**TRƯỜNG ĐẠI HỌC HỒNG ĐỨC**



**BÀI TẬP LỚN**

**LẬP TRÌNH ỨNG DỤNG ANDROID**

**TÌM HIỂU VÀ PHÂN TÍCH KIẾN TRÚC, CHỨC NĂNG VÀ HOẠT ĐỘNG CỦA HỆ ĐIỀU HÀNH ANDROID**

**THANH HÓA, THÁNG 11 /2022**

**TRƯỜNG ĐẠI HỌC HỒNG ĐỨC**

**BÀI TẬP LỚN**

**LẬP TRÌNH ỨNG DỤNG ANDROID**

**TÌM HIỂU VÀ PHÂN TÍCH KIẾN TRÚC, CHỨC NĂNG VÀ HOẠT ĐỘNG CỦA HỆ ĐIỀU HÀNH ANDROID**

Sinh viên thực hiện: Đặng Thị Thuỳ Linh

Lớp: K22 CNTT

Khoa: Công nghệ thông tin và Truyền thông

Giảng viên hướng dẫn: Nguyễn Thị Bích Nhật

**THANH HÓA, THÁNG 11 /2022**

MỤC LỤC

[**I.TỔNG QUAN VỀ HỆ ĐIỀU HÀNH ANDROID** 4](#_Toc119937949)

[1. Giới thiệu chung về hệ điều hành Android 4](#_Toc119937950)

[2. Sự ra đời của Android 4](#_Toc119937951)

[3. Lịch sử phát triển Android 4](#_Toc119937952)

[4. Một số phiên bản của Android 4](#_Toc119937953)

[**II.** **KIẾN TRÚC HỆ ĐIỀU HÀNH ANDROID** 4](#_Toc119937954)

[**III.** **CÁC TÍNH NĂNG NỔI BẬT CỦA HỆ ĐIỀU HÀNH ANDROID** 4](#_Toc119937955)

[**IV.** **HOẠT ĐỘNG CỦA HỆ ĐIỀU HÀNH ANDROID** 6](#_Toc119937956)

[1. Bảo mật dữ liệu ứng dụng Android 6](#_Toc119937957)

[5. Ưu điểm và một số hạn chế của hệ điều hành Android 9](#_Toc119937958)

[a. Ưu điểm 9](#_Toc119937959)

[b. Hạn chế 10](#_Toc119937960)

[6. Quản lý bộ nhớ 10](#_Toc119937961)

[7. Quản lý tiến trình trong hệ điều hành Android 13](#_Toc119937962)

[a. Hệ thống phân cấp 13](#_Toc119937963)

[b. Tính năng tự động quản lý tiến trình của Android 14](#_Toc119937964)

[c. Ứng dụng Android có thể đáp ứng tự đáp ứng các sự kiện 14](#_Toc119937965)

[d. Quản lý tiến trình “cơ bản” 14](#_Toc119937966)

[**V.** **KẾT LUẬN** 15](#_Toc119937967)

# **TỔNG QUAN VỀ HỆ ĐIỀU HÀNH ANDROID**

## Giới thiệu chung về hệ điều hành Android

Android là một “Hệ điều hành” được cài đặt trên một số phần cứng riêng biệt mà ở đây chủ yếu là cài đặt trên các thiết bị smartphone (của Samsung, LG, HTC,…). Cũng giống như các hệ điều hành khác trên điện thoại (Window Phone 7, IOS, Blackberry…), Android là một hệ điều hành dành cho điện thoại được viết từ java, mã nguồn mở hoàn toàn giúp cho các lập trình viên cài đặt ứng dụng trên thiết bị do chính mình viết ra.

## Sự ra đời của Android

Ban đầu, Android được phát triển bởi công ty Android, với sự hỗ trợ tài chính từ Google và sau này được chính Google mua lại vào năm 2005. Android ra mắt vào ngày 5 tháng 11 năm 2007 cùng với tuyên bố thành lập “Liên minh thiết bị cầm tay mở”: Một hiệp hội gồm các công ty phần cứng, phần mềm, và viễn thông với mục tiêu đẩy mạnh các tiêu chuẩn mở cho các thiết bị di động.



Hình 1 Chiếc điện thoại đầu tiên chạy Android được bán vào tháng 10 năm 2008

## Lịch sử phát triển Android

Bước đầu, hệ điều hành Android chỉ đơn thuần là phần mềm trên điện thoại di động.

Tháng 7 năm 2005, Google mua lại Android ,Inc. Những nhà đồng sáng lập của Android chuyển sang làm việc tại Google gồm có Andy Rubin (đồng sáng lập công ty Danger), Rich Miner (đồng sáng lập công ty Wildfire Communications), Nick Sears (từng là phó chủ tịch của T-Mobile), và Chris White (trưởng nhóm thiết kế và phát triển giao diện tại WebTV). Khi đó, có rất ít thông tin về các công việc của Android, ngoại trừ việc họ đang phát triển phần mềm cho điện thoại di động.Tại Google, nhóm do Rubin đứng đầu đã phát triển một nền tảng thiết bị di động dựa trên hạt nhân Linux – như chúng ta đã biết Linux là hệ điều hành mã nguồn mở. Những nhà sáng lập Android từ đầu đã áp dụng nguyên tắc rẻ thay cho đắt, nguyên tắc đẳng thế. Android đã hướng đến đa số người dùng những phần mềm rẻ, chất lượng. Thực tế hiện nay chúng ta đã thấy các smartphone sử dụng hệ điều hành Android có giá phù hợp với túi tiền của đa số người sử dụng.

Sau đó, nền tảng ấy được họ tiếp thị đến các nhà sản xuất thiết bị cầm tay và các nhà mạng trên những tiền đề về việc cung cấp một hệ thống mềm dẻo, có khả năng nâng cấp mở rộng cao dựa trên nguyên tắc linh động có thể đáp ứng được trên nhiều loại thiết bị di động, nguyên tắc liên tục tác động có ích để nâng cấp và phát triển. Google cũng lên danh sách các thành phần phần cứng đáp ứng nền tảng và các đối tác phần mềm, đồng thời cam kết với các nhà mạng rằng họ sẵn sàng hợp tác ở nhiều cấp độ khác nhau. Ở giai đoạn này họ áp dụng nguyên tắc kết hợp, nguyên tắc cầu (tròn) hoá. Kết hợp lại làm nền tảng để tạo ra một chu trình khép kín: phần cứng - phần mềm - các nhà cung cấp dịch vụ đến với người sử dụng. Trên nền tảng đó, hệ điều hành của họ tạo ra sẽ được sống và phát triển bền vững.

