

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1

o0o



BÀI TẬP LỚN

Đề tài 8: Hệ điều hành Android

Môn học: Hệ điều hành

Số thứ tự nhóm: 03

Nguyễn Đắc Thành	MSSV: B21DCCN678
La Thảo Vân	MSSV: B21DCCN126
Trần Tiến Đạt	MSSV: B21DCCN030
Tạ Kiều Yến	MSSV: B21DCCN810
Cao Hồng Đức	MSSV: B21DCCN234
Nguyễn Hồng Đăng	MSSV: B21DCAT051
Ngô Văn Trọng	MSSV: B21DCCN726

Giảng viên hướng dẫn: Ths. Đinh Xuân Trường

HÀ NỘI, 11/2023

ĐÓNG GÓP CỦA CÁC THÀNH VIÊN TRONG NHÓM

Nguyễn Đắc Thành	MSSV: B21DCCN678	100%
La Thảo Vân	MSSV: B21DCCN126	100%
Trần Tiên Đạt	MSSV: B21DCCN030	100%
Tạ Kiều Yến	MSSV: B21DCCN810	100%
Cao Hồng Đức	MSSV: B21DCCN234	100%
Nguyễn Hồng Đăng	MSSV: B21DCAT051	100%
Ngô Văn Trọng	MSSV: B21DCCN726	100%

LỜI CẢM ƠN

Xin kính chào thầy, nhóm 3 môn Hệ điều hành, xin gửi lời cảm ơn chân thành và sâu sắc đến thầy, cũng như đến tất cả các thành viên trong nhóm, vì đã cùng nhau chăm chỉ và quyết tâm thực hiện Bài tập lớn này.

Đầu tiên và quan trọng nhất, chúng em muốn bày tỏ lòng biết ơn đến thầy Đinh Xuân Trường đã hướng dẫn cho chúng em hoàn thành bài tập lớn. Những lời khuyên, sự chỉ bảo, và kiến thức chuyên môn mà thầy chia sẻ đã giúp chúng em hiểu rõ hơn về nội dung môn học và áp dụng kiến thức đó vào Bài tập lớn.

Cảm ơn thầy vì sự kiên nhẫn và tận tâm trong việc giảng dạy, giúp chúng em vượt qua những khó khăn, và hỗ trợ chúng em phát triển kỹ năng cần thiết trong lĩnh vực này.

Đến tất cả các thành viên trong nhóm, xin bày tỏ lòng biết ơn sâu sắc. Mỗi người trong nhóm đã đóng góp không ngừng, từ việc nghiên cứu, lên ý tưởng, đến việc thực hiện và hoàn thiện bài tập lớn. Tinh thần làm việc nhóm tích cực và sự đồng thuận giữa các thành viên đã tạo nên thành công của Bài tập lớn này.

Mỗi thành viên trong nhóm đều đóng góp một phần lớn vào thành công này, và chúng em tự hào về sự đồng đội và sự cam kết của mình.

Cuối cùng, chúng em muốn bày tỏ lòng biết ơn đến bản thân mình và mỗi người trong nhóm. Sự nỗ lực và quyết tâm cá nhân đã kết hợp tốt với tinh thần làm việc nhóm, hoàn thiện bài tập lớn.

Một lần nữa, xin chân thành cảm ơn thầy và mọi người trong nhóm. Sự hợp tác và nỗ lực của mọi người sẽ là nguồn động viên lớn trong các thử thách sắp tới.

TÓM TẮT NỘI DUNG BÀI TẬP LỚN

Hệ điều hành Android, với tầm quan trọng ngày càng tăng lên trong thế giới công nghiệp di động, đang đóng vai trò không thể phủ nhận trong việc thúc đẩy sự phát triển và đa dạng hóa của ứng dụng di động. Điều này tạo ra một tình huống hết sức thú vị và đầy thách thức cho những nghiên cứu liên quan đến hệ điều hành Android.

Lựa chọn đề tài về hệ điều hành Android đồng thời phản ánh sự nhận thức về vai trò quan trọng của Android trong cung cấp nền tảng cho các ứng dụng di động hiện đại. Sự tích hợp sâu rộng của Android với nhiều loại thiết bị và khả năng tương tác người dùng đã làm nền tảng này trở thành một nguồn cảm hứng lớn đối với các nhà phát triển và nghiên cứu viên.

Tại mặt ứng dụng, việc nghiên cứu về hệ điều hành Android mang lại nhiều cơ hội để khám phá và tối ưu hóa hiệu suất ứng dụng di động. Thách thức đặt ra là làm thế nào chúng ta có thể tối ưu hóa trải nghiệm người dùng và hiệu suất của ứng dụng trên các thiết bị sử dụng hệ điều hành Android khác nhau. Đồng thời, việc nghiên cứu này cũng sẽ đưa ra cái nhìn sâu sắc hơn về cách Android tương tác với phần cứng và ứng dụng thông qua các lớp trung gian.

Mặt kỹ thuật của đề tài này đặt ra một loạt các câu hỏi quan trọng về cách Android quản lý tài nguyên, bảo mật, và tương tác với các thành phần phần cứng. Nghiên cứu về cấu trúc nền tảng, cơ chế quản lý tác vụ, và các giao thức tương tác giữa hệ điều hành và ứng dụng sẽ đóng góp vào sự hiểu biết sâu sắc hơn về cách Android hoạt động.

Nội dung chính của đề tài mà nhóm chúng em muốn đề cập đến là cấu trúc hệ thống Android và quản lý hệ điều hành, tạo nền tảng cho việc hiểu rõ về môi trường phát triển ứng dụng di động. Mục 2.1 chi tiết về cấu trúc hệ thống Android, bao gồm Linux Kernel, Libraries, Android runtime, Application Framework, và ứng dụng. Điều này giúp người đọc hiểu về cơ sở hạ tầng cần thiết để chạy ứng dụng trên hệ điều hành này. Mục 2.2 chuyển đến quản lý file, trình bày hệ thống file và phương pháp tổ chức quản lý trong hệ điều hành Android, thúc đẩy sự hiểu biết về cách dữ liệu được tổ chức và quản lý trong môi trường di động. Trong khi đó, Mục 2.3 và Mục 2.4 tập trung vào quản lý tiến trình và tổ chức cơ sở dữ liệu. Người đọc sẽ được hướng dẫn về thuật toán quản lý tiến trình và trạng thái của chúng trong hệ điều hành Android, cũng như cách cơ sở dữ liệu được tổ chức và chia sẻ trong môi trường di động. Mục 2.5 mô tả quản lý bộ nhớ trên hệ điều hành Android, bao gồm các phương thức quản lý và khái niệm về bộ nhớ ảo. Cuối cùng, Mục 2.6 so sánh

dịch vụ của Android với dịch vụ của Windows, giúp người đọc có cái nhìn tổng quan về các tính năng và khả năng của hệ điều hành trong môi trường so sánh. Tóm lại, chương 2 cung cấp một cơ sở hiểu biết sâu rộng về cấu trúc và quản lý của hệ thống Android, hỗ trợ người đọc trong quá trình phát triển ứng dụng di động.

Kết quả đạt được sau cùng từ việc nghiên cứu là sự hiểu biết sâu rộng về cấu trúc hệ thống Android, quản lý file, tiến trình, cơ sở dữ liệu, quản lý bộ nhớ, và dịch vụ hệ điều hành. Điều này cung cấp nền tảng vững chắc cho việc phát triển ứng dụng di động, với khả năng đánh giá và áp dụng kiến thức vào thực tế.

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....	1
1.1 Khái quát lịch sử.....	1
1.2 Thống kê số lượng sử dụng hệ điều hành.....	4
1.3 Một số ứng dụng của hệ điều hành Android	4
1.4 Mục đích sử dụng hệ điều hành.....	5
CHƯƠNG 2. NỘI DUNG CHÍNH CỦA ĐỀ TÀI.....	7
2.1 Cấu hệ thống Android	7
2.1.1 Linux Kernel (Hạt nhân Linux)	7
2.1.2 Libraries (Thư viện)	8
2.1.3 Android runtime (Thời gian chạy Android).....	8
2.1.4 Application Framework (Khung ứng dụng).....	8
2.1.5 Applications (các ứng dụng).....	9
2.2 Quản lý file	9
2.2.1 Hệ thống file trên hệ điều hành android.....	9
2.2.2 Phương pháp tổ chức quản lý của hệ điều hành	13
2.3 Quản lý tiến trình.....	16
2.3.1 Các thuật toán quản lý tiến trình phổ biến trong hệ điều hành android.....	16
2.3.2 Các trạng thái của tiến trình	19
2.4 Tổ chức cơ sở dữ liệu và chia sẻ dữ liệu.....	21
2.4.1 Tổ chức cơ sở dữ liệu	21
2.4.2 Chia sẻ dữ liệu trong android	24
2.4.3 So sánh với DBMS của Window(MySQL)	26

2.5 Android và quản lý bộ nhớ.....	27
2.5.1 Các phương thức quản lý bộ nhớ.....	27
2.5.2 Bộ nhớ ảo.....	35
2.6 Các dịch vụ của hệ điều hành cung cấp.....	36
2.6.1 Khái niệm	36
2.6.2 So sánh dịch vụ của Android với dịch vụ của Windows	39
CHƯƠNG 3. KẾT LUẬN	41
3.1 Kết luận.....	41
3.2 Hướng phát triển trong tương lai	41

DANH MỤC HÌNH VẼ

Hình 1.1	HTC Hoặc T-Mobile G1, thiết bị thương mại đầu tiên chạy Android (2008)	1
Hình 1.2	Android 4.4 KitKat	2
Hình 1.3	Lịch sử Android là một bữa thức ăn đầy màu sắc	3
Hình 1.4	Một số ứng dụng của hệ điều hành Android	5
Hình 2.1	Cấu hệ thống Android	7
Hình 2.2	Quyền hạn các file trên thiết bị	12
Hình 2.3	Cây thư mục	12
Hình 2.4	Android Partition Details	13
Hình 2.5	Các tiến trình xếp theo mức độ ưu tiên	19
Hình 2.6	Hình ảnh biểu thị những dịch vụ của tiến trình	20
Hình 2.7	Hình ảnh biểu thị việc giới hạn những tiến trình nền	21
Hình 2.8	Mô hình client server trong android	22
Hình 2.9	Chia sẻ dữ liệu sử dụng Intent	24
Hình 2.10	Sơ đồ tổng quan về cách nhà cung cấp nội dung quản lý quyền truy cập vào bộ nhớ	25
Hình 2.11	Dùng Android Sharesheet để chia sẻ link tới bạn bè	26
Hình 2.12	Dùng Android Sharesheet để chia sẻ ảnh tới bạn bè	26
Hình 2.13	Cấp phát và thu hồi bộ nhớ ứng dụng	29
Hình 2.14	Chuyển đổi giữa các ứng dụng	31
Hình 2.15	Phân bổ bộ nhớ giữa các tiến trình hệ thống	32
Hình 2.16	Low-memory killer	34
Hình 2.17	Music player sử dụng Foreground Service để chơi nhạc	37
Hình 2.18	Báo thức cho người dùng sử dụng Background Service	38

DANH MỤC BẢNG BIỂU

Bảng 1.1	Thị phần người dùng theo năm	4
Bảng 2.1	Bảng so sánh quy tắc đặt tên với một số hệ điều hành khác . .	10
Bảng 2.2	Các kiểu file trên android	11
Bảng 2.3	Tổ chức quyền sở hữu và quyền hạn trên file	11
Bảng 2.4	Quy trình thuật toán Round Robin	17
Bảng 2.5	Bảng so sánh cách quản lí bộ nhớ giữa Android và iOS	35

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

Android là một hệ điều hành mã nguồn mở được phát triển dựa trên nền tảng Linux. Hệ điều hành này được thiết kế dành cho các thiết bị di động có màn hình cảm ứng như điện thoại thông minh và máy tính bảng. Android có khả năng tùy biến cao và là hệ điều hành di động phổ biến nhất hiện nay, chiếm thị phần sử dụng cao trên toàn thế giới. Kho ứng dụng Google Play với nhiều ứng dụng, trò chơi phong phú. Hệ điều hành Android cũng hỗ trợ nhiều dịch vụ như nhắn tin (SMS và MMS), trình duyệt web, lưu trữ (SQLite), kết nối (GSM, CDMA, Blue Tooth, Wi-Fi).

1.1 Khái quát lịch sử

Hệ điều hành Android là một trong những hệ điều hành di động phổ biến nhất trên thế giới, được phát triển bởi Google. Android đã trải qua nhiều phiên bản kể từ khi ra mắt lần đầu tiên vào năm 2008. Phiên bản mới nhất là Android 14, được phát hành vào tháng 7 năm 2023.

Dưới đây là tóm tắt về lịch sử hình thành của hệ điều hành Android:

1. Thập kỷ 2000: Andy Rubin, Rich Miner, Nick Sears và Chris White thành lập công ty Android Inc. vào tháng 10 năm 2003. Ban đầu, họ nhắm đến việc phát triển một hệ điều hành dành cho các thiết bị di động.
2. Mua bởi Google: Google mua lại Android Inc. vào tháng 8 năm 2005 và công bố sẽ phát triển hệ điều hành Android dựa trên mã nguồn mở.
3. Android 1.0 (2008): Phiên bản đầu tiên của Android được ra mắt cùng với điện thoại HTC Dream (gọi là T-Mobile G1 ở Mỹ). Nó hỗ trợ các tính năng cơ bản như duyệt web, email, và thư mục ảnh.



Hình 1.1: HTC Hoặc T-Mobile G1, thiết bị thương mại đầu tiên chạy Android (2008)

4. Android Market (2008): Google giới thiệu Android Market, một cửa hàng

ứng dụng cho phép người dùng tải về và cài đặt ứng dụng từ bên ngoài.

5. Phiên bản tiếp theo: Các phiên bản Android tiếp theo được phát hành với các tên gọi theo thứ tự là Cupcake (1.5), Donut (1.6), Eclair (2.0/2.1), Froyo (2.2), Gingerbread (2.3), và Honeycomb (3.0/3.1/3.2). Các phiên bản này cung cấp nhiều cải tiến về giao diện, hiệu suất và tính năng.

6. Ice Cream Sandwich và Jelly Bean (2011): Hai phiên bản này mang lại nhiều cải tiến đáng kể về giao diện người dùng và tính năng, bao gồm tích hợp NFC (Near Field Communication) và chức năng nhận diện giọng nói.

7. KitKat (2013): Android 4.4 KitKat được thiết kế để hoạt động trên nhiều loại thiết bị, bao gồm cả các thiết bị có cấu hình thấp hơn. Nó cũng đưa ra cải tiến về hiệu suất và quản lý pin.



Hình 1.2: Android 4.4 KitKat

8. Lollipop (2014): Android 5.0 Lollipop đưa ra một giao diện người dùng mới gọi là Material Design và cải tiến về tích hợp đám mây, bảo mật và hiệu suất.

9. Marshmallow (2015): Android 6.0 Marshmallow tập trung vào quản lý quyền truy cập ứng dụng, cải thiện tuổi thọ pin và tích hợp Google Now.

10. Nougat, Oreo và Pie (2016-2018): Các phiên bản này tiếp tục cải tiến giao diện, bảo mật và hiệu suất của hệ điều hành Android.

11. Android 10 (2019): Android 10 đưa ra nhiều tính năng mới như chế độ tối, quản lý quyền riêng tư tốt hơn và hỗ trợ 5G.

