dùng muốn sử dụng những bức ảnh đó sau khi xóa ứng dụng. Vì vậy ta nên lưu các loại file đó ở External Storage.

*Internal cache file:*

* Nếu ta muốn lưu trữ một số dữ liệu tạm thời, thay vì duy trì lưu dữ liệu liên tục, ta nên sử dụng cache directory để lưu trữ dữ liệu. Mỗi ứng dụng có một bộ nhớ cache riêng dành riêng cho các loại file này. Khi ứng dụng đang hoạt động nhưng hệ thống thiếu bộ nhớ, Android có thể xóa các file trong bộ nhớ cache này để khôi phục dung lượng. Tuy nhiên ta không nên dựa vào hệ thống để dọn dẹp các tệp này cho ta. Ta nên luôn tự duy trì các file trong bộ nhớ cache ở trong giới hạn không gian hợp lý được tiêu thụ, chẳng hạn như 1MB. Khi người dùng gỡ các cài đặt của ứng dụng, các file này cũng bị xóa.

#### **3.1.2. External storage.**

* Mọi thiết bị Android đều hỗ trợ không gian lưu trữ External Storage (bộ nhớ ngoài) mà ta có thể sử dụng để lưu file. Vùng nhớ này được gọi là External (ngoài) là bởi vì nói không phải chắc chắn là một vùng nhớ có thể truy cập được mà nó là một không gian lưu trữ. Là vùng nhớ mà người dùng có thể kết nối với máy tính, thậm chí có thể tháo lắp vật lý, ví dụ như các thẻ nhớ SD. Các file trên External Storage có thể được đọc ở mọi nơi và có thể được người dùng sửa đổi khi chúng cho phép như là một USB để có thể chuyển đổi dữ liệu sang máy tính.

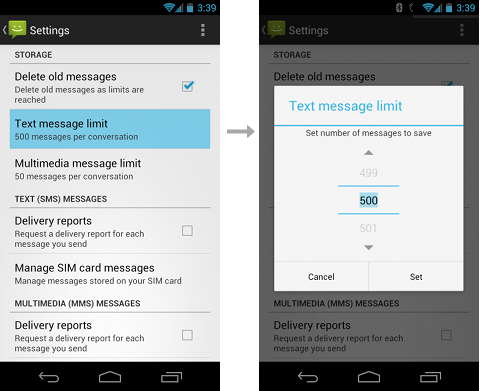


Hình 2.II.2 : Minh họa bộ nhớ ngoài ( SD card )

* Vì vậy trước khi ta cố gắng truy cập truy cập một file trong External Storage trong ứng dụng của ta, ta nên kiểm tra tính khả dụng của các thư mục này cũng như các file trong chúng
* Thông thường, ta nên sử dụng bộ nhớ ngoài cho dữ liệu mà có thể truy cập vào các ứng dụng khác và vẫn còn lưu khi ứng dụng của ta bị người dùng gỡ cài đặt. Chẳng hạn như những bức ảnh được chụp hoặc những file đã được download xuống trước đó. Hệ thống sẽ cung cấp các thư mục public chuẩn cho các loại file này, do đó người dùng có các vị trí cho photos, ringtones, music, ...
* Ta cũng có thể lưu file vào External Storage trong thư mục dành riêng cho ứng dụng của ta mà hệ thống sẽ xóa khi người dùng gỡ cài đặt ứng dụng này trên điện thoại. Đây có thể là giải pháp thay thế cho bộ nhớ trong nếu cần thêm dung lượng, nhưng các file ở đây không đảm bảo có thể truy cập được bởi vì người dùng có thể xóa, hay tháo thẻ nhớ SD. Và các file này có thể đọc được mọi nơi, chúng chỉ sao lưu vào một nơi mà không được chia sẻ với các ứng dụng khác.

#### **3.1.3. *Shared Preferences***

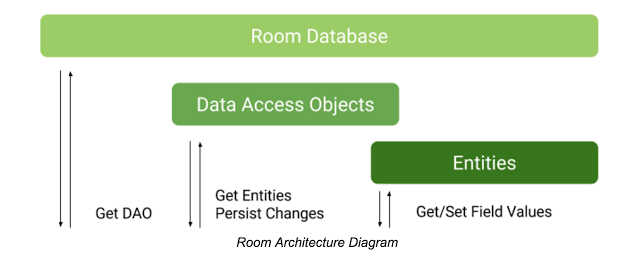
* + Nếu ta không lưu nhiều dữ liệu và dữ liệu đó không yêu cầu cấu trúc, ta nên sử dụng Shared Preferences. Loại lưu trữ này cho phép ta đọc và ghi các cặp key-value của các kiểu dữ liệu nguyên thủy: booleans, floats, ints, longs và strings
  + Các cặp key-value này được ghi vào các tệp XML, luôn tồn tại khi ứng dụng được sử dụng và kể các khi ứng dụng bị buộc dừng. Thường được dùng bởi các lập trình viên để lưu các trạng thái đơn giản của App như cho phép thông báo (bool),điểm cao của game (int),…
  + Tên Shared Preferences đôi khi gây ra một chút hiểu lầm bởi vì nó không phải hoàn toàn để lưu những tùy chọn của người dùng, chẳng hạn như nhạc chuông mà người dùng đã chọn. Ta chỉ có thể dùng Shared Preferences để lưu bất kì loại dữ liệu đơn giản nào, chẳng hạn như điểm số cao của người dùng.