Các nhà phát triển viết ứng dụng cho Android dựa trên ngôn ngữ Java. Sự ra mắt của Android vào ngày 5 tháng 11 năm 2007 gắn với sự thành lập của liên minh thiết bị cầm tay mã nguồn mở (Open Handset Alliance), bao gồm 78 công ty phần cứng, phần mềm và viễn thông nhằm mục đích tạo nên một chuẩn mở cho điện thoại di động trong tương lai.Và như vậy, Android chính thức gia nhập Liên minh thiết bị cầm tay mã nguồn mở (OHA) đồng thời Google đã công vố việc họ bắt tay phát triển hệ điều hành mã nguồn mở cho thiết bị di động nhằm cạnh tranh với Symbian, Windows Mobile và các đối thủ khác. Hãng cũng giới thiệu sản phẩm Android đầu tiên là T-Mobile G1. Đây là một thiết bị di động có hệ điều hành dựa trên nhân Linux phiên bản 2.6. Cạnh tranh lành mạnh, sẽ tạo điều kiện thuận lợi cho người sử dụng có được sản phẩm tốt, giá cả phải chăng, đó cũng chính là một nguyên tắc liên tục tác động có ích vào thị trường, cũng như khắc phục được “sức ì” của nhà cung cấp sản phẩm trước đó.

Từ tháng 10/2008, hệ điều hành Android đã chính thức trở thành phần mềm mã nguồn mở. Theo đó, các công ty thứ ba được phép thêm những ứng dụng của riêng của họ vào Android và bán chúng mà không cần phải hỏi ý kiến Google. Họ đã ứng dụng triệt để nguyên tắc tách khỏi và nguyên tắc phẩm chất cục bộ. Google cung cấp nền, các công ty phát triển ứng dụng có thể tuỳ theo thế mạnh của mình, hoặc tuỳ theo mục đích mà mình muốn hướng đến sẽ xây dựng cái mới dựa trên cái đã có, bổ sung thêm ưu điểm, phát huy các tiềm năng của cái đã có, đồng thời tiếp nhận thêm những cái mới để xây dựng thành một hệ thống mới. Điều đó cũng chứng tỏ rằng nguyên tắc “chứa trong” đã được ứng dụng.

Tháng 11/2008, Liên minh OHA ra mắt gói phát triển phần mềm Android SDK cho người lập trình. SDK là viết tắt của Software Development Kit, thông qua đó developer có thể xây dựng và phát triển các ứng dụng chạy được trên hệ điều hành Android. Nguyên tắc phân nhỏ được ứng dụng trong quyết định của OHA. Họ đã tận dụng được trí tuệ của cộng đồng phát triển và sử dụng Android thông qua việc cung cấp công cụ để người lập trình phát triển ứng dụng trên Android. Bất cứ hệ điều hành nào muốn sống được với thị trường đòi hỏi phải có nhiều ứng dụng ích lợi, hay, hiệu quả, phù hợp với nhu cầu của người sử dụng. Điểu đó cho thấy rằng họ đã sử dụng nguyên tắc kết hợp để kết hợp sức mạnh của tập thể tạo ra kho ứng dụng nhỏ gọn, đa dạng phục vụ nhu cầu sử dụng ngày càng cao của con người và có khả năng linh động. Người sử dụng có thể cài đặt hoặc gỡ bỏ khi cần thiết điều này thể hiện tính linh động của sản phẩm và nguyên tắc linh động đã được áp dụng. Hơn thế nữa ở giai đoạn này, họ cũng đã áp dụng nguyên tắc phân nhỏ, nguyên tắc tách khỏi, nguyên tắc phẩm chất cục bộ, và đặc biệt là nguyên tắc vạn năng. Thông qua các ứng dụng đa dạng được phát triển bởi cộng đồng thiết bị di động sử dụng hệ điều hành Android là một sản phẩm có nhiều chức năng kết hợp: có thể là một máy chụp hình, có khi là một chiếc la bàn, một thiết bị chơi games, máy nghe nhạc…Không phải chỉ riêng hệ điều hành Android cho phép các ứng dụng đa dạng như trên mà các hệ điều hành dành cho thiết bị di động khác cũng không thua kém, nhưng riêng với Android là một phần mềm mã nguồn mở và được phát triển ứng dụng trên đông đảo các nhà lập trình, chính vì thế làm tăng tính đa dạng của các ứng dụng. Một nhóm người phát triển trong nội bộ một công ty không thể nào đa dạng bằng sự hợp lực của nhiều công ty. Và hơn thế nữa, người dùng có kiến thức về lập trình cũng có thể tự tay mình thiết kế một ứng dụng phù hợp với nhu cầu của mình. Qua đó kho ứng dụng của Android cũng nhận được sự liên tục tác động có ích làm cho kho ứng dụng này phát triển một cách nhanh chóng. Đồng thời họ cũng sẽ gỡ bỏ khỏi kho ứng dụng những ứng dụng không tốt thông qua nguyên tắc quan hệ phản hồi giữa nhà cung cấp và người sử dụng. Nguyên tắc chuyển sang chiều khác cũng đã được áp dụng thông qua việc cho ra đời gói sản phẩm Android SDK. Trước đây, cung cấp các ứng dụng trên các thiết bị di động là công việc của những người phát triển sản phẩm, giờ đây cả người dùng có chuyên môn cũng được tự tay xây dựng một ứng dụng trên cái có sẵn và hoàn toàn miễn phí. Quả thật việc cho ra đời Android SDK là một dấu mốc lịch sử trong quá trình phát triển không ngừng và lớn mạnh của Android.

Tháng 12/2008, có thêm 14 thành viên mới gia nhập dự án Android được công bố, gồm có ARM Holdings, Atheros Communications, Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, và Vodafone Group Plc. - Tháng 2/2009, một số công ty trong đó có Qualcomm và Texas Instruments đã có trong tay những con chip chạy các phiên bản đơn giản của hệ điều hành Android, mang đến những trải nghiệm hoàn toàn mới cho người dùng trên toàn thế giới. - Đến năm 2010, số lượng smartphone nền tảng Android tăng trưởng mạnh mẽ. Hàng loạt nhà sản xuất hàng đầu đã bắt tay sản xuất smartphone như Samsung, HTC, Motorola… Thậm chí, Android còn được coi là “cứu cánh” cho nhiều đại gia công nghệ bước sang một trang mới trong việc cải thiện doanh số, bán hàng có lãi sau một thời gian dài trì trệ, tiêu biểu là Motorola.