12. Android 11 (2020): Android 11 tập trung vào quản lý thông báo, quyền riêng

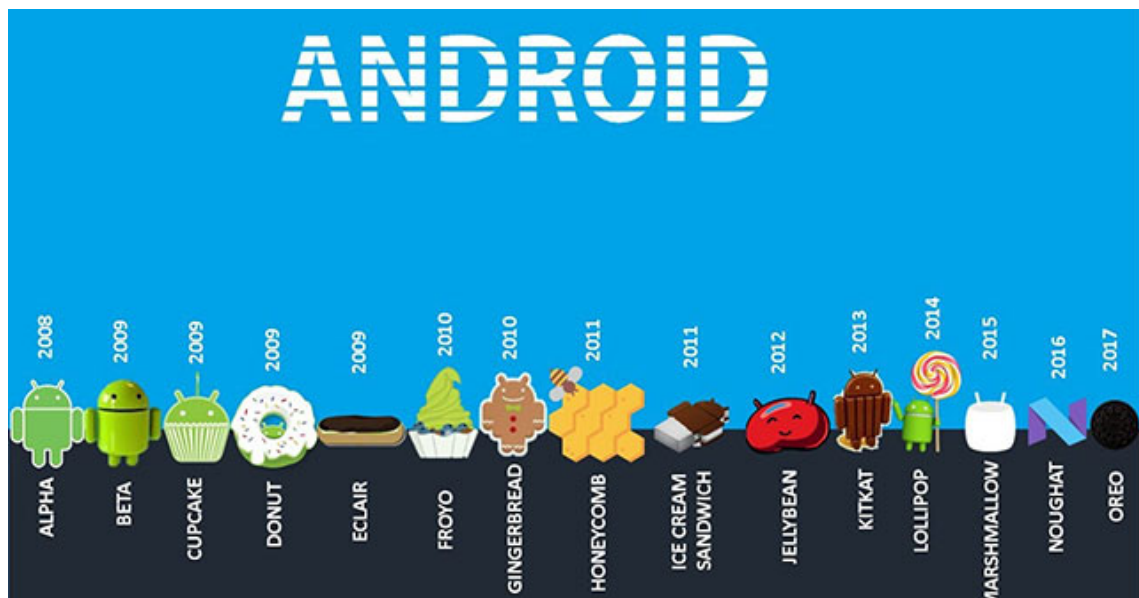
tư và giao diện người dùng.

13. Android 12 (2021): Android 12 mang đến một giao diện người dùng hoàn toàn mới với tên gọi "Material You" và nhiều tính năng cải tiến khác.

14. Android 13 (2022): Android 13 mang đến nhiều tính năng mới và cải tiến cho người dùng, bao gồm bảo mật, quyền riêng tư, và cá nhân hóa.

15. Android 14 (2023): Android 14 chủ yếu cải thiện trải nghiệm cho người dùng, bao gồm cải thiện hiệu năng, thời lượng pin, và cá nhân hóa.

Thông thường, các bản Android chính được phát hành mỗi năm 1 lần (dù không phải khi nào cũng vậy) cùng với đó là các bản cập nhật bảo mật giữa 2 lần phát hành. Thi thoảng Google cũng phát hành các bản cập nhật (.1, .2...) không thường xuyên. Họ cũng phát hành các bản cập nhật lớn, dù không lớn như bản đầy đủ, như cập nhật từ Android 8.0 lên Android 8.1 chẳng hạn.



Hình 1.3: Lịch sử Android là một bàn thức ăn đầy màu sắc

Đi cùng các bản Android là tên riêng, thường được dùng thay cho số phiên bản. Tên có khi được đặt theo món ăn, tráng miệng, móng tráng miệng, bánh ngọt... nhìn chung là cho vui chứ không có ý nghĩa cụ thể nào.

Thời gian cập nhật hệ thống của Android không theo 1 lịch trình cụ thể nào. Nhưng kể từ Ice Cream Sandwich thì OS được cập nhật theo năm.

Android tiếp tục phát triển và được sử dụng rộng rãi trên nhiều loại thiết bị di động, từ điện thoại thông minh đến máy tính bảng và các thiết bị nhúng khác nhau.

Android cũng đã trở thành một trong những hệ điều hành phổ biến cho các ứng dụng di động và đang có tầm ảnh hưởng toàn cầu đối với cuộc sống hàng ngày của

người dùng di động

1.2 Thống kê số lượng sử dụng hệ điều hành

Android là hệ điều hành phổ biến: Android là một trong những hệ điều hành di động phổ biến nhất trên thế giới, được sử dụng trên hàng tỷ thiết bị. Hiểu rõ về Android giúp bạn nắm bắt được một phần quan trọng của hệ thống di động hiện đại.

Theo số liệu thống kê của StatCounter, tính đến tháng 10 năm 2023, Android chiếm 85,2% thị phần hệ điều hành di động trên toàn thế giới. Đứng thứ hai là iOS với 14,8% thị phần. Cụ thể, số lượng người dùng Android trên toàn thế giới là 3,82 tỷ người, trong khi số lượng người dùng iOS là 630 triệu người. Tại Việt Nam, Android cũng là hệ điều hành phổ biến nhất với thị phần hơn 60%. iOS đứng thứ hai với thị phần hơn 38%. Dưới đây là thống kê số lượng sử dụng hệ điều hành Android trên toàn thế giới theo từng năm:

Năm	Thị phần (%)	Số lượng người dùng (tỷ)
2020	74.4	2.81
2021	78.8	3.29
2022	82.2	3.62
2023	85.2	3.82

Bảng 1.1: Thị phần người dùng theo năm

Sự phổ biến của Android có thể được lý giải bởi vì Android là hệ điều hành miễn phí và mã nguồn mở. Điều này cho phép các nhà sản xuất thiết bị di động có thể tùy chỉnh Android theo nhu cầu của mình, từ đó tạo ra nhiều lựa chọn đa dạng cho người dùng. Android có sẵn trên nhiều phân khúc thiết bị, từ điện thoại giá rẻ đến điện thoại cao cấp. Điều này giúp Android tiếp cận được với nhiều đối tượng người dùng hơn. Android có kho ứng dụng khổng lồ. Tính đến tháng 10 năm 2023, Google Play Store có hơn 3,5 triệu ứng dụng và game.

Với những ưu điểm trên, Android được dự đoán sẽ tiếp tục duy trì vị thế là hệ điều hành di động phổ biến nhất trên thế giới trong nhiều năm tới.

1.3 Một số ứng dụng của hệ điều hành Android

Các ứng dụng di động: Android là nền tảng hàng đầu cho các ứng dụng di động. Có hàng triệu ứng dụng có sẵn trên Google Play Store, bao gồm các ứng dụng phổ biến như Facebook, Instagram, WhatsApp, và TikTok.

Các thiết bị đeo được: Android cũng được sử dụng trên các thiết bị đeo được như đồng hồ thông minh và vòng đeo tay thông minh. Các thiết bị này có thể được sử dụng để theo dõi sức khỏe, theo dõi hoạt động, và trả lời cuộc gọi.



Hình 1.4: Một số ứng dụng của hệ điều hành Android

Các thiết bị ô tô: Android cũng đang được sử dụng trên các thiết bị ô tô. Các hệ thống thông tin giải trí trên ô tô sử dụng Android để cung cấp các tính năng như định vị GPS, phát nhạc, và phát video.

1.4 Mục đích sử dụng hệ điều hành

Hệ điều hành Android, phát triển bởi tập đoàn Google, là một hệ điều hành di động phổ biến trên toàn cầu, được thiết kế đặc biệt để hoạt động trên nhiều loại thiết bị, bao gồm điện thoại thông minh và máy tính bảng. Tính linh hoạt và khả năng tương tác cao của Android đã đưa ra một loạt các cơ hội và trải nghiệm đa dạng cho người dùng và nhà phát triển.

Một trong những mục đích chính của hệ điều hành Android là cung cấp một môi trường đa dạng cho việc chạy ứng dụng di động. Với hàng ngàn ứng dụng có sẵn trên Google Play Store, người dùng có thể trải nghiệm độ phong phú từ ứng dụng xã hội, trò chơi, ứng dụng productivity, cho đến các ứng dụng độc đáo dành cho giải trí. Khả năng tương thích và sự đa dạng trong ứng dụng là một trong những điểm mạnh của hệ điều hành này.

Bên cạnh đó, Android cũng tập trung vào việc cung cấp các tính năng liên lạc và gọi điện thoại tiện ích. Người dùng có thể dễ dàng thực hiện cuộc gọi điện thoại, gửi tin nhắn văn bản, và thậm chí tham gia các cuộc trò chuyện trực tuyến thông qua các ứng dụng như WhatsApp, Messenger, mang lại sự tiện lợi và liên kết xã

hội.

Truy cập Internet và thông tin cũng là một khía cạnh quan trọng của trải nghiệm Android. Người dùng có thể duyệt web, xem email, và tìm kiếm thông tin trên mạng một cách thuận tiện, nhờ vào tích hợp các ứng dụng và trình duyệt tiên tiến.

Quản lý thông tin cá nhân là một ưu tiên, và Android cung cấp các ứng dụng quản lý liên lạc và lịch để giúp người dùng tổ chức danh bạ, lịch làm việc và các thông tin cá nhân khác một cách hiệu quả.

Giải trí và đa phương tiện đều được Android hỗ trợ đầy đủ. Người dùng có thể thưởng thức âm nhạc, xem video, chơi trò chơi và quản lý tệp đa phương tiện một cách thuận lợi, biến thiết bị Android thành một trung tâm giải trí di động.

Tính năng định vị GPS tích hợp của Android cho phép người dùng sử dụng dịch vụ bản đồ, điều hướng và tìm kiếm địa điểm một cách chính xác, mang lại trải nghiệm điều hướng toàn diện.

Android không chỉ dừng lại ở đó, nó còn mở rộng khả năng thông qua tính năng tùy chỉnh và mở rộng. Người dùng có thể tùy chỉnh giao diện và cài đặt hệ thống theo ý muốn cá nhân, tạo ra một trải nghiệm sử dụng độc đáo. Việc cài đặt ứng dụng từ các nguồn bên ngoài Google Play Store cũng mở ra nhiều cơ hội mới để mở rộng chức năng của thiết bị.

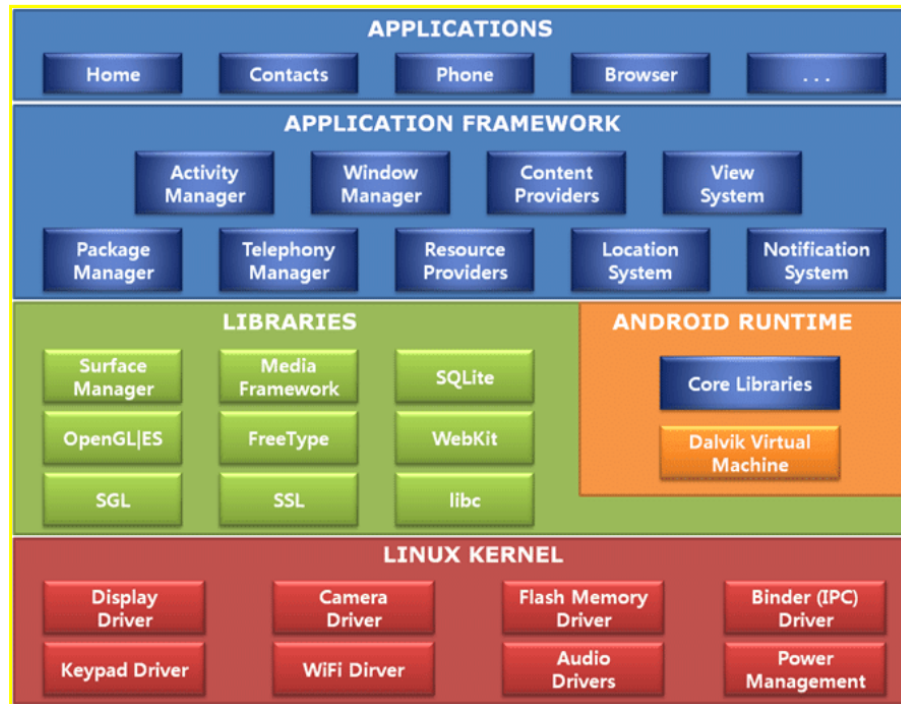
Đối với những nhà phát triển, Android không chỉ là một nền tảng mà còn là một môi trường phát triển ứng dụng mạnh mẽ. Điều này giúp họ xây dựng và phát triển những ứng dụng di động sáng tạo, đáp ứng nhu cầu đa dạng của cộng đồng người dùng Android trên toàn thế giới.

Tóm lại, Android có nhiều mục đích sử dụng, từ việc cung cấp một trải nghiệm di động toàn diện cho người dùng đến việc tạo ra một nền tảng phát triển đa dạng cho các nhà phát triển ứng dụng di động.

CHƯƠNG 2. NỘI DUNG CHÍNH CỦA ĐỀ TÀI

2.1 Cấu hệ thống Android

Android là một hệ điều hành mã nguồn mở dựa trên nền tảng Linux và có thể sử dụng cho nhiều thiết bị khác nhau. Là một tập hợp các thành phần phần mềm được chia thành năm phần và bốn lớp chính như sơ đồ bên dưới:



Hình 2.1: Cấu hệ thống Android

2.1.1 Linux Kernel (Hạt nhân Linux)

Là tầng dưới cùng và là nền tảng trong kiến trúc của hệ điều hành Android. Tuy nhiên, nhân Android và nhân Linux không hoàn toàn giống nhau. Android sử dụng phiên bản sửa đổi của nhân Linux để hỗ trợ kiến trúc thiết bị, vì Android là thiết bị di động chứ không phải PC. Đây là Linux 3.6 với khoảng 115 bản vá. Tầng này cung cấp mức độ trừu tượng giữa phần cứng của thiết bị, và nó chứa tất cả các trình điều khiển phần cứng thiết yếu như máy ảnh, bàn phím, màn hình,.. Nó giúp Android trong các dịch vụ và chức năng hệ thống cốt lõi như phân luồng, quản lý bộ nhớ cấp thấp, bảo mật, quản lý quy trình, mạng và trình điều khiển phần cứng. Ngoài ra, hạt nhân (kernel) xử lý tất cả những thứ mà Linux có thể làm tốt nhất như kết nối mạng và một loạt các trình điều khiển thiết bị, giúp làm giảm bớt khó khăn khi giao tiếp với phần cứng ngoại vi.

2.1.2 Libraries (Thư viện)

Thư viện bao gồm nhiều thư viện lõi C/C++ và thư viện dựa trên Java để cung cấp hỗ trợ cho việc phát triển Android. Các thành phần quan trọng của thư viện C/C++ như media (Hỗ trợ phát và ghi âm thanh, video), graphics (Cung cấp các chức năng đồ họa 2D và 3D) và surface Manager (Quản lý quyền truy cập vào hệ thống con hiển thị). Ngoài ra thư viện Java bao gồm OpenGL (Thư viện đồ họa 2D và 3D, có API đa ngôn ngữ và đa nền tảng), SQLite (Hỗ trợ cơ sở dữ liệu), FreeType (Hỗ trợ phông chữ), Web-Kit (Công cụ trình duyệt web nguồn mở, giúp hiển thị nội dung web và tối ưu hóa quá trình tải trang) và SSL (Lớp cổng bảo mật) là Công nghệ bảo mật để thiết lập liên kết được mã hóa giữa máy chủ web và trình duyệt web, đảm bảo an toàn khi truyền dữ liệu.

Ví dụ: bạn có thể truy cập OpenGL ES thông qua Java của khung công tác Android. API OpenGL để thêm hỗ trợ vẽ và thao tác đồ họa 2D và 3D trong ứng dụng của bạn

2.1.3 Android runtime (Thời gian chạy Android)

Với những thiết bị chạy Android phiên bản 5.0 (API cấp 21) trở lên, mỗi ứng dụng sẽ có quy trình chạy riêng và với thời gian chạy Android (Android Runtime - ART) riêng. Tầng này cung cấp một thành phần chính được gọi là máy ảo Dalvik (Dalvik Virtual Machine-Dalvik VM), đây là một loại máy ảo Java được thiết kế và tối ưu hóa đặc biệt cho Android. Dalvik VM sử dụng các tính năng cốt lõi của Linux như quản lý bộ nhớ và đa lượng, cho phép mọi ứng dụng Android chạy trong quy trình riêng với phiên bản Dalvik VM của riêng nó. Một số tính năng chính của ART như là biên dịch trước thời gian (AOT) và đúng lúc (JIT); thu gom rác tối ưu hóa (GC); trên Android 9 (API cấp 28) trở lên, chuyển đổi tệp định dạng Dalvik Executable (DEX) của gói ứng dụng thành mã máy nhỏ gọn hơn; hỗ trợ debug tốt hơn bao gồm trình biên dịch lấy mẫu chuyên dụng (profiler), các ngoại lệ chẩn đoán chi tiết và báo cáo sự cố cũng như khả năng thiết lập các điểm theo dõi để giám sát các trường cụ thể. Ngoài ra, ART cũng cung cấp một bộ thư viện cốt lõi, cho phép các nhà lập trình viết các ứng dụng Android bằng ngôn ngữ lập trình Java tiêu chuẩn.