Hình 3.2: Ví dụ cho Shared Preferences với cặp key-value tương ứng là {Text message limit,500}

#### **3.1.4. Databases**

* Android cung cấp hỗ trợ đầy đủ cho SQLite Database. Bất kì cơ sở dữ liệu mà ta tạo chỉ có thể truy cập được bởi ứng dụng của ta. Tuy nhiên thay vì sử dụng các SQLite API trực tiếp, thì hiện tại Android khuyến khích các Developers nên tạo và tương tác với cơ sở dữ liệu của mình với *Room Database*
* Có 3 thành phần chính trong Room:
* Lớp cơ sở dữ liệu lưu giữ cơ sở dữ liệu và đóng vai trò là điểm truy cập chính cho đường kết nối cơ bản đến dữ liệu cố định của ứng dụng.
* Thực thể dữ liệu biểu thị các bảng trong cơ sở dữ liệu của ứng dụng.
* Đối tượng truy cập dữ liệu (DAO-data access objects) cung cấp các phương thức mà ứng dụng của bạn có thể dùng để truy vấn, cập nhật, chèn và xoá dữ liệu trong cơ sở dữ liệu.
* Lớp cơ sở dữ liệu cung cấp cho ứng dụng của bạn các thực thể của DAO được liên kết với cơ sở dữ liệu đó. Đổi lại, ứng dụng có thể dùng DAO để truy xuất dữ liệu từ cơ sở dữ liệu dưới dạng thực thể của đối tượng thực thể dữ liệu được liên kết. Ứng dụng cũng có thể dùng các thực thể dữ liệu đã xác định để cập nhật các hàng trong bảng tương ứng hoặc tạo hàng mới để chèn dữ liệu. Hình 1 minh hoạ mối quan hệ giữa nhiều thành phần của Room.



Hình 3.3: Mô hình minh họa cấu trúc RoomDatabase

* Room cung cấp một lớp trừu tượng ánh xạ các đối tượng cho phép truy cập cơ sở dữ liệu nhanh chóng, tốc độ giúp phát huy toàn bộ sức mạnh của SQLite Database.
* Mặc dù ta có thể sử dụng với SQLite Database nhưng hiện thì đã xem như tốn rất nhiều thời gian và công sức để sử dụng :
* Các truy vấn cứng SQL thì không được Compiler time checking.
* Khi cấu trúc cơ sở dữ liệu thay đổi thì ta phải update một cách thủ công. Quá trình này gây tốn thời gian vã dễ bị lỗi.
* Ta phải viết nhiều mã để chuyển đổi giữa các truy vấn SQL và các đối tượng dữ liệu java.
* Thư viện Room sẽ giải quyết hết những vấn đề này bằng cách cung cấp các lớp trừu tượng trong SQLite.

### **3.2. QUẢN LÝ BỘ NHỚ**

* Máy ảo Android Runtime (ART) và Dalvik sử dụng các tùy chọn phân trang và ánh xạ bộ nhớ (mapping) để quản lý bộ nhớ. Điều này có nghĩa là mọi bộ nhớ mà một ứng dụng sửa đổi (cho dù bằng cách phân bổ các đối tượng mới hay chạm vào các trang được liên kết) sẽ vẫn nằm trong RAM và không thể phân trang. Cách duy nhất để giải phóng bộ nhớ khỏi một ứng dụng là giải phóng các tệp tham chiếu đối tượng mà ứng dụng lưu giữ, cung cấp bộ nhớ này cho trình thu gom rác. Có một trường hợp ngoại lệ là bất kỳ tệp nào được ánh xạ mà không cần chỉnh sửa, chẳng hạn như mã, đều có thể bị phân trang ra khỏi RAM nếu hệ thống muốn dùng bộ nhớ đó ở nơi khác.

#### **3.2.1. Thu gom rác ( Garbage Collection)**

* Một môi trường bộ nhớ được quản lý, như máy ảo ART hoặc Dalvik, sẽ tiếp tục theo dõi từng lượt phân bổ bộ nhớ. Sau khi xác định một phần bộ nhớ không còn được chương trình sử dụng, chương trình sẽ giải phóng phần bộ nhớ không dùng đó mà không cần sự can thiệp của lập trình viên. Cơ chế thu hồi bộ nhớ không sử dụng trong môi trường bộ nhớ được quản lý được gọi là *thu gom rác*. Việc thu gom rác có hai mục tiêu: tìm các đối tượng dữ liệu trong một chương trình không thể truy cập trong tương lai; và thu hồi tài nguyên mà các đối tượng đó sử dụng.
* Bộ nhớ khối của Android là một bộ nhớ mang phân vùng, nghĩa là có nhiều nhóm phân bổ mà ứng dụng theo dõi, dựa trên tuổi thọ và kích thước của một đối tượng được phân bổ. Ví dụ: các đối tượng được phân bổ gần đây thuộc về *Khu vực nhớ tạm thời*. (vùng nhớ chứa các đối tượng mới được khởi tạo). Nếu đối tượng đó vẫn tiếp tục được lưu và sử dụng thì hệ thống sẽ di chuyển xuống khu vực nhớ dài hạn, sau đó là khu vực nhớ vĩnh viễn.
* Mỗi khu vực của vùng nhớ khối có giới hạn trên riêng về dung lượng bộ nhớ mà các đối tượng ở đó có thể chiếm. Bất cứ khi nào một khu vực nhớ bắt đầu đầy, hệ thống sẽ thực thi một sự kiện thu gom rác để giải phóng bộ nhớ. Thời gian thu gom rác tùy thuộc vào khu vực đối tượng mà nó đang thu thập và số lượng đối tượng đang hoạt động trong mỗi khu vực.
* Mặc dù việc thu gom rác có thể khá nhanh, nhưng vẫn có thể ảnh hưởng đến hiệu suất của ứng dụng. Ta thường không kiểm soát được thời điểm xảy ra sự kiện thu gom rác bên trong mã của mình. Hệ thống có một bộ tiêu chí đang chạy để xác định thời điểm tiến hành thu gom rác. Khi các tiêu chí được đáp ứng, hệ thống sẽ ngừng thực thi quy trình này và bắt đầu thu gom rác. Nếu việc thu gom rác xảy ra ở giữa vòng lặp xử lý chuyên sâu như ảnh động hoặc trong khi phát nhạc, thì thời gian xử lý có thể kéo dài ra.
* Ngoài ra, mã (code) của ta có thể thực hiện những loại tác vụ buộc các sự kiện thu gom rác xảy ra thường xuyên hơn hoặc kéo dài lâu hơn bình thường. Chẳng hạn như nếu ta phân bổ nhiều đối tượng ở phần trong cùng của một vòng lặp update của chương trình, ta có thể làm ô nhiễm vùng nhớ khối vì có quá nhiều đối tượng được tạo ra liên tục. Trong trường hợp đó, trình thu gom rác phải thực hiện nhiều sự kiện thu gom rác và có thể làm giảm hiệu suất của ứng dụng.