## Một số phiên bản của Android

Từ lúc ra mắt phiên bản đầu tiên cho tới nay, Android đã có rất nhiều bản nâng cấp. Đa số đều tập trung vào việc vá lỗi và thêm những tính năng mới.

Android những thế hệ đầu tiên 1.0 ( 9/2008) và 1.1 ( 2/2009) chưa có tên gọi chính thức. Từ thế hệ tiếp theo, mỗi bản nâng cấp đều được đặt với những mã tên riêng dựa theo các món ăn hấp dẫn theo thứ tự bảng chữ cái từ “C-D-E-F-G-H-I”. Hiện tại các phiên bản chính của Android bao gồm:

**Android 1.5 (Cupcake):** Ra mắt tháng 4/2009: Phiên bản này có một số tính năng đáng chú ý như: khả năng ghi lại và xem video thông qua chế độ máy ghi hình, tải video lên YouTube và ảnh lên Picasa trực tiếp từ điện thoại, tích hợp bàn phím ảo với khả năng đoán trước văn bản, tự động kết nối với một thiết bị Bluetooth trong một khoảng cách nhất định, các widget và thư mục mới có thể cài đặt linh động trên màn hình chủ. Ice cream Sandwich 4.0.

**Android 1.6 (Donut)** : Ra mắt tháng 9/2009: Phiên bản này giúp Nâng cao trải nghiệm trên kho ứng dụng Android Market, tích hợp giao diện tùy biến cho phép người dùng xóa nhiều ảnh cùng lúc, nâng cấp Voice Search, nâng cấp khả năng tìm kiếm bookmarks, history, contacts và web trên màn hình chủ, bước đầu hỗ trợ màn hình độ phân giải WVGA. Nguyên tắc chứa trong và nguyên tắc vạn năng, nguyên tắc phản hồi được sử dụng trong suốt quá trình nâng cấp và phát triển Android.

**Android 2.0/2.1 (Eclair)**: Ra mắt tháng 10/2009. Phiên bản này có sự cải thiện rõ rệt trong giao diện người dùng, tối ưu hóa tốc độ phần cứng, hỗ trợ nhiều kích cỡ và độ phân giải màn hình hơn, thay đổi giao diện duyệt web và hỗ trợ chuẩn HTML5, Exchange ActiveSync 2.5, nâng cấp Google Maps 3.1.2, camera zoom kĩ thuật số tích hợp đèn flash, nâng cấp bàn phím ảo và kết nối Bluetooth 2.1. Nguyên tắc phẩm chất cục bộ, nguyên tắc thay đổi màu sắc, nguyên tắt kết hợp, nguyên tắc vạn năng, nguyên tắc đồng nhất được ứng dụng kết hợp trong giai đoạn nâng cấp này. Đồng nhất để phù hợp với qui luật phát triển của bộ phận có liên quan để chúng tương thích nhau. Thay đổi giao diện, màu sắc để không làm chai ì khả năng khám phá của con người, thay đổi kích cỡ để phù hợp với nhu cầu của người dùng thông qua nguyên tắc phản hồi. Kết hợp nhiều chức năng, tối ưu cái đã có ở phiên bản trước thể hiện nguyên tắc sáng tạo vạn năng và nguyên tắc chứa trong, nguyên tắc tách khỏi.

**Android 2.2 (Froyo)**: Ra mắt tháng 5/2010: Phiên bản này chú trọng nâng cấp tốc độ xử lí, giới thiệu engine Chrome V8 JavaScript, hỗ trợ Adobe Flash10.1, thêm tính năng tạo điểm truy cập Wi-Fi. Một tính năng đáng chú ý khác hỗ trợ chuyển đổi nhanh chóng giữa các ngôn ngữ và từ điển trên bàn phím đồng thời cho phép cài đặt và cập nhật ứng dụng ở các thiết bị mở rộng bộ nhớ. Một trong những smartphone đầu tiên chạy phiên bản Android 2.2 Froyo là LG Optimus One.

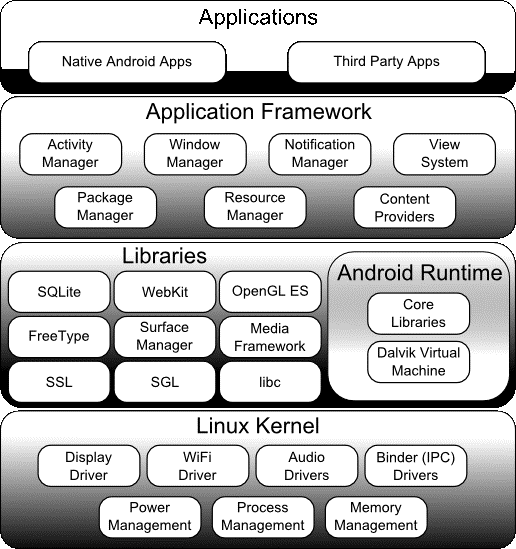
**Android 2.3 (Gingerbread)**: Ra mắt tháng 12/2010: Phiên bản này đã nâng cấp đáng kể giao diện người dùng, cải thiện bàn phím ảo, thêm tính năng copy/paste, hỗ trợ công nghệ giao tiếp tầm sóng ngắn NFC, hỗ trợ chuẩn video WebM và nâng cao tính năng copy–paste. Cùng với phiên bản Gingerbread, Google cũng ra mắt điện thoại đầu tiên của hãng sử dụng nền tảng này là Google Nexus S. Trong phiên bản này nguyên tắc thay đổi màu sắc đã được áp dụng. Không giống như người tiền nhiệm Android 2.2 tập trung vào hiệu năng xử lý, nền tảng mới chú ý nhiều hơn đến khả năng tinh chỉnh giao diện và hỗ trợ người dùng. Dễ thấy nhất là sự biến mất của các thanh màu xám thuộc các trình đơn thông báo mà thay vào đó là các thanh màu đen xuất hiện xuyên suốt cả hệ thống. Google cho biết đây là những cải tiến phù hợp với thị giác của người sử dụng cũng như sẽ giúp tiết kiệm điện năng bởi đặc điểm của màn hình Super AMOLED. Android 2.3 cung cấp gói hỗ trợ sóng vô tuyến ngắn với hiệu năng nổi bật giúp khách hàng thanh toán bằng smartphone thay cho thẻ tín dụng. Tuy nhiên, không dừng ở đó, người dùng còn có thể đọc nhanh các thông báo, áp-phích, quảng cáo từ NFC để tìm hiểu thêm về các chương trình khuyến mãi hay nhiều thứ khác mà chỉ cần đưa chú dế của mình lướt qua những thiết bị tương thích. Ngoài ra công cụ Copy/Paste có thể trở nên tiện dụng hoặc rắc rối với bất kỳ hệ điều hành nào. Nhưng trong Gingerbread, người sử dụng chỉ cần giữ một từ đủ lâu để thay cho các thao tác như trước kia. Đây là tính năng khá khác biệt so với Froyo - phiên bản Android 2.2. Phiên bản nàycòn cung cấp các thiết lập và tùy chọn đa năng, nền tảng có vai trò tích cực hơn trong việc đóng các ứng dụng không cần thiết, qua đó giúp kéo dài thời lượng pin đáng kể. Gingerbread quản lý ứng dụng thông minh hơn: để tiết kiệm thời gian, nó cung cấp tính năng Shortcut giúp khách hàng tìm được thứ mình muốn hết sức nhanh chóng. Người sử dụng cũng sẽ trở nên chủ động hơn khi đóng các ứng dụng đang hoạt động thông qua trình quản lý do hệ điều hành cung cấp.

