

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

o0o



BÀI TẬP LỚN

**Đề tài 5: Tìm hiểu chung về 2 hệ điều hành macOS
và iOS**

Môn học: Hệ điều hành
Số thứ tự nhóm: 3

Phạm Đức Chính	MSV: B21DCCN181
Nguyễn Văn Hòa	MSV: B21DCCN380
Nguyễn Đức An	MSV: B21DCCN001
Phạm Hữu Đoàn	MSV: B21DCCN229
Đỗ Đắc Huy	MSV: B21DCCN431
Bùi Thanh Tùng	MSV: B21DCAT214

Giảng viên hướng dẫn: Ths. Đinh Xuân Trường

HÀ NỘI, 11/2023

LỜI CẢM ƠN

Chúng em, nhóm sinh viên lớp Hệ điều hành, xin gửi lời cảm ơn chân thành nhất đến thầy Đinh Xuân Trường đã tận tình hướng dẫn và giúp đỡ chúng em trong suốt quá trình thực hiện bài tập lớn môn Hệ điều hành.

Trong quá trình học tập và tìm hiểu môn học này, chúng em đã được thầy truyền đạt rất nhiều kiến thức quý báu về hệ điều hành, giúp chúng em có thêm những hiểu biết sâu sắc về lĩnh vực này. Đặc biệt, những lời chia sẻ và định hướng của thầy đã giúp chúng em định hình được hướng nghiên cứu và hoàn thành bài tập lớn một cách trọn vẹn.

Đặc biệt, trong quá trình thực hiện bài tập lớn về tìm hiểu chung về 2 hệ điều hành macOS và iOS, chúng em đã gặp không ít khó khăn. Tuy nhiên, với sự hướng dẫn tận tình của thầy, chúng em đã dần dần vượt qua được những khó khăn đó và hoàn thành bài tập lớn một cách thành công.

Chúng em xin chân thành cảm ơn thầy!

TÓM TẮT NỘI DUNG BÀI TẬP LỚN

Bài tập lớn của nhóm chúng em tập trung vào việc tìm hiểu hệ điều hành macOS và iOS. Lý do chúng em chọn đề tài này là vì macOS và iOS đang trở thành hai trong những hệ điều hành phổ biến nhất trên thị trường và đó là một hướng phát triển quan trọng trong lĩnh vực phát triển ứng dụng di động và máy tính cá nhân. Nội dung chính của đề tài bao gồm nghiên cứu về kiến thức cơ bản về hệ điều hành macOS và iOS, khám phá các yếu tố kỹ thuật cơ bản của chúng, và phát triển ứng dụng dựa trên hai hệ điều hành này. Trong quá trình tìm hiểu chúng em đã nghiên cứu về kiến trúc cơ bản của hệ điều hành macOS và iOS, bao gồm cấu trúc hệ thống tệp tin, quản lý bộ nhớ quy trình thực thi và ứng dụng hiệu suất. Chúng em đã tìm hiểu về cách làm việc với giao diện người dùng, quản lý tài nguyên, và tích hợp các tính năng đặc biệt của từng hệ điều hành. Kết quả đạt được của đề tài này là sự hiểu biết sâu hơn về hệ điều hành macOS và iOS, khả năng phát triển ứng dụng trên cả hai nền tảng, và khả năng tùy chỉnh ứng dụng để phù hợp với yêu cầu cụ thể của từng hệ điều hành. Điều này sẽ giúp chúng em trong việc phát triển các ứng dụng di động và máy tính cá nhân trong tương lai, đáp ứng nhu cầu của thị trường và người dùng.

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU VỀ MACOS VÀ IOS	1
1.1 Hệ điều hành macOS	1
1.1.1 Lịch sử và nguồn gốc của hệ điều hành macOS.....	1
1.1.2 Ưu điểm và nhược điểm của hệ điều hành macOS	2
1.2 Hệ điều hành iOS	3
1.2.1 Lịch sử và nguồn gốc của hệ điều hành iOS	3
1.2.2 Ưu điểm và nhược điểm của hệ điều hành iOS	4
CHƯƠNG 2. CẤU TRÚC HỆ THỐNG	6
2.1 macOS.....	6
2.1.1 Hệ thống cơ bản (Core OS)	6
2.1.2 Hệ thống đồ họa (Graphics Subsystem).....	7
2.1.3 Hệ thống ứng dụng (Application Subsystem)	8
2.1.4 Giao diện người dùng (User interface)	9
2.2 iOS.....	10
2.2.1 Lõi OS (Core OS):	11
2.2.2 Dịch vụ lõi (Core Services):.....	11
2.2.3 Phương tiện (Media):.....	12
2.2.4 Cocoa Touch:.....	12
2.3 So sánh cấu trúc hệ thống macOS và IOS	13
2.3.1 Core OS:	13
2.3.2 Hệ thống đồ họa (Graphics Subsystem):.....	13
2.3.3 Hệ thống ứng dụng (Application Subsystem):	14
2.3.4 Giao diện người dùng (User Interface):	14

CHƯƠNG 3. HÀM SHELL VÀ LỜI GỌI HỆ THỐNG..... 15

3.1 Hàm Shell trong MacOS và IOS	15
3.1.1 Terminal là gì?.....	15
3.1.2 Giới thiệu về thiết bị Terminal macOS	15
3.1.3 Mở Terminal trên MacOS	15
3.1.4 Cách điều hướng trong macOS Terminal	18
3.1.5 Cấu trúc câu lệnh Shell.....	19
3.1.6 Các lệnh Terminal cơ bản cần biết	20
3.1.7 Các nguyên tắc khi sử dụng lệnh Terminal.....	21
3.1.8 Cách thoát Terminal	21
3.2 Các công cụ trên iOS	21

CHƯƠNG 4. QUẢN LÝ BỘ NHỚ..... 24

4.1 Nguyên tắc quản lý bộ nhớ, cơ chế xử lý tiến trình và quản lý bộ nhớ ứng dụng trên macOS và iOS.....	24
4.1.1 Nguyên tắc quản lý bộ nhớ.....	24
4.1.2 Cơ chế xử lý tiến trình	28
4.2 Đánh giá sự khác biệt và tương đồng về quản lý bộ nhớ giữa hai hệ thống..	30
4.2.1 Sự tương đồng	30
4.2.2 Sự khác biệt.....	31

CHƯƠNG 5. QUẢN LÝ FILE..... 33

5.1 Cách quản lý file.....	33
5.2 Truy cập file.....	42
5.3 Bảo mật file	43
5.4 Quyền truy cập	46

CHƯƠNG 6. SO SÁNH VÀ ĐẶC ĐIỂM RIÊNG..... 48

6.1 Các điểm tương đồng và khác biệt chính giữa macOS và iOS về cấu trúc hệ thống, hàm Shell, quản lý bộ nhớ và quản lý file 48

 6.1.1 Các điểm tương đồng về cấu trúc hệ thống, hàm Shell, quản lý bộ nhớ và quản lý file 48

 6.1.2 Khác biệt chính giữa macOS và iOS về cấu trúc hệ thống, hàm Shell, quản lý bộ nhớ và quản lý file 49

6.2 Ưu điểm và hạn chế của hai hệ điều hành về cấu trúc hệ thống, hàm Shell, quản lý bộ nhớ và quản lý file 50

 6.2.1 Về ưu điểm 50

 6.2.2 Về mặt hạn chế 51

CHƯƠNG 7. ỨNG DỤNG VÀ HIỆU SUẤT 53

7.1 Hiệu suất ứng dụng trên macOS và iOS 53

 7.1.1 So sánh hiệu suất 53

 7.1.2 Ví dụ về hiệu suất của ứng dụng iMovie thiết bị macOS và iOS thông thường 54

7.2 Cách phát triển ứng dụng 55

CHƯƠNG 8. BẢO MẬT VÀ QUẢN LÝ THIẾT BỊ..... 58

8.1 Bảo mật trên macOS vs iOS 58

8.2 Quản lý thiết bị 60

 8.2.1 Quản lý cấu hình và dịch vụ 60

 8.2.2 Quản lý ứng dụng 63

CHƯƠNG 9. HỖ TRỢ HỆ THỐNG VÀ CẬP NHẬT 65

9.1 Hỗ trợ hệ thống 65

 9.1.1 Dịch vụ hỗ trợ macOS và iOS 65

 9.1.2 Cộng đồng hỗ trợ macOS và iOS 65

9.2 Cập nhật hệ thống 65

CHƯƠNG 10. KẾT LUẬN..... 69

DANH MỤC HÌNH VẼ

Hình 1.1	Macbook Air sử dụng hệ điều hành macOS	1
Hình 1.2	Hệ điều hành iOS	3
Hình 1.3	Các phiên bản iOS	4
Hình 2.1	Biểu đồ mô tả cấu trúc macOS	6
Hình 2.2	Cấu trúc lõi Darwin	6
Hình 2.3	Giao diện người dùng Aqua	9
Hình 2.4	Kiến trúc lớp iOS	10
Hình 2.5	Lõi OS của IOS	11
Hình 2.6	Dịch vụ lõi IOS	11
Hình 2.7	Tầng phương tiện của IOS	12
Hình 2.8	Cocoa Touch của IOS	12
Hình 4.1	Virtual Memory	24
Hình 4.2	Life cycle	26
Hình 4.3	Grand Central Dispatch	27
Hình 4.4	Automatic Reference Counting	27
Hình 4.5	Round Robin Scheduling	28
Hình 4.6	Shortest Job First	29
Hình 4.7	Multilevel Feedback Queue	30
Hình 4.8	Paging và Swapping	31
Hình 5.1	Cấu trúc của HFS+.	34
Hình 5.2	Cấu trúc của APFS.	36
Hình 5.3	Minh họa về Space Sharing.	36
Hình 5.4	Clone file trong APFS.	37
Hình 5.5	Vị trí của Bảo mật và Quyền riêng tư.	38
Hình 5.6	Ứng dụng Files.	39
Hình 5.7	Cách thiết lập iCloud Drive.	40
Hình 5.8	Minh họa về AirDrop trên iPhone.	41
Hình 5.9	AirDrop trên Macbook.	42
Hình 5.10	Các chế độ xem trên Macbook.	42
Hình 5.11	Thay đổi quyền cho thư mục	44
Hình 8.1	Quản lý cho phép ứng dụng theo dõi trên iOS	59
Hình 8.2	Quản lý cấu hình và dịch vụ trong ứng dụng cài đặt của iOS	61
Hình 8.3	Tính năng MDM	61

Hình 8.4	Mở ứng dụng cài đặt trong iOS	62
Hình 8.5	Cài đặt dịch vụ wifi	63
Hình 8.6	Bật tắt wifi	63
Hình 9.1	Thông báo cập nhật macOS	66
Hình 9.2	Thông báo cập nhật iOS	66

CHƯƠNG 1. GIỚI THIỆU VỀ MACOS VÀ IOS

1.1 Hệ điều hành macOS

1.1.1 Lịch sử và nguồn gốc của hệ điều hành macOS

Hệ điều hành macOS là một hệ điều hành dựa trên Unix được phát triển bởi Apple Inc và được trang bị cho các dòng máy tính xách tay của hãng Apple.



Hình 1.1: Macbook Air sử dụng hệ điều hành macOS

MacOS là tên viết tắt xuất phát từ cụm từ Macintosh operating system. Bắt đầu từ phiên bản sơ khai nhất được giới thiệu vào năm 1984, hệ điều hành macOS đã trải qua một thời gian dài với nhiều tên gọi khác nhau trước khi được biết đến rộng rãi với cái tên hiện tại.

- Mac OS X(2001-2012): được thiết kế với giao diện cửa sổ và có nhiều tính năng và công nghệ tiên tiến như quản lý đa nhiệm, bảo mật cao, và tích hợp các ứng dụng và dịch vụ của Apple.
- OS X(2012-2016): kết hợp các tính năng của hệ điều hành trước đó với nền tảng Unix. OS X đem lại các cải tiến về hiệu suất và tính năng, bao gồm tích hợp iCloud, AirDrop và nhiều cải tiến khác.
- macOS(2016-nay): để phù hợp với các hệ điều hành khác của họ như iOS, watchOS và tvOS, Apple đã quyết định thay đổi tên gọi của

hệ điều hành thành "macOS". Bên cạnh đó Apple cũng cải tiến giao diện người dùng, hiệu suất và tối ưu hóa hệ thống, tích hợp tốt với các thiết bị và dịch vụ Apple, mức độ bảo mật cao, và cung cấp các ứng dụng và tính năng (trợ lý ảo của Apple Siri,...).

Hệ điều hành macOS đã trải qua nhiều sự thay đổi và cải tiến trong suốt hơn ba thập kỷ. Nó đã trở thành một hệ điều hành mạnh mẽ và phổ biến trong ngành công nghiệp công nghệ thông tin và thiết bị di động.

1.1.2 Ưu điểm và nhược điểm của hệ điều hành macOS

- **Ưu điểm**

- Giao diện dễ nhìn: giao diện mà Apple tạo ra luôn thu hút một tập người dùng lớn trên toàn cầu. Hệ điều hành macOS sở hữu một giao diện với tính thẩm mỹ rất cao được tối ưu hóa theo trải nghiệm của khách hàng.
- Hiệu năng ổn định và bền bỉ: macOS được xây dựng trên cơ sở Unix giúp nó hoạt động mượt mà và ổn định hơn so với các hệ điều hành khác.
- Bảo mật: macOS được thiết kế với nhiều tính năng bảo mật như Gatekeeper, XProtect, và sandboxing để ngăn chặn ứng dụng độc hại, bảo vệ dữ liệu và quyền riêng tư của người dùng.
- App Store: Cho phép người dùng tải xuống và cài đặt ứng dụng một cách dễ dàng và an toàn. Ứng dụng trên App Store thường được kiểm duyệt để đảm bảo tính bảo mật.
- Tích hợp dễ dàng với các sản phẩm Apple khác: macOS tương thích tốt với các sản phẩm và dịch vụ khác của Apple như iPhone, iPad, iCloud và Apple Watch, tạo điều kiện thuận lợi cho việc đồng bộ hóa dữ liệu và trải nghiệm người dùng liền mạch.
- Trợ lý ảo Siri: là một trợ lý ảo mạnh mẽ có khả năng tương tác với nhiều khía cạnh của cuộc sống hàng ngày, giúp bạn thực hiện các tác vụ một cách thuận tiện và nhanh chóng thông qua giọng nói.

- **Nhược điểm**

- Giá thành cao: Các dòng máy sử dụng hệ điều hành macOS có giá cả khá cao so với các máy tính chạy hệ điều hành Windows hoặc Linux với cấu hình tương tự.
- Giới hạn phần cứng: macOS chỉ chạy trên các máy tính của Apple,

nhiều MacBooks, iMacs và Mac Pros. Điều này có nghĩa là bạn không thể cài đặt macOS trên các máy tính cá nhân khác hoặc máy tính xách tay của các nhà sản xuất khác.

- Phạm vi ứng dụng giới hạn: macOS có một số ứng dụng tùy chỉnh, nhưng cửa hàng ứng dụng Mac App Store không có số lượng ứng dụng lớn như Google Play Store hoặc Microsoft Store.

1.2 Hệ điều hành iOS

1.2.1 Lịch sử và nguồn gốc của hệ điều hành iOS

Cũng như macOS, iOS là hệ điều hành di động được phát triển bởi Apple Inc dành cho các thiết bị di động của họ bao gồm iPhone, iPad và iPod Touch.



Hình 1.2: Hệ điều hành iOS

iOS ban đầu được gọi là "iPhone OS" và được giới thiệu lần đầu vào năm 2007, đó cũng là lúc chiếc iPhone thế hệ thứ nhất xuất hiện trên thế giới. Phiên bản đầu tiên của hệ điều hành này không hỗ trợ các ứng dụng của bên thứ ba, đi kèm với giao diện đơn giản và đánh dấu sự xuất hiện của các tính năng cơ bản như điều hướng cảm ứng, trình duyệt Safari, và ứng dụng âm nhạc.

Từ năm 2010, hệ điều hành này được đổi tên thành "iOS" để phản ánh việc hỗ trợ cho nhiều thiết bị di động của Apple hơn chỉ iPhone.

Cho đến ngày nay, iOS đã trở thành một trong những hệ điều hành di động phổ biến và ảnh hưởng nhất trên thế giới. Nó đã trải qua nhiều phiên

bản và nâng cấp lớn. Các phiên bản mới của iOS thường được giới thiệu hàng năm và cung cấp cải tiến về giao diện người dùng, tính năng mới, hiệu suất và bảo mật.

Các phiên bản hệ điều hành	Ngày phát hành
iOS 1	29/6/2007
iOS 2	7/2008
iOS 3	17/6/2009
iOS 4	21/6/2010
iOS 5	6/6/2011
iOS 6	11/6/2012
iOS 7	10/6/2013
iOS 8	2/6/2014
iOS 9	8/6/2015
iOS 10	13/6/2016
iOS 11	5/6/2017
iOS 12	4/6/2018
iOS 13	3/6/2019
iOS 14	22/06/2020
iOS 15	08/06/2021
iOS 16	07/06/2022

Hình 1.3: Các phiên bản iOS

1.2.2 Ưu điểm và nhược điểm của hệ điều hành iOS

- **Ưu điểm**

- Giao diện: iOS được thiết kế với giao diện đẹp mắt, dễ sử dụng và trực quan. Các biểu tượng và các thành phần của giao diện được tối ưu hóa để cung cấp trải nghiệm người dùng tốt.
- Ổn định và bảo mật: Hệ điều hành này được kiểm tra kỹ lưỡng trước khi phát hành và chỉ chạy trên các thiết bị được chứng nhận bởi Apple giúp giảm thiểu rủi ro từ các lỗ hổng bảo mật và đảm bảo rằng người dùng có trải nghiệm tốt.
- Tích hợp tốt với các dịch vụ Apple: iOS được tích hợp tốt với các dịch vụ của Apple như iCloud, iMessage, FaceTime và Apple Pay. Người dùng có thể dễ dàng đồng bộ hóa dữ liệu và truy cập vào các dịch vụ này trên các thiết bị khác nhau.
- Cập nhật phiên bản liên tục: Phiên bản mới sẽ được ra mắt thường xuyên (ít nhất 1 lần/năm). Do đó, nếu gặp lỗi gì ở hệ điều hành hiện tại, người dùng hoàn toàn có thể nâng cấp lên bản mới để khắc phục lỗi.
- Ứng dụng phong phú: App Store của Apple cung cấp một hệ sinh

thái ứng dụng phong phú với hàng ngàn ứng dụng và trò chơi. Người dùng có thể tìm thấy các ứng dụng từ các lĩnh vực khác nhau như giáo dục, giải trí, sức khỏe, kinh doanh,...

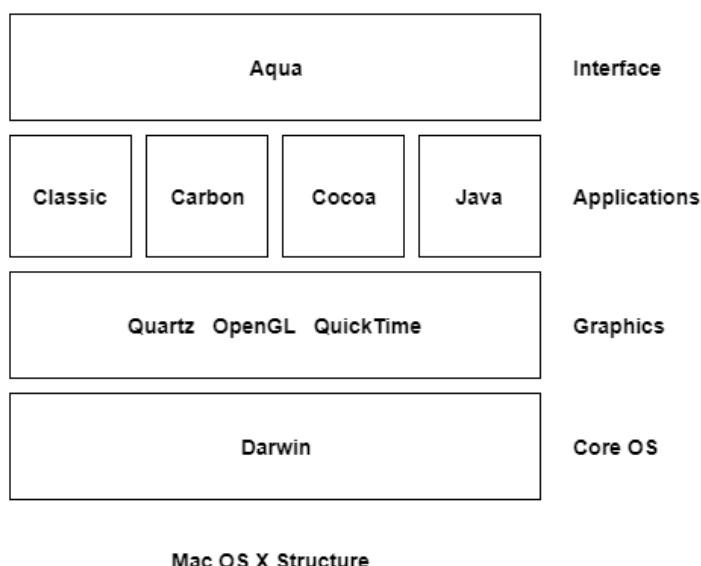
- Nhược điểm

- Giới hạn trong việc chia sẻ dữ liệu: iOS có nhiều hạn chế khi chia sẻ dữ liệu giữa các ứng dụng và thiết bị. Apple thiết kế hệ thống bảo mật mạnh mẽ, nhưng điều này có thể làm cho việc chia sẻ dữ liệu giữa các ứng dụng trở nên khó khăn.
- Hạn chế sự linh hoạt: iOS được thiết kế để hoạt động trên các thiết bị của Apple và không cho phép sự tùy chỉnh mức độ cao như các hệ điều hành mã nguồn mở khác. Điều này có nghĩa rằng người dùng không thể thay đổi giao diện, cài đặt các ứng dụng từ các nguồn không chính thống và làm nhiều thay đổi hệ thống.
- Giá thành cao: Thiết bị chạy iOS, như iPhone và iPad, thường có giá cao hơn so với các thiết bị chạy hệ điều hành khác. Điều này có thể là một rào cản đối với một số người tiêu dùng.

CHƯƠNG 2. CẤU TRÚC HỆ THỐNG

2.1 macOS

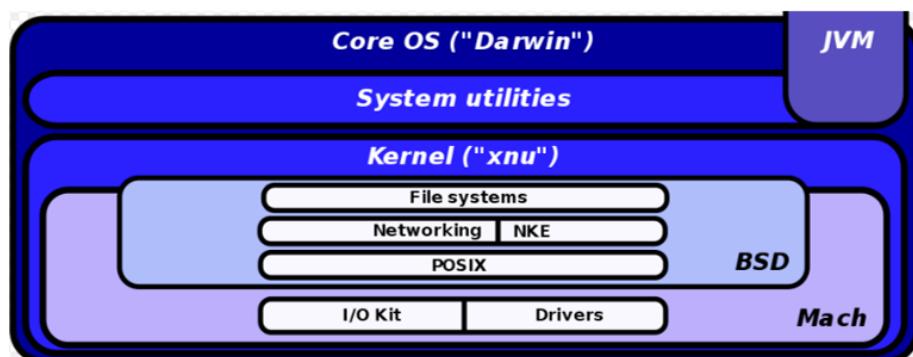
- Cấu trúc của macOS bao gồm nhiều lớp. Lớp cơ bản là Darwin, là lõi Unix của hệ thống. Lớp tiếp theo là hệ thống đồ họa, bao gồm Quartz, OpenGL và QuickTime. Sau đó là lớp ứng dụng với bốn thành phần, gồm Classic, Carbon, Cocoa và Java. Lớp trên cùng là Aqua, giao diện người dùng.
- Một biểu đồ mô tả cấu trúc của macOS như sau:



Hình 2.1: Biểu đồ mô tả cấu trúc macOS

Chi tiết về các thành phần khác nhau của cấu trúc macOS X như được thể hiện trong hình ảnh trên là như sau:

2.1.1 Hệ thống cơ bản (Core OS)



Hình 2.2: Cấu trúc lõi Darwin

- Lõi Darwin của macOS là một hệ điều hành mã nguồn mở được xây dựng dựa trên phiên bản UNIX của BSD (Berkeley Software Distribution). Hệ điều hành BSD là một biến thể của UNIX, một hệ thống máy tính mạnh mẽ và đa dạng được phát triển từ cuối những năm 1960. Trong ngữ cảnh của Darwin, hệ điều hành cơ sở của macOS, khi nói về Unix, người ta thường hiểu theo nghĩa của một hệ điều hành tuân theo các chuẩn như POSIX (Portable Operating System Interface) và ISO/IEC 9945.
- Darwin bao gồm nhiều thành phần quan trọng, trong đó Mach kernel là một phần trọng yếu. Mach kernel là một loại kernel (lõi hệ điều hành) phát triển tại Đại học Carnegie Mellon. Kernel quản lý quy trình, phân bổ và quản lý bộ nhớ, và thực hiện các thao tác quan trọng khác như luồng dữ liệu từ và đến CPU.
- Darwin là một dự án mã nguồn mở, điều này có nghĩa là bất kỳ ai đều có thể truy cập mã nguồn của nó, thay đổi và sử dụng nó dưới các điều kiện của Giấy phép Công cộng GNU (GNU General Public License - GPL) và Giấy phép Apache.
- Các phiên bản khác nhau của Darwin có thể được sử dụng để cải thiện hệ điều hành macOS X. Darwin cung cấp nền tảng cho nhiều tính năng quan trọng của macOS, bao gồm quản lý bộ nhớ được bảo vệ, quản lý bộ nhớ tự động, lập lịch đa nhiệm tiên đoán, quản lý bộ nhớ ảo tiên tiến, và dịch vụ I/O cho hệ điều hành Mac OS X. Nó cũng hỗ trợ các tính năng như cắm và chơi, thay thế nóng, và quản lý năng lượng để tối ưu hóa hiệu suất hệ thống.
- Tóm lại, Darwin là cơ sở của macOS, với sự kết hợp của các thành phần UNIX và kernel Mach, cung cấp một nền tảng mạnh mẽ và đáng tin cậy cho hệ điều hành của Apple.