2.1.4 Application Framework (Khung ứng dụng)

Lớp Khung ứng dụng chứa giao diện API và cung cấp nhiều dịch vụ cấp cao hơn cho các ứng dụng dưới dạng các lớp Java. Các nhà phát triển ứng dụng được phép sử dụng các dịch vụ này trong ứng dụng của họ. Khung Android bao gồm các dịch vụ chính là trình quản lý hoạt động (Kiểm soát tất cả các khía cạnh của vòng đời ứng dụng và ngăn xếp hoạt động), nhà cung cấp nội dung (Cho phép ứng dụng

xuất bản và chia sẻ dữ liệu với các ứng dụng khác.), trình quản lý tài nguyên (Cung cấp quyền truy cập vào các tài nguyên được nhưng không có mã như chuỗi, cài đặt màu và bố cục giao diện người dùng), trình quản lý thông báo (Cho phép ứng dụng hiển thị cảnh báo và thông báo cho người dùng), hệ thống quan sát (Một tập hợp các view có thể mở rộng được sử dụng để tạo giao diện người dùng ứng dụng).

2.1.5 Applications (các ứng dụng)

Các ứng dụng được cài đặt sẵn như Email, Nhắn tin SMS, Lịch, Trình duyệt Internet, Danh bạ, Máy ảnh, Thư viện, v.v. và các ứng dụng của bên thứ ba được tải xuống từ cửa hàng Play như ứng dụng trò chuyện, trò chơi, v.v. sẽ chỉ được cài đặt trên lớp này. Nó chạy trong thời gian chạy Android với sự trợ giúp của các lớp và dịch vụ do khung ứng dụng cung cấp. Các ứng dụng hệ thống hoạt động vừa là ứng dụng cho người dùng vừa cung cấp các chức năng chính mà nhà phát triển có thể truy cập từ ứng dụng của riêng họ. Ví dụ: nếu muốn ứng dụng của mình gửi tin nhắn SMS, bạn không cần phải tự mình xây dựng chức năng đó. Thay vào đó, bạn có thể gọi bất kỳ ứng dụng SMS nào đã được cài đặt để gửi tin nhắn đến người nhận mà bạn chỉ định.

Tổng kết, vì Android cung cấp Máy ảo Dalvik, đảm bảo rằng Android là nền tảng Java nên các dịch vụ miễn phí được cung cấp nhằm khuyến khích các nhà phát triển sử dụng nền tảng Java. Ngoài ra, các thư viện khác nhau cần thiết để phát triển ứng dụng cho Android đều có sẵn, do đó, nền tảng này có vẻ hấp dẫn hơn rất nhiều so với ios. Giá xuất bản ứng dụng có thể là một vấn đề đối với một số nhà phát triển. Nếu đúng như vậy, Android sẽ được ưu ái hơn vì tốn số tiền thấp hơn đáng kể so với mức mà Apple yêu cầu. Hơn nữa với mức tiền đó, ứng dụng có thể được tiếp thị trên bất kỳ nền tảng nào mà nhà phát triển mong muốn vì Android không hạn chế chỉ xuất bản trên Play Store.

Tóm lại, có thể thấy các nhà phát triển ứng dụng ưa thích sử dụng Android hơn iOS.

2.2 Quản lý file

2.2.1 Hệ thống file trên hệ điều hành android

a, Tổng quan về hệ thống file trên android

Trong android các file được tổ chức thành các thư mục, theo mô hình phân cấp. Tham chiếu đến một file bằng tên đường dẫn. Các câu lệnh thao tác file cho phép thực hiện các chức năng như dịch chuyển, sao chép toàn bộ thư mục cùng với các thư mục con chứa trong nó... Về tên file thì có thể sử dụng các ký tự dấu gạch dưới, chữ số, dấu chấm, và dấu phẩy để đặt tên file nhưng không được bắt đầu một tên file bằng dấu chấm hay chữ số. Những ký tự khác như “/”, “?”, “*”, là ký tự đặc biệt

được dành riêng cho hệ thống. Chiều dài của tên file có thể tới 256 kí tự. Ngoài ra trong hệ điều hành android có sự phân biệt tên chữ hoa và chữ thường, điều đó có nghĩa là trong cùng 1 thư mục có thể tồn tại những file có tên là File, FILE, file,... và chúng là những file khác nhau.

So sánh quy tắc đặt tên với một số hệ điều hành khác:

Hệ điều hành	Độ dài tối đa	Phân biệt chữ hoa chữ thường	Cho phép sử dụng dấu cách	Các ký tự cấm
MS-DOS	8 cho tên file, 3 cho mở rộng	không	không	Bắt đầu bằng chữ cái hoặc số, không được chứa các ký tự / \ [] : ; = , ? @
Windows NT FAT	255 ký tự cho cả tên file và đường dẫn	không	có	Bắt đầu bằng chữ cái hoặc số, không được chứa các ký tự / \ [] : ; = , ? @
Windows NT NTFS	255	không	có	Không được chứa các ký tự / \ < > * :
Linux (EXT3)	256	có	Có (nếu tên file chứa trong ngoặc kép)	Không được chứa các ký tự ! @ \$ % * () [] " ' / \ ; < > ' :
Android	256	có	Có (nếu tên file chứa trong ngoặc kép)	Không được chứa các ký tự \ : * ? "

Bảng 2.1: Bảng so sánh quy tắc đặt tên với một số hệ điều hành khác

Tất cả các file trong android có chung cấu trúc vật lý là chuỗi các byte (byte stream). Cấu trúc thống nhất này cho phép android áp dụng khái niệm file cho mọi thành phần dữ liệu trong hệ thống. Thư mục cũng như các thiết bị được xem như file. Chính việc xem mọi thứ như các file cho phép android quản lý và chuyển đổi dữ liệu một cách dễ dàng. Một thư mục chứa các thông tin về thư mục, được tổ chức theo một định dạng đặc biệt. Các thành phần được xem như cá file, chúng được phân biệt dựa trên kiểu file: ordinary file, directory file, character device file, và block device file.

b, Các kiểu file trên android

Trong nhiều hệ điều hành như window, người ta phân biệt rõ file (tập tin) và folder (hay directory : thư mục) là 2 thành phần khác hẳn nhau. Tuy nhiên trên hệ

điều hành android (cũng như Linux) thì coi directory cũng là file và nó là một loại file đặc biệt. Thực tế còn một số loại file nữa có thể liệt kê theo bảng sau:

Chữ cái biểu diễn	Kiểu file
d	Thư mục (Directory)
b	File kiểu khối (block-type special file)
c	File kiểu ký tự (character-type special file)
l	Liên kết tượng trưng (symbolic link)
p	File đường ống (pipe)
s	Socket
-	File bình thường (regular file)

Bảng 2.2: Các kiểu file trên android

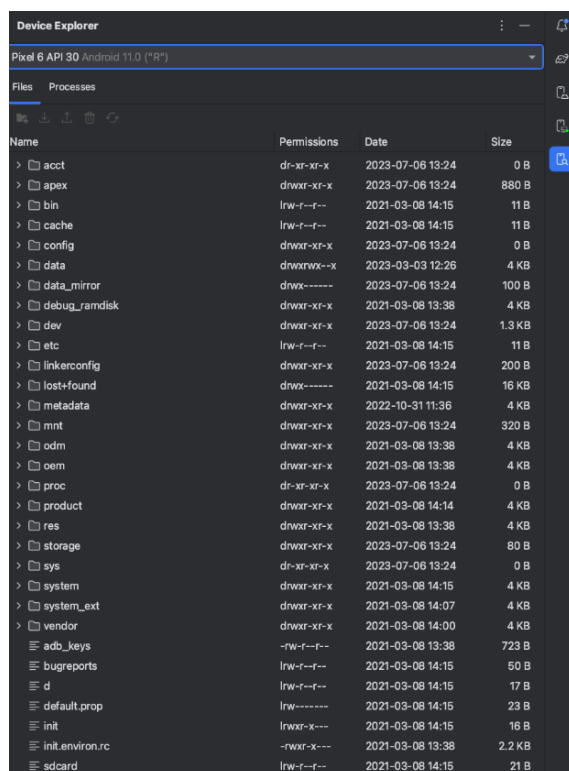
Tổ chức quyền sở hữu và quyền hạn trên file: Tương tự trên hệ thống Linux, trên hệ điều hành android, một file có thể liên kết với một người sử dụng và một nhóm người sử dụng. Sự liên kết đó là một tập hợp các quyền hạn truy cập bao gồm quyền được phép đọc (read), được phép ghi (write) và được phép thực thi (execute). Cụ thể là một file sẽ có những quyền hạn tương ứng với 9 ký tự theo mẫu sau: Với ký tự r w x nghĩa là quyền tương ứng với ký tự viết tắt đó, nghĩa là không có quyền hạn đó.

Owner	Owner Group	Other
r/ - w/ - x/-	r/- w/ -x/-	r/- w/ -x/-

Bảng 2.3: Tổ chức quyền sở hữu và quyền hạn trên file

- * 3 ký tự đầu tiên là quyền hạn chủ nhân file
- * 3 ký tự giữa là quyền hạn của nhóm tài khoản sở hữu file
- * 3 ký tự cuối là quyền hạn của những người không thuộc nhóm sở hữu file.

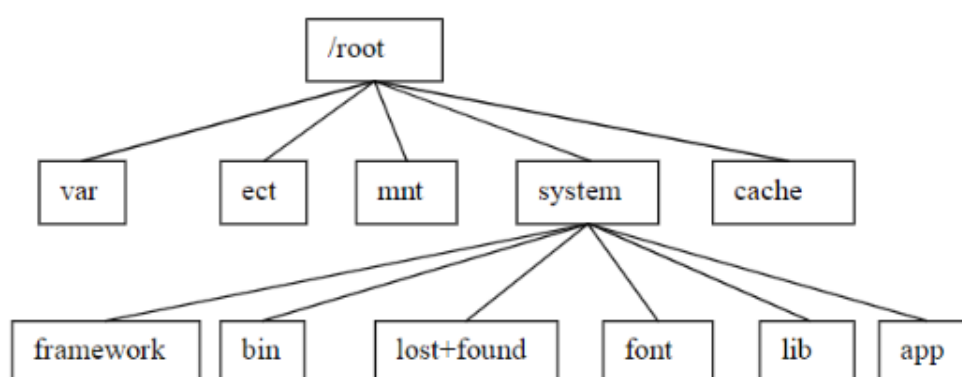
Trên hệ thống android, để biết xem được quyền hạn đó, ta có thể sử dụng câu lệnh ls-l-d.



Name	Permissions	Date	Size
> acct	dr-xr-xr-x	2023-07-06 13:24	0 B
> apex	drwxr-xr-x	2023-07-06 13:24	880 B
> bin	lrwxr--r--	2021-03-08 14:15	11 B
> cache	lrwxr--r--	2021-03-08 14:15	11 B
> config	drwxr-xr-x	2023-07-06 13:24	0 B
> data	drwxrwx--x	2023-03-03 12:26	4 KB
> data_mirror	drwx-----	2023-07-06 13:24	100 B
> debug_ramdisk	drwxr-xr-x	2021-03-08 13:38	4 KB
> dev	drwxr-xr-x	2023-07-06 13:24	1.3 KB
> etc	lrwxr--r--	2021-03-08 14:15	11 B
> linkerconfig	drwxr-xr-x	2023-07-06 13:24	200 B
> lost+found	drwx-----	2021-03-08 14:15	16 KB
> metadata	drwxr-xr-x	2022-10-31 11:36	4 KB
> mnt	drwxr-xr-x	2023-07-06 13:24	320 B
> odm	drwxr-xr-x	2021-03-08 13:38	4 KB
> oem	drwxr-xr-x	2021-03-08 13:38	4 KB
> proc	dr-xr-xr-x	2023-07-06 13:24	0 B
> product	drwxr-xr-x	2021-03-08 14:14	4 KB
> res	drwxr-xr-x	2021-03-08 13:38	4 KB
> storage	drwxr-xr-x	2023-07-06 13:24	80 B
> sys	dr-xr-xr-x	2023-07-06 13:24	0 B
> system	drwxr-xr-x	2021-03-08 14:15	4 KB
> system_ext	drwxr-xr-x	2021-03-08 14:07	4 KB
> vendor	drwxr-xr-x	2021-03-08 14:00	4 KB
≡ adb_keys	-rw-r--r--	2021-03-08 13:38	723 B
≡ bugreports	lrwxr--r--	2021-03-08 14:15	50 B
≡ d	lrwxr--r--	2021-03-08 14:15	17 B
≡ default.prop	lrw-----	2021-03-08 14:15	23 B
≡ init	lrwxr-x---	2021-03-08 14:15	16 B
≡ init.environ.rc	-rwxr-x---	2021-03-08 13:38	2.2 KB
≡ sdcard	lrwxr--r--	2021-03-08 14:15	21 B

Hình 2.2: Quyền hạn các file trên thiết bị

Cây thư mục trên hệ điều hành android: Thư mục (hay có thể gọi là file) root là thư mục gốc của tất cả các file thư mục còn lại. Dưới nó có chứa một số file thư mục hệ thống. Mỗi thư mục (trừ thư mục root) đều có một thư mục cha chứa nó, bản thân nó cũng có thể có nhiều file thư mục con. Cấu trúc đó có thể mô tả bằng một cây thư mục có dạng như sau:



Hình 2.3: Cây thư mục

Giới thiệu một vài thư mục tiêu biểu:

/root : Là thư mục gốc. Là thư mục duy nhất không có thư mục cha.

/mnt : Thư mục chứa thiết bị lưu động (removeable).

/system : Chứa những thành phần cơ bản nhất của hệ thống.

/ect : Chứa những file cấu hình của hệ thống, nó cực kì quan trọng vì sự hoạt động của hệ thống đều bị chi phối ở những file cấu hình này.

/system/lost+found : Chứa những tập tin bị mất lúc khởi động máy.

/system.font : Chứa những font chữ hiển thị được.

/system.lib : Chứa các thư viện để các phần mềm hoạt động (các phần mềm viết bằng ngôn ngữ java).

/system/app : Chứa các file apk của phần mềm (các file cài đặt ứng dụng, kiểu như MSI trong window hay dev trong Linux).

/system/bin : Chứa các chương trình nội trú của hệ thống.

...

2.2.2 Phương pháp tổ chức quản lý của hệ điều hành

a, Phương pháp tổ chức quản lý

Hệ thống tệp Android được chia thành nhiều phân vùng, mỗi phân vùng có mục đích cụ thể. Ngoài ra còn có sự khác biệt giữa bộ nhớ trong và bộ nhớ ngoài trên thiết bị Android. Có sáu phân vùng trong bộ nhớ trong là /boot, /system, /recovery, /data, /cache và /misc. Hai phân vùng trong bộ nhớ ngoài là /sdcard và /sd-ext. Mỗi phân vùng này có bộ quyền và hạn chế riêng.

Android Partition Details



Hình 2.4: Android Partition Details

/boot

Phân vùng /boot bao gồm hình ảnh khởi động và kernel cần thiết để khởi động hệ điều hành Android. Thiết bị không thể khởi động nếu không có phân vùng /boot. Phân vùng này ở chế độ chỉ đọc và người dùng không thể sửa đổi. Để sửa đổi nó, thiết bị phải được root và phải cài đặt recovery tùy chỉnh. Khi thiết bị đã được root, phân vùng /boot có thể được truy cập và sửa đổi bằng trình quản lý tệp hoặc giao diện dòng lệnh. Bắt buộc phải giữ nguyên các tệp trong phân vùng này để hệ thống có thể khởi động chính xác mà không gặp vấn đề gì. Việc xóa phân vùng /boot (nếu cần thiết, mặc dù không khuyến khích) phải luôn được thực hiện kèm theo cài đặt

một ROM mới, cuối cùng sẽ tạo ra một phân vùng /boot mới.