**Android 3.0 (Honeycomb)**: Ra mắt tháng 2/2011: Đây là phiên bản hệ điều hành dành riêng cho máy tính bảng tablet với giao diện mới tối ưu hóa cho tablet, từ các thao tác đều phụ thuộc màn hình cảm ứng (như lướt web, duyệt mail..). Honeycomb hỗ trợ bộ xử lí đa nhân và xử lý đồ họa đồng thời hỗ trợ nhiều màn hình home khác nhau, cho phép người dùng dễ dàng tùy biến giao diện nếu muốn. Nguyên tắc phẩm chất cục bộ đã áp dụng ở lần nâng cấp này và chỉ phụ vụ cho tablet.

**Android 4.0 (Ice-cream sandwich)**: ra mắt vào 11/2011, là sự kết hợp giữa Gingerbread và Honeycomb và chạy trên tất cả các thiết bị như điện thoại, máy tính bảng, máy tính xách tay. Chủ đề chính của phiên bản này sẽ là "một HĐH hoạt động khắp nơi" - trên điện thoại, trên máy tính bảng với mọi kích cỡ và hình dạng.…Bước nâng cấp này nguyên tắc chứa trong, nguyên tắc kết hợp, nguyên tắc phẩm chất cục bộ, nguyên tắc thay đổi màu sắc…và nhiều nguyên tắc khác đã áp dụng ở những lần nâng cấp trước. Một số đặc điểm nổi bật của Ice- cream sandwich là giao diện người dùng hoàn toàn mới, quản lý đa nhiệm tốt hơn các phiên bản trước. Thao tác copy, paste được thực hiện dễ dàng hơn, cho phép mở khoá bằng khuôn mặt, hạn chế truy cập wifi yếu, cho phép chụp ảnh màn hình không cần cài thêm phần mềm…

# **KIẾN TRÚC HỆ ĐIỀU HÀNH ANDROID**

Android bao gồm 4 tầng từ dưới lên trên là tầng hạt nhân Linux, Tầng Libraries & Android Runtine, Tầng Application Framework và tầng trên cùng là Tầng Application.



Hình 2 Mô hình kiến trúc nền tảng của Android

* 1. Tầng hạt nhân Linux (Linux Kernel layer)

Hệ điều hành Android được phát triển dựa trên hạt nhân linux, cụ thể là hạt nhân linux phiên bản 2.6, điều đó được thể hiện ở mức dưới cùng. Tất cả mọi hoạt động của điện thoại muốn thi hành được thì đều được thực hiện ở mức cấp thấp ở lớp 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 (proccess).

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 cầm tay như hạn chế về bộ vi xử lý, dung lượng bộ nhớ, kích thước màn hình, nhu cầu kết nối mạng không dây…

Tầng này có các thành phần chủ yếu:

* Display Driver: Điều khiển việc hiển thị lên màn hình cũng như thu nhận những điều khiển của người dùng lên màn hình (di chuyển, cảm ứng…)
* Camera 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 ciệc thu phát sóng wifi.
* Audio Driver: Điều khiển các bộ thu phát âm thanh, giải mã các tín 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 nhứ như thẻ SD, flash.
* Power Management: Giám sát việc tiêu thụ điện năng.
  1. Tầng Libraries & Android Runtine

Gồm 2 thành phần là phần Library và Android Runtine.

1. Phần Library

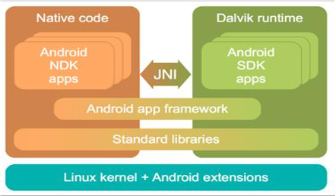
Phần này có nhiều thư viện được viết bằng C/C++ để các phần mềm có thể sử dụng, các thư viện đó được tập hợp thành một số nhóm như:

* Thư viện hệ thống (System C Library): thư viện dựa trên chuẩn C, được sử dụng chỉ bởi hệ điều hành.
* Thư viện Media (Media Library): có nhiều code để hỗ trợ việc phát và ghi các loại định dạng âm thanh, hình ảnh, video thông dụng.
* Thư viện Web (LibWebCore): đây là thành phần để xem nội dung trên web, được sử dụng để xây dựng phần mềm duyệt web (Android Browse) cũng như để các ứng dụng khác có thể nhúng vào. Nó cực kỳ mạnh, hỗ trợ nhiều công nghệ mạnh mẽ như HTML5, JavaScript, CSS, DOM, AJAX…
* Thư viện SQLite: hệ cơ sở dữ liệu để các ứng dụng có thể sử dụng.
* …

1. Phần Android Runtine

Phần này chứa các thư viện mà một chương trình viết bằng ngôn ngữ Java có thể hoạt động. Phần này có 2 bộ phận tương tự như mô hình chạy Java trên máy tính thường. Thứ nhất là các thư viện lõi (Core Library), chứa các lớp như JAVA IO, Collection, File Access. Thứ hai là máy ảo java (Dalvik Virtual Machine).

Mặc dù cũng được viết từ ngôn ngữ Java nhưng một ứng dụng Java của hệ điều hành Android không được chạy bằng JRE của Sun (nay là Oracle) (JVM) mà là chạy bằng máy ảo Dalvik do Google phát triển.