2.1.2 Hệ thống đồ họa (Graphics Subsystem)

- Quartz: Quartz là thành phần quản lý đồ họa 2D quan trọng trong hệ thống macOS, có nhiệm vụ xử lý và hiển thị hình ảnh, văn bản, và đồ họa giao diện người dùng. Cung cấp chức năng vẽ hình, hiển thị phông chữ, xử lý hình ảnh, và hỗ trợ hiệu ứng đồ họa 2D, làm cho các ứng dụng và giao diện người dùng trên macOS trở nên trực quan và hấp dẫn. Quartz là một framework quan trọng trong hệ điều hành, bao gồm cả API như Quartz Composer, ImageKit, và PDFKit, hỗ trợ xử lý đồ họa và tương tác với nhiều định dạng nội

dung đa phương tiện.

- OpenGL: OpenGL là một tiêu chuẩn đồ họa 3D mạnh mẽ và đa năng. Trong hệ thống đồ họa của macOS, OpenGL cung cấp hỗ trợ cho đồ họa 3D, bao gồm ánh xạ texture, độ trong suốt, chống răng cửa (antialiasing), hiệu ứng khí quyển, hiệu ứng đặc biệt, và nhiều tính năng đồ họa 3D khác. Nó cho phép các ứng dụng và trò chơi 3D chạy trên macOS tận dụng sức mạnh của đồ họa 3D.
- QuickTime: QuickTime là một khung phương tiện kỹ thuật số đa phương tiện được sử dụng trong macOS. Nó hỗ trợ nhiều loại phương tiện kỹ thuật số như video số, âm thanh, hình ảnh, và cả video trực tuyến. QuickTime cung cấp thư viện và công cụ để phát lại, chỉnh sửa và xử lý phương tiện số. Nó còn cho phép các ứng dụng sáng tạo như iMovie và iTunes tương tác với các phương tiện số một cách dễ dàng.
- Các thành phần này làm cho hệ thống đồ họa trong macOS mạnh mẽ và đa dạng, cho phép các ứng dụng và giao diện người dùng hiển thị đồ họa 2D và 3D, xử lý phương tiện số, và cung cấp trải nghiệm người dùng tốt hơn.

2.1.3 Hệ thống ứng dụng (Application Subsystem)

- Carbon: Carbon là một môi trường phát triển ứng dụng dành cho các ứng dụng cổ điển đã được viết cho phiên bản trước của hệ điều hành Mac, chẳng hạn như Mac OS 9. Nó giúp đảm bảo rằng các ứng dụng cổ điển có thể tiếp tục hoạt động mượt mà trên macOS mà không cần viết lại toàn bộ mã nguồn. Quá trình chuyển đổi ứng dụng cổ điển sang môi trường Carbon được gọi là "carbon hóa ứng dụng." Một số ứng dụng nổi tiếng trong môi trường Carbon trước khi Apple chuyển sang Cocoa như: Quicken 2007, Final Cut Pro 7, iTunes,...
- Cocoa: Cocoa là môi trường phát triển ứng dụng hướng đối tượng. Nó cung cấp một cách tiếp cận hiện đại cho việc phát triển ứng dụng trên macOS. Cocoa sử dụng ngôn ngữ lập trình Objective-C và hỗ trợ một loạt các khung làm việc và giao diện người dùng để xây dựng ứng dụng với tính năng cao cấp và hiệu suất tốt. Các ứng dụng Cocoa tận dụng mạnh mẽ cấu trúc của macOS X. Một số ứng dụng nổi tiếng trong môi trường Cocoa như: Finder, Safari,

Xcode,...

- Java: Java là môi trường phát triển ứng dụng cho các ứng dụng và applet được viết bằng ngôn ngữ lập trình Java. Nó cho phép các ứng dụng Java chạy trên macOS và tương thích với nhiều nền tảng khác nhau. Java cung cấp tính di động và độc lập nền tảng, cho phép các ứng dụng Java chạy trên nhiều hệ điều hành khác nhau. Tuy nhiên, hiện nay Apple đã ngừng tích hợp hỗ trợ Java runtime mặc định. Người dùng cần tự cài đặt Java runtime và theo dõi thông báo về giảm bớt hỗ trợ trong các phiên bản macOS sắp tới.
- Hệ thống ứng dụng (Application Subsystem) của macOS cung cấp môi trường phát triển đa dạng cho các nhà phát triển, cho phép họ xây dựng các loại ứng dụng từ các ứng dụng cổ điển đến các ứng dụng hướng đối tượng và ứng dụng Java, đáp ứng nhu cầu và yêu cầu của họ. Các môi trường này giúp đảm bảo tính linh hoạt và sự phong phú trong việc phát triển ứng dụng trên macOS.

2.1.4 Giao diện người dùng (User interface)



Hình 2.3: Giao diện người dùng Aqua

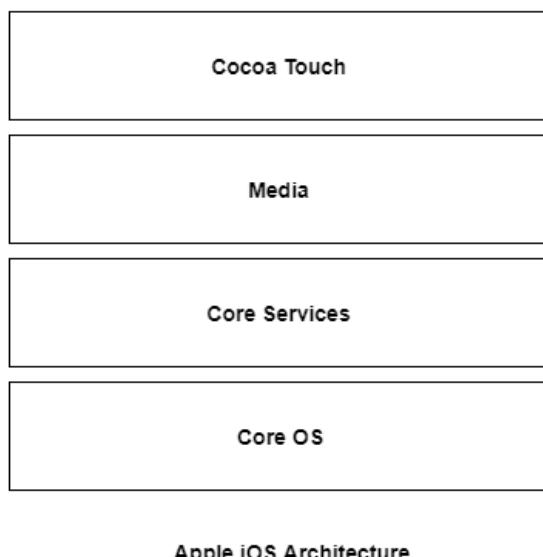
- Giao diện người dùng (User Interface - UI) của macOS được thiết kế để cung cấp tính năng trực quan tốt và thuận tiện cho người dùng. Nó chứa nhiều màu sắc, biểu tượng chi tiết, và hiệu ứng đặc biệt để làm cho việc sử dụng máy tính trở nên đẹp mắt và dễ dàng. Aqua là một giao diện đồ họa, ngôn ngữ thiết kế và chủ đề trực

quan cho hệ điều hành macOS của Apple.

- Tên "Aqua" không chỉ nói lên tính hấp dẫn của giao diện người dùng mà còn tạo nên một nhận thức đặc biệt cho thương hiệu và sự độc đáo của macOS trong việc cung cấp trải nghiệm người dùng độc nhất vô nhị.

2.2 iOS

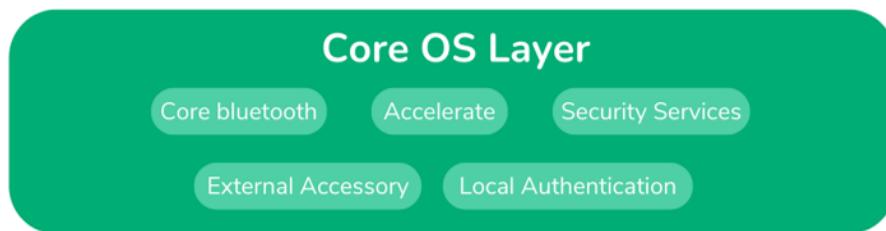
- Kiến trúc của iOS có cấu trúc lớp. Nó chứa một lớp trung gian giữa các ứng dụng và phần cứng để họ không giao tiếp trực tiếp với nhau. Các lớp ở phía dưới trong iOS cung cấp các dịch vụ cơ bản và các lớp ở phía trên cung cấp giao diện người dùng và đồ họa phức tạp.
- Kiến trúc lớp của iOS được trình bày như sau:



Hình 2.4: Kiến trúc lớp iOS

Tất cả các yếu tố của kiến trúc iOS được chia thành các lớp khác nhau để cung cấp các tính năng và dịch vụ cụ thể cho các ứng dụng di động. Dưới đây là mô tả chi tiết về từng lớp trong kiến trúc iOS:

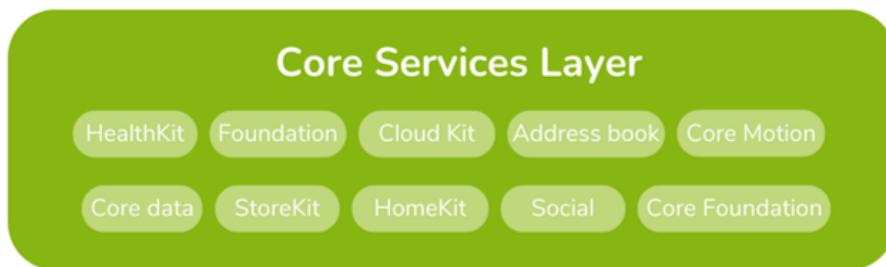
2.2.1 Lõi OS (Core OS):



Hình 2.5: Lõi OS của IOS

- Lớp Lõi OS cung cấp các dịch vụ cơ bản cho hệ thống iOS.
- Các công nghệ quan trọng được xây dựng trên các tính năng cấp thấp của lớp này, bao gồm Core Bluetooth Framework cho việc kết nối Bluetooth, External Accessory Framework để tương tác với các thiết bị ngoại vi, Accelerate Framework cho tính toán nhanh, Security Services Framework để quản lý bảo mật, Local Authorisation Framework cho quyền truy cập địa phương, và nhiều công nghệ khác.

2.2.2 Dịch vụ lõi (Core Services):



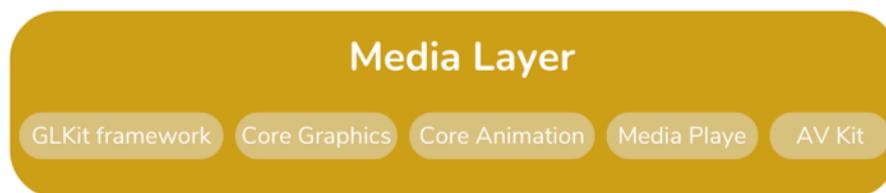
Hình 2.6: Dịch vụ lõi IOS

- Lớp Dịch vụ Lõi bao gồm nhiều khung (frameworks) cho các dịch vụ và quản lý dữ liệu.
- Một số khung quan trọng trong lớp này bao gồm:
 - Cloudkit Framework: Cho phép di chuyển dữ liệu giữa ứng dụng và iCloud.
 - Core Foundation Framework: Cung cấp quản lý dữ liệu và tính năng dịch vụ cho ứng dụng iOS.
 - Core Data Framework: Quản lý mô hình dữ liệu trong các ứng dụng theo kiến trúc Mô hình - Xem - Điều khiển.
 - Address Book Framework: Cung cấp quyền truy cập vào cơ sở

dữ liệu danh bạ của người dùng.

- Core Motion Framework: Cho phép truy cập dữ liệu dựa trên chuyển động trên thiết bị.
- Healthkit Framework: Xử lý thông tin liên quan đến sức khỏe của người dùng.
- Core Location Framework: Cung cấp thông tin vị trí và hướng cho các ứng dụng.

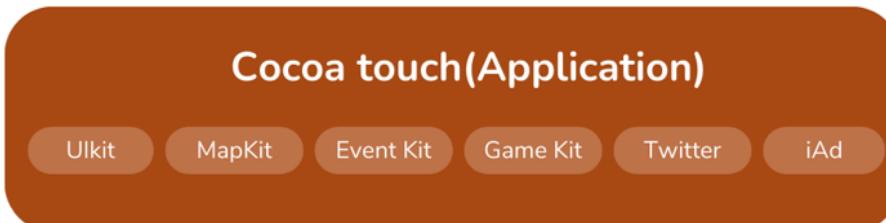
2.2.3 Phương tiện (Media):



Hình 2.7: Tầng phương tiện của IOS

- Lớp Phương tiện quản lý công nghệ đồ họa, âm thanh và video của hệ thống.
- Một số khung quan trọng bao gồm:
 - UIKit Graphics: Hỗ trợ thiết kế hình ảnh và tạo hoạt hình cho nội dung chế độ xem.
 - Core Graphics Framework: Hỗ trợ vẽ dựa trên vector 2D và dựa trên hình ảnh, là công cụ vẽ mặc định cho ứng dụng iOS.
 - Core Animation: Tối ưu hóa trải nghiệm hoạt hình trong ứng dụng.
 - Media Player Framework: Hỗ trợ phát danh sách phát và cho phép người dùng truy cập thư viện iTunes của họ.
 - AV Kit: Cung cấp giao diện dễ sử dụng cho trình bày video.

2.2.4 Cocoa Touch:



Hình 2.8: Cocoa Touch của IOS

- Lớp chạm (Cocoa Touch) chứa các khung và dịch vụ cung cấp các tính năng cụ thể cho ứng dụng di động.
- Một số khung quan trọng bao gồm:
 - EventKit Framework: Hiển thị giao diện hệ thống cho việc xem và chỉnh sửa sự kiện liên quan đến lịch.
 - GameKit Framework: Hỗ trợ người dùng chia sẻ dữ liệu liên quan đến trò chơi trực tuyến thông qua Game Center.
 - MapKit Framework: Cung cấp một bản đồ có thể cuộn được tích hợp vào giao diện ứng dụng.

Các lớp này tạo nên kiến trúc phức tạp của iOS và cho phép các ứng dụng di động phát triển với nhiều tính năng và khả năng đa dạng.

2.3 So sánh cấu trúc hệ thống macOS và iOS

Cấu trúc hệ thống của macOS và iOS có một số điểm tương đồng, bao gồm việc chia thành nhiều lớp để cung cấp các tính năng, dịch vụ cụ thể cho các ứng dụng và đều được dựa trên Unix. Tuy nhiên, cả hai hệ điều hành đều có mục tiêu và ứng dụng chính khác nhau, nên cấu trúc của chúng cũng có một số điểm khác biệt quan trọng. Dưới đây là một số điểm so sánh chính giữa cấu trúc hệ thống của macOS và iOS:

2.3.1 Core OS:

- Cả macOS và iOS đều chia sẻ một nền tảng cơ bản là Darwin, hệ điều hành cơ bản dựa trên mã nguồn mở, với hạt nhân XNU và thành phần của BSD. Tuy nhiên, có những khác biệt quan trọng giữa Core OS của macOS và iOS khi chúng được triển khai trong các hệ điều hành cụ thể.
- Core OS trên macOS tập trung vào dịch vụ cho máy tính cá nhân, bao gồm đa nhiệm và quản lý tệp. Trong khi đó, Core OS trên iOS tối ưu hóa cho môi trường di động, chú trọng vào quản lý tệp, mạng, và bảo mật để hỗ trợ trải nghiệm người dùng cảm ứng trên iPhone và iPad.

2.3.2 Hệ thống đồ họa (Graphics Subsystem):

- macOS có một hệ thống đồ họa mạnh mẽ với các thành phần như Quartz cho đồ họa 2D và OpenGL cho đồ họa 3D. QuickTime cũng được sử dụng để quản lý phương tiện số.
- iOS cũng có hệ thống đồ họa, nhưng chúng tập trung vào giao diện

người dùng đa phương tiện, đặc biệt là UIKit, với hỗ trợ cho đồ họa vector 2D. Nó không sử dụng OpenGL cho ứng dụng bình thường, mà thường sử dụng Metal cho đồ họa 3D.

2.3.3 Hệ thống ứng dụng (Application Subsystem):

- macOS có môi trường ứng dụng đa dạng với các môi trường phát triển như Carbon (để hỗ trợ các ứng dụng cổ điển), Cocoa (để xây dựng ứng dụng hướng đối tượng), và hỗ trợ cho ứng dụng Java.
- iOS có môi trường phát triển chủ yếu dựa trên Cocoa Touch, mà thường sử dụng ngôn ngữ lập trình Objective-C hoặc Swift. Nó không hỗ trợ ứng dụng cổ điển và chủ yếu tập trung vào việc phát triển ứng dụng di động hiện đại.

2.3.4 Giao diện người dùng (User Interface):

- Aqua là giao diện người dùng chính của macOS, với thiết kế đẹp và hiệu quả. Tên "Aqua" đã tạo ra sự nhận thức và thương hiệu cho macOS.
- iOS tập trung vào cung cấp giao diện người dùng đa phương tiện cho thiết bị di động, sử dụng giao diện Cocoa Touch. Không có tên cụ thể cho giao diện người dùng của iOS như "Aqua."

Tóm lại, cấu trúc hệ thống của macOS và iOS có nhiều điểm tương đồng, nhưng cũng tồn tại nhiều sự khác biệt quan trọng. Cả hai hệ điều hành đều dựa trên các lớp cơ bản tương tự, nhưng macOS hướng tới việc cung cấp môi trường đa dạng cho các ứng dụng trên máy tính cá nhân, trong khi iOS tập trung vào việc cung cấp trải nghiệm di động cho các thiết bị cầm tay và máy tính bảng. Sự khác biệt này thể hiện qua thiết kế, hệ thống đồ họa, hệ thống ứng dụng, giao diện người dùng và số lượng khung cho hai hệ điều hành.

CHƯƠNG 3. HÀM SHELL VÀ LỜI GỌI HỆ THỐNG

3.1 Hàm Shell trong MacOS và IOS

3.1.1 Terminal là gì?

Terminal là một giao diện dạng dòng lệnh, cho phép người dùng có thể sử dụng các câu lệnh Unix để có thể hoàn thành các tác vụ trên máy một cách nhanh chóng so với việc sử dụng giao diện GUI, với công cụ lệnh này, người dùng có thể mở tệp, cải thiện được hiệu suất máy của mình nhiều hơn.

3.1.2 Giới thiệu về thiết bị Terminal macOS

Terminal cho phép người dùng tương tác với hệ thống macOS thông qua giao diện dòng lệnh thay vì giao diện đồ họa. Nó cung cấp quyền truy cập vào Unix shell (Unix shell là một chương trình giao diện dòng lệnh (CLI - Command Line Interface) cho phép người dùng tương tác với hệ thống máy tính dựa trên Unix hoặc Unix-like thông qua việc nhập lệnh văn bản. Shell đóng vai trò như một giao diện giữa người dùng và hệ thống, cho phép thực hiện các lệnh, chạy các chương trình, quản lý tập tin và thư mục, xử lý dữ liệu, và thực hiện các tác vụ hệ thống khác), thường là Bash shell (Bash (Bourne Again SHell) là một trong những shell phổ biến nhất trên các hệ thống Unix và Unix-like. Nó là một phiên bản mở rộng và cải tiến của Bourne shell (sh), mặc định trên hầu hết các phiên bản Linux và cũng được sử dụng rộng rãi trên macOS. Bash cung cấp các tính năng mạnh mẽ bao gồm cú pháp mở rộng, biến, vòng lặp, điều kiện, và nhiều công cụ hữu ích khác để thực hiện và tự động hóa các tác vụ hệ thống).

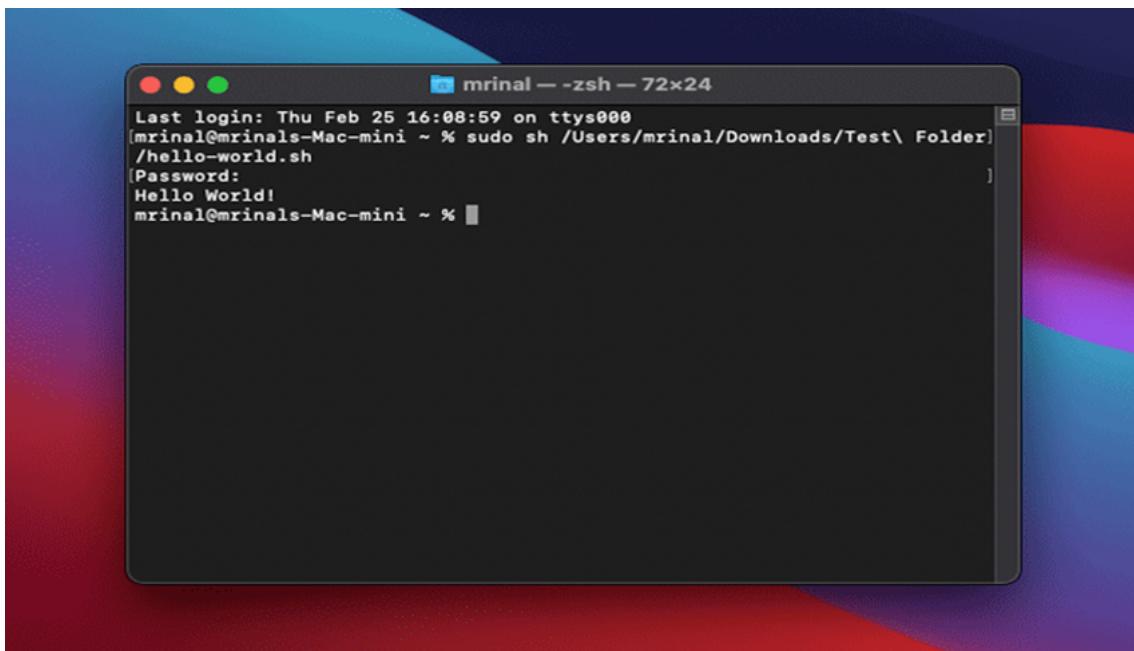
Người dùng có thể nhập các lệnh, chạy các công cụ dòng lệnh, thực thi các tập lệnh (shell scripts) để tự động hóa các tác vụ và quản lý hệ thống.

Terminal cho phép nhiều phiên bản cửa sổ dòng lệnh để chạy đồng thời nhiều quy trình. Nó hỗ trợ các tính năng như hoàn thành lệnh tự động, lịch sử lệnh, màu sắc phân biệt cú pháp, kiểm soát việc chạy các tiến trình nền.