#### **3.2.2. Share Memory**

* Để đáp ứng được mọi yêu cầu về RAM, Android sẽ cố gắng chia sẻ các trang RAM giữa các tiến trình. Android có thể làm như vậy theo các cách sau:
* Mỗi tiến trình của ứng dụng đều được tách ra từ một tiến trình hiện có được gọi là Zygote. Quá trình Zygote bắt đầu khi hệ thống khởi động và tải mã khung cũng như các tài nguyên thông thường (chẳng hạn như các chủ đề hoạt động). Để bắt đầu một tiến trình mới đối với ứng dụng, hệ thống sẽ phân tách phát triển nhánh tiến trình Zygote, sau đó tải và chạy mã của ứng dụng trong tiến trình mới. Phương pháp này cho phép hầu hết các trang RAM được phân bổ cho mã khung và tài nguyên dùng chung trên tất cả các tiến trình của ứng dụng.
* Hầu hết dữ liệu tĩnh được liên kết vào một tiến trình. Kỹ thuật này cho phép chia sẻ dữ liệu giữa các tiến trình, đồng thời cho phép phân phát dữ liệu khi cần thiết. Ví dụ về dữ liệu tĩnh bao gồm: Mã Dalvik (bằng cách đặt mã đó vào tệp .odex được liên kết trước để ánh xạ trực tiếp), tài nguyên ứng dụng (bằng cách thiết kế bảng tài nguyên thành một cấu trúc có thể được ánh xạ và căn chỉnh các mục nhập zip của APK) và các phần tử dự án truyền thống như mã gốc trong tệp .so.
* Ở nhiều nơi, Android chia sẻ cùng một RAM động trên các tiến trình bằng cách sử dụng vùng bộ nhớ dùng chung được phân bổ rõ ràng. Ví dụ như các nền tảng cửa sổ sử dụng bộ nhớ dùng chung giữa trình kết hợp dữ liệu màn hình và ứng dụng, còn bộ đệm con trỏ sử dụng bộ nhớ dùng chung giữa nhà cung cấp nội dung và ứng dụng.
* Do việc sử dụng rộng rãi bộ nhớ dùng chung, nên việc xác định dung lượng bộ nhớ mà ứng dụng của ta đang sử dụng cần phải được xử lý cẩn thận.

#### **3.2.3. Phân bổ và thu hồi bộ nhớ ứng dụng**

* Vùng nhớ khối Dalvik bị hạn chế trong một phạm vi bộ nhớ ảo duy nhất cho mỗi tiến trình ứng dụng. Điều này xác định kích thước vùng nhớ khối hợp lý, kích thước này có thể tăng lên khi cần, nhưng chỉ đến một giới hạn mà hệ thống xác định cho từng ứng dụng.
* Kích thước hợp lý của vùng nhớ khối không giống với dung lượng bộ nhớ thực mà vùng nhớ khối sử dụng. Khi kiểm tra vùng nhớ khối của ứng dụng, Android sẽ tính toán một giá trị được gọi là Kích thước cài đặt theo tỷ lệ (PSS). Giá trị này tính cho cả trang bẩn và trang sạch đã chia sẻ với các tiến trình khác – nhưng chỉ trong một số lượng tương ứng với số lượng ứng dụng chia sẻ RAM đó. Tổng số (PSS) này là giá trị mà hệ thống coi là dấu vết bộ nhớ thực.
* Bộ nhớ khối Dalvik không thu gọn kích thước hợp lý của vùng nhớ khối, nghĩa là Android không phân đoạn vùng nhớ khối để đóng dung lượng. Android chỉ có thể thu nhỏ kích thước vùng nhớ khối hợp lý khi có dung lượng chưa được sử dụng ở cuối vùng nhớ khối. Tuy nhiên, hệ thống vẫn có thể giảm bộ nhớ thực mà vùng nhớ khối sử dụng. Sau khi thu gom rác, Dalvik sẽ di chuyển qua vùng nhớ khối và tìm các trang không sử dụng, sau đó sử dụng madvise để trả các trang đó về kernel. Vì vậy, việc phân bổ theo cặp và phân bổ từng phần lớn sẽ dẫn đến tình trạng thu hồi tất cả (hoặc gần như tất cả) bộ nhớ thực đã dùng. Tuy nhiên, việc thu hồi bộ nhớ từ các tiến trình phân bổ nhỏ có thể kém hiệu quả hơn nhiều, vì trang được dùng để phân bổ nhỏ vẫn có thể được chia sẻ với một nội dung khác chưa được giải phóng.