Hình 3 Mô hình hợp tác giữa máy ảo Dalvik và Native code

* 1. Tầng Application Framework

Tầng này xây dựng bộ công cụ - các phần tử ở mức cao để các lập trình viên có thể nhanh chóng xây dựng ứng dụng. Nó được viết bằng java, có khả năng sử dụng chung để tiết kiệm tài nguyên.

Đây là mộ nền mở có ưu điểm:

* Với những hãng sản xuất điện thoại: có thể tuỳ biến để phù hợp với cấu hình điện thoại mà họ sản xuất như để có nhiều mãu mã, style hợp thị hiếu người dùng. Vì thế nên tuy cùng chung nền tảng Android mà điện thoại của Google có thể khác hản HTC, T-Mobile, Samsung…
* Với lập trình viên: cho phép lập trình viên có thể sử dụng các API ở tầng trên mà không cần phải hiểu rõ cấu trúc bên dưới, tạo điều kiện cho lập trình viên tự do sáng tạo bởi vì chỉ cần quan tâm đến nội dung mà ứng dụng và học làm việc. Một tập hợp API rất hữu ích được xây dựng sẵn như hệ thống định vị, các dịch vụ chạy nền, liên lạc giữa các ứng dụng, các thành phần giao diện cấp cao…

Giới thiệu một số thành phần của phần này:

* Activity Manager: quản lý các chu kỳ sống của một ứng dụng cũng như cung cấp công cụ điều khiển các Activity.
* Telephony Manager: cung cấp công cụ để thực hiện liên lạc như gọi điện thoại.
* XMPP Service: cung cấp công cụ để liên lạc trong thời gian thực.
* Location Manager: cho phép xác định vị trí điện thoại dựa vào hệ thống định vị toàn cầu GPS và Google Maps.
* Window Manager: quản lý việc xây dựng và hiển thị các giao diện người dùng cũng như tổ chức quản lý các giao diện giữa các ứng dụng.
* Resourse Manager: quản lý tài nguyên tĩnh của các ứng dụng bao gồm các file hình ảnh, âm thanh, layout, string. (Những thành phần không được viết bởi ngôn ngữ lập trình).
* Notication Manager: quản lý việc hiển thị các thông báo (Như báo có tin nhắn, email mới…)
* …
  1. Tầng Application

Đây là lớp ứng dụng giao tiếp với người dùng, bao gồm các ứng dụng như:

* Các ứng dụng cơ bản, được cài đặt đi liền với hệ điều hành là gọi điện (Phone), quản lý danh bạ (Contacts), duyệt web (Browser), nhắn tin (SMS), lịch làm việc (Calender), đọc email (Email-Client),…
* Các ứng dụng được cài thêm như các phần mềm trò chơi, từ điển…

Các chương trình có các đặc điểm:

* Viết bằng Java, phần mở rộng 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 ngòi dùng hoặc là một background: chương trình chạy nền hay là dịch vụ.
* Android là hệ điều hành đa nhiệm, điều đó có nghĩa là trong cùng một thời điểm, có thể có nhiều chương trình cùng chạy một lúc, tuy nhiên, với mỗi ứng dụng thì có duy nhất một thực thể (Instance) được phép chạy mà thôi. Điều đó có tác dụng hạn chế sự lạm dụng tài nguyên, giúp hệ thống hoạt động tốt hơn.
* Các ứng dụng được gán số ID của người sử dụng nhằm phân định 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 đó chỉ có một hạn chế nhỏ đó là không được 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.

# **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. Ví dụ: nếu người dùng nhấn vào nút điều hướng, thiết bị sẽ rung.

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.

Android chạy trên cả hai tiêu chuẩn cellular (thiết bị hỗ trợ khe cắm sim) được triển khai rộng rãi nhất là GSM/HSDPA và CDMA/EV-DO. Android cũng hỗ trợ:

* Bluetooth
* Các giao thức truyền thông 3G như EVDO và HSDPA
* Wifi
* Tự động vá lỗi
* Tin nhắn SMS và MMS
* Camera tĩnh/Camera kỹ thuật số
* GPS
* Bản đồ
* Ứng dụng đa nhiệm/đa tác vụ (multitask)

# **HOẠT ĐỘNG CỦA HỆ ĐIỀU HÀNH ANDROID**

## Bảo mật dữ liệu ứng dụng Android

Android đã có chức năng bảo mật tích hợp trong hệ điều hành, và dựa vào các chức năng đó, chúng ta có thể xây dựng ứng dụng với các chế độ bảo mật của android.

Các chức năng bảo mật giúp ta xây dựng bảo mật ứng dụng bao gồm:

* Ứng dụng Android Sandbox, mà nó cô lập dữ liệu ứng dụng và code thực thi từ các ứng dụng khác.
* Ứng dụng FrameWork với các chức năng bảo mật như cryptography, permissions, và secure IPC.
* Công nghệ như ASLR, NX, ProPolice, safe\_iop, OpenBSD dlmalloc, OpenBSD Calloc, và Linux mmap\_min\_addr để giảm thiểu các rủi ro liên quan đến các lỗi quản lý bộ nhớ thông thường.
* Một hệ thống tập tin được mã hóa có thể được kích hoạt để bảo vệ dữ liệu trên các thiết bị bị mất hoặc bị đánh cắp.
* Cung cấp cho phép người dùng hạn chế truy cập vào các tính năng hệ thống và dữ liệu người dùng.
* Cho phép kiểm soát dữ liệu ứng dụng trên mỗi ứng dụng cơ sở.

1. **Lưu trữ dữ liệu**

Lưu trữ dữ liệu là một trong những tính năng bảo mật quan trọng nhất của ứng dụng trên Android, các dữ liệu được lưu trên thiết bị có thể truy cập vào các úng dụng khác.