/system

Phân vùng /system là một thành phần quan trọng của Hệ thống tệp Android vì nó chứa toàn bộ hệ điều hành Android, bao gồm cả các thành phần GUI cũng như tất cả các ứng dụng được cài đặt sẵn trên thiết bị. Trong phân vùng /system chứa các thư mục và tệp như app, bin, fonts, framework, v.v. . . Đây là phân vùng chỉ đọc và chỉ có thể được sửa đổi bằng thiết bị đã root để xóa mọi ứng dụng được cài đặt sẵn hoặc tùy chỉnh HĐH Android. Việc xóa phân vùng này sẽ xóa tất cả các ứng dụng hệ thống được cài đặt sẵn và chỉ nên thực hiện khi người dùng muốn cài đặt ROM tùy chỉnh hoặc khắc phục sự cố với HĐH.

/recovery

Phân vùng /recovery là một phân vùng nhỏ bao gồm image khôi phục để khôi phục thiết bị nếu xảy ra lỗi phần mềm. Đây lại là một thư mục chỉ đọc khác và không thể sửa đổi được. Trong phân vùng này chứa hình ảnh khôi phục và các tập lệnh để tự động khởi động thiết bị vào chế độ khôi phục (được sử dụng để bảo trì hệ thống như cài đặt bản cập nhật phần mềm, sao lưu hoặc khôi phục dữ liệu hoặc thực hiện khôi phục cài đặt gốc) và để cập nhật hệ điều hành Android.

/data

Phân vùng /data chứa tất cả dữ liệu người dùng và ứng dụng. Dữ liệu người dùng có thể ở dạng ảnh, video, nhạc, danh bạ, tùy chọn, cài đặt, ứng dụng Android, v.v. đã được cài đặt vào thiết bị. Đây là một phân vùng có khả năng đọc và ghi, có thể được sửa đổi bởi dữ liệu từ các ứng dụng đã cài đặt. Việc sao lưu định kỳ các tệp tin trong phân vùng này là quan trọng. Toàn bộ phân vùng này sẽ bị xóa nếu thực hiện việc đặt lại cài đặt gốc, đưa thiết bị về trạng thái ban đầu giống như một thiết bị mới mua.

/cache

Phân vùng /cache chứa tất cả các tệp tạm thời được hệ điều hành và ứng dụng sử dụng. Đây là phân vùng đọc-ghi và cũng có thể được hệ thống hoặc người dùng xóa. Các tệp có trong này có thể được truy cập nhanh chóng mà không cần phải truy xuất chúng từ bộ nhớ trong hoặc thẻ SD bên ngoài. Làm như vậy sẽ cải thiện hiệu suất và tốc độ của thiết bị bằng cách giảm thiểu thời gian cần thiết để truy cập vào dữ liệu được sử dụng thường xuyên. Việc xóa phân vùng /cache có thể giúp giải phóng một phần dung lượng trong thiết bị và dữ liệu được lưu trong bộ nhớ đệm bắt đầu xuất hiện trở lại trong khi chúng ta sử dụng thiết bị.

/misc

Phân vùng /misc chứa một tệp có tên là misc.img bao gồm thông tin cần thiết dành riêng cho thiết bị, chẳng hạn như trạng thái bộ nạp khởi động để biết thiết bị đang ở trạng thái khoá hay mở khoá. Tệp misc.img cũng chứa các lỗi xảy ra trong quá trình khởi động. Phân vùng này cũng lưu trữ dữ liệu cho các bản cập nhật qua mạng (OTA), thông báo xem thiết bị có được cập nhật hay cần nâng cấp hay không. Việc thiếu phân vùng /misc có thể dẫn đến hoạt động không đúng của một số tính năng của thiết bị.

/sdcard

phân vùng /sdcard trong Hệ thống Tệp Android thường là một thiết bị lưu trữ ngoại vi ảo mà cả người dùng và hệ thống Android có thể truy cập. Thiết bị lưu trữ ngoại vi ảo cho phép ứng dụng truy cập một phần của bộ nhớ trong của thiết bị như là bộ nhớ ngoại vi và cung cấp tính tương thích với các ứng dụng cũ không thể truy cập trực tiếp vào bộ nhớ trong. Phân vùng này cho phép người dùng dễ dàng chuyển các tập tin giữa thiết bị và máy tính hoặc thiết bị Android. Có thể kết nối thiết bị này với PC thông qua Giao thức truyền phương tiện (MTP) để truyền tệp của mình. Ngoài ra, đầu đọc thẻ cũng có thể được sử dụng cho mục đích tương tự. Nhiều ứng dụng Android được thiết kế để lưu nội dung do người dùng tạo vào phân vùng /sdcard theo mặc định, giúp người dùng truy cập và quản lý tệp của họ dễ dàng hơn. Việc xóa mọi dữ liệu khỏi phân vùng này sẽ không gây ra bất kỳ sự cố nào cho thiết bị.

/sd-ext

Phân vùng /sd-ext là phân vùng tùy chọn có thể được sử dụng để cung cấp thêm dung lượng lưu trữ cho thiết bị Android nếu bộ nhớ trong có hạn, nó thường được sử dụng cùng với phân vùng /data. Phân vùng này cho phép người dùng cài đặt nhiều ứng dụng hơn trên thiết bị trong trường hợp dung lượng bộ nhớ trong bị hạn chế, bởi vì phân vùng này có khả năng nhanh chóng bị đầy với dữ liệu ứng dụng. Ngoài ra, phân vùng /sd-ext thường không được tất cả các thiết bị Android hỗ trợ và thậm chí còn yêu cầu các bước thiết lập bổ sung trong một số trường hợp. Nó được gắn dưới dạng /mnt/sd-ext hoặc /storage/sdcard1.

Làm cách nào để truy cập tệp gốc trong Android?

Để truy cập các tệp gốc trong Hệ thống tệp Android, cần bật quyền truy cập root trên thiết bị Android. Điều này cung cấp các đặc quyền quản trị để truy cập và sửa đổi các tệp hệ thống cũng như cài đặt hệ thống.

Đây là cách có thể truy cập các tệp gốc trong Hệ thống tệp Android:

Bước 1: Root thiết bị Android

Bước 2: Cài đặt trình quản lý tệp gốc

Bước 3: Cấp quyền truy cập root cho trình quản lý tệp

Bước 4: Điều hướng đến thư mục gốc

Bây giờ chúng ta đã điều hướng đến thư mục gốc, chúng ta có thể truy cập và sửa đổi các tệp và thư mục hệ thống theo nhu cầu của mình.

Tuy nhiên, việc root thiết bị Android có thể nguy hiểm vì một số lý do được liệt kê ở đây:

- Việc root khiến thiết bị gặp rủi ro bảo mật khi cho phép phần mềm độc hại truy cập vào các tệp và dữ liệu ngoài tầm hiểu biết của người dùng.
- Nó làm mất hiệu lực bảo hành vì nhà sản xuất không khuyến khích sửa đổi phần mềm của thiết bị.
- Nó có thể dẫn đến việc thiết bị Android bị brick, có nghĩa là thiết bị có thể không hoạt động được một phần hoặc hoàn toàn.

Hơn nữa, các ứng dụng ngân hàng có thể phát hiện các thiết bị đã root và ngăn chặn việc sử dụng chúng trên các thiết bị đó vì nó làm tổn hại đến bảo mật và cho phép đánh cắp thông tin nhạy cảm như thông tin xác thực ngân hàng dễ dàng hơn.

2.3 Quản lý tiến trình

2.3.1 Các thuật toán quản lý tiến trình phổ biến trong hệ điều hành android

2.3.1.1 Quản lý theo kiểu " Công bằng "

Thuật Toán RR(ROUND ROBIN):

a, Giới thiệu:

Round Robin là một trong những thuật toán lập lịch tiến trình phổ biến nhất trong hệ điều hành Android và nhiều hệ điều hành khác, cho phép các tiến trình sử dụng CPU một cách công bằng.

b, Nguyên Tắc Hoạt Động :

Cơ chế : dựa trên nguyên tắc chia sẻ thời gian (time-sharing). Khi một tiến trình được tạo hoặc cần được thực thi, nó được thêm vào cuối hàng đợi (queue) chờ đợi. Trong thuật toán này, mọi quy trình được thực thi theo một cách tuần hoàn.

Time Slice (Quantum): Mỗi tiến trình trong hàng đợi được gán cho CPU một time slice cố định, được gọi là "quantum" hoặc lượng tử, lát cắt thời gian. Quantum thường có độ dài ngắn, chẳng hạn 10-100 mili giây. Nếu quá trình thực thi được hoàn thành trong thời gian đó thì quá trình sẽ kết thúc nếu không quá trình sẽ quay

P1	P2	P3	P4
P2	P3	P4	P1
P3	P4	P1	P2
P4	P1	P2	P3

Bảng 2.4: Quy trình thuật toán Round Robin

lại hàng đợi sẵn sàng và đợi lượt tiếp theo hoàn thành việc thực hiện.

c, Ưu và nhược điểm của thuật toán này :

Ưu điểm:

Thuật toán không phụ thuộc vào Burst Time, điều này làm cho nó thực hiện được trong hệ thống mà không cần thông tin chi tiết về thời gian thực hiện của các tiến trình. Vấn đề chết đói và hiệu ứng đoàn xe không xuất hiện, đảm bảo tất cả các công việc đều có cơ hội sử dụng CPU.

Nhược điểm:

Thứ nhất, khi lượng tử thời gian tăng, thời gian phản hồi trong hệ thống cũng tăng lên, có thể làm giảm hiệu suất. Thứ hai, khi lượng tử thời gian giảm, chi phí chuyển đổi ngữ cảnh trong hệ thống sẽ tăng, tăng áp lực lên hệ thống. Cuối cùng, quyết định về một lượng tử thời gian hoàn hảo là một nhiệm vụ khó khăn, đòi hỏi sự đánh đổi cẩn thận giữa thời gian phản hồi và chi phí chuyển đổi ngữ cảnh trong hệ thống.

2.3.1.2 Quản Lí Kiểu Ưu Tiên

Thuật Toán Priority Scheduling(Ưu tiên theo độ ưu tiên định sẵn) :

a, Giới thiệu :

Ngoài sự đa nhiệm, công bằng ra thì Android cũng cần những thuật toán để tối ưu trải nghiệm người dùng, do đó thuật toán Priority Scheduling được sử dụng , tiến trình được ưu tiên dựa trên mức độ ưu tiên của nó. Tiến trình có mức độ ưu tiên cao sẽ được xử lý trước.

b, Nguyên tắc hoạt động:

Tại thời điểm xuất hiện một quá trình trong hàng đợi sẵn sàng, Mức độ ưu tiên của nó được so sánh với mức độ ưu tiên của các quá trình khác có trong hàng đợi sẵn sàng cũng như với một quá trình đang được CPU thực thi tại thời điểm đó của thời gian. Cái có mức độ ưu tiên cao nhất trong số tất cả các quy trình có sẵn sẽ được cấp cho CPU tiếp theo.

Sự khác biệt giữa lập kế hoạch ưu tiên trước và lập lịch ưu tiên không ưu tiên là,

trong Priority Scheduling, công việc đang được thực hiện có thể bị dừng lại khi có công việc có mức độ ưu tiên cao hơn

c, Ưu và nhược điểm :

Ưu điểm : Ưu tiên ứng dụng quan trọng: Priority Scheduling cho phép ứng dụng quan trọng được ưu tiên xử lý trước. Điều này đảm bảo rằng các ứng dụng quan trọng như cuộc gọi điện thoại hoặc ứng dụng y tế có thể đáp ứng kịp thời và không bị ảnh hưởng bởi các tiến trình khác.

Nhược điểm : Priority Scheduling có nguy cơ tiến trình ở mức ưu tiên thấp không bao giờ được xử lý nếu tiến trình ở mức ưu tiên cao luôn có công việc để làm. Điều này dẫn đến hiện tượng starvation(đói).

Ngoài ra thì còn nhiều thuật toán quản lý theo kiểu ưu tiên như trên như First-Come First-Served (FCFS) - ưu tiên theo thời điểm ,Shortest Job First(SJF) ưu tiên theo thời gian thực hiện,.....

2.3.1.3 Multilevel Queue Scheduling (Lập lịch hàng đợi đa cấp)

a, Giới thiệu:

Để dung hòa được ưu và nhược điểm của mỗi thuật toán giúp hệ điều hành Android vừa có thể đa nhiệm mà vừa tăng trải nghiệm người dùng và hiệu suất tốt hơn thì ta dùng một thuật toán mà có thể kết hợp được nhiều thuật toán khác nhau để phù hợp với nhiều loại tiến trình

b, Cơ chế hoạt động:

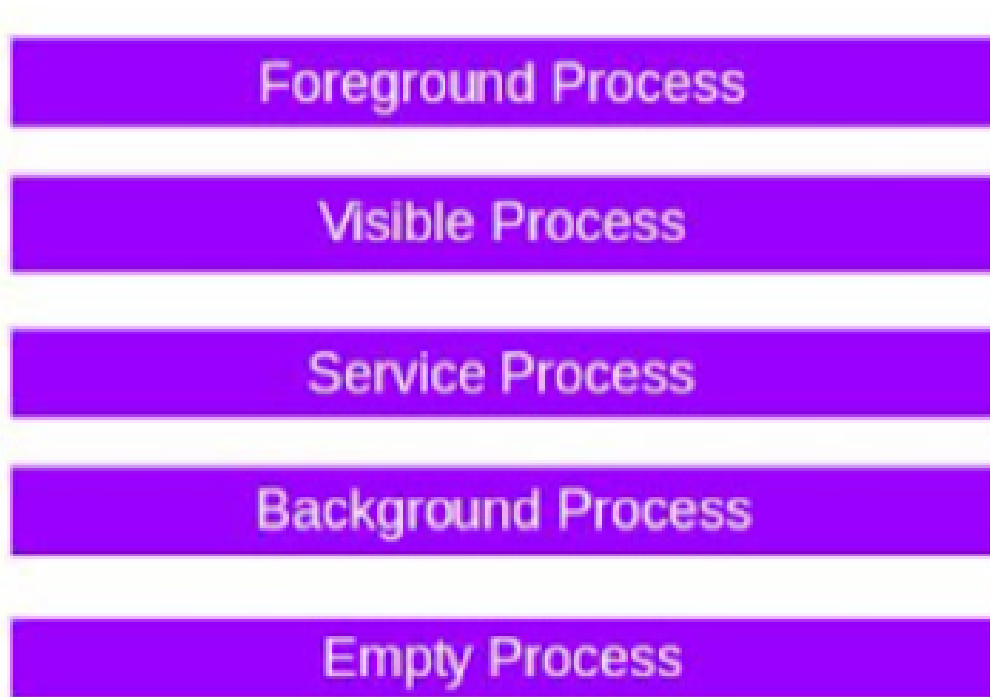
Tiến trình được chia thành nhiều hàng đợi (queues), mỗi hàng đợi tương ứng với một mức độ ưu tiên hoặc loại công việc cụ thể. Các hàng đợi này có thể được sắp xếp theo thứ tự ưu tiên từ cao đến thấp hoặc ngược lại. Mỗi hàng đợi có thể sử dụng một thuật toán lập lịch khác nhau, chẳng hạn như Round Robin, FCFS, SJF, hoặc bất kỳ thuật toán nào khác phù hợp với mức độ ưu tiên hoặc loại công việc cụ thể. Một hàng đợi có thể ưu tiên công việc ngắn hơn, trong khi hàng đợi khác có thể ưu tiên công việc

2.3.1.4 Đánh giá chung

Đây là ba loại thuật toán cơ bản nhất , nền tảng nhất , phổ biến nhất trong các phiên bản khác nhau của hệ điều hành Android bởi nó vừa đáp ứng được sự đa nhiệm, công bằng, vừa tối ưu được hiệu suất cũng như trải nghiệm người dùng với những tiến trình ưu tiên. Tuy vậy sự đa nhiệm này của Android có nhược điểm khi toàn bộ ứng dụng được chạy đồng thời, tuy có sự linh hoạt trong hệ thống đa nhiệm nhưng sẽ làm giảm hiệu suất hệ thống , dễ bị giật lag . Bên cạnh đó thì hệ điều hành

IOS có sự quản lí đa nhiệm một cách chặt chẽ hơn, giới hạn các ứng dụng được chạy , cải thiện hiệu suất hệ thống , cho nên trải nghiệm thực tế của người dùng về độ mượt giữa hai hệ điều hành thì IOS vẫn luôn nhỉnh hơn.