#### **3.2.4. Hạn chế bộ nhớ của ứng dụng**

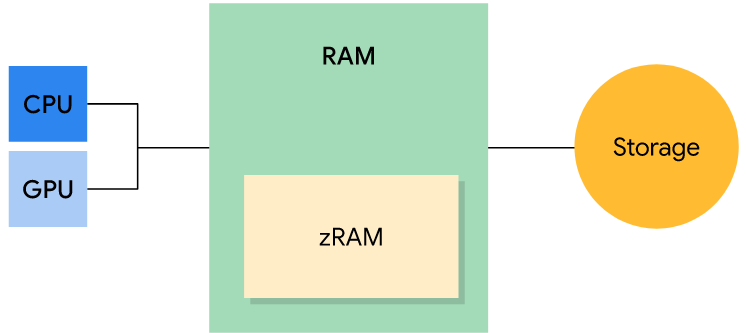
* Để duy trì một môi trường hoạt động đa nhiệm các chức năng, Android đã đặt giới hạn cố định về kích thước vùng nhớ khối cho từng ứng dụng. Giới hạn kích thước vùng nhớ khối chính xác sẽ khác nhau giữa các thiết bị dựa trên dung lượng RAM mà thiết bị có sẵn về tổng thể. Nếu một ứng dụng đã đạt đến mức giới hạn vùng nhớ khối và đang cố gắng phân bổ thêm bộ nhớ, ứng dụng có thể sẽ bị OutOfMemoryError (hết bộ nhớ) .

#### **3.2.5. Phân bổ bộ nhớ giữa các tiến trình**

* Android platform chạy trên tiền đề rằng "Bộ nhớ trống là bộ nhớ bị lãng phí" (free memory is wasted memory). Nó cố gắng sử dụng toàn bộ memory có sẵn mọi lúc. Ví dụ: hệ thống sẽ giữ các ứng dụng trong bộ nhớ sau khi chúng bị đóng để người dùng có thể nhanh chóng quay lại. Vì lý do này, các thiết bị Android thường chạy với rất ít bộ nhớ trống. Quản lý bộ nhớ là rất quan trọng để phân bổ bộ nhớ hợp lý giữa các tiến trình hệ thống (system tiến trình) quan trọng và các ứng dụng người dùng.

#### **3.2.6. Các loại bộ nhớ**

* Thiết bị Android chứa ba loại bộ nhớ khác nhau: RAM, zRAM và storage. Lưu ý rằng cả CPU và GPU đều truy cập cùng RAM.



Hình 3.4: Sơ đồ bộ nhớ Android

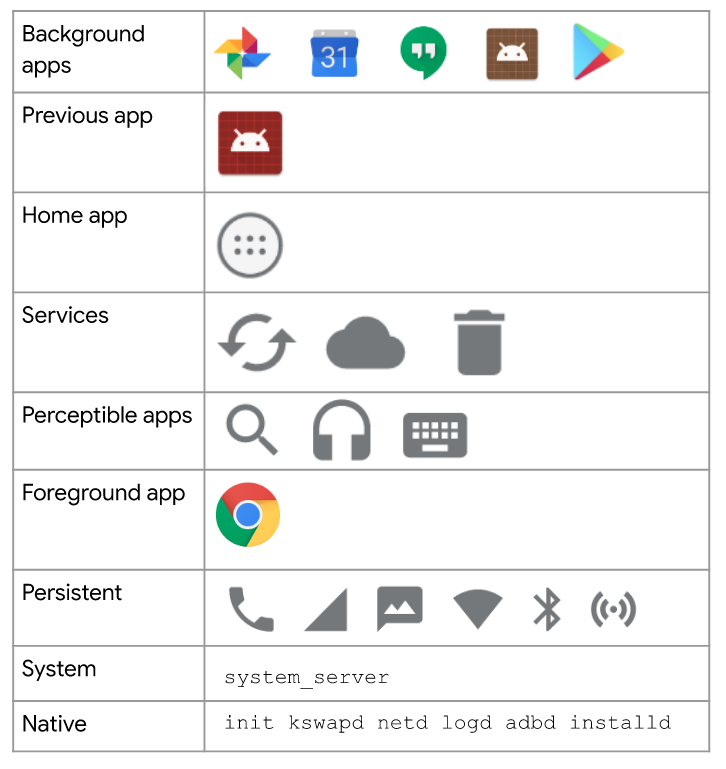
1. RAM là loại bộ nhớ nhanh nhất, nhưng thường bị giới hạn về kích thước. Các thiết bị cao cấp thường có dung lượng RAM lớn.
2. zRAM là một phân vùng RAM được sử dụng cho không gian trao đổi (swap space). Mọi thứ được nén lại khi được đặt vào zRAM, và sau đó được giải nén khi sao chép ra khỏi zRAM. Phần RAM này tăng hoặc giảm kích thước khi các trang bộ nhớ được chuyển vào hoặc lấy ra khỏi zRAM. Các hãng sản xuất thiết bị có thể đặt kích thước tối đa cho zRAM.
3. Storage chứa tất cả các dữ liệu ổn định (persistent data) như file hệ thống và tất cả các ứng dụng, thư viện và platform. Storage có dung lượng lớn hơn nhiều so với hai loại bộ nhớ còn lại. Trên Android, Storage không được sử dụng để swap space như trên các triển khai Linux khác vì việc writing thường xuyên có thể gây hao mòn bộ nhớ này và rút ngắn tuổi thọ nó.