Có 3 các cơ bản để lưu trữ dự liệu trên thiết bị:

* **Sử dụng bộ nhớ trong**: Mặc định, các tập tin tự tạo ra trên bộ nhớ trong có thể truy cập duy nhất từ ứng dụng của chính mình. Đây là chức năng được hỗ trợ bởi tính năng bảo mật Android và có trên hầu hết các ứng dụng. Thông thường khi lưu trữ dữ liệu, ta nên tránh sử dụng chế độ **MODE\_WORLD\_WRITEABLE** hoặc chế độ **MODE\_WORLD\_READABLE** cho các tập tin IPC vì chúng không cung cấp khả năng để hạn chế truy cập dữ liệu cho các ứng dụng cụ thể, và cũng không cung cấp bất kỳ việc kiểm soát trên định dạng dữ liệu. Nếu ta muốn chia sẻ dữ liệu của ta với các ứng dụng khác ta có thể sử dụng Content provider. để bảo vệ các dữ liệu nhạy cảm ta có thể mã hóa các tập tin sử dụng key mà không thể truy cập trực tiếp vào ứng dụng cụ thể.
* **Lữu trữ bộ nhớ ngoài:** Các tập tin được lưu trên bộ nhớ ngoài như SD Card, khi lưu trữ trên bộ nhớ ngoài thì người dùng có thể được sửa đổi, xóa, thêm,.. bởi người dùng. Không nên lưu trữ các tập tin thực thi hay lớp dữ liệu trên bộ nhớ ngoài, nếu ứng dụng của ta không lấy các tập tin thực thi từ lưu trữ ngoài, các tập tin nên được mã hóa trước khi sử dụng.
* **Sử dụng Content providers:** cung cấp một cơ chế lưu trữ có cấu trúc có thể được giới hạn ứng dụng của riêng ta hoặc xuất khẩu cho phép truy cập bởi các ứng dụng khác, Nếu ta không có ý định để cung cấp các ứng dụng khác truy cập vào ContentProvider của ta, ta sử dụng thuộc tính: android:exported="false", trong tập tin manifest. nếu android:exported="android:exported" sẽ cho phép các ứng dụng khác truy cập dữ liệu của ta. Nếu ta muốn chia sẻ dữ liệu duy nhất trong ứng dụng của ta thì sử dụng thuộc tính: android:protectionLevel="signature".khi truy cập Content provider, sử dụng các tham số truy vấn dữ liệu như query(), update(), và delete() để tránh truy suất dữ liệu không tin cậy.

1. **Sử dụng permission**

Chúng ta nên giảm thiểu tối đa số lượng request permission trong ứng dụng. Điều này sẽ làm giảm thiểu rủi ro mất dữ liệu cho ứng dụng, nếu permission không cần thiết cho ứng dụng thì không nên cung cấp permisison.ngoài việc yêu cầu permisison, ứng dụng của ta có thể sử dụng quyền bảo vệ permission này sẽ cho phép ứng dụng của ta tiếp xúc với cá ứng dụng khác.

1. **Sử dụng IP Networking**

Chúng ta nên sử dụng HTTPS trên HTTP bất cứ nơi nào mà HTTPS được hỗ trợ trên các máy chủ, vì thiết bị di động thường xuyên kết nối vào mạng mà không được bảo vệ, chẳng hạn như các điểm nóng Wi-Fi công cộng. Mã hóa giao tiếp ổ cắm cấp có thể được thực hiện dễ dàng bằng cách sử dụng các lớp SSLSocket. Với tần suất các thiết bị Android kết nối với mạng không dây không an sử dụng Wi-Fi, việc sử dụng mạng an toàn được khuyến khích cho tất cả các ứng dụng truyền thông trên mạng. Chúng ta không nên sử dụng localhost để sử lý dữ liệu.

1. **Sử dụng Telephony Networking**

Các giao thức tin nhắn SMS được thiết kế chủ yếu để giao tiếp với người sử dụng, do những hạn chế của SMS nên nên chúng ta nên sử dụng Google Cloud Messaging (GCM) và mạng IP để gửi thông điệp dữ liệu từ một máy chủ web đến ứng dụng của ta trên thiết bị.

* 1. **Sử dụng Webview**

Chúng ta sử dụng Webview để hiện thị nội dung HTML và Javascript, chính vì vậy nếu chúng ta dùng không đúng cách có thể dẫn đến không an toàn dữ liệu. Nếu ứng dụng của ta không trực tiếp sử dụng Javascript trong một Webview thì không nên gọi **setJavaScriptEnabled ().** Bởi mặc định, webview không thực thi javascript.nếu ứng dụng của ta truy cập dữ liệu với webview, ta nên gọi phương thức clearCache() để xóa các tập tin đã lưu trữ trên local.

* 1. **Sử dụng mã hóa dữ liệu**

Android cung cấp các thuật toán để mã hóa dữ liệu, nếu ta muốn sử dụng một tập tin an toàn từ việc kết nối internet ta nên sử dụng các giao thức **HttpsURLConnection** hoặc **SSLSocket** và các thuật toán mã hóa AES hoặc RSA được cung cấp trong lớp Cipher.

* 1. **Sử dụng Intent**

Intent cơ chế ưu tiên cho không đồng bộ IPC trong Android. Tùy thuộc vào yêu cầu của ứng dụng của ta, ta có thể sử dụng sendBroadcast (), sendOrderedBroadcast (), hoặc một Intent rõ ràng để một thành phần ứng dụng cụ thể. Nếu dữ liệu trong broadcast intent ta nên áp dụng một permission để đảm bảo rằng các ứng dụng độc hại không thể đăng ký để nhận được những tin nhắn mà không có quyền thích hợp, trong trường hợp này nên sử dụng Intent trực tiếp, thay vì gọi broadcast.

* 1. **Sử dụng Service**

Service thường được sử dụng để cung cấp các chức năng cho các ứng dụng khác để sử dụng. Mỗi lớp dịch vụ phải được khai báo tương ứng trong file manifest của nó. Theo mặc định Service không được gọi bởi các ứng dụng khác, tuy nhiên ta có thể thêm <ues-permission> trong file manifest để start, stop or bind tới một dịch vụ.

* 1. **Sử dụng binder and messenger interfaces**

Sử dụng Binder hoặc Messenger là các cơ chế ưu tiên cho RPC-style IPC trong Android. Chúng cung cấp một giao diện được xác định rõ cho phép xác thực lẫn nhau của các thiết bị đầu cuối, nếu có yêu cầu. Binder và Messenger đối tượng không được khai báo bên trong manifest ứng dụng. Nếu ta đang tạo ra một giao diện mà yêu cầu xác thực và / hoặc truy cập điều khiển, những người điều khiển phải được bổ sung rõ ràng là mã trong giao diện Binder hoặc Messenger. Nếu cung cấp một giao diện mà không yêu cầu kiểm soát truy cập, sử dụng checkCallingPermission () để xác định người gọi có permission hay không.