2.3.2 Các trạng thái của tiến trình



Hình 2.5: Các tiến trình xếp theo mức độ ưu tiên

2.3.2.1 Foreground process

Ta có thể gọi nó là tiến trình trước mặt hay tiền cảnh , là những ứng dụng đang hiển thị trước mặt mà chúng ta đang tương tác. Ví dụ ta đang sử dụng ứng dụng Facebook. Trong trường hợp này, tiến trình của ứng dụng Facebook được coi là tiến trình trước mặt.

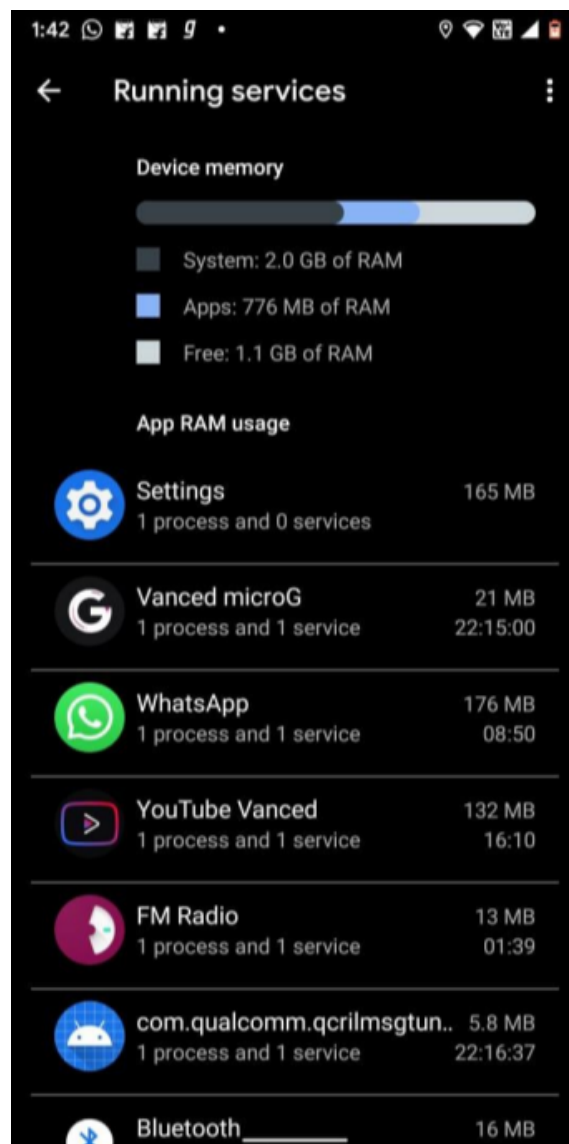
2.3.2.2 Visible process

Một tiến trình hiển thị là một tiến trình khi mà hoạt động có thể nhìn thấy được bởi người dùng. Người dùng không tương tác trực tiếp với tiến trình này những tiến trình này chỉ bị hủy khi việc giữ cho chúng tồn tại trở nên không thể để tiến trình "trước mặt" tiếp tục thực thi.

Ví dụ: Khi một ứng dụng cần quyền truy cập như quyền truy cập camera, quyền truy cập bộ nhớ, v.v., một hộp thoại hoặc hộp thoại sẽ xuất hiện và yêu cầu quyền cần thiết. Lúc này, tiến trình tương ứng với hoạt động của ứng dụng đang chạy trước đó sẽ chuyển sang trạng thái hiển thị.

2.3.2.3 Service process

Một tiến trình được coi là tiến trình dịch vụ khi nó đang ở trạng thái chạy và không thuộc vào các loại tiến trình "trước mặt" (foreground) và "hiển thị" (visible). Những tiến trình này không xuất hiện trực tiếp trước mặt người dùng của ứng dụng. Tiến trình dịch vụ thường hữu ích cho các ứng dụng thực hiện các nhiệm vụ nền như tải lên hoặc tải xuống dữ liệu qua mạng. Hệ thống sẽ duy trì tiến trình dịch vụ cho đến khi không thể duy trì được các tiến trình "trước mặt" và "hiển thị" nữa.

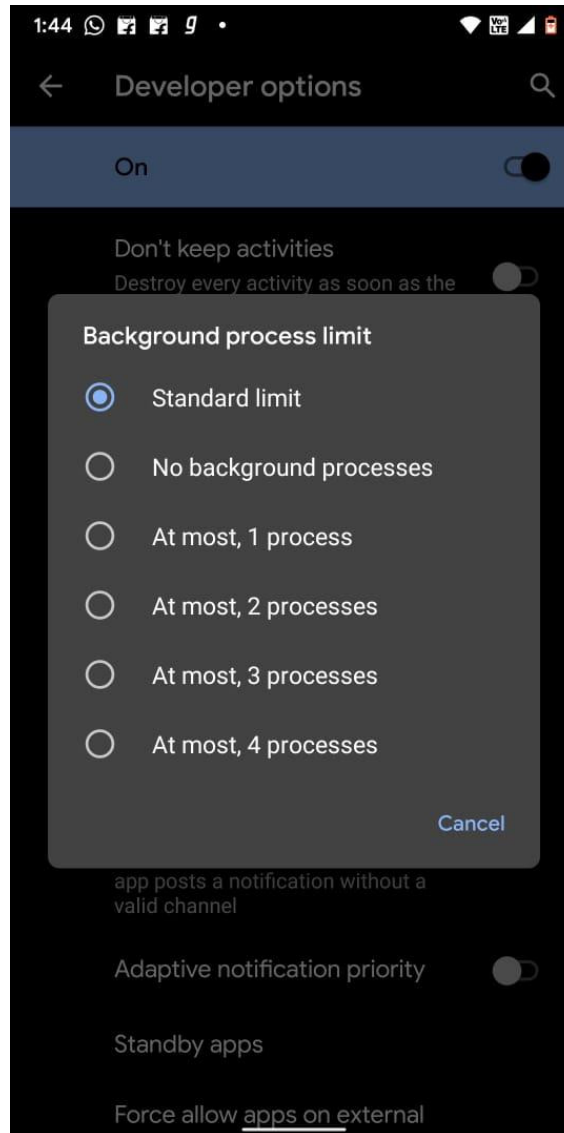


Hình 2.6: Hình ảnh biểu thị những dịch vụ của tiến trình

2.3.2.4 Background process

Giả sử người dùng đang sử dụng một ứng dụng và đột ngột nhấn nút home, vì hành động này, tiến trình chuyển từ trạng thái trước mặt sang trạng thái nền. Khi ứng dụng chuyển từ trạng thái trước mặt sang trạng thái nền, nó sẽ được đặt vào hàng đợi cache LRU và sẽ được đặt ở phía trước của hàng đợi. Khi người dùng quay

lại ứng dụng đó, tiến trình sẽ trở lại từ trạng thái nền sang trạng thái trước mặt.



Hình 2.7: Hình ảnh biểu thị việc giới hạn những tiến trình nền

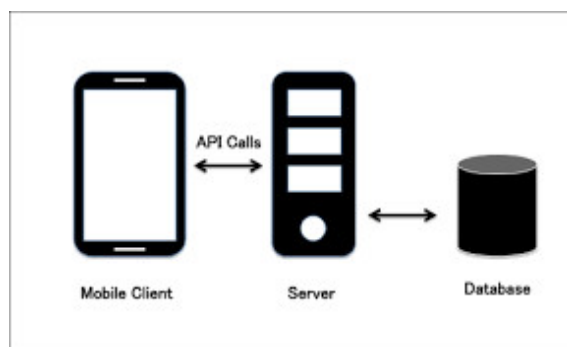
2.4 Tổ chức cơ sở dữ liệu và chia sẻ dữ liệu

2.4.1 Tổ chức cơ sở dữ liệu

Android cung cấp một số công nghệ cơ sở dữ liệu và giao diện lập trình ứng dụng (APIs) để làm việc với các cơ sở dữ liệu.

Ứng dụng Android thường sử dụng kiến trúc client-server, trong đó cơ sở dữ liệu nằm trên một máy chủ từ xa và ứng dụng Android sử dụng các API và giao thức như HTTP để giao tiếp với cơ sở dữ liệu. Điều này cho phép ứng dụng Android truy cập và quản lý dữ liệu từ xa thông qua mạng.

Tổ chức cơ sở dữ liệu trong một ứng dụng Android phụ thuộc vào loại cơ sở dữ liệu được sử dụng và yêu cầu cụ thể của ứng dụng. Thường thì, dữ liệu trong ứng dụng Android được tổ chức thành các bảng hoặc tập tin, với mỗi bảng hoặc tập tin



Hình 2.8: Mô hình client server trong android

đại diện cho một loại dữ liệu cụ thể. Dữ liệu trong bảng có thể được lưu trữ và truy cập thông qua các thao tác SQL hoặc các phương thức cung cấp bởi thư viện cơ sở dữ liệu.

Trong ứng dụng Android, quản trị cơ sở dữ liệu có thể được tổ chức và triển khai theo một số cách khác nhau. Dưới đây là một số phương pháp chính để tổ chức quản trị cơ sở dữ liệu trong Android:

- **SQLite Database:** Android cung cấp hỗ trợ tích hợp cho SQLite Database, một cơ sở dữ liệu nhỏ, nhúng và phổ biến. Chúng ta có thể tạo và quản lý các bảng SQLite, và thực hiện các thao tác CRUD (Create, Read, Update, Delete) thông qua các API cung cấp bởi lớp SQLiteDatabase. Thông thường, chúng ta sẽ tạo một lớp dữ liệu (Data Model) để đại diện cho các bảng và sử dụng lớp SQLiteOpenHelper để khởi tạo và quản lý cơ sở dữ liệu.
- **Object Relational Mapping (ORM):** Android hỗ trợ sử dụng các thư viện ORM như Room hoặc GreenDAO để tự động ánh xạ giữa các đối tượng Java và cơ sở dữ liệu SQLite. Các thư viện ORM này giúp giảm thiểu việc viết code SQL và cung cấp một cách tiện lợi để truy cập và quản lý dữ liệu.
- **Content Provider:** Content Provider là một thành phần của Android cho phép một ứng dụng chia sẻ dữ liệu với các ứng dụng khác thông qua giao diện chuẩn. Content Provider cung cấp một cách tiếp cận cho ứng dụng khác truy cập và thao tác với dữ liệu. Điều này rất hữu ích khi chúng ta muốn chia sẻ dữ liệu trong ứng dụng của mình với các ứng dụng khác hoặc khi muốn truy cập vào dữ liệu được cung cấp bởi các ứng dụng khác.

Ngoài ra, Android cũng hỗ trợ các cơ sở dữ liệu khác như MySQL và PostgreSQL thông qua các thư viện và giao diện lập trình ứng dụng bên thứ ba. Bằng cách sử dụng các thư viện này, ứng dụng Android có thể truy cập và quản lý dữ liệu trong các cơ sở dữ liệu được triển khai trên máy chủ.

Android hỗ trợ sử dụng cơ sở dữ liệu SQL (Structured Query Language) thông qua SQLite, một cơ sở dữ liệu nhỏ, nhúng và phổ biến.

Android cung cấp hệ quản trị cơ sở dữ liệu quan hệ đầy đủ thông qua thư viện SQLite, mà không có bất kỳ hạn chế nào

Sử dụng SQLite, chúng ta có thể tạo độc lập, cơ sở dữ liệu quan hệ cho mỗi ứng dụng. Sử dụng chúng để lưu trữ và quản lý cấu trúc dữ liệu phức tạp của ứng dụng. Tất cả cơ sở dữ liệu của Android được lưu trữ trong thư mục /data/data/<tên package>/databases trong thiết bị hoặc máy ảo. Mặc định tất cả cơ sở dữ liệu là riêng tư chỉ có ứng dụng tạo ra chúng mới có quyền truy cập. Tuy nhiên, để chia sẻ cơ sở dữ liệu của ứng dụng ra ngoài chúng ta có thể dùng Content Providers.

Giới thiệu SQLite

SQLite là hệ quản trị cơ sở dữ liệu quan hệ (Relational database management system – RDBMS).

SQLite có nhiều điểm mạnh, với kích thước nhỏ , cấu trúc đơn tầng và tuân theo quy chuẩn về cơ sở dữ liệu , không yêu cầu máy chủ riêng, SQLite dễ tích hợp và triển khai.

Nó được cài đặt như một thư viện nhỏ gọn và là một phần trong bộ phần mềm Android. Ngoài ra, SQLite cung cấp các chức năng như thông qua một thư viện chứ không phải là một quá trình riêng biệt. Mỗi cơ sở dữ liệu sẽ trở thành một phần được tích hợp trong ứng dụng. Điều này giúp làm giảm sự phụ thuộc vào bên ngoài, tối thiểu độ trễ và đơn giản hóa các giao dịch và đồng bộ.

SQLite nổi tiếng là cơ sở dữ liệu cực kỳ đáng tin cậy và là sự lựa chọn cho nhiều thiết bị điện tử bao gồm máy nghe nhạc MP3, Iphone, iPod về hệ thống cơ sở dữ liệu.

Kích thước nhẹ và mạnh mẽ, SQLite khác với nhiều với nhiều cơ sở dữ liệu khác bằng cách sử dụng hướng tiếp cận mềm dẻo để định nghĩa cột. Thay vì yêu cầu giá trị của cột phải phù hợp với 1 loại duy nhất, giá trị trong mỗi hàng cho mỗi cột. Dẫn đến là không có sự yêu cầu chặt chẽ nào cho giá trị của mỗi cột trong một hàng.

Các bước trong Tổ chức quản trị cơ sở dữ liệu trong Android

Thiết kế cơ sở dữ liệu

Đây là bước quản trị cơ sở dữ liệu quan trọng nhất, đặc biệt là trong việc đảm bảo chức năng, hiệu suất, và an ninh. Trong quá trình thiết kế cơ sở dữ liệu, cần xem xét các yếu tố như số lượng bảng, trường trong mỗi bảng, và mối quan hệ giữa

chúng. Ràng buộc chính, ràng buộc ngoại, và ràng buộc toàn vẹn cũng là các yếu tố cần quan tâm. Các chỉ mục, sử dụng cho truy vấn và hiệu suất, đóng một vai trò quan trọng trong việc thiết kế cơ sở dữ liệu.

Trong khi thực thi truy vấn, việc tối ưu hóa bằng cách sử dụng chỉ mục, hàm tổng hợp và tập hợp là quan trọng. Xử lý lỗi, cả truy vấn và dữ liệu, cũng đóng một phần quan trọng.

Quản lý dữ liệu là một khía cạnh quan trọng, bao gồm thêm, chỉnh sửa, xóa, sao lưu và phục hồi dữ liệu. Cuối cùng, bảo mật cơ sở dữ liệu đảm bảo an toàn với việc cấp quyền truy cập cho người dùng và ứng dụng, sử dụng khóa cơ sở dữ liệu, và áp dụng xác thực và ủy quyền để kiểm soát truy cập vào cơ sở dữ liệu.

2.4.2 Chia sẻ dữ liệu trong android

Chia sẻ dữ liệu là một thành phần quan trọng của Android cho phép các ứng dụng khác nhau giao tiếp với nhau và chia sẻ dữ liệu. Bao gồm chia sẻ dữ liệu người dùng, chia sẻ nội dung, chia sẻ dữ liệu ứng dụng. Có thể chia sẻ dữ liệu bằng các cách như sử dụng các API(giao diện lập trình ứng dụng)của nhà phát triển. API cung cấp khả năng truy xuất đến một tập các hàm hay dùng. Và từ đó có thể trao đổi dữ liệu giữa các ứng dụng. Ngoài ra có thể chia sẻ dữ liệu thông qua các dịch vụ nền tảng.

a, Sử dụng Intent

Intent là một cách để gửi và nhận dữ liệu giữa các thành phần của ứng dụng Android. Nó dùng để chuyển dữ liệu giữa các Activity hoặc giữa Activity và Service trong ứng dụng Android. Thông qua các extras để đính kèm dữ liệu.