#### **3.2.7. Kernel swap daemon**

* Kernel swap daemon (kswapd) là một phần của Linux kernel , chuyển đổi bộ nhớ đã sử dụng thành bộ nhớ trống. Daemon sẽ active khi bộ nhớ trống trên thiết bị sắp hết. Linux kernel duy trì một ngưỡng bộ nhớ trống mức thấp và cao. Khi bộ nhớ trống giảm xuống dưới ngưỡng thấp, kswapd bắt đầu lấy lại bộ nhớ. Khi bộ nhớ trống đạt đến ngưỡng cao, kswapd sẽ ngừng lấy lại bộ nhớ.
* kswapd có thể lấy lại các clean page bằng cách xóa chúng vì chúng được backup bởi storage và chưa được sửa đổi. Nếu một tiến trình cố gắng address một clean page đã bị xóa, hệ thống sẽ sao chép page từ storage sang RAM. Hoạt động này được gọi là phân trang nhu cầu (demand paging).
* kswapd có thể di chuyển các riêng tư dirty page được lưu trong bộ nhớ cache và các anonymous dirty page sang zRAM, nơi chúng được nén. Làm như vậy sẽ giải phóng bộ nhớ khả dụng trong RAM (free page). Nếu một tiến trình cố gắng chạm vào một dirty page trong zRAM, page đó sẽ không bị nén và được chuyển ngược trở lại vào RAM. Nếu tiến trình liên kết với một page nén bị kill, thì page đó sẽ bị xóa khỏi zRAM.
* Nếu bộ nhớ trống giảm xuống dưới một ngưỡng nhất định, hệ thống sẽ bắt đầu hủy các tiến trình.

#### **3.2.8. Low-memory killer**

* Nhiều lần, kswapd không thể giải phóng đủ bộ nhớ cho hệ thống. Trong trường hợp này, hệ thống sử dụng onTrimMemory() để thông báo cho ứng dụng rằng bộ nhớ sắp hết và nó sẽ giảm việc phân bổ. Nếu điều này là không đủ, kernel bắt đầu hủy các tiến trình để giải phóng bộ nhớ. Nó sử dụng low-memory killer (LMK) để làm điều này.
* Để quyết định tiến trình nào bị hủy, LMK sử dụng điểm "hết bộ nhớ" ("out of memory" score) được gọi là oom\_adj\_score để ưu tiên các tiến trình đang chạy. Các tiến trình với số điểm cao bị hủy đầu tiên. Các ứng dụng nền bị hủy trước tiên và các tiến trình hệ thống sẽ bị hủy cuối cùng. Bảng sau liệt kê các loại điểm LMK từ cao đến thấp. Các item trong mục có điểm cao nhất, ở hàng một, sẽ bị hủy trước:



Hình 3.5: Bảng tính điểm LMK ( từ cao đến thấp)

* Background apps: Ứng dụng đã chạy trước đây và hiện không hoạt động. LMK trước tiên sẽ hủy các ứng dụng nền có oom\_adj\_score cao nhất.
* Previous app: Ứng dụng nền được sử dụng gần đây nhất. Ứng dụng trước có mức độ ưu tiên cao hơn (điểm thấp hơn) so với ứng dụng chạy nền vì nhiều khả năng người dùng sẽ chuyển sang ứng dụng này hơn là một trong những ứng dụng chạy nền.
* Home app: Đây là ứng dụng launcher. Hủy cái này sẽ làm cho hình nền biến mất.
* Services: Service được start bởi các ứng dụng và có thể bao gồm syncing hoặc uploading lên cloud.
* Perceptible app: Các ứng dụng non-foreground mà người dùng có thể cảm nhận được bằng một cách nào đó, chẳng hạn như tiến trình tìm kiếm hiển thị một UI nhỏ hoặc tiến trình nghe nhạc.
* Foreground app: Ứng dụng hiện đang được sử dụng. Hủy ứng dụng nền trước trông giống như một crash ứng dụng có thể cho người dùng biết rằng có sự cố xảy ra với thiết bị.
* Persistent (services): Đây là những core service cho thiết bị, như telephony và wifi.
* System: Các tiến trình của hệ thống. Khi các tiến trình này bị hủy, điện thoại có thể xuất hiện hiện tượng khởi động lại.
* Native: Các low-level tiến trình được sử dụng bởi hệ thống (ví dụ: kswapd).
* Các hãng sản xuất thiết bị có thể thay đổi hành vi của LMK.

### **3.3. QUẢN LÝ TIẾN TRÌNH**

#### **3.3.1. Hệ thống phân cấp**

* Thứ bậc của tiến trình trong Android được chia thành 5 cấp theo 5 mức độ quan trọng từ cao đến thấp:
* Foreground process (tạm dịch: tiến trình nổi bật): Đây chính là những tiến trình dùng để chạy ứng dụng (app) bạn đang dùng. Những tiến trình khác cũng có thể được xem là foreground khi chúng có liên hệ trức tiếp với tiến trình “xử lý” app đang chạy. Tại một thời điểm cũng chỉ có vài foreground process .
* Visible process (tạm dịch: tiến trình nhìn thấy được): Visible process không liên quan đến app đang chạy nhưng có tác động đến những gì thể hiện trên màn hình. Ví dụ, foreground process có tính năng “trong suốt” (transparent) và những ứng dụng được hiển thị đằng sau chính là visible process. Phổ biến nhất đó chính là bong bóng chat của Facebook Messenger,bộ lọc ánh sáng xanh màn hình.
* Service process (tạm dịch: tiến trình dịch vụ): Tiến trình dạng này không liên quan đến bất kì ứng dụng cả đang chạy và “dưới” đang chạy nào. Chúng thực hiện công việc một cách âm thầm như chơi nhạc hay tải tập tin. Ví dụ bạn đang nghe nhạc và muốn chuyển sang chơi game, khi bạn mở game cũng là lúc tiến trình phát nhạc trở thành service process, vẫn tiếp tục chơi nhạc khi bạn làm việc khác.
* Background process (tạm dịch: tiến trình nền): Background process không xuất hiện, cũng không thực hiện vai trò dễ nhận ra (như chơi nhạc), chúng không ảnh hưởng đến trải nghiệm sử dụng. Tại một thời điểm, có rất nhiều background process đang chạy và bạn có thể xem chúng là những ứng dụng đang “tạm dừng”. Background process vẫn sử dụng RAM, cho phép bạn nhanh chóng chuyển đổi trở lại nhưng không sử dụng thêm tài nguyên phần cứng nào khác. Ví dụ khi dùng Chrome và bấm phím Home, Chrome trở thành background process và khi mở lại Chrome, nó cũng lập tức mở lại tab đang xem.
* Empty process (tạm dịch: tiến trình rỗng): Tiến trình này không còn tiêu tốn tài nguyên nào nữa. Chúng được giữ lại nhằm mục đích tạo bộ nhớ đệm cho lần khởi động sau và hệ thống có thể tùy ý loại bỏ chúng.