* 1. **Sử dụng broadcast receivers**

Một BroadcastReceiver xử lý yêu cầu không đồng bộ khởi tạo bởi một Intent. Receiver có thể được gọi bởi bất kỳ một ứng dụng nào, nếu BroadcastReceiver ta muốn sử dụng cho ứng dụng khác ta nên cung cấp cho nó **permisison <receiver>** trong tập tin manifest. Điều này sẽ ngăn chặn các ứng dụng mà không có quyền thích hợp từ việc gửi một Intent đến BroadcastReceiver.

* 1. **Sử lý dữ liệu người dùng**

Phương pháp tốt nhất để bảo mật dữ liệu người dùng là để giảm thiểu việc sử dụng các API truy cập vào dữ liệu người dùng nhạy cảm hoặc cá nhân, Nếu ta có quyền truy cập vào dữ liệu người dùng và có thể tránh lưu trữ hoặc truyền thông tin, không lưu trữ hoặc truyền tải dữ liệu. Ta cũng nên xem xét liệu ứng dụng của ta có thể vô tình để lộ thông tin cá nhân cho các bên khác như thành phần của bên thứ ba cho quảng cáo của bên thứ ba hoặc các dịch vụ được sử dụng bởi ứng dụng của ta. Nếu ta không biết lý do tại sao một phần hoặc dịch vụ đòi hỏi một thông tin cá nhân, không cung cấp nó. Nói chung, việc giảm các quyền truy cập vào thông tin cá nhân bằng cách ứng dụng của ta sẽ làm giảm tiềm năng cho các vấn đề trong lĩnh vực này. Hãy cẩn thận khi viết cho các bản ghi trên thiết bị. Trong Android, các bản ghi là một nguồn tài nguyên chia sẻ, và có sẵn cho một ứng dụng với sự cho phép READ\_LOGS. Mặc dù các dữ liệu nhật ký điện thoại chỉ là tạm thời và xóa trên khởi động lại, các thông tin người dùng vô tình có thể rò rỉ dữ liệu người dùng cho các ứng dụng khác.trên đây là một số thành phần cơ bản của ứng dụng android, và cách bảo mật khi sử dụng chúng.

## Ưu điểm và một số hạn chế của hệ điều hành Android

### Ưu điểm

* **Kho ứng dụng đa dạng**

Với kho ứng dụng Google Play, hệ điều hành Android đáp ứng tốt các nhu cầu của người dùng, từ chơi game online cho đến chỉnh sửa ảnh, video, làm việc với gần 3 triệu ứng dụng khác nhau.

* **Nhiều mẫu mã**

Hệ điều hành Android được nhiều nhiều nhà sản xuất lớn sử dụng cho thiết bị điện tử của họ. Do đó, người dùng có thể thoải mái lựa chọn giữa nhiều mẫu mã khác nhau, từ các mẫu giá rẻ cho đến các mẫu thuộc phân khúc cao cấp.

* **Cho phép mở rộng bộ nhớ bằng thẻ nhớ**

Với các thiết bị của Apple, ta chỉ có thể sử dụng bộ nhớ trong có sẵn của máy. Còn với phần lớn các thiết bị Android, ta sẽ có lựa chọn mở rộng bộ nhớ có sẵn với các loại thẻ nhớ dung lượng cao.

* **Khả năng tùy biến cao**

Do bản chất là mã nguồn mở, bất kỳ ai cũng có thể xem được mã nguồn của hệ điều hành Android. Điều này đồng nghĩa với việc các nhà cung cấp hay kể cả lập trình viên riêng lẻ đều có thể tự do tùy biến Android để đạt hiệu năng tốt nhất hoặc lược bỏ những tính năng không cần thiết.

* **Được nhiều người dùng ưa chuộng**

Hệ điều hành này có cộng đồng người dùng đông đảo, do đó nếu ta gặp vấn đề về thiết bị hay về phiên bản Android đang sử dụng, ta có thể được hỗ trợ nhanh chóng từ phía cộng đồng.

### 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.

## Quản lý bộ nhớ

Máy ảo Android Runtime (ART) và Dalvik sử dụng các tùy chọn phân trang và ánh xạ bộ nhớ (mmapping) 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.

* **Thu gom rác:**

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 bộ nhớ trở lại vùng nhớ khối xếp 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 xếp của Android là một bộ nhớ mang tính thế hệ, 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 *Thế hệ trẻ*. (vùng nhớ chứa các đối tượng mới được khởi tạo). Khi một đối tượng hoạt động đủ lâu, đối tượng này có thể được quảng bá cho một thế hệ cũ, theo sau là một thế hệ vĩnh viễn.

Mỗi thế hệ của vùng nhớ khối xếp có giới hạn trên riêng về dung lượng bộ nhớ mà các đối tượng ở đó có thể chiếm. Bất cứ khi nào một thế hệ 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 thế hệ đối tượng mà nó đang thu thập và số lượng đối tượng đang hoạt động trong mỗi thế hệ.

* **Bộ nhớ chung**

Để đáp ứng được mọi yêu cầu về RAM, Android sẽ cố gắng chia sẻ các trang RAM trên các quy trình. Ta có thể làm như vậy theo các cách sau:

* Mỗi quy trình của ứng dụng đều được tách ra từ một quy 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 quy trình mới đối với ứng dụng, hệ thống sẽ phân tách phát triển nhánh quy trình Zygote, sau đó tải và chạy mã của ứng dụng trong quy 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 quy trình của ứng dụng.
* Hầu hết dữ liệu tĩnh được liên kết vào một quy trình. Kỹ thuật này cho phép chia sẻ dữ liệu giữa các quy 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 quy trình bằng cách sử dụng vùng bộ nhớ dùng chung được phân bổ rõ ràng (với ashmem hoặc gralloc). 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.

* **Phân bổ và thu hồi bộ nhớ ứng dụng**

Vùng nhớ khối xếp Dalvik bị hạn chế trong một phạm vi bộ nhớ ảo duy nhất cho mỗi quy trình ứng dụng. Điều này xác định kích thước vùng nhớ khối xếp 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 xếp không giống với dung lượng bộ nhớ thực mà vùng nhớ khối xếp sử dụng. Khi kiểm tra vùng nhớ khối xếp 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 quy 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 của ta.

Bộ nhớ khối xếp Dalvik không thu gọn kích thước hợp lý của vùng nhớ khối xếp, nghĩa là Android không phân đoạn vùng nhớ khối xếp để đóng dung lượng. Android chỉ có thể thu nhỏ kích thước vùng nhớ khối xếp hợp lý khi có dung lượng chưa được sử dụng ở cuối vùng nhớ khối xếp. Tuy nhiên, hệ thống vẫn có thể giảm bộ nhớ thực mà vùng nhớ khối xếp sử dụng. Sau khi thu gom rác, Dalvik sẽ di chuyển qua vùng nhớ khối xếp 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 quy 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.