Terminal rất hữu ích cho các nhà phát triển, quản trị viên hệ thống để có thể truy cập sâu vào macOS. Nó cũng hữu ích cho người dùng thông thường trong một số tác vụ cụ thể. Terminal được tích hợp sẵn trong mọi phiên bản macOS nên người dùng có thể sử dụng luôn mà không cần cài đặt thêm phần mềm nào.

3.1.3 Mở Terminal trên Mac OS

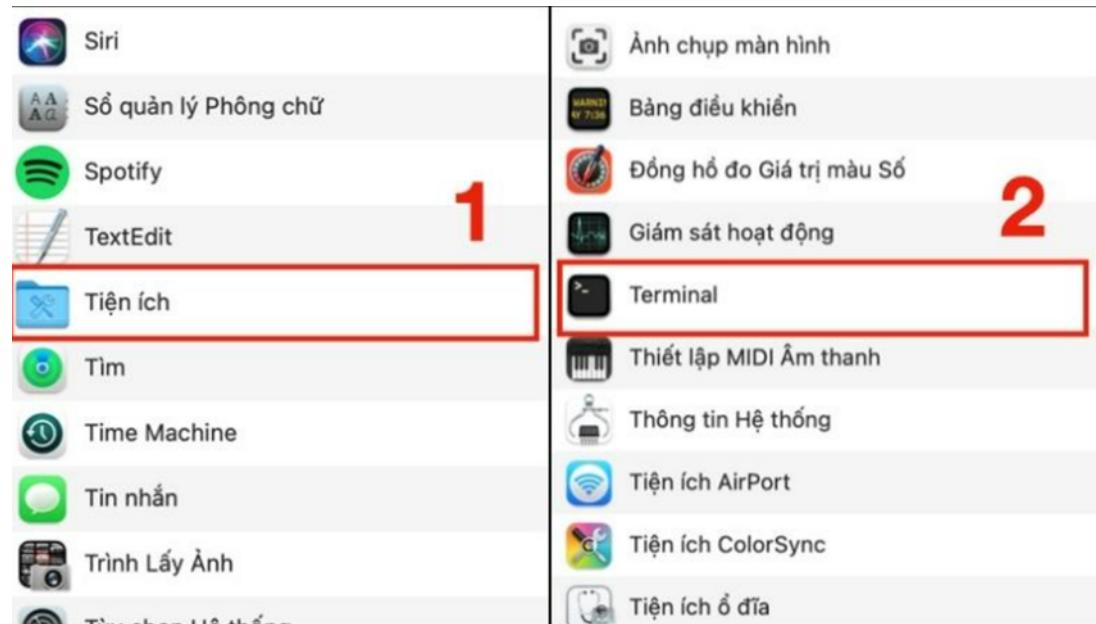
Mở Terminal từ Finder



Bước 1: Truy cập vào Finder -> sau đó bấm vào thư mục Ứng dụng(Aplications)

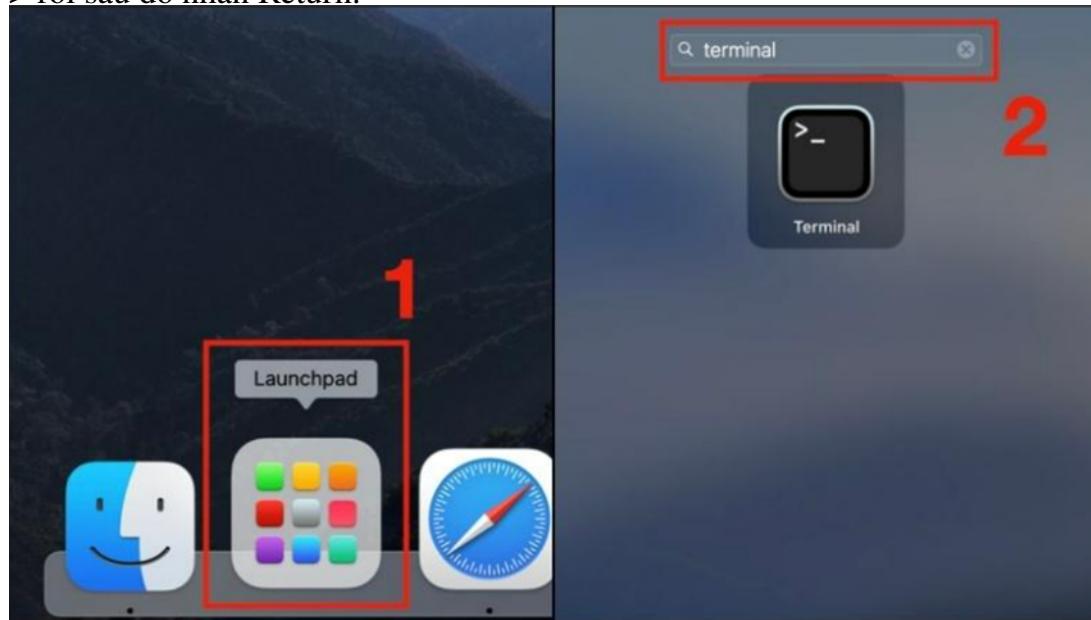


Bước 2: Sau đó hãy chọn vào dòng mục Tiện ích (Utilities) > rồi sau đó tìm kiếm và mở ứng dụng Terminal .



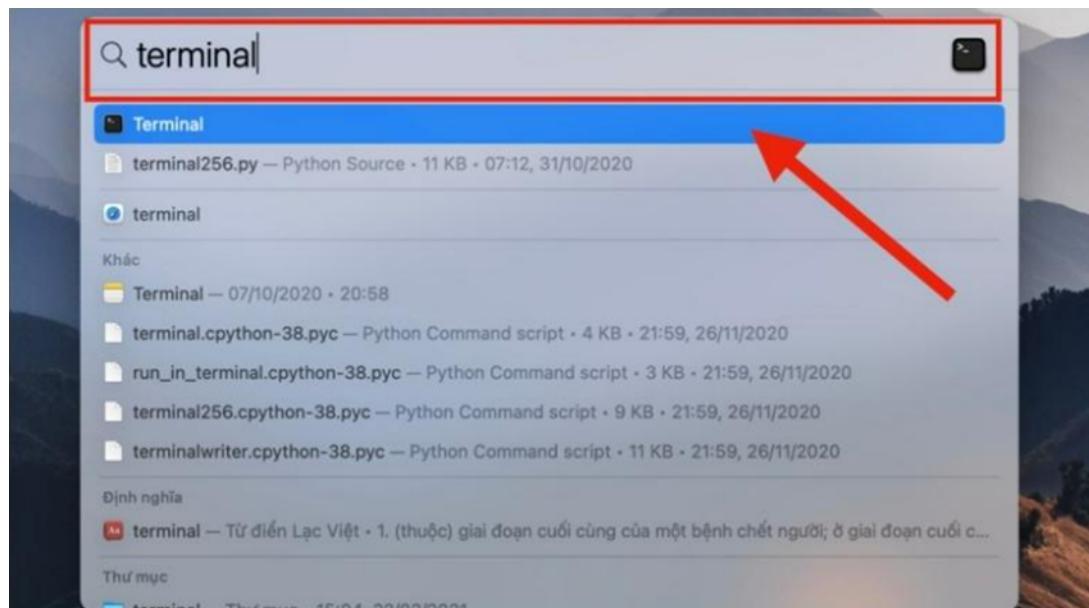
Mở Terminal từ Launchpad

Đầu tiên bấm vào Launchpad hoặc có thể chụm 4 ngón tay đặt trên Touchpad để mở > sau đó hãy gõ vào thanh Tìm kiếm trên hộp thoại tìm kiếm Terminal > rồi sau đó nhấn Return.



Mở Terminal từ Spotlight

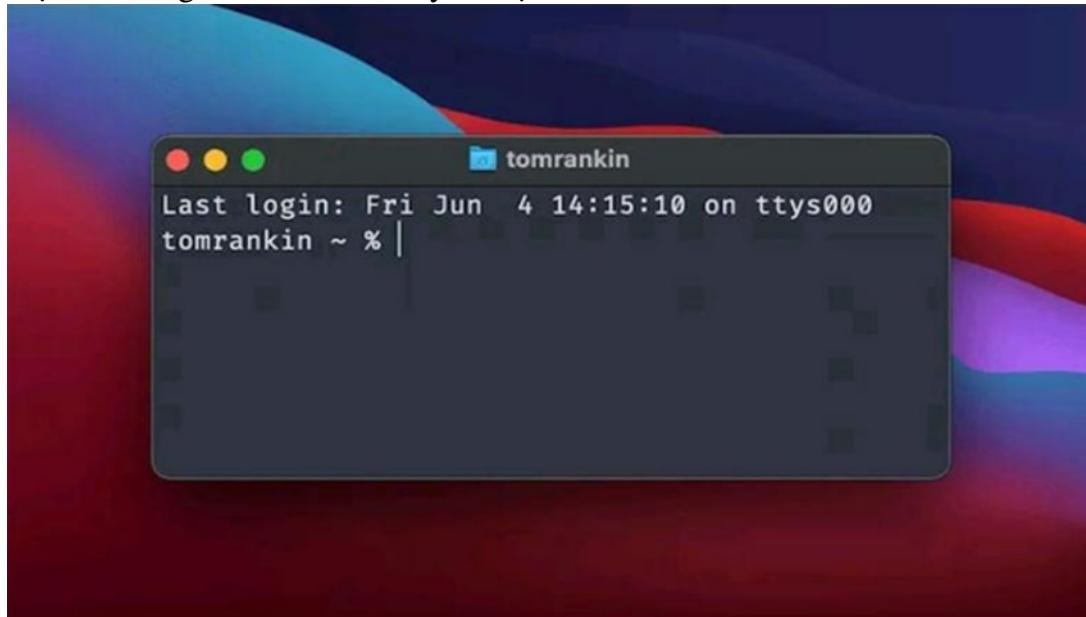
Đầu tiên bấm tổ hợp phím tắt Command + Space > rồi sau đó gõ vào thanh tìm kiếm Terminal > rồi bấm Return.

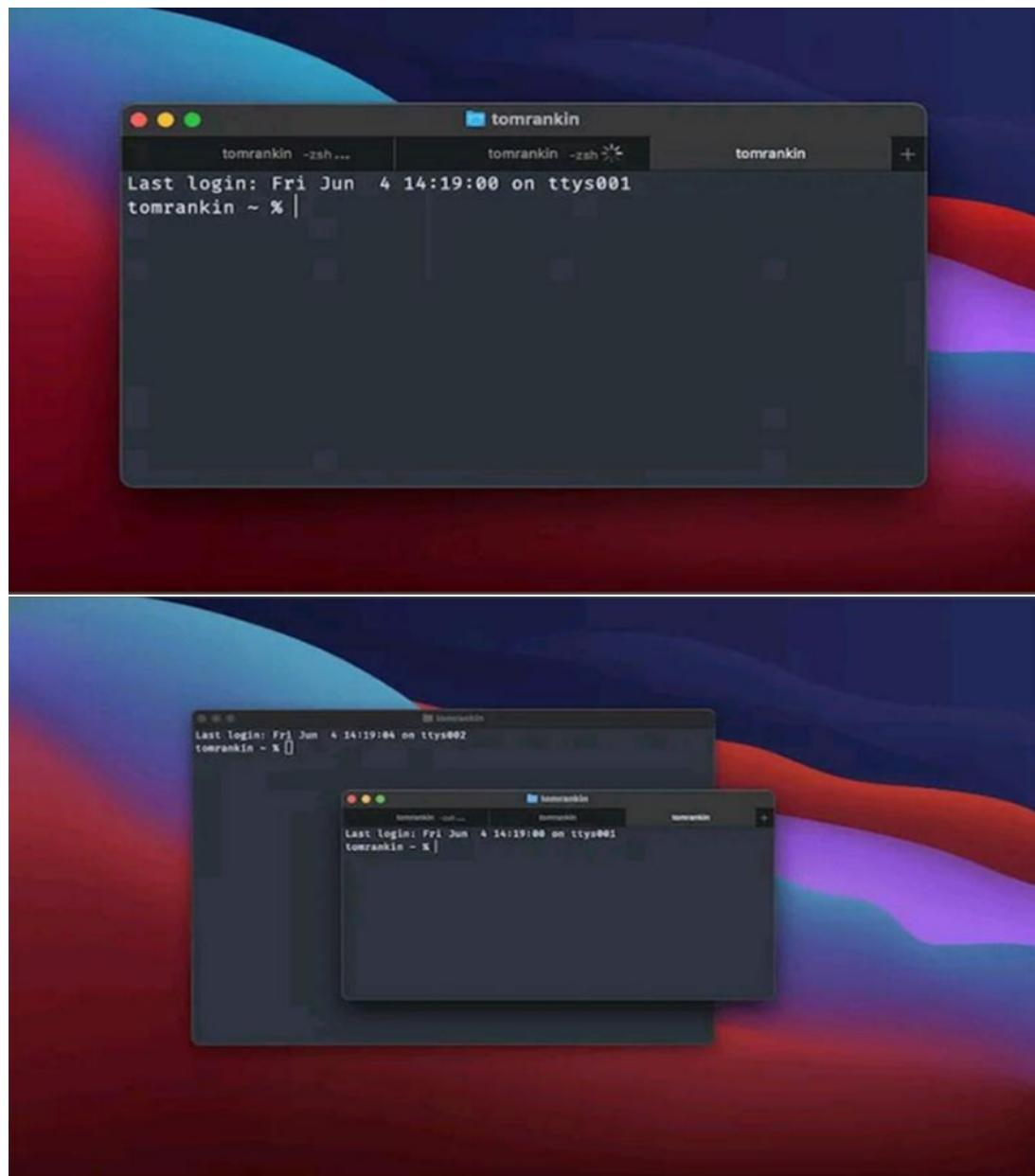


3.1.4 Cách điều hướng trong macOS Terminal

Terminal macOS về cơ bản không quá khó để sử dụng, giao diện của công cụ này sẽ là thanh tiêu đề hiển thị thông tin chi tiết của người dùng và thư mục ở hiện tại. Trong khi màn hình làm việc chính sẽ là nơi người dùng nhập những câu lệnh và đọc kết quả ở đầu ra.

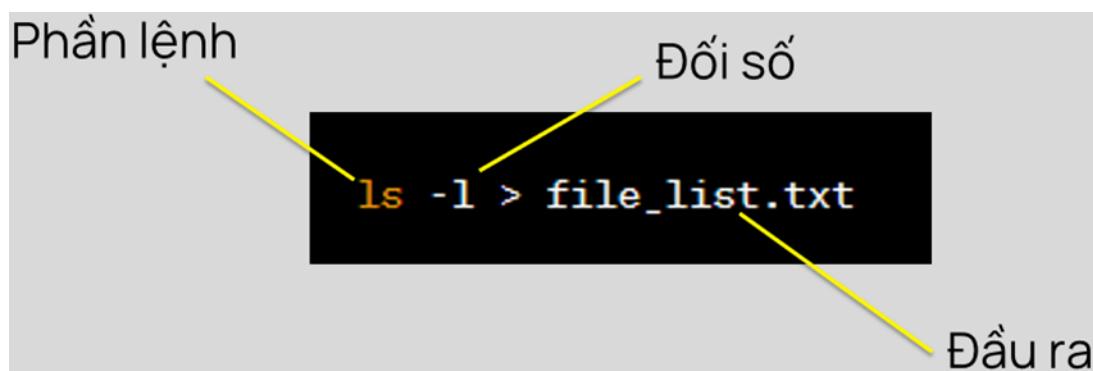
Người dùng có thể mở nhiều tab mới ngay trên macOS và chúng sẽ đại diện cho từng cửa sổ Shell chuyên biệt.





3.1.5 Cấu trúc câu lệnh Shell

Terminal macOS sẽ đưa ra các lệnh khác nhau để có thể thực thi các tác vụ cụ thể tương ứng ở trên hệ thống. Mỗi lệnh của Terminal đều sẽ có 3 phần như sau:



- Phần lệnh:** Đây là nơi để người dùng nhập các ký tự thực tế vào cửa sổ của Terminal để tiến hành thực hiện lệnh. Cần hết sức cẩn thận khi sử dụng lệnh, vì mỗi câu lệnh sẽ có thể xóa đi tệp hoặc gây ra các thiệt hại cho hệ thống máy và có thể làm cho máy ngừng hoạt động.
- Đối số:** Đây là một trong những phần cho biết rằng câu lệnh sẽ hoạt động trên những tài nguyên nào. Ví dụ như một cp hoặc lệnh sao chép, có nên sao chép cửa sổ này hay một cửa sổ khác?
- Một câu chọn sửa đổi đầu ra:** Đây là một chỉ thị về nơi kết quả của một câu lệnh cụ thể nào đó sẽ xuất hiện.

3.1.6 Các lệnh Terminal cơ bản cần biết

Cấu trúc chung của một câu lệnh di chuyển đến thư mục sẽ là “cd + Địa chỉ thư mục”.

- Di chuyển đến thư mục muôn thao tác (cd):** Cấu trúc chung của một câu lệnh di chuyển đến thư mục sẽ là “cd + Địa chỉ thư mục”.

Các câu lệnh phổ biến người dùng nên biết ở dưới bảng sau đây:

Cấu trúc lệnh với cd	Thao tác tương ứng với câu lệnh
Câu lệnh cd [Địa chỉ thư mục]	Đây là thao tác di chuyển đến thư mục được nhập sau câu lệnh.
cd ~	Câu lệnh này dùng để di chuyển đến thư mục người dùng
cd ./[Tên thư mục]	Câu lệnh này thực hiện di chuyển tệp đến thư mục con của thư mục hiện tại
cd ../	Câu lệnh di chuyển tệp đến thư mục mẹ của thư mục hiện tại
cd -	Câu lệnh di chuyển tệp đến thư mục vừa mới truy cập trước đó

- Hiển thị địa chỉ thư mục hiện tại (pwd):** Để kiểm tra lại xem là người dùng đã di chuyển đúng địa chỉ thư mục mà mình muốn hay chưa, chỉ cần gõ “pwd” trên Terminal và bấm vào Return.
- Liệt kê tệp và thư mục (ls):** Câu lệnh ls giúp người dùng liệt kê hết tất cả các tệp và thư mục con ở trong cùng một địa chỉ thư mục, được liệt kê ở bảng dưới đây:

Cấu trúc câu lệnh với ls	Thao tác tương ứng với câu lệnh
ls [Địa chỉ thư mục]	Câu lệnh này giúp liệt kê các tệp và thư mục con hiện đang ở trong thư mục mà bạn nhập
ls -l [Địa chỉ thư mục]	Câu lệnh này giúp liệt kê tất cả các tệp, cũng như thư mục con có trong thư mục mà bạn nhập, bao gồm cả người sở hữu và thời gian mà tệp được tạo.
ls -la [Địa chỉ thư mục]	Câu lệnh này liệt kê tất cả các tệp và thư mục con có trong thư mục bạn nhập, bao gồm cả tệp, thư mục đã được ẩn đi, chủ sở hữu và thời gian đã tạo ra thư mục và tệp.

- Xóa tệp (rm):** Câu lệnh Terminal macOs để xóa tệp đơn giản là “rm +

Địa chỉ tệp”.

- **Tạo và xóa thực mục (mkdir và rmdir):** Câu lệnh để tạo thư mục đơn giản là “mkdir + Địa chỉ thư mục”. Câu lệnh để thực hiện thao tác xóa các tệp thư mục cũ là “rmdir + Địa chỉ thư mục”.
- **Sao chép file (cp):** Cú pháp câu lệnh sao chép file cp đơn giản là “cp + Địa chỉ thư mục kèm với file gốc + Địa chỉ thư mục đích và tên file mới”
- **Xem lịch sử trên Terminal (history):** Để xem được lịch sử các câu lệnh mà người dùng đã dùng trên Terminal macOS, gõ History và bấm vào mục Return. Hoặc có thể sử dụng câu lệnh “history -c” để xóa lịch sử khi cần.

3.1.7 Các nguyên tắc khi sử dụng lệnh Terminal

Sau khi đã tìm hiểu về một số câu lệnh cơ bản khi sử dụng Terminal macOS, cũng đồng thời cần phải lưu ý thêm một số nguyên tắc dưới đây để áp dụng:

- Khi sử dụng Terminal thì người dùng luôn phải nhập câu lệnh liên quan và bấm Enter hoặc Return trên bàn phím để thực hiện lệnh.
- Không thể sử dụng chuột để tương tác trên cửa sổ Terminal ngoài việc sử dụng 3 nút là đóng, mở rộng và ẩn ở ngay góc trái của cửa sổ.
- Để ngắt lệnh đang được chạy gõ Control + C.
- Để thoát khỏi Terminal nhanh chóng không cần thông qua chuột thì người dùng có thể sử dụng tổ hợp phím tắt Command + Q.

3.1.8 Cách thoát Terminal

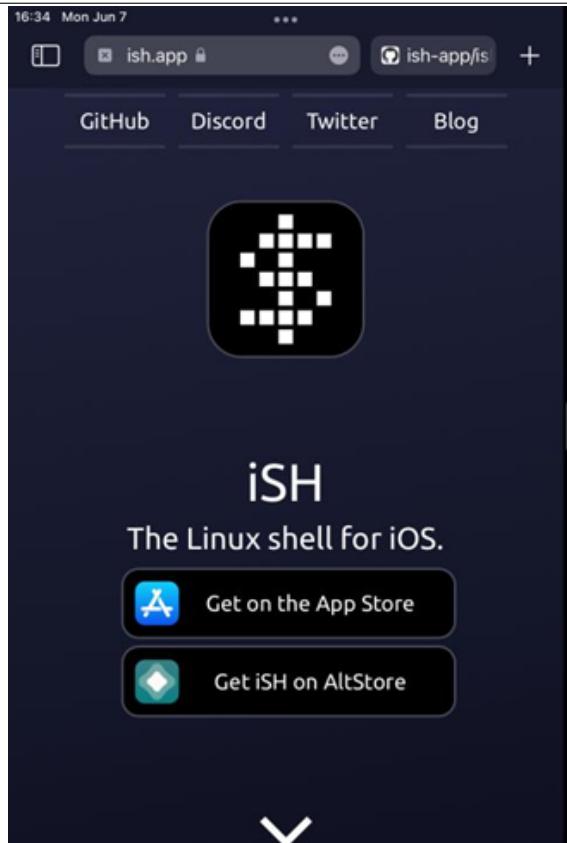
- Trong ứng dụng Terminal bấm chọn vào Terminal > và chọn thoát Terminal.
- Trong cửa sổ đang chạy qua trình Shell mà người dùng muốn thoát, thì hãy nhập lệnh “exit” rồi sau đó bấm Return.