Hình 2.9: Chia sẻ dữ liệu sử dụng Intent

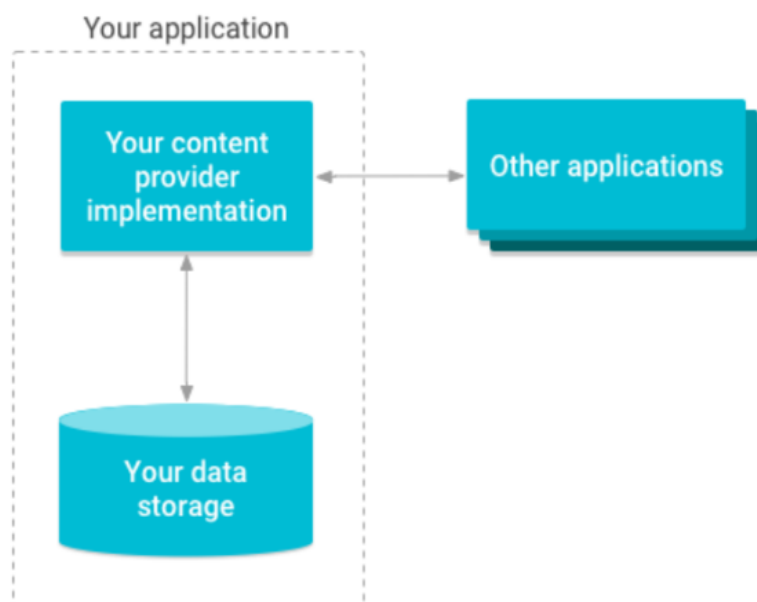
Phương thức Intent có ưu điểm đơn giản và dễ sử dụng, phù hợp cho truyền dữ

liệu nhỏ hoặc trạng thái ngắn gọn nhưng hạn chế của nó là không phù hợp cho truyền dữ liệu lớn hoặc giữa các ứng dụng khác nhau.

b, Content Providers (Nhà cung cấp nội dung)

Nhà cung cấp nội dung có thể giúp ứng dụng quản lý quyền truy cập vào dữ liệu được lưu trữ bởi chính nó hoặc được lưu trữ bởi các ứng dụng khác và cung cấp cách chia sẻ dữ liệu với các ứng dụng khác. Chúng đóng gói dữ liệu và cung cấp các cơ chế để xác định bảo mật dữ liệu. Nhà cung cấp nội dung là giao diện chuẩn kết nối dữ liệu trong một quy trình với mã chạy trong quy trình khác.

Việc triển khai một nhà cung cấp nội dung có nhiều lợi thế. Quan trọng nhất là có thể định cấu hình nhà cung cấp nội dung để cho phép các ứng dụng khác truy cập và sửa đổi dữ liệu ứng dụng một cách an toàn.

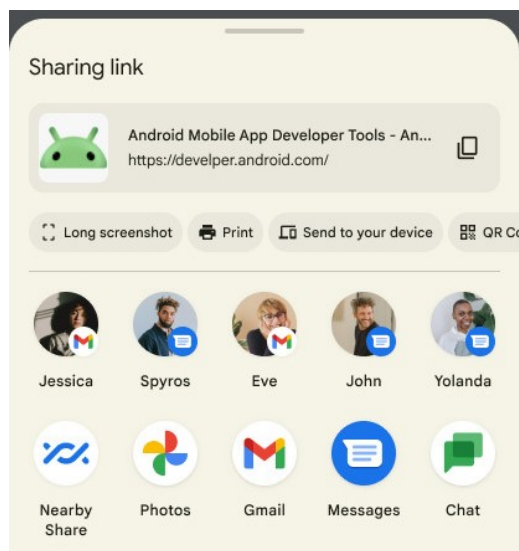


Hình 2.10: Sơ đồ tổng quan về cách nhà cung cấp nội dung quản lý quyền truy cập vào bộ nhớ

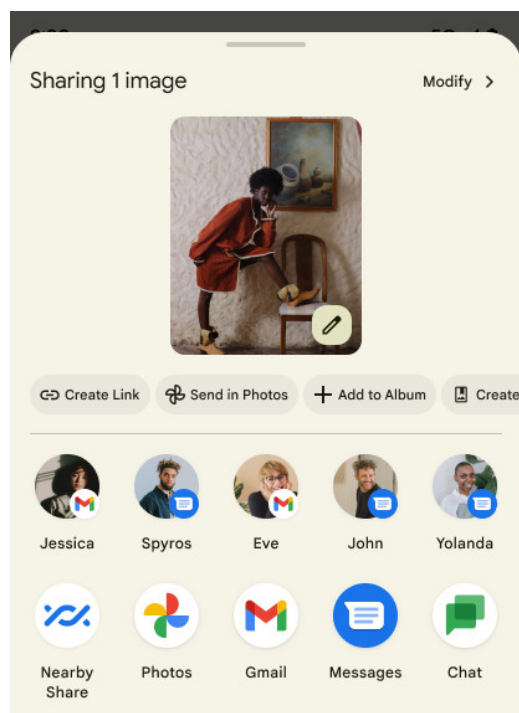
Ưu điểm của phương thức này là cung cấp giao diện chung để truy cập dữ liệu, bảo mật và kiểm soát quyền truy cập tốt, tuy nhiên mặt hạn chế lại thì nó phải cần triển khai nhiều mã và cấu hình.

c, Sharing Files

File Sharing là quá trình chia sẻ và truyền tải các tệp tin giữa các thiết bị hoặc ứng dụng khác nhau. Nó dùng để chia sẻ hình ảnh, video, văn bản và các loại tệp tin khác giữa các ứng dụng hoặc thiết bị khác nhau.



Hình 2.11: Dùng Android Sharesheet để chia sẻ link tới bạn bè



Hình 2.12: Dùng Android Sharesheet để chia sẻ ảnh tới bạn bè

Ưu điểm phương thức này là nó phù hợp cho việc chia sẻ tệp tin giữa các ứng dụng và thiết bị. Tuy nhiên thì việc này phát sinh nhiều hạn chế là cần kiểm soát quyền truy cập và xử lý các vấn đề bảo mật.

d, Firebase Realtime Database và Cloud Firestore

Firebase Realtime Database và Cloud Firestore là dịch vụ cơ sở dữ liệu trực tuyến của Google. Nó được sử dụng để lưu trữ và đồng bộ dữ liệu thời gian thực giữa các thiết bị và máy chủ, thích hợp cho ứng dụng đa người dùng, ứng dụng xã hội và ứng dụng cần cập nhật dữ liệu nhanh chóng.

Ưu điểm của phương thức này là cung cấp khả năng đồng bộ dữ liệu thời gian thực và tích hợp chặt chẽ với nền tảng Firebase. Bên cạnh đó nó lại yêu cầu kết nối internet để tương tác với dữ liệu trực tuyến cho nên vấn đề này đã nảy sinh ra một hạn chế nhỏ.

2.4.3 So sánh với DBMS của Window(MySQL)

Tất cả hệ quản trị cơ sở dữ liệu (DBMS) phục vụ mục đích lưu trữ và quản lý dữ liệu, nhưng có sự khác biệt đáng kể giữa SQLite trong Android và MySQL trong môi trường Windows, dưới đây là một so sánh chi tiết:

SQLite và MySQL là hai hệ quản trị cơ sở dữ liệu với đặc điểm và ứng dụng riêng biệt. SQLite, là hệ quản trị cơ sở dữ liệu nhúng nhẹ, không yêu cầu cài đặt riêng biệt và thích hợp cho ứng dụng di động và các ứng dụng nhúng có quy mô

nhỏ. Nó tích hợp sẵn trong Android, dễ quản lý với một tệp dữ liệu đơn giản và phù hợp cho việc lưu trữ dữ liệu cục bộ.

Trái ngược, MySQL là hệ quản trị cơ sở dữ liệu mã nguồn mở mạnh mẽ và phân tán, thường cài đặt trên máy chủ riêng biệt. Nó được ưa chuộng trong các ứng dụng web và hệ thống máy chủ với quy mô lớn và tải truy cập cao. Tuy nhiên, MySQL đòi hỏi quá trình cài đặt và cấu hình MySQL Server, cũng như kiến thức về quản trị cơ sở dữ liệu và bảo mật hệ thống.

Về quy mô và hiệu suất, SQLite thích hợp cho ứng dụng nhỏ với dữ liệu ít và tải truy cập thấp. MySQL, ngược lại, hỗ trợ quy mô lớn và có khả năng xử lý nhiều truy vấn đồng thời, cung cấp hiệu suất cao và tùy chọn cấu hình cho các tác vụ phức tạp.

Ngôn ngữ truy vấn cũng là điểm khác biệt, khi SQLite sử dụng một phạm vi hạn chế của SQL, trong khi MySQL hỗ trợ nhiều tính năng SQL phong phú và câu lệnh mở rộng.

Cả hai hệ quản trị cơ sở dữ liệu đều cung cấp cơ chế bảo mật, nhưng MySQL có tính năng bảo mật phong phú hơn và khả năng quản lý quyền truy cập tốt hơn.

Tùy thuộc vào nhu cầu và yêu cầu cụ thể của dự án, chúng ta có thể chọn SQLite hoặc MySQL. SQLite thường được sử dụng trong phát triển ứng dụng di động hoặc dự án nhỏ, trong khi MySQL phù hợp cho các dự án web và hệ thống máy chủ quy mô lớn hơn.

2.5 Android và quản lý bộ nhớ

2.5.1 Các phương thức quản lý bộ nhớ

Máy ảo Android Runtime và Dalvik sử dụng phân trang và ánh xạ bộ nhớ để quản lý bộ nhớ. Phân trang bộ nhớ (paging) trong Android đề cập đến việc quản lý không gian lưu trữ trên thiết bị thành các phân đoạn hay trang. Các phân trang này giúp tối ưu hóa quản lý bộ nhớ và làm cho việc truy cập dữ liệu nhanh chóng hơn. Ánh xạ bộ nhớ (Memory mapping/ mmapping) là một cơ chế cho phép ứng dụng truy cập dữ liệu lưu trữ trên đĩa mà không cần phải đọc hoặc ghi dữ liệu một cách trực tiếp. Trong Android, cơ chế này thường được sử dụng để tối ưu hóa hiệu suất của ứng dụng. Như vậy bất kỳ bộ nhớ nào mà ứng dụng sửa đổi - phân bổ mới các đối tượng hay sử dụng các trang được ánh xạ - 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 tham chiếu đối tượng mà ứng dụng chứa, làm cho bộ nhớ tạm dụng được khả năng của bộ thu gom rác. Tuy nhiên có một ngoại lệ là bất kỳ tập tin nào được thêm vào mà không sửa đổi, chẳng hạn như code, đều có thể được phân trang ra

khỏi RAM nếu hệ thống muốn sử dụng bộ nhớ đó ở nơi khác.

a, Thu gom rác (garbage collection)

Môi trường quản lý bộ nhớ (như ART hay máy ảo Dalvik) theo dõi từng lượt cấp phát bộ nhớ. Sau khi xác định một phần bộ nhớ không còn được 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 quản lý bộ nhớ được gọi là thu gom rác. Việc thu gom rác có hai mục tiêu: tìm các đối tượng dữ liệu trong chương trình mà không thể không truy cập trong tương lai; và thu hồi tài nguyên mà các đối tượng đó sử dụng.

Vùng nhớ khối xếp của Android là một vùng nhớ mang tính thể hệ, nghĩa là nó theo dõi nhiều tiến trình phân bổ bộ nhớ khác nhau, dựa trên thời gian tồn tại và kích thước dự kiến của một đối tượng được cấp phát. Ví dụ: các đối tượng mới được cấp phát vào vùng nhớ sẽ nằm ở Young generation (vùng nhớ chứa các đối tượng mới được khởi tạo). Khi một đối tượng hoạt động đủ lâu, nó sẽ dần trở thành older generation (vùng nhớ chứa các đối tượng cũ), và cuối cùng là trở thành một permanent generation (vùng nhớ chứa các đối tượng vĩnh viễn).

Mỗi thể hệ của vùng nhớ khối xếp đều có giới hạn trên (upper limit) về lượng bộ nhớ mà các đối tượng ở đó có thể sử dụng. Bất cứ khi nào một thể 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 thể hệ của các đối tượng mà nó đang thu thập và số lượng đối tượng đang hoạt động trong mỗi thể hệ.

Mặc dù việc thu gom rác có thể khá nhanh, nhưng vẫn có thể ảnh hưởng đến hiệu suất của ứng dụng. Nhìn chung, chúng ta không kiểm soát được thời điểm xảy ra sự kiện thu gom rác bên trong code của mình. Hệ thống có một bộ tiêu chí được dùng 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 diễn ra ở chính giữa vòng lặp xử lý chuyên sâu như ảnh động hoặc trong khi phát nhạc, thì thời gian xử lý có thể bị kéo dài thêm, điều này sẽ làm quá trình thực thi các dòng lệnh trong ứng dụng của chúng ta vượt quá ngưỡng 16ms được đề xuất để khung hình có thể hiển thị một cách hiệu quả và mượt mà.

Ngoài ra, quy trình code của chúng 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 chúng ta cấp phát nhiều đối tượng ở phần trong cùng của một vòng lặp trong mỗi khung hình của ảnh động kết hợp alpha, chúng ta đã vô tình làm đầy vùng nhớ khối xếp của mình vì có quá nhiều đối tượng. Trong trường hợp đó, trình thu gom rác phải thực hiện nhiều sự kiện thu gom rác và từ đó làm giảm

hiệu suất của ứng dụng.

b, Chia sẻ bộ nhớ

Để phù hợp với mọi thứ cần thiết trong RAM, Android cố gắng chia sẻ các trang RAM qua các tiến trình.

Mỗi tiến trình ứng dụng được lấy từ một tiến trình gọi là Zygote. Tiến trình Zygote được thực thi khi hệ thống khởi động, và tải các tài nguyên, mã nguồn framework. Để bắt đầu một process ứng dụng mới, hệ thống sẽ lấy tiến trình Zygote, sau đó tải và chạy mã nguồn của ứng dụng trong tiến trình mới. Cách tiếp cận này cho phép hầu hết các trang RAM được cấp phát cho mã nguồn framework và tài nguyên được chia sẻ trên tất cả các tiến trình ứng dụng.

Hầu hết các dữ liệu tĩnh được ánh xạ bộ nhớ (mmaped) vào một tiến trình. Kỹ thuật này cho phép dữ liệu được chia sẻ giữa các tiến trình và cũng cho phép dữ liệu được xóa đi khi cần.

c, Cấp phát và thu hồi bộ nhớ ứng dụng



Hình 2.13: Cấp phát và thu hồi bộ nhớ ứng dụng

Bộ nhớ khối xếp Dalvik bị giới hạn cho mỗi tiến trình ứng dụng. Nghĩa là kích thước của bộ nhớ khối xếp có thể tăng theo nhu cầu nhưng chỉ đến giới hạn mà hệ thống xác định cho mỗi ứng dụng.

Kích thước của bộ nhớ khối xếp không giống với dung lượng bộ nhớ vật lý được sử dụng bởi nó. Khi kiểm tra bộ nhớ khối xếp của ứng dụng, Android sẽ tính toán một giá trị được gọi là kích thước cài đặt theo tỷ lệ (Proportional Set Size - PSS),

gồm cả các dirty và clean page được chia sẻ với các tiến trình khác. Tổng số (PSS) này là những gì mà hệ thống coi là physical memory footprint.

Bộ nhớ khối xếp Dalvik không nén kích thước hợp lý của nó, có nghĩa là Android không giảm phân mảnh bộ nhớ để dồn không gian bộ nhớ. Android chỉ có thể thu nhỏ kích thước bộ nhớ hợp lý khi có không gian chưa sử dụng ở cuối. Tuy nhiên, hệ thống vẫn có thể giảm bộ nhớ vật lý được sử dụng bởi bộ nhớ khối xếp. Sau khi thu gom rác, Dalvik đi qua bộ nhớ khối xếp và tìm các trang không sử dụng, sau đó trả lại các trang đó cho kernel bằng cách sử dụng madvise. Vì vậy, các cấp phát bộ nhớ đôi và giải phóng tài nguyên bộ nhớ của các chunk (dữ liệu điều khiển) lớn sẽ dẫn đến việc lấy lại tất cả (hoặc gần như tất cả) bộ nhớ vật lý được sử dụng. Tuy nhiên, việc lấy lại bộ nhớ từ các cấp phát nhỏ có thể kém hiệu quả hơn nhiều vì trang được sử dụng cho cấp phát nhỏ vẫn có thể được chia sẻ với một thứ khác chưa được giải phóng.

d, Hạn chế bộ nhớ của ứng dụng

Để duy trì môi trường đa tác vụ, Android đặt giới hạn cứng cho kích thước bộ nhớ khối xếp cho mỗi ứng dụng. Giới hạn kích thước bộ nhớ khối xếp chính xác khác nhau giữa các thiết bị dựa trên tổng lượng RAM mà thiết bị có sẵn. Nếu ứng dụng đã đạt đến dung lượng bộ nhớ khối xếp tối đa và cố gắng cấp phát thêm bộ nhớ, ứng dụng có thể bị OutOfMemoryError.