#### **3.3.2. Tính năng tự động quản lý tiến trình của Android**

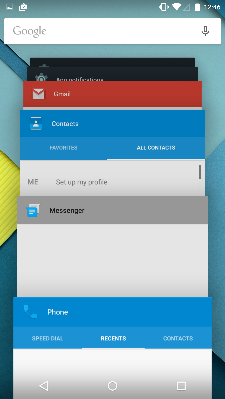
* Android có khả năng quản lý tiến trình một cách tự động, do đó bạn không cần cài đặt bất kì ứng dụng “quản lý ứng dụng” nào.
* Khi cần nhiều tài nguyên, Android tự động xóa các tiến trình ít quan trọng nhất, bắt đầu từ empty và background process. Khi cần nhiều tài nguyên hơn nữa như lúc chơi game nặng, hệ thống tự động loại bỏ thêm service process. Chơi game nặng, nhạc của bạn sẽ tự tắt, tập tin tải về tự động dừng.
* Trong đa số trường hợp, bạn không cần lo về lượng RAM của máy. Nhiều người lo rằng máy mình chỉ còn 1 ít RAM và “đổ thừa” cho hệ thống là không đúng. Cơ chế quản lý thông minh của Android tự động lưu giữ các ứng dụng và dữ liệu khác trên RAM cho bạn trải nghiệm tốt nhất có thể.
* Dĩ nhiên, sự linh hoạt cũng có điểm yếu. Nếu một ứng dụng được lập trình không tốt, một service process có thể tiếp tục sử dụng tài nguyên CPU và RAM khiến máy nóng lên và tụt pin nhanh chóng. Về mặt kĩ thuật, tình trạng này được gọi là “memory leaked” (rò rỉ bộ nhớ).

#### **3.3.3. Ứng dụng Android có thể tự nhận biết các sự kiện**

* Các lập trình viên có thể cho ứng dụng của mình quyền tự khởi động cùng hệ thống hay chạy một dịch vụ ngầm. Ứng dụng có thể tự khởi động bởi nhiều loại “sự kiện” khác nhau: khi mở máy, khi chụp ảnh, khi thay đổi mạng. Khả năng này cho phép ứng dụng làm việc mà không cần tiêu tốn tài nguyên “chạy nền” (background process). Ví dụ Facebook Messenger biết khi nào bạn mở wifi để tự nhận tin nhắn.

#### **3.3.4. Recent Apps**

* Là menu hiển thị các ứng dụng đang ở trạng thái “background process”. Dùng tay “quét bỏ” sẽ loại bỏ hoàn toàn ứng dụng khỏi bộ nhớ RAM của máy. Cách này thực sự cũng không quá cần thiết nhưng cũng có thể giải quyết các trường hợp ứng dụng gặp lỗi (đứng hình, vẫn chạy nhưng mất cảm ứng…).
* Kích hoạt menu này bằng cách nhấn và giữ Home, nhấn liên tiếp Home 2 lần, chọn biểu tượng chỉ định (thường là 2 hình chữ nhật xếp chồng)… tùy vào loại máy.
* Hoặc cũng có thể vào mục Settings > Apps > chọn ứng dụng và bấm Force stop để tắt chúng đi.



Hình 2.IV.3: Recent Apps của Android

# **II. CÁC TÍNH NĂNG NỔI BẬT CỦA HỆ ĐIỀU HÀNH ANDROID**

* Giao diện người dùng mặc định của Android sử dụng các thao tác trực tiếp như chạm, vuốt và kéo để bắt đầu giao tiếp. Thiết bị cung cấp phản hồi xúc giác cho người dùng thông qua các cảnh báo như rung để phản hồi các hành động.
* Khi người dùng khởi động thiết bị, hệ điều hành Android sẽ hiển thị màn hình chính, đây là trung tâm điều hướng chính cho thiết bị Android, bao gồm các tiện ích và biểu tượng ứng dụng. Widget là màn hình thông tin tự động cập nhật nội dung như thời tiết hoặc tin tức. Màn hình chính có thể khác nhau tùy theo nhà sản xuất của thiết bị. Người dùng cũng có thể chọn các chủ đề khác nhau cho màn hình chính thông qua các ứng dụng của bên thứ ba trên Google Play .
* Thanh trạng thái ở phía trên cùng màn hình chính hiển thị các thông tin và trạng thái kết nối của thiết bị: mạng Wi-Fi đang kết nối, sóng 3G, cường độ sóng điện thoại. Người dùng có thể vuốt thanh trạng thái theo chiều đi xuống để xem màn hình thông báo
* Hệ điều hành Android cũng sở hữu các tính năng giúp tiết kiệm pin. Hệ điều hành tạm dừng các ứng dụng không được sử dụng để tiết kiệm pin và mức sử dụng CPU. Android có các tính năng quản lý bộ nhớ giúp tự động đóng các trình xử lý không hoạt động lưu trong bộ nhớ