3.2 Các công cụ trên iOS

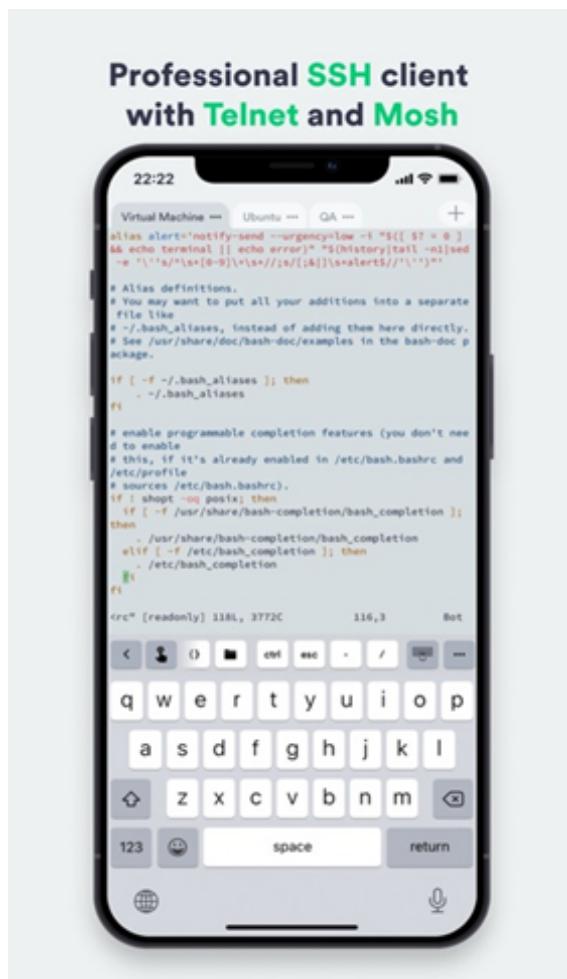
Trái ngược với macOS, iOS không có một ứng dụng Terminal tích hợp sẵn. Tuy nhiên, người dùng vẫn có thể truy cập giao diện dòng lệnh trên iOS thông qua một số ứng dụng thứ 3:

- iSH: Đây là một ứng dụng mô phỏng giao diện dòng lệnh Linux trên iOS. Nó cung cấp bash shell và có thể chạy nhiều lệnh Linux tiêu chuẩn. Tuy nhiên, khả năng của nó bị hạn chế bởi cơ chế sandbox của iOS.

CHƯƠNG 3. HÀM SHELL VÀ LỜI GỌI HỆ THỐNG



- Termius: Là một ứng dụng SSH client, cho phép kết nối từ xa tới các máy chủ UNIX/Linux và truy cập dòng lệnh.



- Prompt 2: Cung cấp giao diện dòng lệnh đơn giản cho phép thực thi các lệnh cơ bản trên iOS. Nó không đầy đủ tính năng như một terminal thực sự.



- DomTerm: Là một ứng dụng terminal mã nguồn mở cho iOS. Nó hỗ trợ nhiều tính năng terminal như split screen, tùy chỉnh phím tắt,...

Nhìn chung, do bị hạn chế bởi tính chất đóng của hệ điều hành, các ứng dụng terminal trên iOS không thể cung cấp đầy đủ khả năng như trên macOS. Chúng thường chỉ cung cấp truy cập hạn chế vào shell.

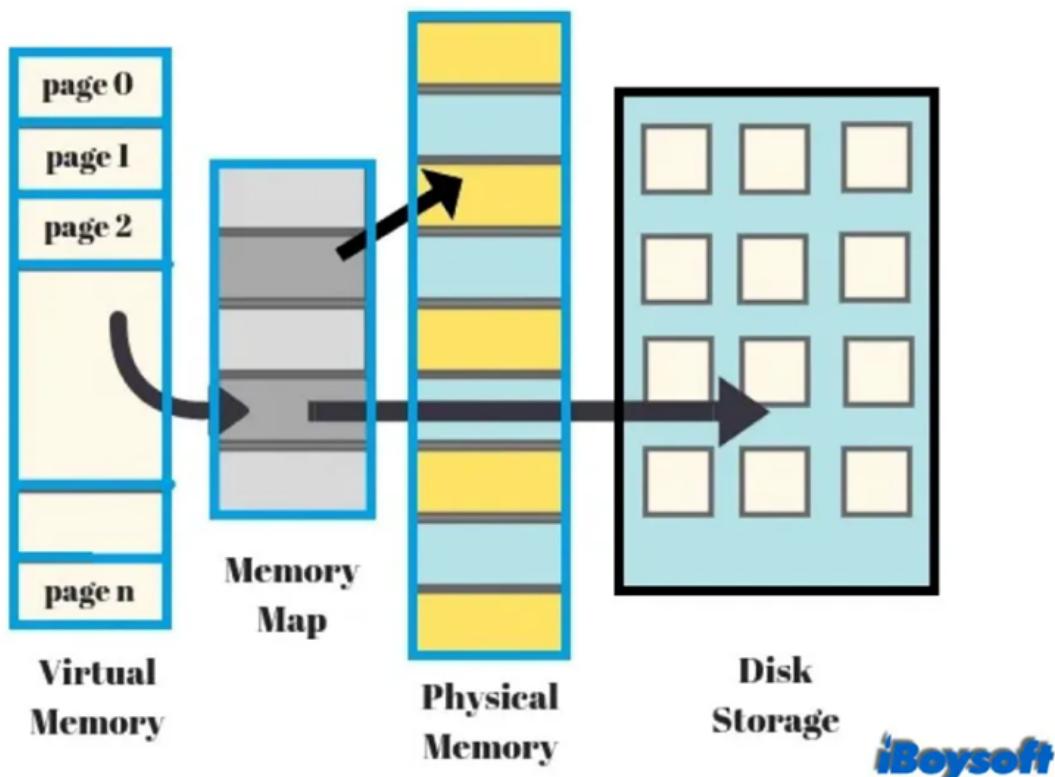
CHƯƠNG 4. QUẢN LÝ BỘ NHỚ

4.1 Nguyên tắc quản lý bộ nhớ, cơ chế xử lý tiến trình và quản lý bộ nhớ ứng dụng trên macOS và iOS

4.1.1 Nguyên tắc quản lý bộ nhớ

- macOS:

- Bộ Nhớ ảo (Virtual Memory): macOS sử dụng bộ nhớ ảo để tạo ra một không gian bộ nhớ ảo lưu trữ dữ liệu khi bộ nhớ RAM cạn kiệt. Dữ liệu trong bộ nhớ ảo được lưu trữ tạm thời trên ổ đĩa cứng và được sử dụng khi cần thiết.



Hình 4.1: Virtual Memory

- Quản lí bộ nhớ vật lý và ảo: Hệ điều hành macOS quản lý việc sử dụng bộ nhớ vật lý và bộ nhớ ảo thông qua các cơ chế như Memory Compression để tối ưu hóa sử dụng bộ nhớ và giữ cho hệ thống hoạt động trơn tru. (Memory Compression là việc nén dữ liệu trực tiếp trong bộ nhớ để tiết kiệm không gian và tăng hiệu suất hệ thống khi bộ nhớ RAM gặp áp lực.)
- Memory Management Unit (MMU): MMU quản lý việc ánh xạ dữ

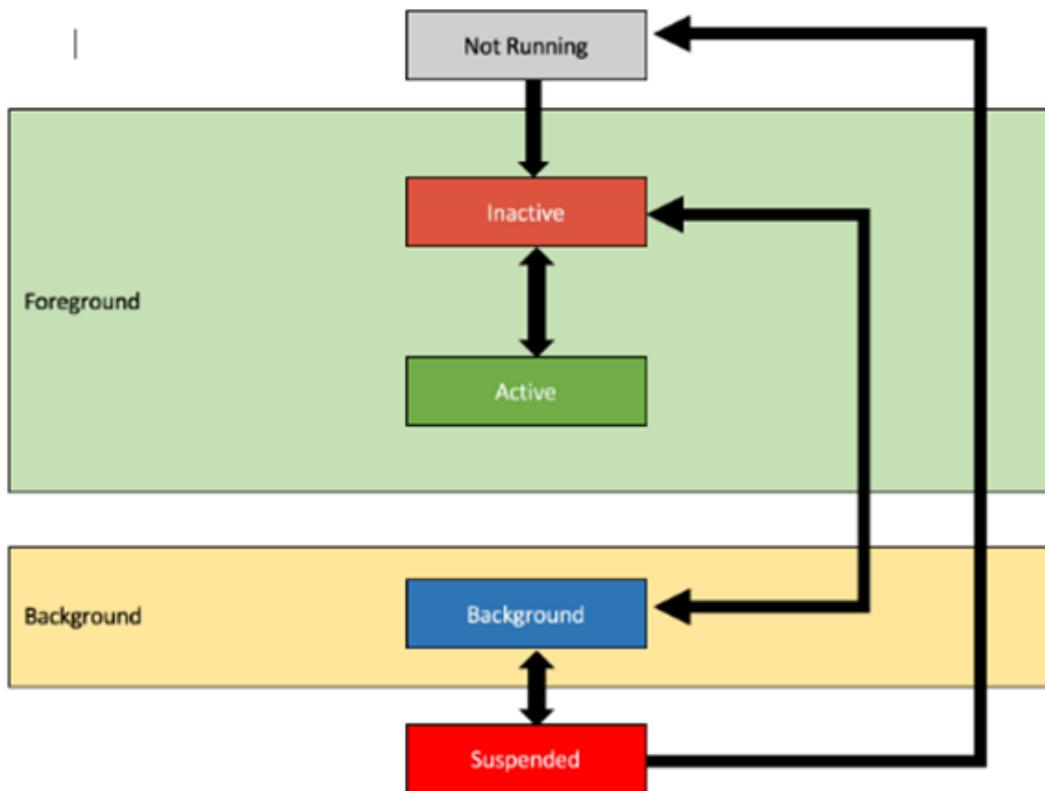
liệu giữa bộ nhớ ảo và bộ nhớ vật lý, đảm bảo việc truy cập vào bộ nhớ là hiệu quả và an toàn.

- Phân trang (Phân trang là một kỹ thuật trong hệ thống quản lý bộ nhớ ảo, cho phép hệ điều hành tạo ra một không gian địa chỉ logic lớn hơn dung lượng RAM vật lý có sẵn bằng cách chia không gian địa chỉ thành các phần nhỏ gọi là trang. Khi một ứng dụng truy cập vào một địa chỉ trên trang bộ nhớ không có sẵn trong RAM vật lý, xảy ra lỗi trang. Quá trình phân trang xử lý lỗi trang này bằng cách di chuyển dữ liệu giữa RAM và đĩa để đảm bảo trang cần thiết có sẵn trong RAM để chương trình có thể truy cập bình thường):
 - * Quá trình Phân Trang: Khi một trang bộ nhớ không có sẵn trong RAM vật lý, hệ thống kích hoạt trình xử lý lỗi trang. Trình xử lý dừng mã chương trình đang thực thi, tải trang từ đĩa, cập nhật bảng trang và trả lại quyền điều khiển cho chương trình.
 - * Phân Trang Ra (Paging Out) và Phân Trang Vào (Paging In): Trong macOS, dữ liệu thường được chuyển từ RAM sang backing store (Là nơi lưu trữ các trang không còn sử dụng trong RAM. Khi RAM đầy, các trang không sử dụng sẽ được ghi vào đĩa để giải phóng bộ nhớ cho dữ liệu cần thiết hiện tại) trên đĩa khi không cần thiết (phân trang ra) và ngược lại khi cần thiết (phân trang vào).

- iOS:

- Phân trang(Paging):
 - * Quá trình Phân Trang: Tương tự macOS, khi xảy ra lỗi trang, hệ thống tải trang từ đĩa và cập nhật bảng trang. Tuy nhiên, iOS không sử dụng backing store để lưu trữ trang bộ nhớ không cần thiết.
 - * Không có Phân Trang Ra đối với Disk (No Paging Out): Trong iOS, trang không bao giờ được ghi ra đĩa, nhưng các trang chỉ đọc có thể được tải vào từ đĩa khi cần thiết.
- Phân chương (Segmentation): iOS không sử dụng cơ chế phân chương tương tự như macOS. Thay vào đó, nó tập trung chủ yếu vào phân trang để quản lý không gian địa chỉ của các quá trình.
- Quản lí vòng đời ứng dụng: iOS quản lý việc khởi động, tạm dừng (suspend), kích hoạt (resume) và kết thúc ứng dụng một cách tự

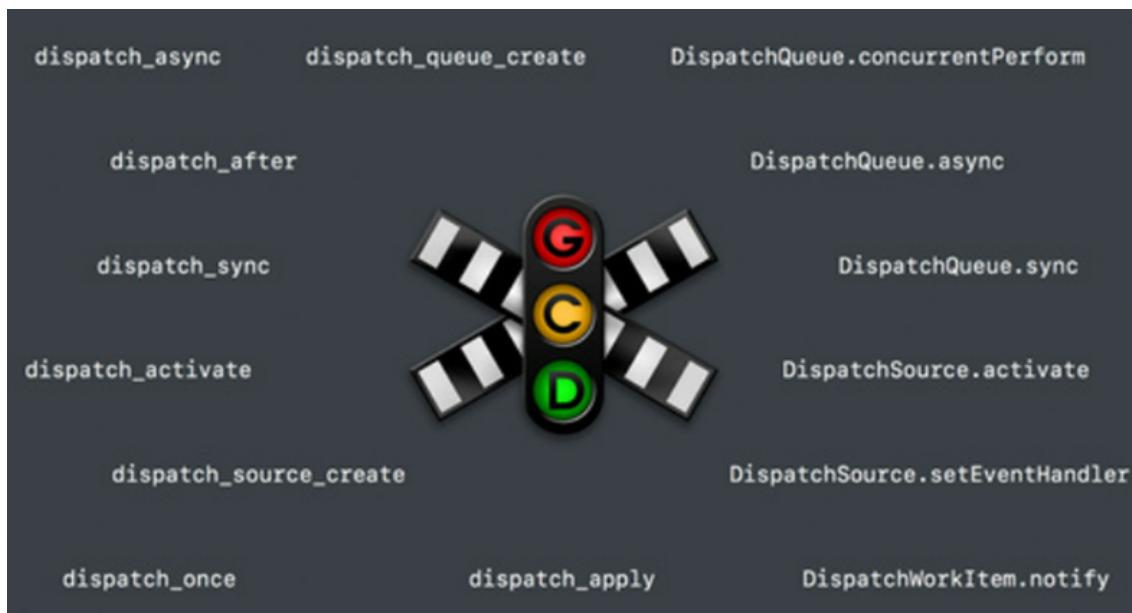
dộng. Khi người dùng chuyển sang ứng dụng khác, iOS có thể tạm dừng ứng dụng để giải phóng tài nguyên và cung cấp cho ứng dụng khác sử dụng.



Hình 4.2: Life cycle

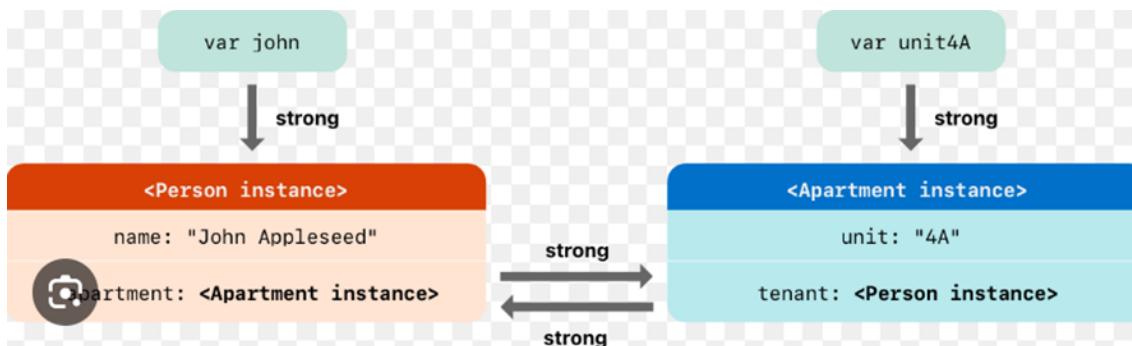
- Grand Central Dispatch (GCD): iOS sử dụng GCD để quản lý hàng đợi và thực thi các tác vụ bất đồng bộ, giúp ứng dụng chạy mượt mà mà không bị đơ (GCD là một thư viện Apple cung cấp nhằm hỗ trợ việc chạy những tasks song song nhằm tối ưu hiệu năng cho những thiết bị có bộ xử lý đa lõi (multi core processor). GCD hoạt động dựa trên cơ chế Thread Pool: thay vì phải tạo các threads trực tiếp thì chúng ta sẽ đưa các tasks vào Queues. Queues sẽ tự động tạo và huỷ threads, còn việc tạo bao nhiêu threads thì sẽ do hệ thống quyết định dựa trên công việc của CPU đang thực thi. Queues(hàng đợi): để quản lý những tasks submit tới và chạy chúng theo thứ tự FIFO. Điều này đảm bảo rằng task đầu tiên được thêm vào queue sẽ là task đầu tiên được bắt đầu sau đó lần lượt các task được bắt đầu theo thứ tự đến hết queue. GCD cung cấp 2 loại hình của DispatchQueue

là: Serial Queues và Concurrent Queues).



Hình 4.3: Grand Central Dispatch

- Quản lý bộ nhớ động: iOS sử dụng ARC (Automatic Reference Counting) trong Objective-C và Swift để tự động quản lý việc giải phóng bộ nhớ cho các đối tượng không còn sử dụng (Trong iOS, khi object được khởi tạo, nó sẽ được quản lý bởi ARC. Dựa vào số lượng reference đến mà object đó được gán một số được gọi là Reference counting. Mỗi khi object được reference bởi object khác, số reference counting sẽ tăng thêm, và giảm khi nó không còn được reference đến nữa. Một object sẽ được xoá bỏ và trả lại bộ nhớ cho hệ thống khi nó không còn được reference bởi bất kỳ object nào khác, hay reference counting bằng 0).



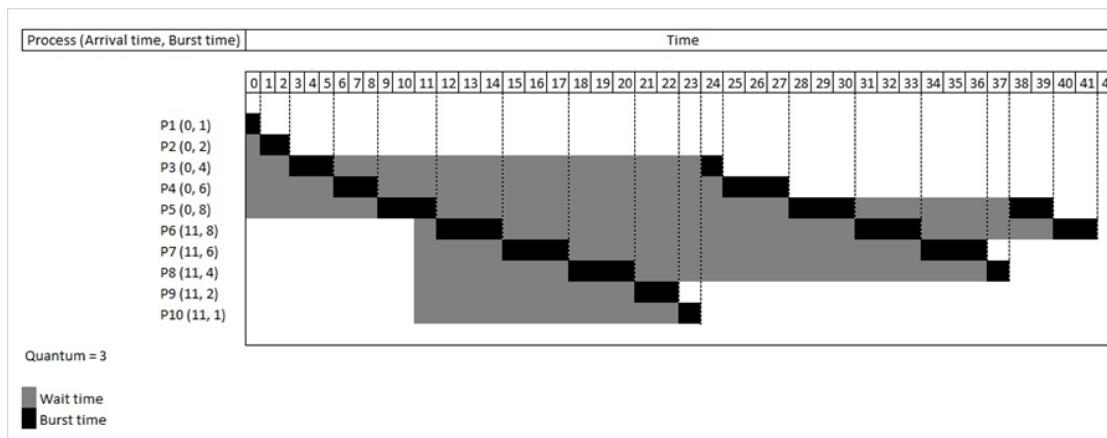
Hình 4.4: Automatic Reference Counting

4.1.2 Cơ chế xử lý tiến trình

- macOS:

MacOS sử dụng một số thuật toán quản lý và lập lịch tiến trình để điều phối việc chạy các tiến trình trên hệ thống. Dưới đây là một số thuật toán chính:

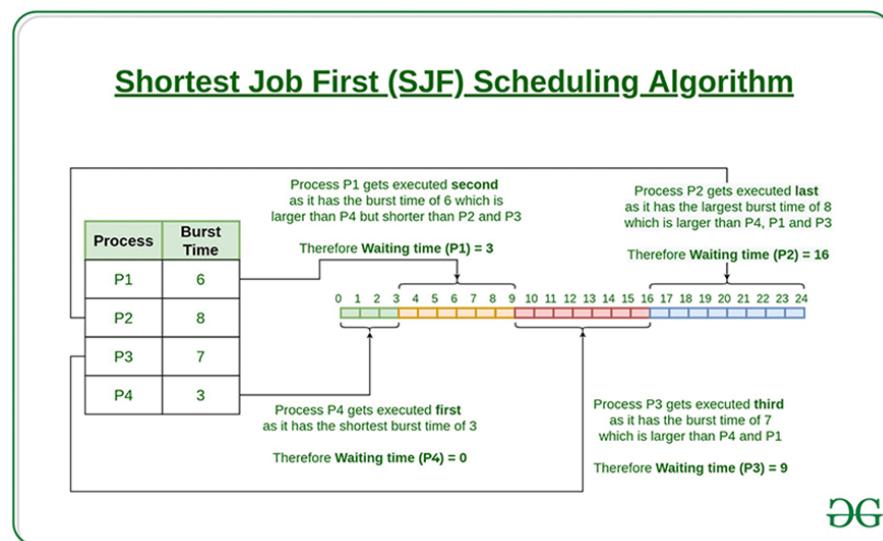
- Round Robin Scheduling: Thuật toán Round Robin là một trong những thuật toán lập lịch CPU phổ biến nhất trong hệ thống điều khiển lập lịch CPU và nó cũng là thuật toán được sử dụng nhiều nhất trong macOS. Thuật toán này phân chia thời gian CPU thành các đoạn nhỏ, gọi là quantum, và sau đó lập lịch các tiến trình theo trình tự quay vòng, mỗi tiến trình được chạy trong một quantum. Thuật toán Round Robin hoạt động bằng cách lập lịch các tiến trình trong một danh sách. Mỗi tiến trình được chạy trong một khoảng thời gian được xác định trước gọi là quantum. Khi quantum kết thúc, tiến trình sẽ được đưa lại vào danh sách và tiến trình tiếp theo sẽ được chạy. Quá trình này tiếp tục đến khi tất cả các tiến trình hoàn thành.



Hình 4.5: Round Robin Scheduling

- Shortest Job First (SJF): Thuật toán ưu tiên thực thi tiến trình có thời gian thực thi ngắn nhất trước. Điều này giúp tối ưu hóa thời gian chờ đợi và cải thiện hiệu suất hệ thống. Quá trình này bắt đầu bằng việc xác định thời gian xử lý của từng tiến trình trong danh sách. Sau đó, hệ thống chọn tiến trình có thời gian xử lý ngắn nhất từ danh sách các tiến trình đang chờ xử lý hoặc đã sẵn sàng để thực thi. CPU bắt đầu thực thi tiến trình được chọn và tiếp tục cho đến khi tiến trình

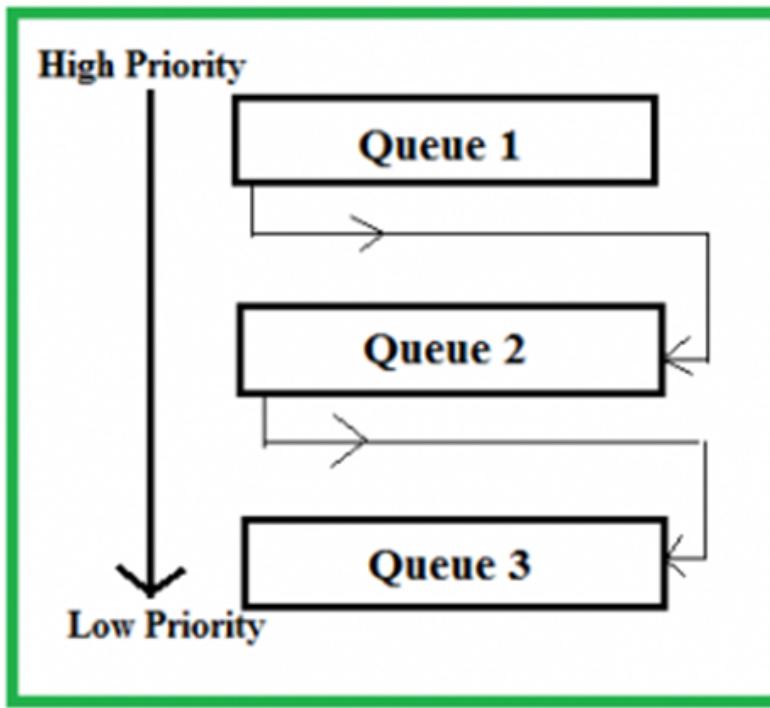
hoàn thành công việc của mình. Khi tiến trình hoàn thành, CPU chuyển sang tiến trình tiếp theo có thời gian xử lý ngắn nhất trong danh sách. Quá trình này tiếp tục cho đến khi tất cả các tiến trình đã hoàn thành công việc của mình. SJF có thể được thực hiện có ưu tiên (preemptive) hoặc không có ưu tiên (non-preemptive), điều này ảnh hưởng đến khả năng gián đoạn của CPU để thực thi tiến trình mới có thời gian xử lý ngắn hơn.