Trong một số trường hợp, nếu muốn truy vấn hệ thống để xác định chính xác dung lượng bộ nhớ khối xếp chúng ta có trên thiết bị hiện tại, có thể truy vấn hệ thống bằng cách gọi `getMemoryClass()`.

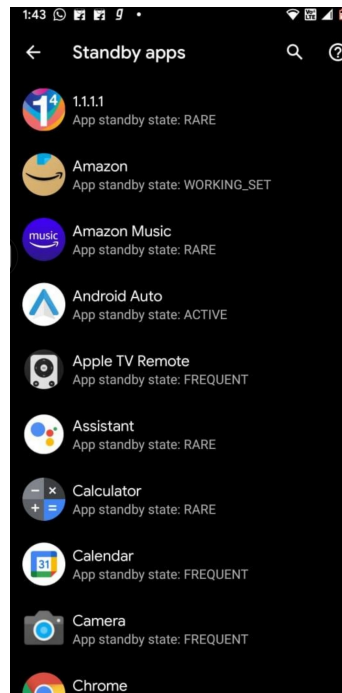
e, Chuyển đổi giữa các ứng dụng

Khi người dùng chuyển đổi giữa các ứng dụng, Android sẽ giữ các ứng dụng không phải là foreground - tức ứng dụng không thể thấy bởi người dùng hoặc chạy ngầm như trình phát nhạc - trong một bộ đệm LRU cache (Least-recently used).

Nếu ứng dụng có một tiến trình được lưu trong bộ nhớ cache và nó vẫn giữ bộ nhớ mà hiện tại nó không cần, thì ngay cả khi người dùng không sử dụng nó, vẫn ảnh hưởng đến hiệu suất chung của hệ thống. Khi hệ thống sắp hết bộ nhớ, nó sẽ hủy các tiến trình trong LRU cache, bắt đầu với tiến trình ít được sử dụng gần đây nhất. Hệ thống cũng chiếm các tiến trình giữ nhiều bộ nhớ nhất và có thể dừng chúng để giải phóng RAM.

f, Phân bổ bộ nhớ giữa các tiến trình hệ thống

Android chạy trên tiền đề: "Bộ nhớ trống là bộ nhớ bị lãng phí" (free memory is wasted memory). Nó cố gắng sử dụng toàn bộ bộ nhớ có sẵn mọi lúc. Ví dụ: hệ



Hình 2.14: Chuyển đổi giữa các ứng dụng

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 quan trọng và các ứng dụng người dùng.

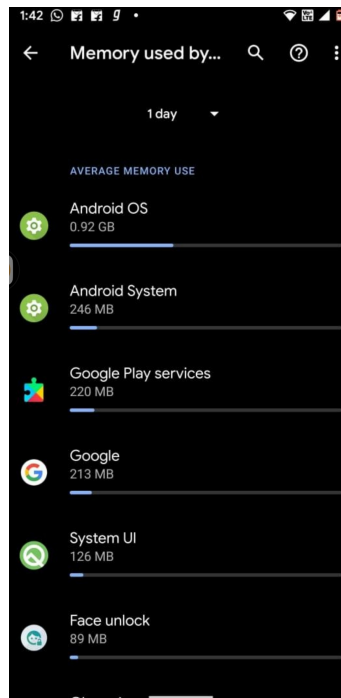
g, Các dạng bộ nhớ

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

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.

zRAM là một phân vùng RAM được sử dụng cho trao đổi không gia (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.

Storage chứa tất cả các dữ liệu ổn định (persistent data) như file hệ thống và tất cả các ứng dụng và thư viện. Storage có dung lượng lớn hơn nhiều so với hai loại bộ nhớ còn lại. Trên Android, Storage không được sử dụng để trao đổi không gian như trên các triển khai Linux khác vì việc viết thường xuyên có thể gây hao mòn bộ nhớ này và rút ngắn tuổi thọ nó.



Hình 2.15: Phân bố bộ nhớ giữa các tiến trình hệ thống

h, Trang bộ nhớ

RAM được chia thành các trang. Thông thường mỗi trang là 4KB bộ nhớ.

Bộ nhớ được chia thành các phân trang liên tiếp.

Hệ điều hành quản lý không gian lưu trữ bằng cách sử dụng bảng trang (page table) để ánh xạ giữa địa chỉ ảo và địa chỉ vật lý của các trang.

Cached: Bộ nhớ được hỗ trợ bởi lưu trữ tệp. Có hai loại bộ nhớ đệm. Thứ nhất là bộ nhớ đệm riêng tư (Private) là của 1 tiến trình duy nhất và không thể chia sẻ. Bộ nhớ này bao gồm bộ nhớ Clean là một phiên bản chưa sửa đổi của file trên lưu trữ, có thể bị xóa bởi swapd(kernel swap daemon) để tăng bộ nhớ trống và bộ nhớ Modified là một bản sao đã được sửa đổi của file trên lưu trữ, có thể di chuyển hoặc nén trong zRAM bằng kswapd để tăng bộ nhớ trống. Thứ hai là bộ nhớ đệm có thể chia sẻ (Shared), nó được sử dụng trong nhiều tiến trình. Bao gồm bộ nhớ Clean là một phiên bản chưa được sửa đổi của file trên storage, có thể bị xóa bởi kswapd để tăng bộ nhớ trống và bộ nhớ Revision là một bản sao đã được sửa đổi của file trên lưu trữ, cho phép các thay đổi được ghi lại để tăng bộ nhớ trống bằng kswapd, hoặc sử dụng tường minh bằng cách sử dụng msync() hoặc munmap().

Anonymous: Bộ nhớ không được hỗ trợ bởi lưu trữ tệp. Có bộ nhớ Dirty là loại bộ nhớ có thể di chuyển / nén trong zRAM bởi kswapd để tăng bộ nhớ trống

i, Kiểm soát bộ nhớ thấp

Android có hai cơ chế chính để xử lý các tình huống bộ nhớ thấp: kernel swap daemon và low-memory killer.

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 sao lưu bởi bộ nhớ và chưa được sửa đổi. Nếu một tiến trình cố gắng tìm một clean page đã bị xóa, hệ thống sẽ sao chép trang đó từ lưu trữ sang RAM. Hoạt động này được gọi là phân trang nhu cầu (demand paging).

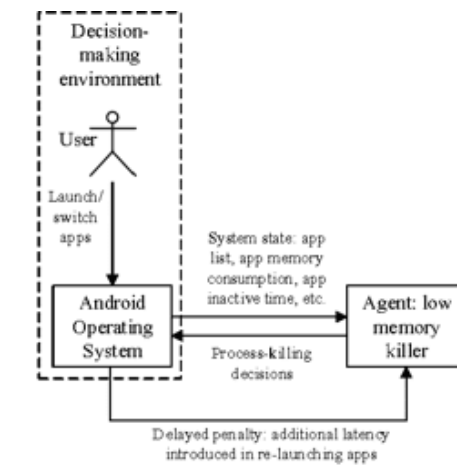
kswapd có thể di chuyển các private dirty page được lưu trong bộ nhớ đệm và các anonymous dirty page sang zRAM, nơi chúng được nén. Làm như vậy sẽ giải phóng bộ nhớ khả dụng trong RAM (free page). Nếu một tiến trình cố gắng truy cập vào một dirty page trong zRAM, trang đó sẽ không bị nén và được chuyển ngược trở lại vào RAM. Nếu tiến trình liên kết với một trang nén bị xóa, thì trang đó cũng 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.

Low-memory killer

Nếu kswapd không thể giải phóng đủ bộ nhớ cho hệ thống thì hệ thống sử dụng `onTrimMemory()` để thông báo cho ứng dụng rằng bộ nhớ sắp hết và nó sẽ giảm việc phân bổ. Nếu vẫn chưa đủ, kernel bắt đầu hủy các tiến trình để giải phóng bộ nhớ bằng low-memory killer (LMK).

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 process 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. Đoạn 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.

Đi đầu là Background apps là những 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. Đúng vị trí thứ hai là Previous app, là các 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



Hình 2.16: Low-memory killer

những ứng dụng chạy nền. Home app là ứng dụng launcher. Hủy cái này sẽ làm cho hình nền biến mất. Tiếp đó là Services, loại này được khởi động bởi các ứng dụng và có thể bao gồm đồng bộ hóa hoặc uploading lên cloud. Perceptible app là các ứng dụng non-foreground mà người dùng có thể cảm nhận được bằng một cách nào đó, chẳng hạn như process tìm kiếm hiển thị một UI nhỏ hoặc tiến trình nghe nhạc. Tiếp nữa là Foreground app, chính là các ứng dụng hiện đang được sử dụng. Hủy ứng dụng nền trước trông giống như một lỗi ứng dụng có thể cho người dùng biết rằng có sự cố xảy ra với thiết bị. Persistent (services) là những dịch vụ cốt lõi cho thiết bị, như telephony và wifi. System là các tiến trình của hệ thống. Khi các tiến trình này bị hủy, điện thoại có thể phải khởi động lại. Và cuối cùng là native, đó là Các tiến trình mức độ thấp được sử dụng bởi hệ thống (ví dụ: kswapd).

	Android	iOS
Quản lý Ứng dụng và Dịch vụ Nền	Cho phép ứng dụng chạy nền và thậm chí có thể chạy các dịch vụ ngầm. Điều này có thể ảnh hưởng đến tài nguyên hệ thống và tiêu tốn bộ nhớ	Có hạn chế hơn về quyền hạn của ứng dụng chạy nền. Ứng dụng iOS thường bị hạn chế trong việc thực hiện các tác vụ nền và chỉ được phép chạy một số dịch vụ cụ thể
Quản lý Bộ nhớ RAM	Sử dụng mô hình quản lý bộ nhớ tự động, cho phép ứng dụng sử dụng bộ nhớ linh hoạt dựa trên nhu cầu. Tuy nhiên, điều này cũng có thể dẫn đến việc tiêu tốn nhiều bộ nhớ khi có quá nhiều ứng dụng đang chạy	Có một hệ thống quản lý bộ nhớ hiệu suất cao và khá tự động. Hệ điều hành iOS có khả năng quản lý tài nguyên hệ thống và ưu tiên tác vụ quan trọng để giữ hiệu suất cao.
Quản lý Bộ nhớ Lưu trữ	Người dùng Android có khả năng mở rộng lưu trữ bằng cách sử dụng thẻ nhớ mở rộng, giúp họ dễ dàng mở rộng dung lượng lưu trữ khi cần thiết	Không hỗ trợ thẻ nhớ mở rộng, do đó người dùng phải chọn mua các phiên bản iPhone có dung lượng lớn hơn nếu họ cần nhiều lưu trữ hơn
Quản lý Cache và Dữ liệu	Cung cấp nhiều tùy chọn cho việc quản lý cache và dữ liệu ứng dụng. Người dùng có thể dễ dàng xóa cache và dữ liệu ứng dụng thông qua cài đặt	Cho phép người dùng xóa cache và dữ liệu ứng dụng, nhưng quy trình này phức tạp hơn so với Android
Tối ưu hóa Hiệu suất	Android chia sẻ tài nguyên giữa nhiều ứng dụng, có thể dẫn đến sự đa nhiệm linh hoạt nhưng cũng có thể ảnh hưởng đến hiệu suất.	iOS được tối ưu hóa cho các thiết bị cụ thể của Apple, điều này giúp tối ưu hóa hiệu suất và trải nghiệm người dùng.

Bảng 2.5: Bảng so sánh cách quản lý bộ nhớ giữa Android và iOS

2.5.2 Bộ nhớ ảo

a, Định nghĩa

Bộ nhớ ảo của hệ điều hành Android là một tính năng cho phép các ứng dụng chạy trên bộ nhớ vật lý ít hơn mức cần thiết. Điều này được thực hiện bằng cách sử dụng một phần của bộ nhớ lưu trữ để lưu trữ các phần của ứng dụng không được sử dụng hiện tại. Khi ứng dụng cần truy cập các phần này, chúng sẽ được tải lại vào bộ nhớ vật lý.

Bộ nhớ ảo có thể giúp cải thiện hiệu suất của thiết bị bằng cách cho phép nhiều

ứng dụng chạy cùng lúc. Nó cũng có thể giúp kéo dài thời lượng pin bằng cách giảm lượng bộ nhớ vật lý cần thiết để chạy các ứng dụng.

b, Các loại bộ nhớ ảo

Có hai loại bộ nhớ ảo trên Android. Một là bộ nhớ ảo được quản lý, loại này được quản lý bởi hệ điều hành Android. Hệ điều hành sẽ tự động quyết định ứng dụng nào nên được lưu trữ trong bộ nhớ ảo và ứng dụng nào nên được lưu trữ trong bộ nhớ vật lý. Thứ hai là bộ nhớ ảo được người dùng quản lý, loại này được quản lý bởi người dùng. Người dùng có thể chọn ứng dụng nào nên được lưu trữ trong bộ nhớ ảo và ứng dụng nào nên được lưu trữ trong bộ nhớ vật lý.

c, Ưu nhược điểm

Bộ nhớ ảo của Android có 3 ưu điểm chính. Một là cải thiện hiệu suất, bộ nhớ ảo cho phép nhiều ứng dụng chạy cùng lúc mà không làm giảm hiệu suất của thiết bị từ đó tăng khả năng sử dụng bộ nhớ. Thứ hai là tăng thời lượng pin, bộ nhớ ảo giúp giảm lượng bộ nhớ vật lý cần thiết để chạy các ứng dụng từ đó giúp kéo dài thời lượng pin của thiết bị. Và cuối cùng là tăng dung lượng bộ nhớ, bộ nhớ ảo cho phép lưu trữ các ứng dụng trên bộ nhớ lưu trữ thay vì bộ nhớ vật lý.

Tuy nhiên, bộ nhớ ảo cũng có một số nhược điểm. Đầu tiên là tăng độ trễ, có thể có độ trễ khi ứng dụng cần truy cập các phần được lưu trữ trong bộ nhớ ảo. Thứ hai là tăng khả năng xung đột: Các ứng dụng có thể xung đột với nhau nếu chúng cố gắng truy cập cùng một phần của bộ nhớ ảo. Và cuối cùng là tăng khả năng mất dữ liệu nếu bộ nhớ ảo bị hỏng, các ứng dụng có thể bị mất dữ liệu.