# **III. MỘT SỐ HẠN CHẾ**

* Mặc dù Android cung cấp cho người dùng thêm một giải pháp thay thế tiện ích so với các hệ điều hành di động khác, nhưng HĐH vẫn còn một số hạn chế. Đối với các nhà phát triển, việc viết mã các giao diện và trải nghiệm người dùng thường phức tạp và khó khăn, phải phụ thuộc nhiều hơn vào Java so với Objective-C.
* Vì là hệ điều hành mở nên Android OS có tính bảo mật thấp và khiến người dùng dễ bị lộ thông tin cá nhân
* Hệ điều hành Android dễ bị phân mảnh. Bản chất mã nguồn mở linh hoạt của Android dẫn đến nhiều biến thể của phần cứng và phần mềm. Sự phân mảnh thiết bị mang đến thách thức cho các nhà phát triển vì rất khó để phát triển các ứng dụng hoạt động trên tất cả các loại hình và phiên bản thiết bị. Phân mảnh cũng là một vấn đề của các doanh nghiệp: Nhân viên CNTT không thể dễ dàng bảo mật và quản lý các thiết bị chạy trên nhiều loại phần cứng và phần mềm.
* Một hạn chế khác khi nói đến hệ điều hành Android là các ứng dụng Android có thể dễ dàng bị vi phạm bản quyền.

# **IV. CÁC VẤN ĐỀ AN NINH AN TOÀN CỦA GOOGLE ANDROID**

## **CẬP NHẬT BẢO MẬT ĐỊNH KỲ**

* Cập nhật bảo mật định kỳ là một trong những biện pháp quan trọng nhất để đảm bảo an ninh và an toàn cho hệ điều hành Android. Google thường cung cấp các bản cập nhật bảo mật hàng tháng cho Android, được gửi đến các nhà sản xuất thiết bị và nhà mạng di động để triển khai cho người dùng cuối.
* **Tần suất cập nhậ**t: Google đã cam kết cung cấp các bản cập nhật bảo mật hàng tháng cho Android, nhằm vá các lỗ hổng bảo mật mới và cung cấp bảo mật tốt nhất cho người dùng.
* **Nội dung cập nhậ**t: Các bản cập nhật bảo mật thường bao gồm các bản vá cho các lỗ hổng bảo mật đã biết, cùng với các cải tiến và điều chỉnh bảo mật khác nhằm ngăn chặn các cuộc tấn công tiềm ẩn và cải thiện tính ổn định của hệ điều hành.
* **Triển khai**: Mặc dù Google cung cấp các bản cập nhật bảo mật định kỳ, việc triển khai cập nhật này phụ thuộc vào các nhà sản xuất thiết bị và nhà mạng di động. Điều này có nghĩa là không phải tất cả các thiết bị Android nhận được cập nhật đúng lúc, và một số thiết bị có thể bị trễ hoặc không nhận được cập nhật.
* **Nguy cơ từ việc không cập nhật**: Việc không cập nhật thiết bị Android đúng lúc tạo ra nguy cơ về bảo mật cho người dùng. Các hacker có thể tận dụng các lỗ hổng bảo mật đã biết trong các phiên bản cũ của hệ điều hành để tấn công người dùng, đánh cắp thông tin cá nhân hoặc gây ra các vấn đề khác đối với an ninh của thiết bị.
* **Hỗ trợ dài hạn**: Để giảm thiểu nguy cơ từ việc không nhận được cập nhật đúng lúc, Google đã tăng cường hỗ trợ dài hạn cho các phiên bản Android cũ. Điều này có nghĩa là người dùng có thể tiếp tục nhận được các bản cập nhật bảo mật dài hạn sau khi phiên bản chính thức của họ không còn được hỗ trợ nữa.

## **2. ỨNG DỤNG ĐỘC HẠI VÀ PHẦN MỀM GIẢ TẠO**

* Trong hệ sinh thái Android, sự phổ biến của ứng dụng di động đã mở ra cánh cửa cho sự lây lan của các ứng dụng độc hại và phần mềm giả mạo. Mặc dù Google Play Store, cửa hàng ứng dụng chính thức của Android, đã triển khai các biện pháp kiểm duyệt nghiêm ngặt, nhưng vẫn còn những ứng dụng không đáng tin cậy hoặc độc hại có thể xuất hiện, đe dọa an ninh và quyền riêng tư của người dùng.

### **2.1. Ứng dụng Độc Hại**

* Mục Tiêu: Các ứng dụng độc hại thường được thiết kế để đánh cắp thông tin cá nhân của người dùng, thực hiện giao dịch gian lận, hiển thị quảng cáo gây phiền toái, hoặc thậm chí kiểm soát thiết bị từ xa mà không có sự chấp thuận của người dùng.
* Phát Tán: Các ứng dụng độc hại có thể được phát tán thông qua Google Play Store hoặc các cửa hàng ứng dụng không chính thức, và thậm chí được phân phối qua các trang web độc lập, email rác, hoặc tin nhắn lừa đảo.
* Camouflaging: Một số ứng dụng độc hại có thể giả mạo thành các ứng dụng hữu ích hoặc phổ biến để lừa đảo người dùng tải về và cài đặt chúng, trước khi tiết lộ tính năng độc hại của chúng.

### **2.2. Phần Mềm Giả Mạo**

* Mục Tiêu: Phần mềm giả mạo thường là các bản sao của các ứng dụng phổ biến, nhưng được thay đổi để chứa mã độc hại hoặc quảng cáo không mong muốn. Mục đích chính của chúng là kiếm tiền bằng cách lừa đảo người dùng hoặc hiển thị quảng cáo gây phiền toái.
* Nguy Cơ: Khi người dùng tải về và cài đặt các bản sao giả mạo của các ứng dụng phổ biến từ các nguồn không đáng tin cậy, họ có thể không chỉ bị lừa đảo mà còn mở ra cánh cửa cho việc cài đặt mã độc hại lên thiết bị của mình.