ĐG

Hình 4.6: Shortest Job First

- Multilevel Feedback Queue: Thuật toán này sử dụng nhiều hàng đợi với các mức độ ưu tiên khác nhau. Các tiến trình có thể di chuyển giữa các hàng đợi dựa trên hành vi thực thi của chúng. Điều này giúp tối ưu hóa thời gian chờ đợi và ưu tiên các tiến trình ứng dụng. (Multilevel Feedback Queue hoạt động theo cách mỗi hàng đợi đại diện cho một cấp độ ưu tiên, với các tiến trình được xếp vào hàng đợi có ưu tiên thấp nhất ban đầu. Khi một tiến trình không hoàn thành trong một khoảng thời gian nhất định, nó sẽ bị giảm độ ưu tiên và đẩy xuống hàng đợi có ưu tiên thấp hơn. Ngược lại, các tiến trình hoàn thành sẽ được tăng độ ưu tiên và đẩy lên các hàng đợi có ưu tiên cao hơn. CPU lựa chọn tiến trình ở hàng đợi có ưu tiên cao nhất để thực thi và quản lý sự thay đổi độ ưu tiên dựa trên hiệu suất thực tế của từng tiến trình, tạo điều kiện linh hoạt để xử lý các tiến trình có yêu cầu thời gian xử lý khác nhau một cách hiệu quả).

**Hình 4.7:** Multilevel Feedback Queue

- iOS:

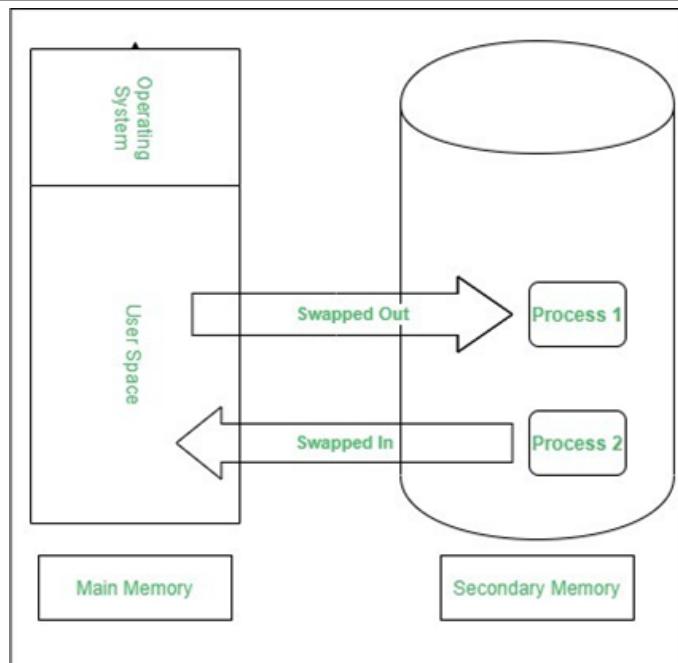
- Thông tin cụ thể về các thuật toán xử lý tiến trình được sử dụng trong iOS không được Apple công bố hoặc công khai rộng rãi. Apple giữ các chi tiết cụ thể về cách thức quản lý tiến trình trong hệ điều hành iOS. Nhưng có thể iOS sẽ cũng sử dụng một số thuật toán xử lý tiến trình tương tự như macOS.

4.2 Đánh giá sự khác biệt và tương đồng về quản lý bộ nhớ giữa hai hệ thống

4.2.1 Sự tương đồng

- Cơ chế quản lý bộ nhớ ảo:

- Memory Compression: Cả macOS và iOS đều sử dụng kỹ thuật Memory Compression để giảm tải cho bộ nhớ RAM bằng cách nén dữ liệu trực tiếp trong bộ nhớ khi cần.
- Paging và Swapping: Cả hai hệ điều hành đều sử dụng việc chuyển dữ liệu giữa RAM và ổ đĩa (paging và swapping) để quản lý bộ nhớ khi cần thiết.
- macOS và iOS là cả hai đều sử dụng quá trình phân trang để quản lý bộ nhớ ảo và cung cấp truy cập vào không gian địa chỉ logic lớn hơn RAM vật lý.



Hình 4.8: Paging và Swapping

- Cơ chế xử lý tiến trình:
 - Grand Central Dispatch (GCD): Đây là một cơ chế quan trọng trong việc quản lý luồng và thực thi tác vụ bất đồng bộ trên cả macOS và iOS, giúp tối ưu hóa việc sử dụng CPU và tăng hiệu suất của hệ thống.
 - Quản Lý Tiến Trình: Cả hai hệ điều hành đều có cơ chế quản lý tiến trình để đảm bảo tài nguyên hệ thống được phân phối cân bằng, tránh các vấn đề xung đột và cải thiện hiệu suất.
- Quản lý bộ nhớ vật lý:
 - ARC và Quản Lý Bộ Nhớ Động: Sử dụng ARC (Automatic Reference Counting) để tự động quản lý bộ nhớ, giải phóng bộ nhớ khi đối tượng không còn được sử dụng.
 - Quản Lý Tài Nguyên Hệ Thống: Cả hai hệ điều hành đều quản lý tài nguyên hệ thống như bộ nhớ, CPU và thiết bị ngoại vi để đảm bảo hiệu suất và sử dụng tối ưu các tài nguyên.

4.2.2 Sự khác biệt

- Tài nguyên bộ nhớ:
 - macOS: macOS thường được cài đặt trên máy tính cá nhân với tài nguyên bộ nhớ RAM và ổ cứng lớn, cho phép quản lý bộ nhớ linh hoạt hơn và hỗ trợ cho các ứng dụng đa nhiệm và phức tạp.
 - iOS: iOS chạy trên các thiết bị di động với bộ nhớ RAM hạn chế. Hệ

thống phải tối ưu hóa quản lý bộ nhớ để đảm bảo hiệu suất và tiết kiệm năng lượng.

- **Ứng dụng và tiến trình:**

- macOS: macOS cho phép chạy nhiều ứng dụng và tiến trình đồng thời, với quản lý ưu tiên và tiến trình đa nhiệm mạnh mẽ.
- iOS: iOS cũng cho phép chạy nhiều ứng dụng đồng thời, nhưng có hạn chế hơn do tài nguyên hạn chế. Hệ thống iOS sử dụng cơ chế tắt ứng dụng chạy trong nền để tiết kiệm tài nguyên.

- **Quản lý bộ nhớ ứng dụng:**

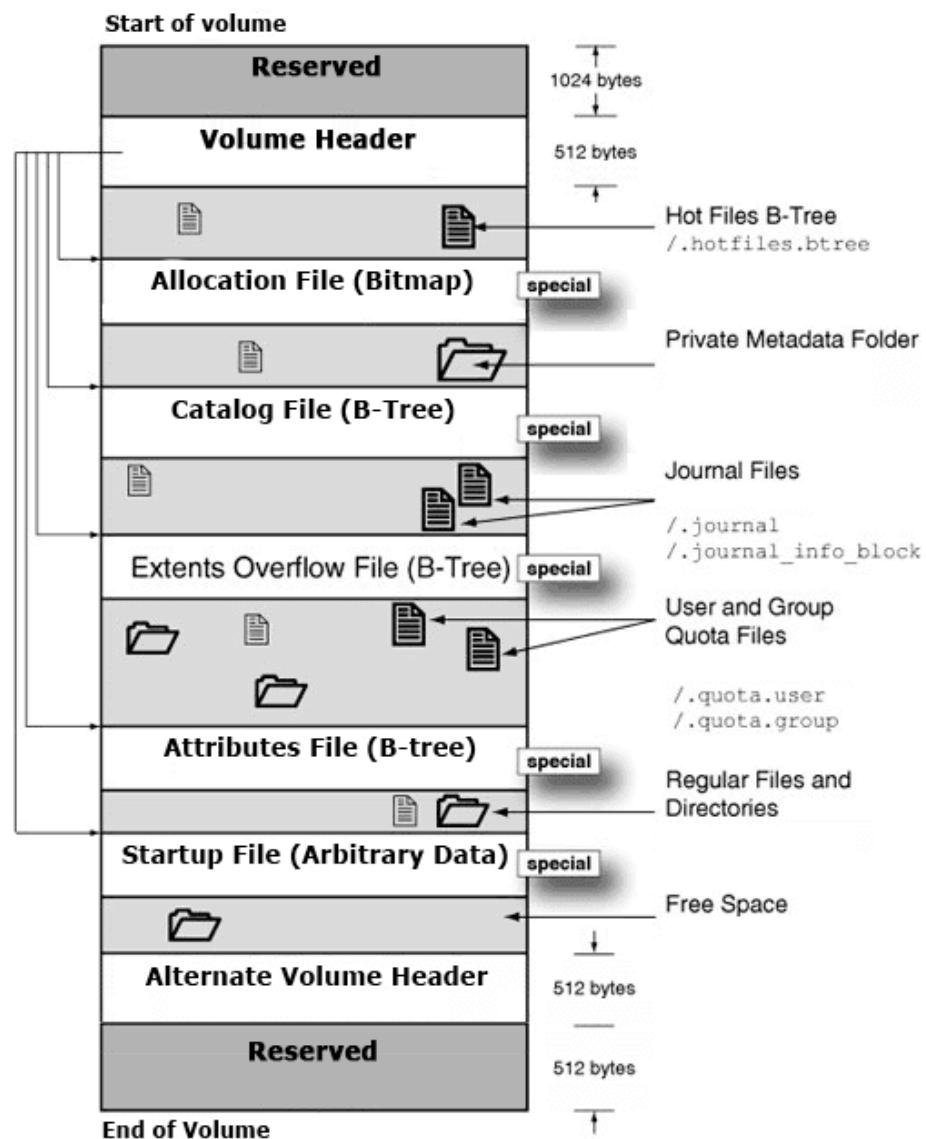
- macOS: macOS cho phép ứng dụng sử dụng bộ nhớ một cách tương đối rộng rãi và đáng tin cậy.
- iOS: iOS quản lý bộ nhớ ứng dụng chặt chẽ hơn để đảm bảo hiệu suất của thiết bị di động.

CHƯƠNG 5. QUẢN LÝ FILE

5.1 Cách quản lý file

Cả macOS và iOS hiện nay đều sử dụng APFS (Apple File System) làm hệ thống tệp tin mặc định. Trước đó thì macOS đã sử dụng HFS+ làm hệ thống tệp tin mặc định cho tới macOS 10.12.

- HFS+ (Hierarchical File System Plus) hay Mac OS Extended là hệ thống file được sử dụng trên Mac từ 1998. Nó được sử dụng trên tất cả các ổ đĩa cơ và ổ đĩa lai. Cấu trúc của HFS+ có 9 phần chính gồm:
 - Volume Header: để lưu trữ thông tin ổ đĩa như ngày tạo, số file,...
 - Allocation File: Kiểm tra một khối phân bổ đang được sử dụng hay trống.
 - Catalog File: Mô tả cấu trúc tệp trên một ổ đĩa.
 - Attributes File: Cây dữ liệu B-tree (cây tìm kiếm tự cân bằng) với thông tin nhánh bổ sung.
 - Extents Overflow File: B-tree cho phần còn lại của các phần mở rộng mà tệp mô tả không được lưu trữ.
 - Startup File: để khởi động máy tính không phải Mac.
 - Alternate Volume Header: lưu trữ thông tin về cấu trúc thư mục chính của hệ thống tệp tin.
 - Boot Blocks: các khối khởi động.
 - Phân đoạn cuối cùng của Apple được dành cho việc sử dụng trong quá trình sản xuất máy tính.



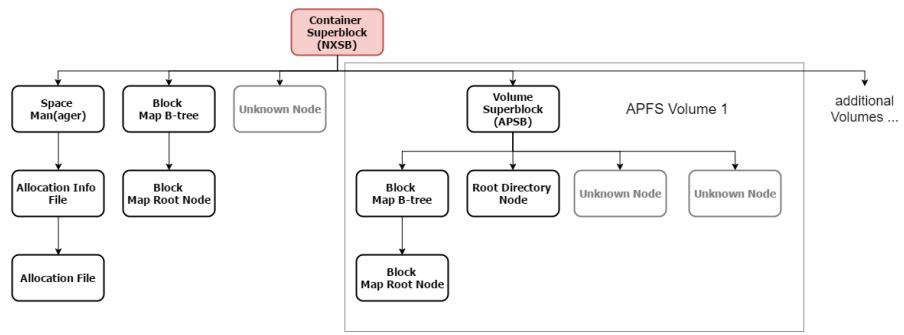
Hình 5.1: Cấu trúc của HFS+.

- Về cách hoạt động của HFS+, nhờ có Catalog File mà lưu trữ thông tin về các tệp tin và thư mục trong một B-tree, giúp tìm kiếm hiệu quả. Ngoài ra nếu không đủ chỗ trong Catalog File, sử dụng Extents Overflow File để chứa thông tin về các phần mở rộng của tệp tin. HFS+ sử dụng kiểu cấp phát dựa trên phần mở rộng, nơi các khối dữ liệu của một tệp tin được liên kết liền kề trên đĩa và sử dụng Allocation File để theo dõi khối dữ liệu đã được cấp phát và khối nào còn trống. Hỗ trợ nhiều kích thước khôi, tăng hiệu suất và tối ưu hóa việc sử dụng đĩa. Ngoài ra, các tệp “nóng” (Hot Files) - thường xuyên truy cập được tập trung gần nhau trên đĩa để tăng tốc độ đọc/ghi. Khả năng bảo toàn dữ liệu của HFS+ khá tuyệt vời khi có Journal File, hỗ trợ giảm thiểu rủi ro mất dữ liệu do sự cố hệ thống hoặc mất điện. Metadata (các siêu dữ liệu) có thể được nén

để tiết kiệm không gian đĩa. Đồng thời, HFS+ có khả năng tương thích ngược, đọc các định dạng HFS và HFSX (HFS Extended) giúp đảm bảo tương thích với các hệ thống tệp tin cũ hơn. HFS+ cung cấp sự ổn định và hiệu suất tốt cho hệ thống Mac. Tuy nhiên, nó được thay thế bởi Apple File System (APFS) trên macOS High Sierra và các phiên bản mới hơn để đáp ứng nhu cầu ngày càng tăng của công nghệ và các tính năng mở rộng.

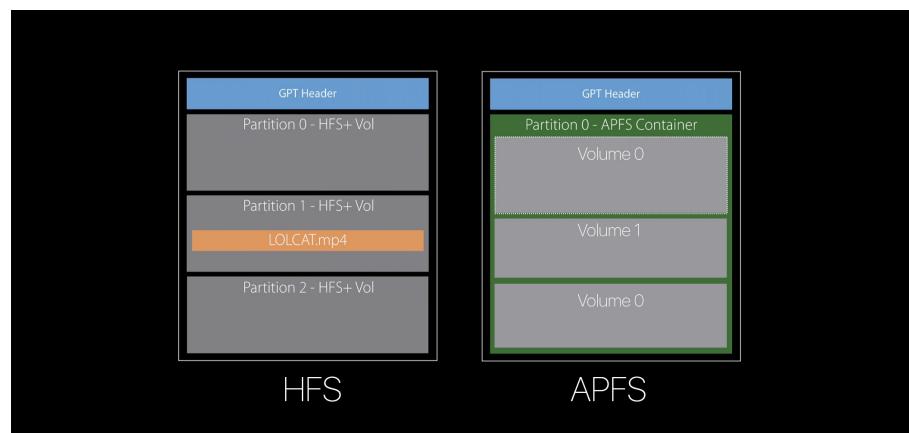
- Bắt đầu từ bản macOS 10.13 và iOS 10.3 thì APFS đã thay thế cho HFS+ trở thành hệ thống tệp tin mặc định. So với HFS+, APFS có nhiều lợi ích rất rõ ràng như tối ưu cho SSD, cho phép copy file mà không tốn thêm dung lượng, hỗ trợ tính năng snapshot lại phiên bản file rồi quay trở lại sau đó hoặc tạo các phân vùng không cần liên tiếp. Ngoài ra, APFS cũng rất linh hoạt khi Apple có thể thêm chức năng mới bất kì khi nào họ muốn mà không phải chạy áp dụng lại cho tất cả các file trong ổ. APFS rất tiện dụng nhờ có:

- Cấu trúc của APFS như là một vùng chứa (container), chứa một hoặc nhiều volume, mỗi volume đại diện cho một hệ thống tập tin APFS và chúng có các thuộc tính và cấu trúc riêng biệt. Mỗi một cấu trúc hệ thống tập tin khởi đầu với một Block header. Blockheader sử dụng Checksum (thuật toán tổng kiểm tra Fletcher) cho toàn khối và cũng sử dụng kỹ thuật Copy-on-write phiên bản, ID và loại khối. Một Container Superblock bao gồm thông tin về kích thước của khối, số lượng các khối và các con trỏ đến trình quản lý không gian cho tác vụ này, ID khối của tất cả các ổ đĩa và một con trỏ tới bản đồ B-tree. Các Node được sử dụng để lưu trữ các loại đầu vào khác nhau, nó có thể là một phần của B-tree hoặc không và có thể chứa các đầu vào có kích thước cố định hoặc không cố định. Space manager quản lý các khối được phân bổ trong APFS container và lưu trữ số lượng khối trống cùng với một con trỏ tới tệp thông tin phân bổ. Allocation info file lưu trữ độ dài, phiên bản và độ lệch của tệp phân bổ. B-tree quản lý các nút khác nhau và chứa phần bù của nút gốc. Một Volume Superblock chứa tên của volume, ID và dấu thời gian.



Hình 5.2: Cấu trúc của APFS.

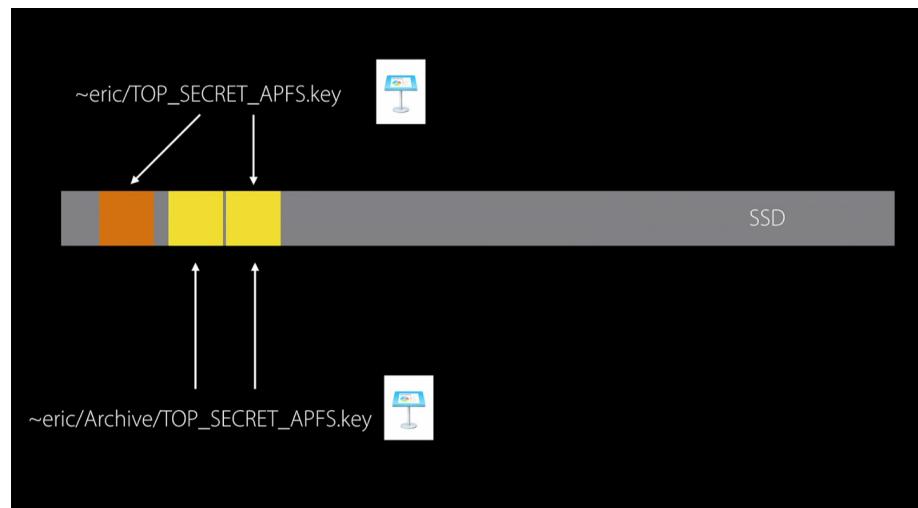
- Space Sharing - phân vùng không cần liên tiếp: Ví dụ một ổ đĩa được phân thành nhiều phân vùng như là A, B và một đoạn C trống. Sau khi đã phân xong, nếu người dùng muốn tăng dung lượng phân vùng A thì không được vì B đã nằm kế tiếp, các khối nhớ đã được phân chia xong. Chính vì vậy nên phải di chuyển B xuống C rồi mới mở rộng A ra được. Nhưng APFS quản lý phân vùng theo cách khác. Ổ đĩa như là một container bao quanh, trong này chỉ có một phân vùng vật lý duy nhất. Phân vùng đó lại được chia thành nhiều volume nhỏ hơn. Vì những volume này đều được chia theo logic nên khi cần tăng kích thước của A thì APFS chỉ cần lấy phần C còn trống và gán cho A. Điều này giúp không cần tốn thời gian chờ đợi di chuyển B. Đồng thời, khả năng mã hóa dữ liệu ở mức tệp tin và hỗ trợ Secure Erase giúp bảo vệ thông tin cá nhân của người dùng.



Hình 5.3: Minh họa về Space Sharing.

- Clone – copy file không tốn dung lượng: đối với APFS, khi copy file thì nó sẽ tham chiếu file mới về lại file cũ nên không làm tăng thêm dung lượng. MacOS vẫn xem đây là hai file riêng biệt nhưng về mặt lưu trữ vật lý thì nó chỉ là một file. Chỉ khi người

dùng chỉnh sửa 1 trong 2 file thì macOS mới tạo ra bản sao thật sự của file, khi đó phần chiếm dụng trên ổ mới tăng lên. Ví dụ như một file nhạc vừa có thể nằm trong folder iTunes, vừa nằm trong thư mục Dropbox để sao lưu. Ngoài ra, APFS còn sử dụng kỹ thuật Copy-on-Write (COW), chỉ ghi lại dữ liệu khi cần thiết để tránh việc trùng lặp dữ liệu và tăng hiệu suất.



Hình 5.4: Clone file trong APFS.

- Chức năng Snapshot và Reversion. Các ứng dụng, kể cả app của bên thứ ba có thể tạo ra bản sao lưu của cả phân vùng ở chế độ read only. Khi người dùng lỡ tay xóa hay chỉnh sửa sai file thi có thể phục hồi lại bản trước đó bằng cách vào snapshot lấy file cũ lại ra. Chức năng này rất hữu ích cho những ứng dụng giúp sao lưu, phục hồi file hay bảo trì hệ thống.
- APFS tối ưu hóa cho ổ đĩa flash và tích hợp của Multithreading giúp tăng tốc độ đọc và ghi dữ liệu. Ngoài ra, APFS sử dụng Metadata Quotas giúp quản lý hiệu quả không gian metadata, trong khi đó Checksums được sử dụng để đảm bảo tính toàn vẹn của dữ liệu.

Ngoài ra, macOS và iOS còn có một số điểm khác biệt.