* **Hạn chế bộ nhớ 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 xếp cho từng ứng dụng. Giới hạn kích thước vùng nhớ khối xếp chính xác sẽ khác nhau giữa các thiết bị dựa trên dung lượng RAM mà thiết bị có sẵn về tổng thể. Nếu ứng dụng của ta đã đạt đến hạn mức vùng nhớ khối xếp và đang cố gắng phân bổ thêm bộ nhớ, ứng dụng có thể sẽ nhận một OutOfMemoryError.

Trong một số trường hợp, ta có thể muốn truy vấn hệ thống để xác định chính xác dung lượng vùng nhớ khối xếp sẵn có trên thiết bị hiện tại. Ví dụ như để xác định lượng dữ liệu an toàn để lưu trong một bộ nhớ đệm. Ta có thể truy vấn hệ thống cho minh hoạ này bằng cách gọi [getMemoryClass()](https://developer.android.com/reference/android/app/ActivityManager#getMemoryClass()). Phương thức này trả về một số nguyên cho biết số megabyte có sẵn cho vùng nhớ khối xếp của ứng dụng.

* **Chuyển đổi giữa các 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 xếp cho từng ứng dụng. Giới hạn kích thước vùng nhớ khối xếp chính xác sẽ khác nhau giữa các thiết bị dựa trên dung lượng RAM mà thiết bị có sẵn về tổng thể. Nếu ứng dụng của ta đã đạt đến hạn mức vùng nhớ khối xếp và đang cố gắng phân bổ thêm bộ nhớ, ứng dụng có thể sẽ nhận một [OutOfMemoryError](https://developer.android.com/reference/java/lang/OutOfMemoryError).

Trong một số trường hợp, ta có thể muốn truy vấn hệ thống để xác định chính xác dung lượng vùng nhớ khối xếp sẵn có trên thiết bị hiện tại. Ví dụ như để xác định lượng dữ liệu an toàn để lưu trong một bộ nhớ đệm. Ta có thể truy vấn hệ thống cho minh hoạ này bằng cách gọi [getMemoryClass()](https://developer.android.com/reference/android/app/ActivityManager#getMemoryClass()). Phương thức này trả về một số nguyên cho biết số megabyte có sẵn cho vùng nhớ khối xếp của ứng dụng.

## Quản lý tiến trình trong hệ điều hành Android

### Hệ thống phân cấp

Thứ bậc của process trong Android được chia thành 5 cấp theo 5 mức độ quan trọng từ cao đến thấp:

**Foreground process:** Đây chính là những process dùng để chạy ứng dụng (app) ta đang dùng. Những process khác cũng có thể được xem là foreground khi chúng có liên hệ trức tiếp với process “xử lý” app đang chạy. Tại một thời điểm cũng chỉ có vài foreground process mà thôi.

**Visible process:** 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. Dễ thấy nhất là khi cài các theme hỗ trợ khả năng “làm mờ” ứng dụng hoặc “ghim” ứng dụng lên màn hình.

**Service process:** 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 ta đang nghe nhạc và muốn chuyển sang chơi game, khi ta mở game cũng là lúc process phát nhạc trở thành service process, vẫn tiếp tục chơi nhạc khi ta làm việc khác.

**Background process:** 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 ta có thể xem chúng là những ứng dụng đang “tạm dừng”. Background process vẫn sử dụng RAM, cho phép ta 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:** 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.

### Tính năng tự động quản lý tiến trình của Android

Android có khả năng quản lý process một cách tự động, do đó ta không cần cài đặt bất kì ứng dụng “quản lý ứng dụng” nào.

Khi cần nhiều tài nguyên, Android tự động xóa các process 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 ta sẽ tự tắt, tập tin tải về tự động dừng.

Trong đa số trường hợp, ta 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 ta 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).

Một số ứng dụng được lập trình đặt icon trên thanh thông báo notification như avast!, TrustGo Antivirus hay… Advanced Task Killer. Nếu ta cố tắt icon này, ứng dụng “chống lại”. Bằng cách luôn giữ icon trên thanh thông báo, chúng cho biết mình không phải là background process để khỏi bị hệ thống loại bỏ khi cần tài nguyên.

### Ứng dụng Android có thể đáp ứng tự đáp ứng 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 ta mở wifi để tự nhận tin nhắn.

### Quản lý tiến trình “cơ bản”

Người dùng không cần tự mình làm việc này, nhưng cũng có vài cách để ta “làm chủ” thiết bị. Có thể dùng menu đa nhiệm của Android 4.0 trở lên (thường gọi Recent Apps). 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.

Những ứng dụng hiện lên là chúng 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…).

# **KẾT LUẬN**

Hệ điều hành là một phần quan trọng của hầu hết các hệ thống máy tính, điện tử. Nó điều khiển và phối hợp việc sử dụng phần cứng cho những ứng dụng khác nhau, cung cấp môi trường mà các chương trình có thể làm việc hữu hiệu trên đó. Hệ điều hành có thể được coi như bộ phân phối tài nguyên của thiết bị, hoạt động như một bộ quản lý tài nguyên và phân phối chúng cho các chương trình và người sử dụng khi cần thiết. Sau khi hoàn thành bài tiểu luận em đã hiểu rõ hơn về hệ điều hành Android và cách thức hoạt động của nó. Trong suốt quá trình thực hiện bài tiểu luận với sự hướng dẫn tận tình của cô Nguyễn Thị Bích Nhật em đã hoàn thành bài tiểu luận này. Đề tài này đã giúp chúng em củng cố thêm kiến thức về nhiều mặt, đặc biệt là hiểu rõ hơn về một nền tảng công nghệ có tính ứng dụng cao. Trong quá trình làm tiểu luận chắn hẳn vẫn còn nhiều thiếu sót, em rất mong nhận được sự góp ý của quý thầy cô và các bạn đề bài tiểu luận đạt kết quả tốt hơn.

Em xin chân thành cảm ơn!

Thanh Hóa, ngày….tháng….năm 2022

|  |  |  |
| --- | --- | --- |
| **Khoa CNTT&TT** | **Giáo viên Hướng Dẫn**  **Nguyễn Thị Bích Nhật** | **Sinh viên thực hiện**  **Đặng Thị Thuỳ Linh** |