### **2.3.Biện Pháp Phòng Ngừa**

* Kiểm Duyệt Nghiêm Ngặt: Google Play Protect, công nghệ kiểm duyệt tự động của Google, thường quét và loại bỏ các ứng dụng độc hại và phần mềm giả mạo từ Google Play Store. Tuy nhiên, không phải tất cả các ứng dụng độc hại đều bị phát hiện và loại bỏ ngay lập tức.
* Tải Chỉ Từ Nguồn Tin Cậy: Người dùng nên chỉ tải ứng dụng từ nguồn tin cậy như Google Play Store và tránh tải từ các nguồn không chính thức hoặc không đáng tin cậy.
* Kiểm Tra Phản Hồi và Đánh Giá: Trước khi tải xuống ứng dụng, người dùng nên đọc phản hồi từ người dùng khác và xem xét đánh giá để đánh giá mức độ đáng tin cậy của ứng dụng.

## **3. PHƯƠNG THỨC XÁC THỰC VÀ BẢO MẬT DẤU VÂN TAY**

* Android cung cấp nhiều phương thức xác thực như mật khẩu, PIN, mẫu vẽ và dấu vân tay. Tuy nhiên, việc sử dụng không cẩn thận hoặc thiết bị không được bảo vệ đúng cách có thể dẫn đến việc dễ bị truy cập bởi người không có quyền truy cập. Ví dụ, mật khẩu yếu hoặc sử dụng dấu vân tay không đảm bảo có thể bị hack bởi các phương pháp như sử dụng vân tay giả tạo hoặc hack mật khẩu. Điều này có thể dẫn đến việc người dùng bị lộ thông tin cá nhân hoặc truy cập trái phép vào các ứng dụng và dữ liệu.

## **4. QUYỀN TRUY CẬP ỨNG DỤNG**

* Android yêu cầu người dùng cấp quyền truy cập cho các ứng dụng như quyền truy cập vào máy ảnh, dữ liệu liên lạc, vị trí, vv. Tuy nhiên, một số người dùng có thể cấp quyền mà không suy nghĩ kỹ, tạo điều kiện cho các ứng dụng lạm dụng thông tin cá nhân. Việc cấp quyền truy cập không cân nhắc có thể dẫn đến việc ứng dụng sử dụng thông tin cá nhân của người dùng một cách không đúng đắn, gây ra nguy cơ về an ninh và quyền riêng tư.

## **5. CHẾ ĐỘ KHÓA VÀ MÃ HÓA**

* Android cung cấp tính năng mã hóa dữ liệu và chế độ khóa để bảo vệ dữ liệu cá nhân của người dùng. Tuy nhiên, không phải tất cả người dùng kích hoạt hoặc sử dụng chúng đúng cách. Nếu không kích hoạt hoặc sử dụng chúng không đúng cách, dữ liệu cá nhân có thể dễ dàng bị truy cập nếu thiết bị bị mất hoặc đánh cắp. Điều này tạo ra nguy cơ lớn đối với an ninh và quyền riêng tư của người dùng Android.

## **6. CƠ CHẾ SANDBOXING**

### **6.1. Nguyên Tắc Hoạt Động**

* Định Nghĩa: Sandboxing là một cơ chế an ninh mà hệ điều hành Android sử dụng để cô lập và bảo vệ dữ liệu của mỗi ứng dụng khỏi các ứng dụng khác và hệ thống hệ điều hành.
* Mô Hình: Mỗi ứng dụng trong Android chạy trong một môi trường ảo gọi là "sandbox". Trong sandbox, dữ liệu của ứng dụng được lưu trữ và các hoạt động của ứng dụng được thực hiện.
* Nguyên Lý: Sandboxing giúp ngăn chặn các ứng dụng không đáng tin cậy từ truy cập hoặc gây hại đến dữ liệu của các ứng dụng khác hoặc hệ thống hệ điều hành. Mỗi sandbox chỉ có quyền truy cập vào các tài nguyên được cấp phép.

### **6.2. Ưu Điểm và Hạn Chế**

* Ưu Điểm:
* Bảo Vệ Dữ Liệu: Sandboxing giúp bảo vệ dữ liệu cá nhân và quyền riêng tư của người dùng bằng cách ngăn chặn các ứng dụng không đáng tin cậy từ truy cập.
* Giảm Nguy Cơ: Nó giảm nguy cơ từ các ứng dụng độc hại hoặc không đáng tin cậy bằng cách cô lập chúng trong một môi trường an toàn.
* Hạn Chế:
* Lỗ Hổng Bảo Mật: Mặc dù sandboxes giúp cải thiện bảo mật, nhưng vẫn có thể tồn tại các lỗ hổng bảo mật hoặc kỹ thuật tấn công để vượt qua chúng.
* Cần Thiết Lập Chính Xác: Thiết lập sandbox không chính xác có thể dẫn đến các lỗ hổng bảo mật và việc lạm dụng.

### **6.3. Triển Khai Trong Android**

* Linux Kernel: Android sử dụng Linux Kernel và các cơ chế bảo mật của nó để triển khai sandboxing.
* Phân Quyền: Mỗi ứng dụng được gán một người dùng và nhóm người dùng riêng, và quyền truy cập vào tài nguyên hệ thống được kiểm soát thông qua hệ thống phân quyền của Android.
* Công Nghệ Bổ Sung: Android cũng sử dụng các công nghệ bổ sung như SELinux (Security-Enhanced Linux) để cung cấp thêm lớp bảo vệ cho sandboxes.

### **6.4. Ưu Điểm Cho Người Dùng**

* Bảo Vệ Dữ Liệu: Sandboxing giúp người dùng cảm thấy yên tâm khi sử dụng các ứng dụng từ cửa hàng ứng dụng chính thức như Google Play Store, bảo vệ dữ liệu cá nhân và quyền riêng tư của họ.
* Ngăn Chặn Tấn Công: Nó cũng giúp ngăn chặn các tấn công từ các ứng dụng không đáng tin cậy hoặc độc hại, giảm nguy cơ từ các cuộc tấn công mạng.