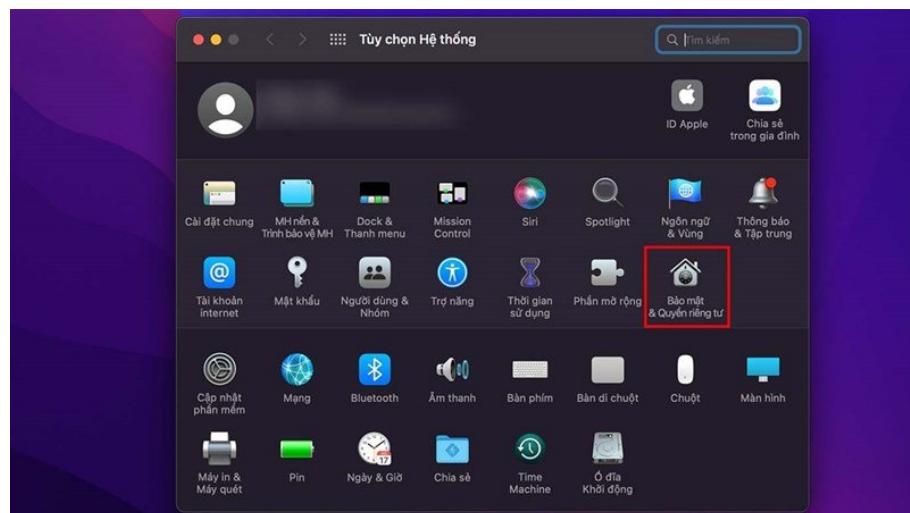
- Đối với macOS:

Người dùng truy cập và quản lý file và thư mục thông qua Finder. Có thể tạo, di chuyển, xóa, sao chép, dán file và thư mục tự do. Đồng thời, họ cũng có quyền lưu file bất kỳ nơi nào trên ổ đĩa máy tính và truy cập bất cứ lúc nào.

Người dùng cũng có thể cài đặt quyền truy cập và bảo mật file một

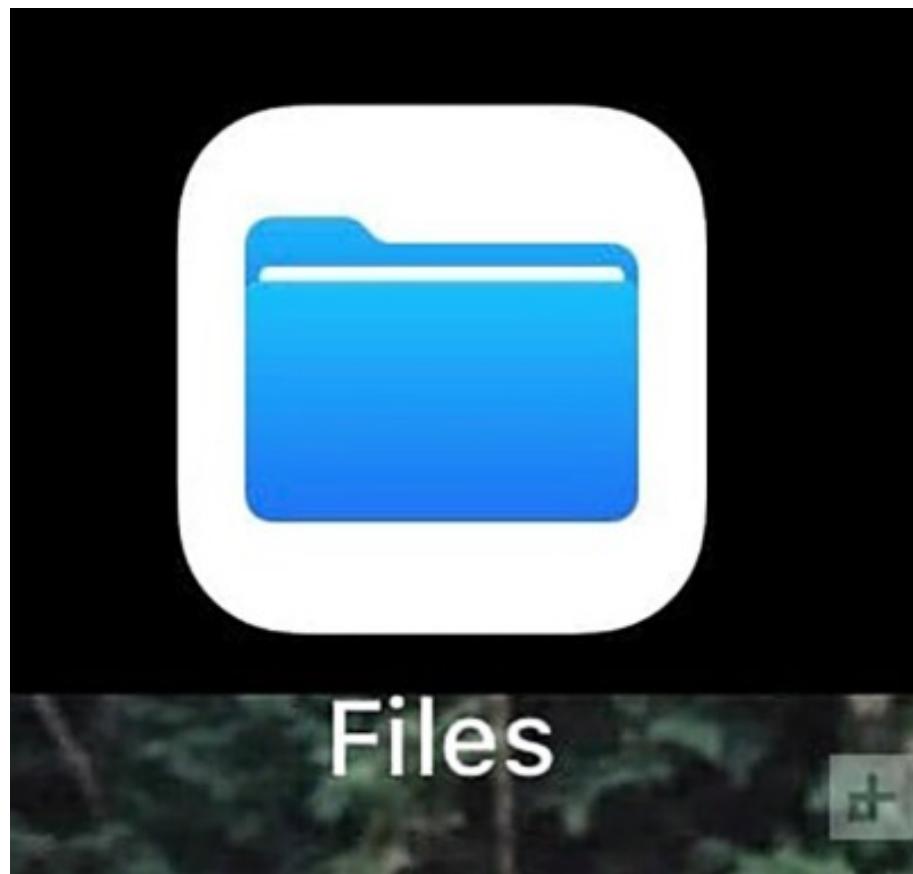
cách chi tiết. Có rất nhiều tùy chọn để người dùng có thể lựa chọn cài đặt về quyền truy cập về bảo mật như là Dịch vụ định vị, Danh bạ, Lịch, Lời nhắc,... Người dùng có thể kiểm soát những ứng dụng và trang web nào có thể được phép truy cập vào các tệp và thư mục tại các vị trí cụ thể bằng cách sau:

- Chọn menu Apple, nhấn vào Cài đặt hệ thống, sau đó bấm vào Quyền riêng tư và bảo mật.
- Bấm vào Tệp và Thư mục ở bên phải.
- Đối với từng ứng dụng trong danh sách, bật hoặc tắt khả năng truy cập vào các tệp và thư mục tại các vị trí cụ thể.



Hình 5.5: Vị trí của Bảo mật và Quyền riêng tư.

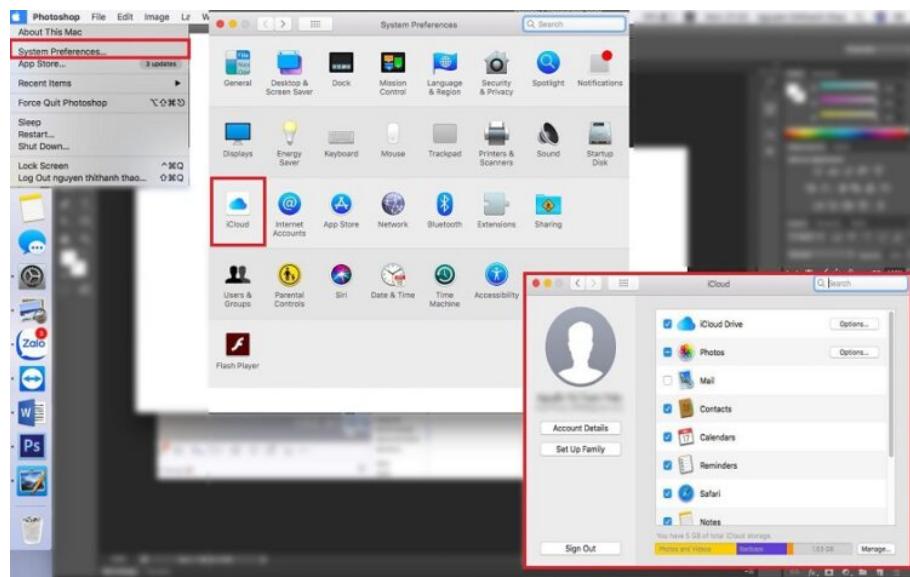
- Đối với iOS: Không cho phép người dùng truy cập hệ thống file tự do mà cần thông qua ứng dụng Files, ngoại trừ một số vùng như iCloud Drive và ứng dụng cụ thể. Không thể trực tiếp lưu trên hệ thống file iOS ngoại trừ một số vùng như vùng Documents của ứng dụng cụ thể. Không có quyền truy cập trực tiếp vào hệ thống file.



Hình 5.6: Ứng dụng Files.

Tuy có nhiều điểm khác biệt nhưng macOS và iOS vẫn có một số điểm tương đồng về việc quản lý file:

- Cả hai hệ thống đều tích hợp chẽ với iCloud Drive, cho phép người dùng lưu trữ và đồng bộ hóa tập tin và thư mục thông qua các thiết bị Apple khác nhau. iCloud Drive có khả năng lưu trữ an toàn tất cả các loại tài liệu trong iCloud và truy cập vào chúng từ bất kỳ thiết bị nào của mình trên web tại iCloud.com. Người dùng có thể thiết lập iCloud Drive bằng các bước sau:
 - Chọn menu Apple -> Cài đặt hệ thống, sau đó bấm vào ID Apple.
 - Bấm vào iCloud ở bên phải, sau đó bấm vào iCloud Drive và ấn vào dòng “Đồng bộ hóa Mac này” đối với Macbook hoặc “Đồng bộ hóa iPhone này” đối với iPhone.



Hình 5.7: Cách thiết lập iCloud Drive.

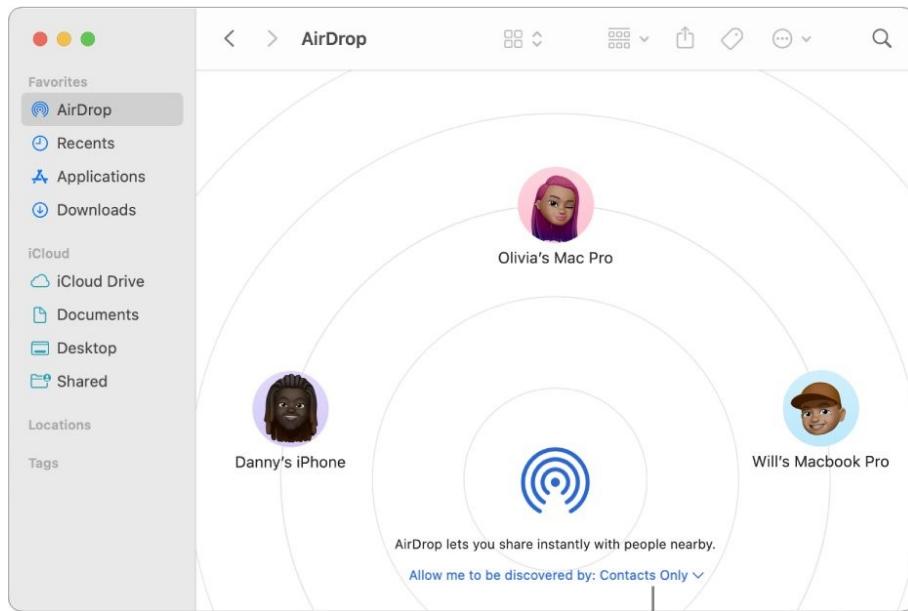
- MacOS và iOS đều cho phép chia sẻ và gửi file qua Email, AirDrop và các ứng dụng khác. Đối với iOS, chỉ cần mở một ứng dụng, nhấn vào Chia sẻ, sau đó nhấn vào AirDrop và chọn đối tượng muốn gửi thư mục đó đến.



Hình 5.8: Minh họa về AirDrop trên iPhone.

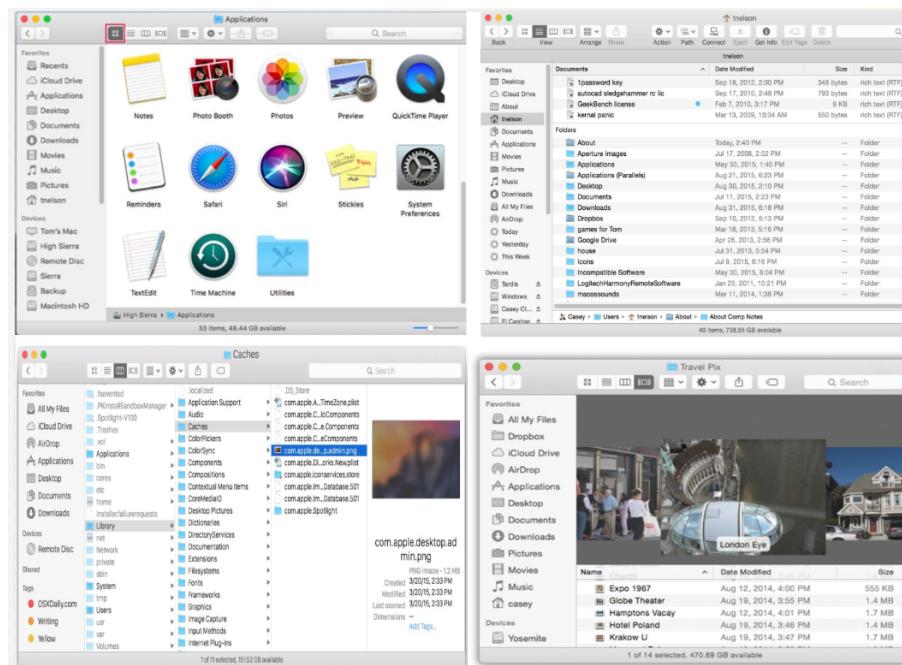
Đối với AirDrop trên máy Mac, người dùng có thể gửi các mục từ Finder, màn hình nền hoặc từ bên trong các ứng dụng như Safari hoặc Bản đồ. Ví dụ như:

- Từ màn hình nền hoặc cửa sổ Finder: Giữ Control khi bấm vào mục muốn gửi, chọn Chia sẻ từ menu phím tắt, sau đó chọn thiết bị muốn gửi mục đó đến.
- Từ Finder: Bấm AirDrop trong thanh bên Finder, sau đó kéo mục vào thiết bị muốn gửi mục đó.
- Từ ứng dụng: Bấm nút Chia sẻ trên thanh công cụ của ứng dụng, chọn AirDrop, sau đó chọn thiết bị muốn gửi mục đó.



Hình 5.9: AirDrop trên Macbook.

Hai hệ thống này đều có thể xem và sắp xếp tập tin, thư mục trong các chế độ khác nhau, bao gồm List View, Icon View, Column View và Cover Flow View.



Hình 5.10: Các chế độ xem trên Macbook.

5.2 Truy cập file

Hệ thống file trên macOS và iOS có một số điểm khác nhau về cách truy cập file như:

- Đối với macOS:

Sử dụng ứng dụng Finder làm trình quản lý file chính. Finder cho phép người dùng duyệt qua toàn bộ hệ thống file thông qua giao diện đồ họa. macOS có cấu trúc hệ thống file truyền thống và linh hoạt giúp cho người dùng dễ dàng quản lý tập tin và thư mục. Ngoài ra, người dùng còn có thể truy cập file thông qua Terminal bằng cách sử dụng các lệnh Unix và Bash.

- Đối với iOS:

Ứng dụng Files đảm nhiệm vai trò truy cập và quản lý file. Files cung cấp trải nghiệm đồng nhất để truy cập vào các tập tin và thư mục từ nhiều nguồn khác nhau như lưu trữ trên thiết bị, iCloud Drive và các dịch vụ lưu trữ đám mây khác.

Một số ứng dụng cụ thể cho phép người dùng truy cập và quản lý tập tin của riêng họ. Ví dụ như ứng dụng Mail cho phép đính kèm tập tin vào email.

Trong iOS, ứng dụng hoạt động trong môi trường “sandbox”, nghĩa là chúng có quyền truy cập vào không gian lưu trữ cụ thể của mình và không thể dễ dàng truy cập vào không gian lưu trữ của các ứng dụng khác. Điều này giúp bảo vệ sự riêng tư và bảo mật, nhưng cũng làm giảm khả năng chia sẻ dữ liệu giữa các ứng dụng và quản lý file.

Ngoài ra, iOS giới hạn sự truy cập vào hệ thống file để đảm bảo tính bảo mật và đơn giản hóa trải nghiệm người dùng. Người dùng không thể truy cập trực tiếp vào các thư mục hệ thống như trên MacOS.

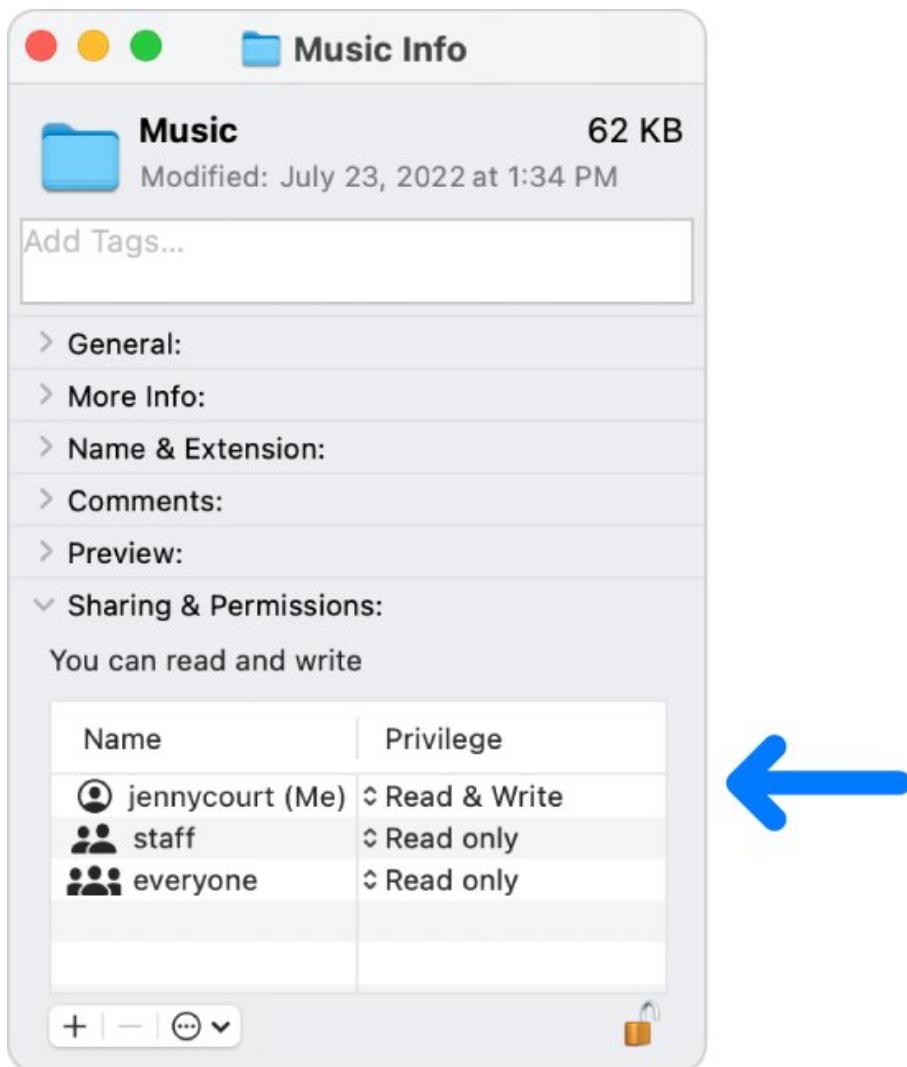
Tổng kết lại, do iOS được tối ưu hóa để sử dụng trên thiết bị di động như iPhone và iPad, trong khi macOS chủ yếu dành cho máy tính cá nhân và máy chủ. Chính vì sự di động và kích thước nhỏ hạn chế đi tính tương tác quản lý file trên iOS so với macOS.

5.3 Bảo mật file

- MacOS cung cấp bảo mật mạnh mẽ và linh hoạt cho việc quản lý tập tin và thư mục. Người dùng có thể thiết lập các quyền truy cập chi tiết, bao gồm quyền đọc, ghi, và thực thi. Người dùng có thể thay đổi cài đặt quyền ở cuối cửa sổ Thông tin cho tệp, thư mục hoặc ổ đĩa trong Finder.

Ví dụ, người dùng có thể thay đổi cài đặt quyền cho thư mục để những người dùng khác đăng nhập vào máy Mac hoặc kết nối với máy Mac đó để chia sẻ tệp có thể xem nhưng không thể thay đổi các

tệp trong thư mục.

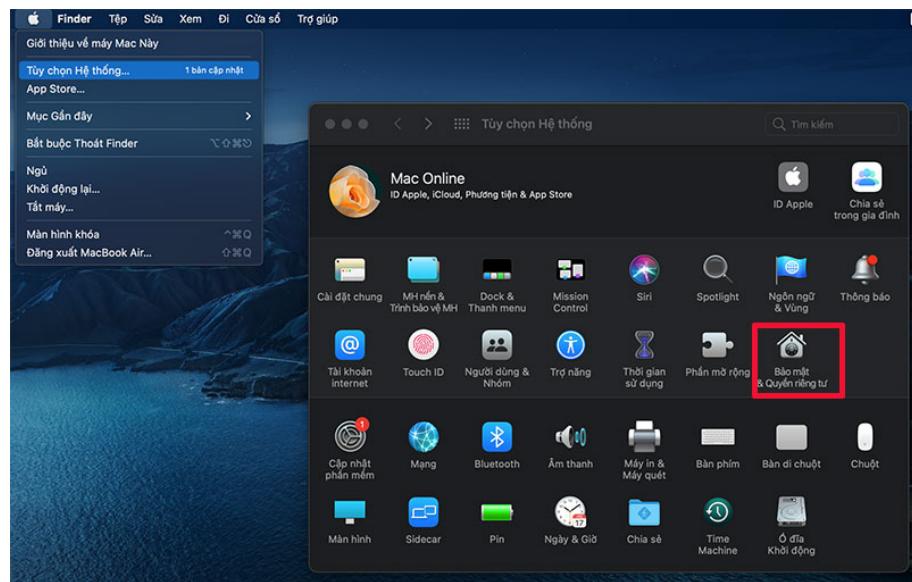


Hình 5.11: Thay đổi quyền cho thư mục

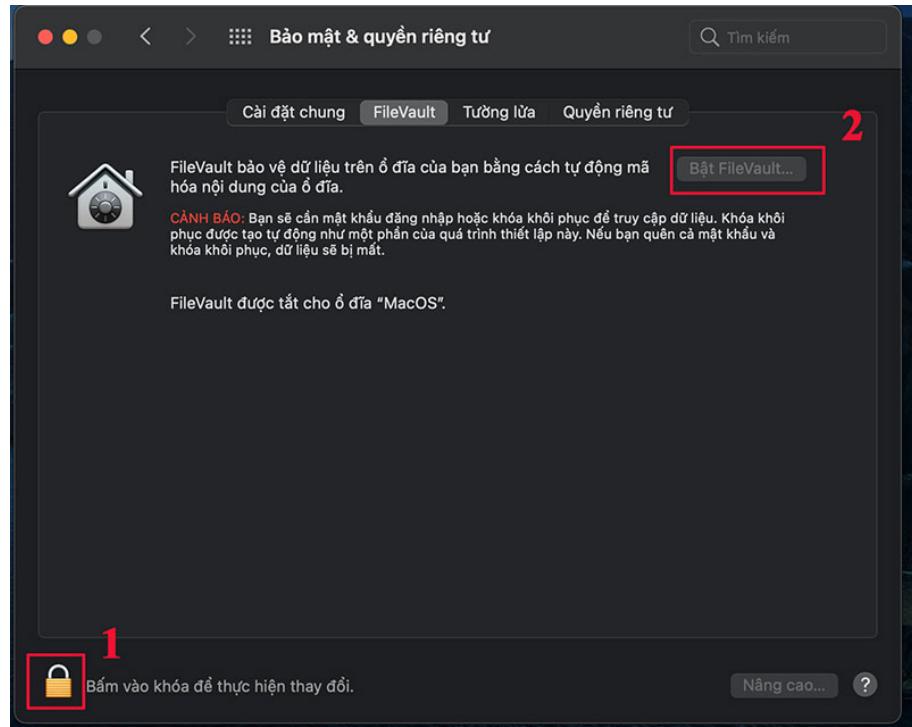
- Người dùng có khả năng mã hóa dữ liệu bằng FileVault để bảo vệ dữ liệu trên ổ đĩa. Điều này đảm bảo rằng tất cả dữ liệu trên máy tính của họ được bảo mật mà không cần phải lo lắng về người truy cập không ủy quyền. FileVault là hệ thống được tích hợp sẵn trên Macbook hỗ trợ mã hóa ổ cứng và chịu trách nhiệm bảo mật ổ cứng HDD và SSD cho Mac. Cơ chế hoạt động của FileVault là cung cấp một lớp bảo mật bổ sung bằng cách không cho người khác giải mã hoặc truy cập được vào dữ liệu của người dùng gốc mà không nhập mật khẩu đăng nhập của họ. Để đảm bảo sự bảo mật khi bật FileVault, các tính năng bảo mật khác cũng được bật, chẳng hạn như người dùng cần nhập mật khẩu để đăng nhập khi máy Mac được đánh thức khỏi chế độ ngủ hoặc sau khi thoát khỏi trình bảo vệ màn hình.
- Dưới đây là cách bật mã hóa ổ cứng trên máy Mac với FileVault:

- Nhấn vào menu Apple hoặc kích vào biểu tượng bánh răng trên

Dock, sau đó chọn Bảo mật và Quyền riêng tư.