Nhìn chung, bộ nhớ ảo là một tính năng hữu ích có thể giúp cải thiện hiệu suất, thời lượng pin và dung lượng bộ nhớ của thiết bị Android

2.6 Các dịch vụ của hệ điều hành cung cấp

2.6.1 Khái niệm

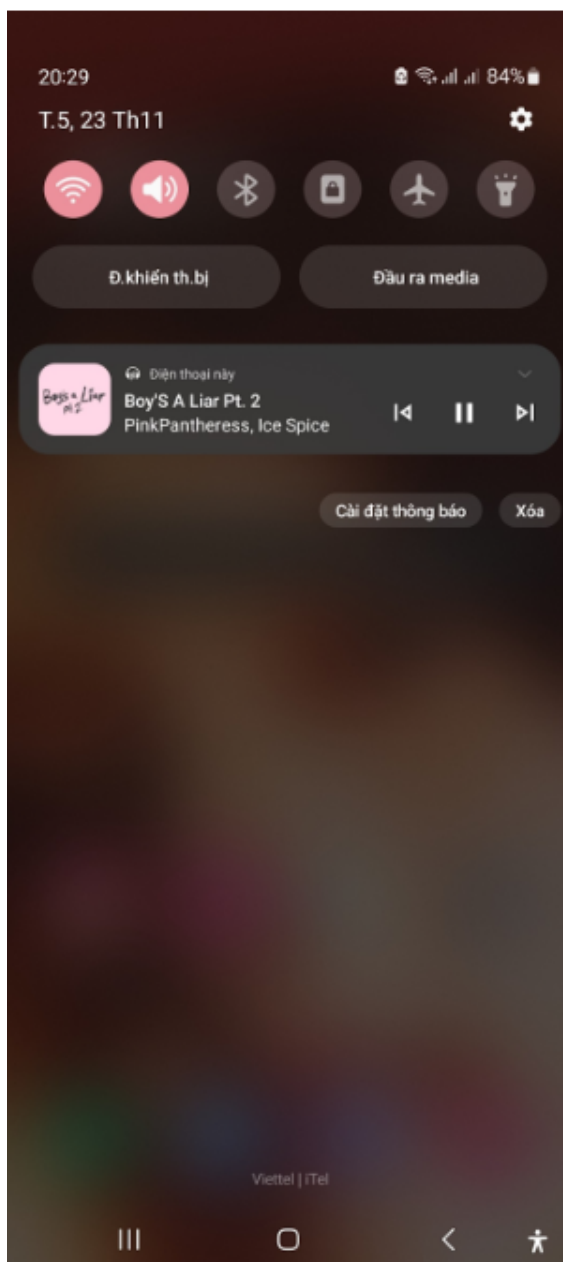
Service là một thành phần ứng dụng (application component) có thể thực hiện các hoạt động lâu dài trong background. Nó không cung cấp giao diện cho người dùng. Sau khi bắt đầu chạy, một Service có thể tiếp tục chạy trong một thời gian, ngay cả khi người dùng chuyển sang ứng dụng khác. Ngoài ra một thành phần có thể liên kết (bind) với một Service để tương tác với Service đó, thậm chí là thực hiện giao tiếp giữa các quá trình IPC (interprocess communication). Ví dụ: một Service có thể thực hiện các giao dịch mạng, chơi nhạc, ra vào file I/O hoặc tương tác với một content provider, tất cả đều từ background

Foreground Service (unbound service)

Là Service mà người dùng biết là Service đang chạy và hệ thống sẽ không kill

Service khi bộ nhớ xuống thấp. Phải cung cấp một notification trên status bar thể hiện Service đang chạy trừ khi Service bị dừng hoặc bị hủy từ foreground.

Ví dụ: music player sẽ sử dụng service để chơi nhạc nên service nên được để ở chế độ chạy foreground vì người dùng biết được bài nhạc đang nghe. Notification trên status bar nên miêu tả bài nhạc hiện tại đang nghe và cho phép người dùng khởi tạo một Activity để tương tác với music player



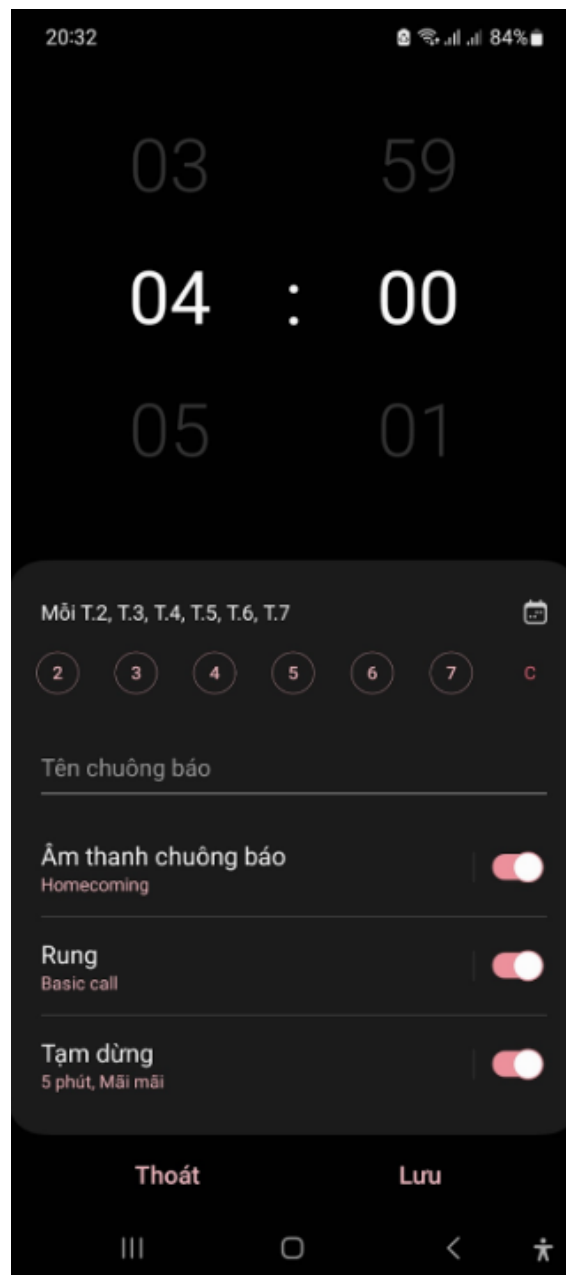
Hình 2.17: Music player sử dụng Foreground Service để chơi nhạc

Background Service (unbound service)

Là Service chạy mà người dùng không ý thức được là Service đang chạy và không có tương tác với người dùng

Ví dụ: nhận tin nhắn, nhận cuộc gọi đến, nhận email,...

Ví dụ: tạo báo thức lúc 4:00 am, khi đồng hồ hệ thống điểm 4:00 am, thiết bị sẽ thông báo báo thức cho người dùng bằng âm thanh và notification.



Hình 2.18: Báo thức cho người dùng sử dụng Background Service

Bound Service

Bound service trong Android là một loại dịch vụ mà các thành phần khác, như hoạt động (activities) hoặc dịch vụ (services) khác, có khả năng liên kết (bind) đến để tương tác với nó thông qua giao diện (interface) được cung cấp bởi dịch vụ đó. Các dịch vụ "bound service" cho phép các thành phần khác truy cập vào các chức năng của dịch vụ và trao đổi dữ liệu với nó thông qua giao diện được định nghĩa trước.

Độ ưu tiên các loại Service:

Hệ thống Android bắt buộc phải dừng một Service khi bộ nhớ ít và phải khôi phục tài nguyên hệ thống cho Activity đang được sử dụng. Nếu Service được ràng buộc với 1 Activity đang sử dụng, nó ít khả năng bị giết (kill). Nếu Service được khai báo và chạy ở chế độ Foreground nó cũng khó bị giết (kill). Do đó, các loại Service sẽ được sắp xếp theo độ ưu tiên sau: Bound Service > Foreground Service > Background Service

2.6.2 So sánh dịch vụ của Android với dịch vụ của Windows

Android và Windows, là hai hệ điều hành phổ biến, đặc biệt thiết kế cho các môi trường và mục đích sử dụng khác nhau. Dưới đây là một so sánh giữa các dịch vụ chính của cả hai hệ điều hành:

Hệ điều hành Android, được phát triển bởi Google, là lựa chọn chủ yếu cho điện thoại thông minh và máy tính bảng. Ngược lại, Windows, sản phẩm của Microsoft, thường xuất hiện trên máy tính cá nhân và máy chủ.

Trong lĩnh vực lưu trữ và đồng bộ hóa dữ liệu, Android sử dụng Google Drive, cung cấp dịch vụ lưu trữ đám mây và đồng bộ hóa dữ liệu trên các thiết bị Android.

Trong khi đó, Windows tích hợp OneDrive của Microsoft để lưu trữ và đồng bộ hóa tệp và dữ liệu giữa các thiết bị chạy hệ điều hành này.

Đối với trình duyệt web, Android sử dụng Google Chrome và các trình duyệt bên thứ ba, trong khi Windows hỗ trợ nhiều trình duyệt như Microsoft Edge, Google Chrome và Mozilla Firefox. Trong khía cạnh truy cập từ xa, Android thường sử dụng ứng dụng và dịch vụ từ xa như AnyDesk hoặc TeamViewer. Ngược lại, Windows hỗ trợ tính năng truy cập từ xa thông qua Remote Desktop Protocol (RDP) cho các phiên bản Windows Professional và Enterprise.

Đối với ứng dụng và cửa hàng ứng dụng, Android sở hữu Google Play Store, nơi người dùng có thể tải về và cài đặt ứng dụng di động và trò chơi. Trong khi đó, Windows có Microsoft Store, cung cấp ứng dụng và trò chơi dành cho máy tính chạy hệ điều hành Windows. Ở lĩnh vực email, Android có thể sử dụng ứng dụng email của bên thứ ba hoặc ứng dụng Gmail của Google. Trong khi đó, Windows cung cấp ứng dụng Mail trong Windows 10 và Microsoft Outlook là ứng dụng email phổ biến cho máy tính Windows.

Về cơ sở dữ liệu, Android không hỗ trợ cơ sở dữ liệu trực tiếp như Windows Server, một hệ điều hành được thiết kế cho môi trường máy chủ và hỗ trợ nhiều hệ thống quản trị cơ sở dữ liệu như Microsoft SQL Server.

Tổng cộng, Android và Windows cung cấp những dịch vụ và tính năng riêng biệt, phù hợp với môi trường và nhu cầu sử dụng cụ thể của họ. Android tập trung

vào di động và kết nối, trong khi Windows đa dạng hóa để hỗ trợ nhiều ứng dụng và dịch vụ cho cả máy tính cá nhân và máy chủ.

Tóm lại, Android và Windows có các dịch vụ và tính năng riêng biệt phù hợp với môi trường và nhu cầu sử dụng cụ thể của họ. Android tập trung vào di động và kết nối trên nền tảng di động, trong khi Windows tập trung vào máy tính cá nhân và máy chủ.

CHƯƠNG 3. KẾT LUẬN

3.1 Kết luận

Dựa trên những nội dung quản lý file, quản lý bộ nhớ, quản lý tiến trình, hệ quản trị cơ sở dữ liệu, và khả năng chia sẻ dữ liệu người dùng, có thể kết luận:

Giao diện người dùng(trong phần cấu trúc hệ thống): Android cung cấp các giao diện người dùng dựa trên cảm ứng và giao diện đồ họa thích hợp cho màn hình cảm ứng của điện thoại thông minh và máy tính bảng. Điều này làm cho việc tương tác với thiết bị di động dễ dàng và hiệu quả.

Quản lý tập tin: Android cung cấp các công cụ quản lý tập tin để quản lý và sắp xếp tài liệu và dữ liệu trên thiết bị di động, điều này quan trọng cho việc lưu trữ và truy cập dữ liệu cá nhân.

Quản lý bộ nhớ: Android quản lý bộ nhớ trong để đảm bảo hiệu suất tốt và sử dụng bộ nhớ lưu trữ hiệu quả trên các thiết bị di động.

Quản lý tiến trình: Android quản lý các tiến trình chạy trên thiết bị, quản lý tài nguyên và đảm bảo rằng các ứng dụng chạy mượt mà và không gây xung đột.

Hệ quản trị cơ sở dữ liệu: Android cung cấp hỗ trợ cho việc quản lý cơ sở dữ liệu trong các ứng dụng, cho phép lưu trữ và truy xuất dữ liệu một cách hiệu quả.

Chia sẻ dữ liệu người dùng: Android cho phép người dùng chia sẻ dữ liệu giữa các ứng dụng và thiết bị một cách dễ dàng, giúp họ truy cập và chia sẻ thông tin một cách thuận tiện.

KẾT LUẬN : Tất cả các tính năng và chức năng này của Android được thiết kế để làm cho nó trở thành một hệ điều hành lý tưởng cho các thiết bị di động. Android cung cấp một môi trường hoàn chỉnh cho việc phát triển và sử dụng các ứng dụng di động và là một nền tảng mạnh mẽ để đáp ứng nhu cầu của người dùng di động. Do vậy , nó hoàn toàn phù hợp với mục đích từ ban đầu đã đưa ra ở phần giới thiệu.

3.2 Hướng phát triển trong tương lai

Trong bối cảnh cuộc sống di động ngày càng đa dạng và phức tạp, Android không chỉ là hệ điều hành di động thông thường mà còn là một người bạn đồng hành đa nhiệm và linh hoạt. Để đáp ứng những thách thức ngày càng cao, Android đang hướng đến việc nâng cao hiệu suất, và tăng cường bảo mật để đảm bảo an toàn cho người dùng. Đồng thời, sự tích hợp mạnh mẽ với công nghệ 5G và IoT là những động lực quan trọng để mở ra những trải nghiệm mới và kết nối sâu sắc với thế giới

xung quanh. Dưới đây là một số xu hướng và khả năng mà Google và cộng đồng phát triển có thể tập trung vào trong tương lai:

- **Nâng cao Hiệu suất và Hiệu quả Năng lượng:** Với sự phổ biến của các ứng dụng đòi hỏi tài nguyên cao và sự gia tăng về đa nhiệm, việc tối ưu hóa hiệu suất và tiêu thụ năng lượng là quan trọng. Cải thiện quản lý tài nguyên hệ thống có thể giúp giảm tiêu thụ pin và tăng trải nghiệm người dùng.
- **Bảo mật và Quyền Riêng Tư:** Bảo mật luôn là một ưu tiên hàng đầu. Hệ điều hành Android có thể phát triển thêm các tính năng bảo mật mới và cập nhật để đối mặt với các mối đe dọa ngày càng tinh vi.
- **Hỗ trợ 5G và Kết nối IoT:** Android có thể tích hợp tốt hơn với công nghệ 5G để cung cấp tốc độ kết nối nhanh hơn và ổn định hơn. Ngoài ra, việc tương tác với các thiết bị Internet of Things (IoT) cũng có thể là một điểm đặc biệt.
- **Trải nghiệm Người dùng:** Cải thiện giao diện người dùng, tối ưu hóa trải nghiệm người dùng, và đơn giản hóa quá trình sử dụng các chức năng của hệ điều hành.
- **Phát triển Ứng dụng và API:** Hỗ trợ cho những công nghệ mới và các API mới để khuyến khích sự đổi mới và phát triển ứng dụng sáng tạo.
- **Hỗ trợ Cho Nền Tảng Giao dịch Tài chính Điện tử:** Với sự phổ biến của thanh toán di động và các ứng dụng giao dịch tài chính điện tử, Android có thể tập trung vào cải thiện tích hợp và bảo mật trong lĩnh vực này.
- **Hỗ trợ Cho Nền tảng Trực tuyến và Đám mây:** Kết nối mạnh mẽ với dịch vụ đám mây và nền tảng trực tuyến để cung cấp trải nghiệm tích hợp và đồng bộ tốt hơn cho người dùng.

TÀI LIỆU THAM KHẢO

[1] Sheikh. AA, Ganai. PT, Malik. NA, Ahmad Dar. KA, “Điện thoại thông minh: Android Vs iOS” (2013) Tạp chí Quốc tế Tiêu chuẩn Giao dịch về Kỹ thuật Khoa học Máy tính và các ứng dụng của nó.

[2] *Platform architecture*. [Online]. Available: <https://developer.android.com/guide/platform> (visited on 02/11/2023).

[3] *Android File System*. [Online]. Available: <https://www.scaler.com/topics/android-file-system/> (visited on 05/11/2023).

[4] *View on-device files with Device Explorer*. [Online]. Available: <https://developer.android.com/studio/debug/device-file-explorer> (visited on 05/11/2023).

[5] *Manage your app's memory*. [Online]. Available: <https://developer.android.com/topic/performance/memory> (visited on 08/11/2023).

[6] *View on-device files with Device Explorer*. [Online]. Available: <https://developer.android.com/studio/debug/device-file-explorer> (visited on 08/11/2023).

[7] *android.database.sqlite*. [Online]. Available: <https://developer.android.com/reference/android/database/sqlite/package-summary> (visited on 08/11/2023).

[8] *Processes and Application Lifecycle in Android*. [Online]. Available: <https://www.geeksforgeeks.org/processes-and-application-lifecycle-in-android/> (visited on 08/11/2023).

[1] *Overview of memory management*. [Online]. Available: <https://developer.android.com/topic/performance/memory-overview> (visited on 09/11/2023).

[9] *Chọn cơ sở dữ liệu: Cloud Firestore hoặc Cơ sở dữ liệu thời gian thực*. [Online]. Available: <https://firebase.google.com/docs/firestore/rtdb-vs-firestore?hl=vi> (visited on 09/11/2023).

[10] *Service Overview*. [Online]. Available: <https://developer.android.com/guide/components/services> (visited on 11/11/2023).