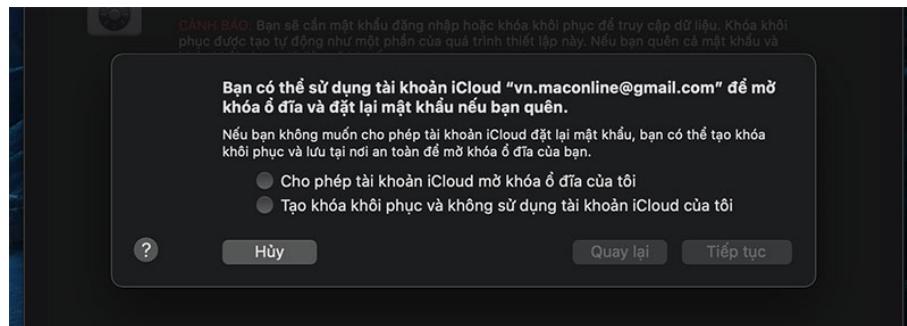


- Trong mục Bảo mật và Quyền riêng tư vừa mở, chọn tab FileVault và nhập vào biểu tượng ổ khóa ở góc dưới cùng bên trái cửa sổ và nhập mật khẩu máy hoặc dùng Touch ID để mở khóa thực hiện thay đổi cài đặt.



- Nhập vào “Bật FileVault...” khi đó máy Mac sẽ hiện lên 2 tùy chọn là “Cho phép tài khoản iCloud mở khóa ổ đĩa của tôi” nếu người dùng muốn sử dụng tài khoản iCloud để mở khóa ổ đĩa và

“Tạo khóa khôi phục và không sử dụng tài khoản iCloud của tôi” để tạo mã gồm một chuỗi các chữ cái và số được tạo riêng cho người dùng.



- iOS cũng cung cấp bảo mật mạnh, nhưng nó tập trung vào tính bảo mật cao hơn và kiểm soát nghiêm ngặt hơn đối với ứng dụng và tài liệu.

Mọi ứng dụng iOS chạy trong môi trường sandbox, và chúng chỉ có quyền truy cập vào dữ liệu của riêng mình hoặc dữ liệu được chia sẻ thông qua các API cụ thể. Môi trường sand box là môi trường thử nghiệm cô lập, cho phép người dùng chạy các chương trình hoạt file thực thi mà không gây hại, làm hỏng các ứng dụng, hệ thống hoặc nền tảng. Chính vì vậy, không có sandbox thì một ứng dụng có thể truy cập không giới hạn vào tất cả dữ liệu người dùng và tài nguyên hệ thống. Nhờ có môi trường sandbox này ngăn chặn ứng dụng khác truy cập dữ liệu của người dùng mà không có sự cho phép.

5.4 Quyền truy cập

- Trên macOS, người dùng có quyền truy cập tự do vào tất cả các tập tin và thư mục trừ khi đã thiết lập các quyền truy cập cụ thể hoặc sử dụng mã hóa dữ liệu. Người dùng có quyền cao hơn và kiểm soát tốt hơn về tài liệu của mình.
- Trên iOS, quyền truy cập vào tập tin và thư mục rất hạn chế. Người dùng chỉ có thể truy cập vào dữ liệu của ứng dụng cụ thể thông qua ứng dụng Files hoặc các ứng dụng khác đã được ứng dụng gốc chấp nhận. Điều này giúp bảo mật thiết bị và dữ liệu, nhưng cũng giới hạn tính linh hoạt trong việc quản lý tập tin.

Tóm lại, macOS và iOS có cách quản lý, bảo mật và truy cập file khá khác nhau. Mac OS cho phép quản lý file truyền thống thông qua Finder và cho phép người dùng có kiểm soát tốt hơn. Trong khi đó, iOS tập trung vào tính bảo mật và sự đơn giản hơn, và không cho phép

CHƯƠNG 5. QUẢN LÝ FILE

truy cập trực tiếp vào hệ thống file, chú trọng vào ứng dụng cụ thể để quản lý tập tin.

CHƯƠNG 6. SO SÁNH VÀ ĐẶC ĐIỂM RIÊNG

6.1 Các điểm tương đồng và khác biệt chính giữa macOS và iOS về cấu trúc hệ thống, hàm Shell, quản lý bộ nhớ và quản lý file

6.1.1 Các điểm tương đồng về cấu trúc hệ thống, hàm Shell, quản lý bộ nhớ và quản lý file

- **Cấu trúc hệ thống:**

Cả macOS và iOS đều chia sẻ hạt nhân Unix, cung cấp một môi trường ổn định và bảo mật cho hệ thống.

Hai hệ điều hành đều có cấu trúc thư mục cơ bản giống nhau, bao gồm thư mục người dùng, thư mục hệ thống và các thư mục hệ thống quan trọng khác.

macOS và iOS đều sử dụng hệ thống tập tin APFS (Apple File System) từ macOS High Sierra trở đi. APFS mang lại nhiều cải tiến về hiệu suất và bảo mật.

- **Hàm Shell:**

Cả macOS và iOS chia sẻ hạt nhân Unix, vì vậy cả hai đều cung cấp môi trường để thực hiện các lệnh và tác vụ hệ thống. Ngoài ra, hai hệ điều hành đều có Terminal, giúp người dùng tương tác với hệ thống thông qua Shell.

Sử dụng một hàm Shell Unix dựa trên Bash (Bourne Again Shell) mặc định. Điều này cho phép người dùng sử dụng các lệnh Shell để thực hiện nhiều tác vụ hệ thống, tuy nhiên macOS Catalina và các phiên bản sau đó đã chuyển sang Zsh(Z Shell) làm shell mặc định.

- **Quản lý bộ nhớ:**

Hai hệ thống đều sử dụng quản lý bộ nhớ thông minh để tối ưu hóa hiệu suất. Điều này bao gồm việc quản lý bộ nhớ ảo và tự động dọn dẹp để giảm tình trạng rác.

macOS và iOS đều có các công cụ kiểm tra lỗi bộ nhớ như Instruments trên Xcode để phát hiện và sửa lỗi liên quan đến bộ nhớ.

Ngoài ra, hai hệ điều hành này còn giới hạn bộ nhớ mà một ứng dụng có thể sử dụng để đảm bảo tính ổn định của hệ thống.

- **Quản lý file:** Đều sử dụng hệ thống tập tin APFS. Hai hệ thống này

đều có ứng dụng quản lý tập tin riêng biệt. Trên macOS có Finder trong khi iOS sử dụng ứng dụng Files. macOS và iOS đều cho phép người dùng quản lý tập tin và thư mục trực tiếp từ giao diện người dùng của họ. Ngoài ra, hai hệ điều hành đều tích hợp với các dịch vụ lưu trữ đám mây như iCloud, Dropbox, Google Drive và OneDrive.

6.1.2 Khác biệt chính giữa macOS và iOS về cấu trúc hệ thống, hàm Shell, quản lý bộ nhớ và quản lý file

- **Cấu trúc hệ thống:**

Về giao diện và trải nghiệm người dùng, macOS được thiết kế cho máy tính cá nhân với giao diện đồ họa phong phú, trong khi iOS được tối ưu hóa cho thiết bị di động và sử dụng giao diện cảm ứng.

macOS có một hệ thống đồ họa mạnh mẽ cho đồ họa 2D và 3D còn iOS tập trung vào giao diện người dùng đa phương tiện, đặc biệt là UIKit để hỗ trợ cho đồ họa 2D.

iOS tích hợp chặt chẽ giữa các ứng dụng có môi trường đóng, giảm tính phức tạp, ngược lại macOS tập trung vào tính đa nhiệm và có khả năng mở nhiều ứng dụng cùng một lúc.

Hệ thống Menu và Dock của macOS giúp truy cập ứng dụng trong khi iOS sử dụng mô hình Home Screen với các ứng dụng được sắp xếp trên màn hình chính.

- **Hàm Shell:**

Đối với macOS, Shell Scripting được sử dụng phổ biến. Trong đó, việc sử dụng Shell trong iOS thường được thực hiện thông qua ứng dụng Terminal từ cộng đồng nhà phát triển hoặc thông qua công cụ jailbreak.

So với được tự do tương tác với hệ thống thông qua Shell để thực hiện nhiều tác vụ quản lý và phát triển trong macOS, iOS giới hạn khả năng tương tác với hệ thống thông qua Shell, hạn chế đối với tính bảo mật và để tối ưu hóa trải nghiệm người dùng cuối.

Ngoài ra, Shell trên macOS thường được sử dụng để thực hiện nhiều tác vụ cùng một lúc và tận dụng tính linh hoạt của hệ thống máy tính.

- **Quản lý bộ nhớ:**

macOS có ứng dụng Activity Monitor để theo dõi và kiểm soát quá

trình hoạt động của hệ thống và sử dụng bộ nhớ. Trong khi đó, iOS không cung cấp quyền truy cập trực tiếp cho người dùng để quản lý bộ nhớ một cách chi tiết. Do đó, macOS có khả năng quản lý bộ nhớ linh hoạt hơn, đặc biệt là trên các máy tính có dung lượng lớn.

Do iOS tập trung vào tối ưu hóa điện năng và hiệu suất nên hạn chế về bộ nhớ trên iOS đôi khi có thể gây ra việc ứng dụng phải tải lại dữ liệu khi người dùng chuyển đổi giữa các ứng dụng, nhưng đồng thời cũng giúp duy trì hiệu suất tổng thể của hệ thống.

- **Quản lý File:**

Quản lý file khác nhau giữa hai hệ thống. Trên macOS, người dùng có quyền truy cập và quản lý file bằng cách sử dụng Finder và các ứng dụng quản lý file. Trong iOS, quản lý file hạn chế hơn và thường thực hiện qua ứng dụng Files hoặc ứng dụng ứng dụng cụ thể, không thể truy cập trực tiếp vào hệ thống file.

6.2 Ưu điểm và hạn chế của hai hệ điều hành về cấu trúc hệ thống, hàm Shell, quản lý bộ nhớ và quản lý file

Dưới đây là một số đánh giá về ưu điểm và hạn chế của cả hai hệ điều hành macOS và iOS trong từng mục về cấu trúc hệ thống, hàm Shell, quản lý bộ nhớ và quản lý file:

6.2.1 Về ưu điểm

- **Cấu trúc hệ thống:**

macOS được thiết kế để chạy trên máy tính cá nhân và máy chủ, mang lại tính linh hoạt trong việc thực hiện nhiều tác vụ đa dạng, tương thích với nhiều loại phần cứng từ nhà sản xuất khác nhau. Cho phép mở nhiều ứng dụng và cửa sổ cùng một lúc, giúp người dùng dễ dàng chuyển đổi giữa các ứng dụng. Ngoài ra, macOS còn cung cấp một môi trường phát triển mạnh mẽ với Xcode, hỗ trợ phát triển ứng dụng cho nền tảng macOS và iOS.

iOS là một hệ điều hành có tính bảo mật cao, được tối ưu hóa cho trải nghiệm di động, mang lại sự dễ sử dụng và hiệu suất tốt trên các thiết bị như iPhone và iPad. Tích hợp chặt chẽ với phần cứng của Apple và các dịch vụ như iCloud, mang lại sự liên kết mạnh mẽ giữa các thiết bị.

- **Hàm Shell:**

macOS cung cấp môi trường Unix mạnh mẽ với Terminal, giúp phát triển và quản lý hệ thống dễ dàng. Trong khi đó, iOS không hỗ trợ trực tiếp môi trường Shell truyền thống, giúp tăng tính bảo mật trên thiết bị di động.

- **Quản lý bộ nhớ:**

Cả hai hệ thống đều có khả năng quản lý bộ nhớ linh hoạt, hỗ trợ quản lý bộ nhớ hiệu quả và tự động thông qua tính năng như thu gom rác và quản lý bộ nhớ tự động. Hỗ trợ kiểm tra lỗi bộ nhớ để tránh lỗi và các lỗi hỏng bảo mật liên quan đến quản lý bộ nhớ.

macOS có khả năng mở nhiều ứng dụng cùng một lúc, giúp tăng linh hoạt nhiệm cho người dùng khi cần xử lý nhiều tác vụ. Còn iOS thiên về tối ưu hóa hiệu suất, giảm tiêu tốn bộ nhớ, đảm bảo rằng các ứng dụng chạy mượt mà và không gặp tình trạng giật lag.

- **Quản lý File:**

macOS thông qua Finder cho phép người dùng truy cập trực tiếp vào hệ thống file để quản lý theo cách linh hoạt và mạnh mẽ. Người dùng có thể dễ dàng tích hợp và quản lý tập tin trên iCloud Drive, đồng bộ hóa dữ liệu giữa các thiết bị của họ. Người dùng có quyền truy cập và quản lý tập tin hệ thống, đồng thời có thể sử dụng nhiều ứng dụng và cửa sổ cùng một lúc giúp tăng tính đa nhiệm.

iOS đem lại trải nghiệm quản lý file dễ dàng thông qua ứng dụng Files. Việc quản lý tập tin thông qua các ứng dụng cụ thể giảm sự phức tạp trong quá trình tìm kiếm và sắp xếp.

6.2.2 Về mặt hạn chế

- **Cấu trúc hệ thống:**

macOS chủ yếu được tối ưu hóa cho máy tính cá nhân, không có sự tích hợp chặt chẽ với các thiết bị di động như iOS.

iOS giới hạn tính linh hoạt so với macOS, người dùng có ít quyền truy cập vào hệ thống và tập tin. Hạn chế việc mở nhiều ứng dụng cùng một lúc và hạn chế về mặt quản lý file so với macOS.

- **Hàm Shell:**

Khi sử dụng Terminal và hàm Shell đòi hỏi phải có kiến thức kỹ thuật và có thể gây rủi ro nếu không biết cách sử dụng khi dùng macOS.

Trên iOS không có hàm Shell trực tiếp khiến hạn chế tính linh hoạt và khả năng sửa đổi hệ thống và việc thực hiện các tác vụ thông qua môi trường dòng lệnh trở nên khó khăn.

- **Quản lý bộ nhớ:**

Trong iOS, tuy việc quản lý bộ nhớ nghiêm ngặt giúp đảm bảo tính ổn định của ứng dụng nhưng đồng thời cũng tạo ra hạn chế về tính linh hoạt. Còn macOS tuy có khả năng quản lý bộ nhớ tốt nhưng không linh hoạt như iOS trong môi trường di động với dung lượng bộ nhớ hạn chế.

- **Quản lý file:**

macOS có cấu trúc phức tạp khiến việc quản lý tập tin khó khăn đối với người dùng. Còn iOS lại giới hạn tính linh hoạt, đặc biệt với những người dùng đòi hỏi kiểm soát chi tiết, yêu cầu đa nhiệm.

Tổng kết lại, iOS tập trung vào tích hợp ứng dụng và đơn giản hóa trải nghiệm người dùng trong khi macOS thiên về tính linh hoạt và khả năng sử dụng cho người dùng.

CHƯƠNG 7. ỨNG DỤNG VÀ HIỆU SUẤT

7.1 Hiệu suất ứng dụng trên macOS và iOS

7.1.1 So sánh hiệu suất

Hiệu suất khi chạy ứng dụng trên macOS và iOS phụ thuộc vào nhiều yếu tố, bao gồm cả phần cứng của thiết bị, phiên bản hệ điều hành, và cách ứng dụng được tối ưu hóa cho mỗi nền tảng.

- macOS tối ưu hóa hiệu suất ứng dụng trên máy tính cá nhân và máy tính xách tay bằng cách quản lý tài nguyên hệ thống thông minh. Hệ điều hành này ưu tiên các ứng dụng đang hoạt động và đảm bảo rằng các ứng dụng không hoạt động trên màn hình không sử dụng tài nguyên quá nhiều.
- iOS tối ưu hóa hiệu suất ứng dụng trên các thiết bị di động thông qua quản lý tài nguyên thông minh, sử dụng bộ nhớ tạm, tối ưu hóa tiêu thụ pin, và quản lý ứng dụng chạy nền. Điều này giúp đảm bảo rằng ứng dụng chạy trơn tru, không gây tình trạng giật lag và tiết kiệm tài nguyên. iOS là một hệ điều hành được thiết kế đặc biệt để cung cấp trải nghiệm mượt mà và hiệu quả trên các thiết bị di động với tài nguyên hạn chế.

Cụ thể:

- macOS:
 - Hệ điều hành macOS có khả năng tối ưu hóa đảm bảo rằng tài nguyên hệ thống như bộ nhớ RAM và CPU được sử dụng một cách hiệu quả. Hệ điều hành cần ưu tiên các ứng dụng đang hoạt động trên màn hình, đồng thời giảm thiểu tác động đến hiệu suất của các ứng dụng không hoạt động hoặc đang chạy ẩn.(chế độ App Nap)
 - Hệ điều hành macOS hỗ trợ đa luồng mạnh mẽ. Điều này cho phép ứng dụng tận dụng sức mạnh của CPU đa nhân và chia sẻ công việc giữa nhiều luồng, tối ưu hóa hiệu suất.
- iOS:
 - Hệ điều hành iOS sử dụng bộ nhớ tạm để lưu trữ dữ liệu tạm thời, giúp tăng tốc quá trình truy cập dữ liệu. Nó cũng tự động xóa dữ liệu tạm khi không còn cần thiết để giải phóng tài nguyên.
 - Hệ điều hành iOS có khả năng quản lý tài nguyên một cách thông minh. Nó sẽ ưu tiên các ứng dụng đang hoạt động trên màn hình,

đồng thời đảm bảo rằng các ứng dụng không hiển thị trên màn hình không sử dụng tài nguyên quá nhiều. Điều này giúp người dùng không gặp tình trạng giật lag khi sử dụng nhiều ứng dụng.

Như vậy, không thể khẳng định hệ điều hành nào tốt hơn mà không xem xét ngữ cảnh sử dụng cụ thể. macOS thường mạnh mẽ hơn về cấu hình phần cứng, nhưng iOS được tối ưu hóa cho thiết bị di động và có hiệu suất tốt cho các nhiệm vụ di động thông thường.

7.1.2 Ví dụ về hiệu suất của ứng dụng iMovie thiết bị macOS và iOS thông thường

- Tốc độ:

- iMovie trên macOS:

- Chất lượng video: iMovie trên macOS cho phép chỉnh sửa video với độ phân giải 4K và 60 khung hình/giây hoặc cao hơn. Điều này có nghĩa rằng người dùng có thể tạo ra video với chất lượng rất cao và độ phân giải sắc nét.
- Tốc độ xuất video: Máy tính macOS thường có cấu hình phần cứng mạnh hơn, cho phép iMovie xuất video nhanh chóng. Xuất video 4K hoặc video với hiệu ứng phức tạp có thể diễn ra nhanh hơn và không gây trễ đợi.

- iMovie trên iOS:

- Chất lượng video: iMovie trên iOS cũng hỗ trợ việc chỉnh sửa video 4K và video với độ phân giải cao, nhưng hiệu suất có thể bị giới hạn bởi cấu hình phần cứng của thiết bị di động. Điều này có nghĩa rằng người vẫn có thể tạo ra video chất lượng cao, nhưng với một số hạn chế như: nhanh chóng làm đầy dung lượng lưu trữ của thiết bị (Video 4K chiếm nhiều dung lượng lưu trữ hơn so với video với độ phân giải thấp hơn.), việc chỉnh sửa video 4K có thể làm giảm thời lượng pin nhanh chóng, và việc làm việc với độ phân giải cao có thể tạo ra nhiệt độ, làm tăng khả năng giảm hiệu suất để tránh quá nhiệt, việc chỉnh sửa video với độ phân giải cao cũng có thể đòi hỏi nhiều tài nguyên xử lý, và thiết bị cũ hơn có thể gặp khó khăn trong việc duy trì hiệu suất ổn định.
- Tốc độ xuất video: Xuất video trên thiết bị di động thường có tốc độ chậm hơn so với máy tính, đặc biệt khi đây là các thiết bị di động có cấu hình thấp hơn. Việc xuất video 4K hoặc video với hiệu ứng có

thể mất nhiều thời gian hơn.

- Tài nguyên tiêu thụ:

• iMovie trên macOS:

- CPU: iMovie trên macOS có thể tận dụng nhiều lõi CPU để xử lý video và hiệu ứng phức tạp. Nó sử dụng CPU mạnh để đảm bảo tốc độ xử lý nhanh chóng.
- Bộ nhớ RAM: Ứng dụng sử dụng một lượng lớn RAM để lưu trữ dữ liệu tạm thời và hiệu ứng. Nếu bạn làm việc với video 4K và hiệu ứng phức tạp, iMovie có thể sử dụng nhiều RAM.

• iMovie trên iOS:

- iMovie trên iOS có thể sử dụng CPU của thiết bị di động, nhưng tài nguyên này có thể giới hạn hơn so với máy tính. Hiệu suất xử lý video và hiệu ứng có thể thấp hơn trên các thiết bị di động.
- Bộ nhớ RAM: Thiết bị di động thường có lượng RAM giới hạn so với máy tính. Việc làm việc với video 4K và hiệu ứng có thể đòi hỏi quản lý tài nguyên tốt để đảm bảo hiệu suất ổn định.

7.2 Cách phát triển ứng dụng

Các bước trong quá trình phát triển ứng dụng cho macOS và iOS khá tương đồng:

Bước 1: Chọn ngôn ngữ lập trình

- Cả 2 hệ điều hành macOS và iOS đều lựa chọn giữa Swift hoặc Objective-C làm ngôn ngữ lập trình chính cho ứng dụng macOS. Swift thường là lựa chọn ưa thích hiện tại.
- Ngoài 2 ngôn ngữ chính trên, Apple cũng hỗ trợ một số ngôn ngữ lập trình khác cho việc phát triển ứng dụng trên các nền tảng của mình như: C/C++ (Cũng được hỗ trợ cho việc phát triển ứng dụng trên các hệ điều hành của Apple.) JavaScript (via React Native): React Native là một framework cho phép sử dụng JavaScript để xây dựng ứng dụng di động, và nó hỗ trợ cả iOS và Android. Đối với iOS, React Native có thể được sử dụng để phát triển một phần hoặc toàn bộ ứng dụng. SwiftUI (Swift UI): SwiftUI là một framework UI được xây dựng bằng Swift. Nó cung cấp một cách khác để xây dựng giao diện người dùng so với UIKit, giúp việc phát triển trở nên linh hoạt và dễ dàng.

hơn.

Bước 2: Cài đặt Xcode

- Tải và cài đặt Xcode để tạo và quản lý dự án, thiết kế giao diện người dùng, viết mã nguồn, và kiểm tra ứng dụng, môi trường phát triển tích hợp (IDE) chính thức của Apple cho cả 2 nền tảng macOS và iOS.
- Mặc dù Xcode là IDE được khuyến khích và hỗ trợ chặt chẽ từ Apple, nhưng vẫn có một số IDE và công cụ phát triển phần mềm khác có thể được sử dụng để phát triển ứng dụng iOS và macOS:

Visual Studio Code: Một IDE nhẹ và đa nền tảng từ Microsoft, hỗ trợ nhiều ngôn ngữ lập trình bao gồm cả Swift và Objective-C thông qua các extension.

AppCode: Được phát triển bởi JetBrains, AppCode là một IDE chuyên nghiệp dành cho phát triển ứng dụng iOS/macOS với hỗ trợ đặc biệt cho Swift và Objective-C.

IntelliJ IDEA: Cũng từ JetBrains, IntelliJ IDEA hỗ trợ phát triển ứng dụng iOS với sự tích hợp Swift và Objective-C.

Eclipse: Một IDE đa nhiệm và mã nguồn mở, có thể được cấu hình để phát triển ứng dụng iOS và macOS. Việc sử dụng một IDE không chính thức có thể đối mặt với một số hạn chế, như thiếu tích hợp sâu vào hệ sinh thái của Apple, hỗ trợ không đầy đủ cho các tính năng mới, và khả năng bảo trì kém. Việc sử dụng Xcode vẫn là lựa chọn được khuyên dùng.

Bước 3: Tạo dự án

- Bắt đầu bằng việc tạo một dự án mới trong Xcode. Chọn loại dự án và cài đặt tên và vị trí cho dự án.

Bước 4: Thiết kế giao diện người dùng:

- macOS: Giao diện người dùng của ứng dụng macOS thường được thiết kế để hoạt động trên màn hình lớn của máy tính để bàn hoặc máy tính xách tay. Các yếu tố như cửa sổ, thanh công cụ và menu được sử dụng rộng rãi.
- iOS: Giao diện người dùng của ứng dụng iOS thiết kế dành riêng cho các thiết bị di động như iPhone và iPad, với các phần khác nhau như các cửa sổ ứng dụng và giao diện cảm ứng.

Bước 5: Lập trình ứng dụng

- Viết mã nguồn ứng dụng bằng ngôn ngữ lập trình (Swift hoặc Objective-C). Sử dụng các API và framework của macOS để xử lý logic ứng dụng, tương tác với hệ thống, và thực hiện các chức năng cụ thể.
- Viết mã nguồn ứng dụng bằng ngôn ngữ lập trình (Swift hoặc Objective-C). Sử dụng các API và framework của iOS để xử lý logic ứng dụng, tương tác với hệ thống, và thực hiện các chức năng cụ thể.

Bước 6: Kiểm tra và gỡ lỗi

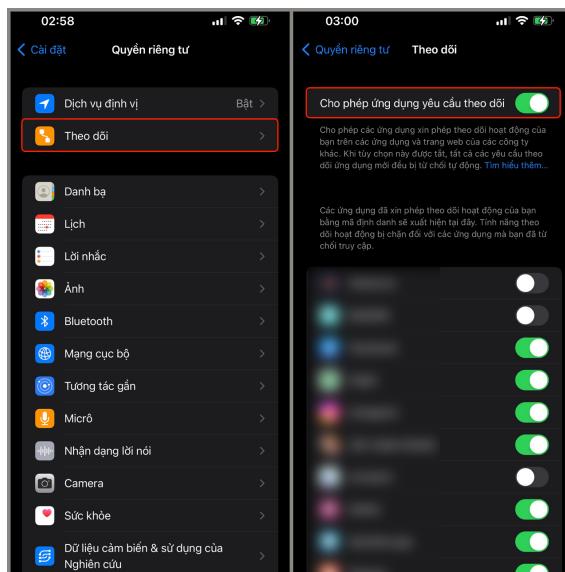
Thực hiện kiểm tra và gỡ lỗi thường xuyên để đảm bảo rằng ứng dụng hoạt động một cách ổn định.

CHƯƠNG 8. BẢO MẬT VÀ QUẢN LÝ THIẾT BỊ

8.1 Bảo mật trên macOS vs iOS

Cả macOS và iOS đều là những hệ điều hành được đánh giá cao về tính bảo mật. Cả hai hệ điều hành đều cung cấp một loạt tính năng bảo mật để giúp bảo vệ người dùng khỏi các mối đe dọa, chẳng hạn như phần mềm độc hại, tấn công mạng và truy cập trái phép.

- **Mã hóa dữ liệu:** Cả macOS và iOS đều hỗ trợ mã hóa dữ liệu để ngăn chặn truy cập trái phép. macOS sử dụng mã hóa AES-256 để mã hóa dữ liệu trên thiết bị. Ngoài ra macOS còn sử dụng mã hóa FileVault để mã hóa toàn bộ ổ cứng, việc bật FileVault cung cấp một lớp bảo mật bổ sung bằng cách không cho người khác giải mã hoặc truy cập được vào dữ liệu của bạn mà không nhập mật khẩu đăng nhập của bạn. Trong khi đó iOS sử dụng mã hóa AES-256 để mã hóa dữ liệu trên thiết bị. AES-256 là một thuật toán về mã hóa đối xứng, có độ dài khóa 256 bits, khi người dùng ghi dữ liệu vào thiết bị lưu trữ, bộ tạo số ngẫu nhiên (RNG) tạo ra một mã khóa đối xứng 256 bit và sau đó chuyển qua cho AES, AES sau đó mã hóa dữ liệu gốc thành dữ liệu mã hóa. Còn tính năng FileVault sử dụng công nghệ mật mã khối XTS-AES 128 và khóa 256. XTS-AES 128 sẽ sử dụng hai khóa AES-128 được tách từ AES-256 để mã hóa dữ liệu, một khối AES-128 sẽ được sử dụng để thực hiện mã hóa AES-128, khối còn lại sẽ được sử dụng để mã hóa “Tweak Value”, “Tweak Value” sẽ được mã hóa sửa đổi thêm các hàm của GF và XOR.
- **Kiểm soát truy cập ứng dụng:** Cả macOS và iOS đều cung cấp các tính năng kiểm soát truy cập để hạn chế quyền truy cập vào dữ liệu và ứng dụng. macOS sử dụng tính năng Gatekeeper, tính năng này có nhiệm vụ ký mã và xác minh những ứng dụng, phần mềm mà người dùng tải xuống, trước khi chạy trên hệ điều hành nhằm ngăn chặn những phần mềm độc hại phát tán virus, tính năng này chỉ cho phép các ứng dụng tải trên App Store hoặc từ các nhà phát triển có liên kết với Apple chạy trên hệ thống. Trong khi iOS sử dụng App Store để kiểm duyệt ứng dụng trước khi phát hành.



Hình 8.1: Quản lý cho phép ứng dụng theo dõi trên iOS

- **Xác thực người dùng:** Cả macOS và iOS đều hỗ trợ một số phương thức xác thực người dùng, bao gồm mật khẩu, Touch ID. Đặc biệt trên iOS có thêm tính năng Face ID. Mật khẩu là phương thức xác thực cơ bản nhất, trong khi Touch ID và Face ID sử dụng sinh trắc học để xác thực người dùng.
- **Quản lý mật khẩu:** Cả macOS và iOS đều có trình quản lý mật khẩu tích hợp sẵn để giúp người dùng lưu trữ và quản lý mật khẩu của họ một cách an toàn. macOS sử dụng tính năng Keychain Access và iCloud Keychain, tính năng Keychain Access này hỗ trợ người dùng lưu trữ mật khẩu và thông tin tài khoản nơi lưu trữ, từ đó giảm số lượng mật khẩu người dùng phải nhớ và quản lý. iOS sử dụng tính năng iCloud Keychain, tính năng này cũng tương tự Keychain Access trên macOS khi nó giúp người dùng lưu trữ, quản lý mật khẩu và thông tin người dùng, các thông tin này đã được Apple mã hóa đầu cuối để bảo mật thông tin. Nhờ các tính năng này giúp người dùng ít phải nhớ mật khẩu hơn, từ đó tạo thói quen cho người dùng đặt mật khẩu phức tạp và khó đoán hơn tăng tính năng bảo mật cho hệ thống. Keychain Access và iCloud Keychain đều được mã hóa dữ liệu bằng một loại mã đặc biệt, mã này được tạo ra từ thông tin dành riêng cho thiết bị và ghép nối với mật khẩu của thiết bị để tăng tính bảo mật cho các dữ liệu của người dùng.
- **Quản lý thiết bị:** Cả macOS và iOS đều cho phép người dùng quản lý thiết bị của họ từ xa. Điều này có thể được sử dụng để áp dụng các chính sách bảo mật và hạn chế truy cập. macOS sử dụng tính năng

Apple Configurator, Remote Management, để dùng tính năng Remote Management người dùng phải kích hoạt tính năng này trong System Preferences. iOS sử dụng tính năng Apple Configurator, để kích hoạt tính năng này trên thiết bị người dùng cần đăng ký giám sát và quản lý thiết bị di động (MDM), từ đó có thể quản lý thiết bị iOS từ xa. Trong đó, Remoto Management sử dụng 2 giao thức TCP và UDP để truyền thông tin và dữ liệu, các dữ liệu này được bảo mật bằng cách mã hóa theo các thuật toán như RSA-2048, AES-128, DH-1024,... tùy theo phiên bản macOS của thiết bị.

Dưới đây là một số điểm giống và khác nhau của tính năng bảo mật trên macOS và iOS:

Tính năng	macOS	iOS
Mã hóa dữ liệu	FileVault, AES-256	AES-256
Kiểm soát truy cập dữ liệu	Định cấu hình quyền truy cập dữ liệu cho từng ứng dụng	Định cấu hình quyền truy cập dữ liệu cho từng ứng dụng
Kiểm soát truy cập ứng dụng	Gatekeeper, App Store	App Store
Xác thực người dùng	Mật khẩu, Touch ID	Mật khẩu, Touch ID và Face ID
Quản lý mật khẩu	Keychain Access, iCloud Keychain	iCloud Keychain
Quản lý thiết bị	RemoteManagement, Apple Configurator	Apple Configurator

Các tính năng bảo mật trên đã được người dùng, lập trình viên và các tổ chức trong lĩnh vực bảo mật công nhận. Do đó dễ dàng nhận thấy cả macOS và iOS đều có tính bảo mật rất cao. Apple có toàn quyền kiểm soát phần cứng và phần mềm nên cũng dễ dàng triển khai khác tính năng bảo mật mới cho cả macOS và iOS.

8.2 Quản lý thiết bị

8.2.1 Quản lý cấu hình và dịch vụ

Cấu hình và dịch vụ của macOS và iOS bao gồm các cài đặt và tùy chọn cho hệ điều hành và các ứng dụng. Hai hệ điều hành trên đều cung cấp tính năng quản lý cấu hình và dịch vụ của mình theo nhiều cách như:

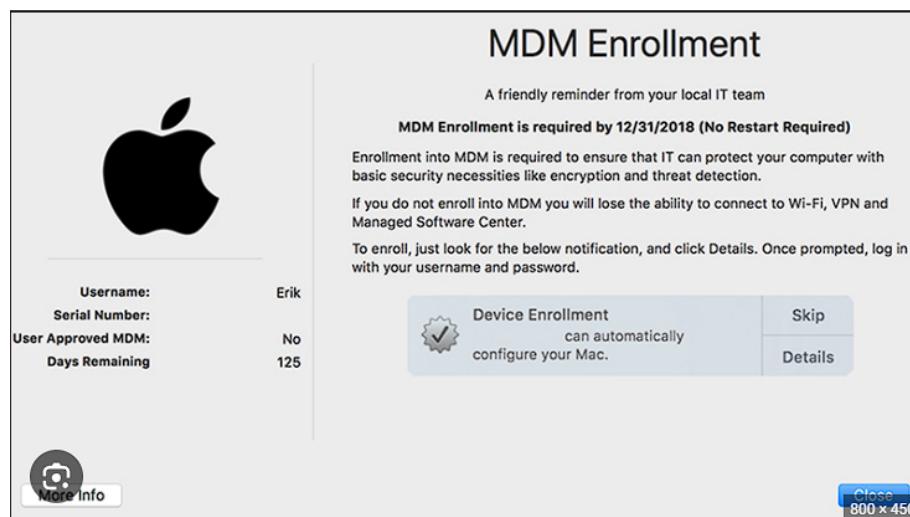
- **Thông qua ứng dụng Cài đặt:** có thể sử dụng ứng dụng Cài đặt để quản

lý nhiều cài đặt và tùy chọn của thiết bị của mình.



Hình 8.2: Quản lý cấu hình và dịch vụ trong ứng dụng cài đặt của iOS

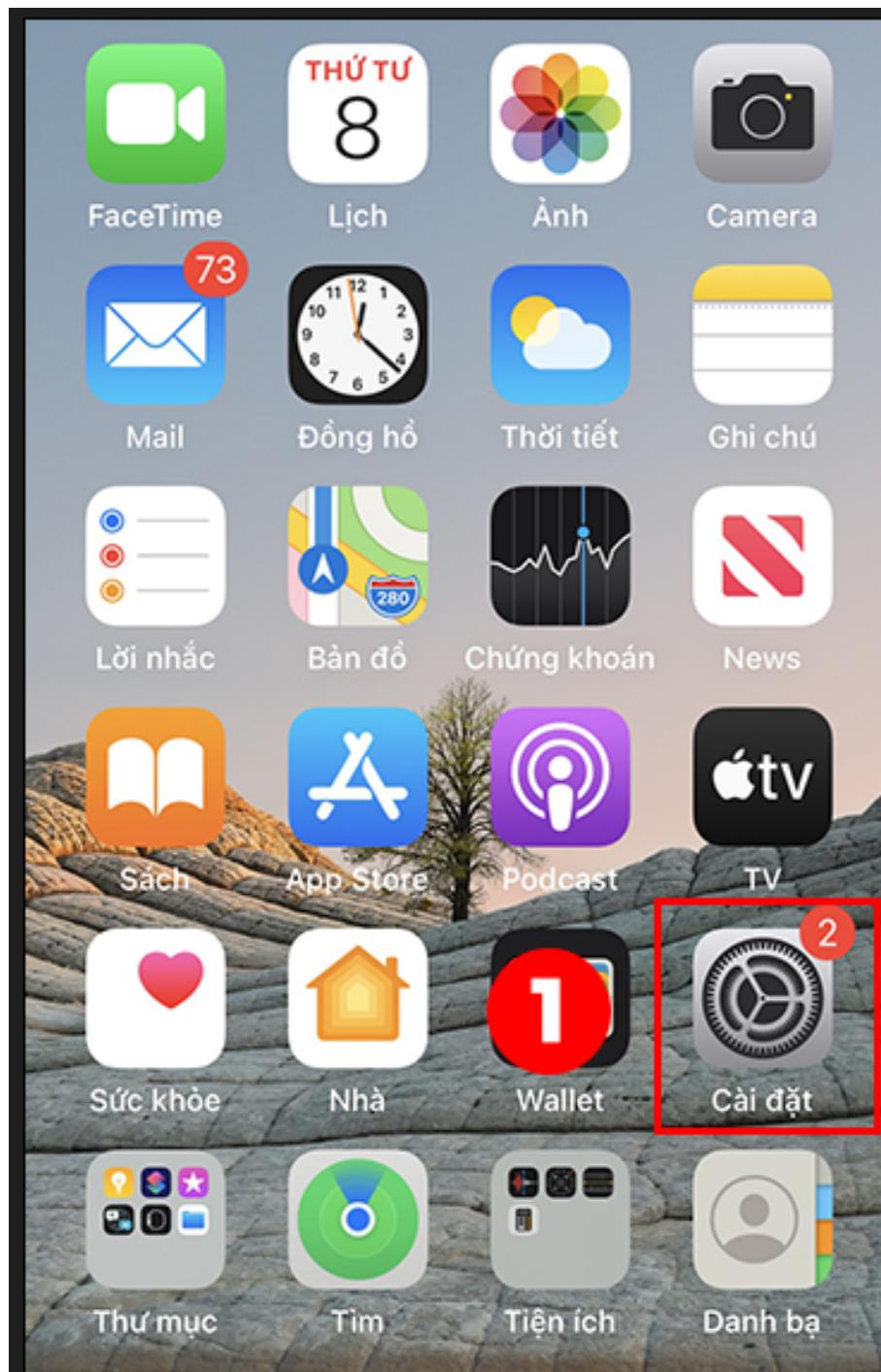
- **Thông qua Trình quản lý thiết bị di động (MDM):** có thể sử dụng Trình quản lý thiết bị di động để quản lý cấu hình thiết bị macOS và iOS của nhiều người dùng



Hình 8.3: Tính năng MDM

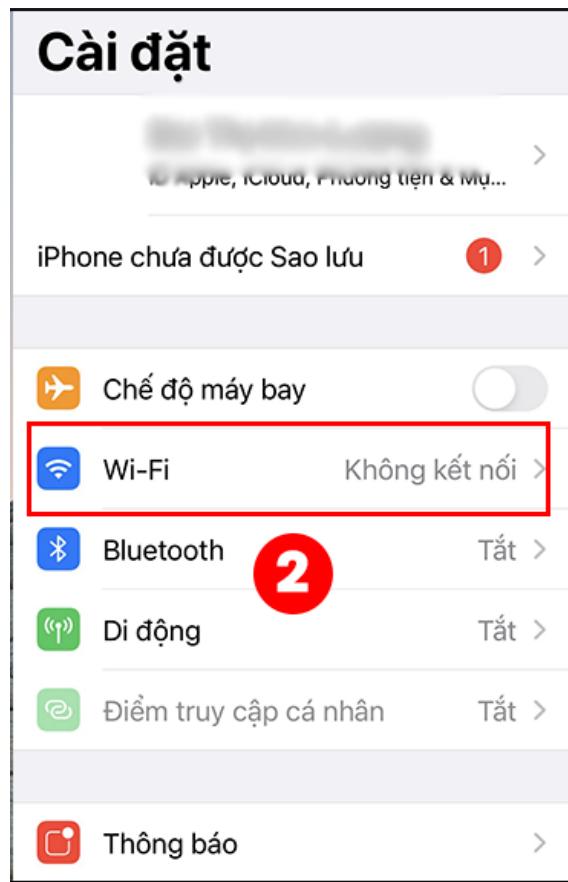
Cách để quản lý cấu hình, dịch vụ của macOS và iOS cũng nhận đơn giản và thân thiện với người dùng. Người dùng có thể thực hiện theo các bước sau để quản lý cấu hình, dịch vụ thông qua ứng dụng Cài đặt:

Bước 1: Mở ứng dụng Cài đặt.



Hình 8.4: Mở ứng dụng cài đặt trong iOS

Bước 2: Nhấn vào mục cài đặt muốn quản lý.



Hình 8.5: Cài đặt dịch vụ wifi

Bước 3: Sử dụng các tùy chọn có sẵn để quản lý cấu hình và dịch vụ.



Hình 8.6: Bật tắt wifi

8.2.2 Quản lý ứng dụng

Cả hai hệ điều hành macOS và iOS đều cung cấp các tính năng quản lý ứng dụng của mình theo nhiều cách như:

- **Cài đặt:** có thể cài đặt ứng dụng từ App Store. Ngoài ra cả macOS và iOS đều có thể cài đặt ứng dụng từ nguồn bên ngoài, tuy vậy người dùng cần cẩn trọng để không gặp vấn đề liên quan tới bảo mật.
- **Gỡ cài đặt:** có thể gỡ cài đặt ứng dụng khỏi thiết bị thông qua ứng dụng

Cài đặt.

- **Cập nhật:** có thể cập nhật ứng dụng lên phiên bản mới nhất.
- **Quản lý quyền:** có thể quản lý quyền truy cập của ứng dụng vào các tính năng của thiết bị của bạn.

CHƯƠNG 9. HỖ TRỢ HỆ THỐNG VÀ CẬP NHẬT

9.1 Hỗ trợ hệ thống

9.1.1 Dịch vụ hỗ trợ macOS và iOS

Apple cung cấp dịch vụ hỗ trợ trực tuyến, qua điện thoại và qua email cho cả macOS và iOS. Dịch vụ hỗ trợ trực tuyến bao gồm các tài nguyên như hướng dẫn, câu hỏi thường gặp và diễn đàn hỗ trợ. Dịch vụ hỗ trợ qua điện thoại và qua email cung cấp hỗ trợ trực tiếp từ nhân viên hỗ trợ của Apple.

Apple có thể hỗ trợ với nhiều ngôn ngữ, từ tính năng cơ bản tới nâng cao. Nhân viên hỗ trợ của Apple được đào tạo bài bản và có thể giúp người dùng giải quyết các vấn đề kỹ thuật.

Cộng đồng hỗ trợ trực tuyến cũng là một nguồn tài nguyên tuyệt vời để tìm kiếm thông tin và giải pháp cho các vấn đề.

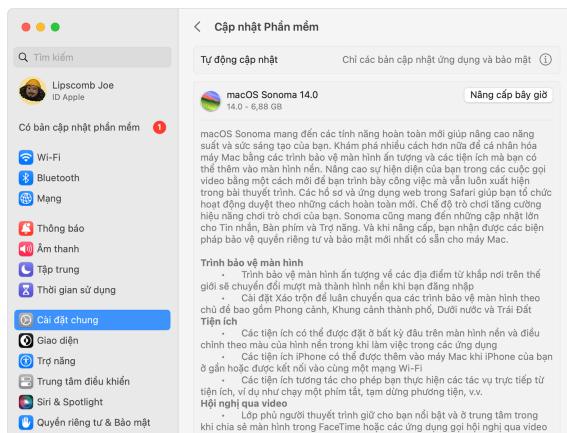
9.1.2 Cộng đồng hỗ trợ macOS và iOS

Cả macOS và iOS đều có cộng đồng hỗ trợ trực tuyến nơi người dùng có thể chia sẻ thông tin và giải pháp cho các vấn đề. Cộng đồng này bao gồm các diễn đàn và nhóm người dùng hệ điều hành macOS và iOS. Cả macOS và iOS đều có cộng đồng với số lượng thành viên đông đảo và mật độ tương tác rất lớn. Chúng ta có thể dễ dàng tìm thấy cộng đồng hỗ trợ hai hệ điều hành này trên internet như Facebook.com, Tinhte.vn, Apple.com,... Vì vậy đây cũng là một phương thức hỗ trợ rất tốt. Tuy nhiên việc nhận hỗ trợ từ cộng đồng không phải lúc nào cũng hiệu quả và chính xác. Do đó, nếu người dùng gặp vấn đề phức tạp hoặc không thể giải quyết được vấn đề của mình bằng cách sử dụng các nguồn tài nguyên này thì người dùng nên liên hệ với bộ phận hỗ trợ chính thức của Apple để được giúp đỡ.

9.2 Cập nhật hệ thống

Cập nhật hệ thống macOS và iOS là một phần quan trọng của việc duy trì hệ thống của bạn an toàn và hoạt động tốt. Các bản cập nhật thường xuyên của Apple bao gồm các bản vá lỗi để khắc phục các vấn đề đã biết, các bản cập nhật tính năng để thêm các tính năng mới và các bản cập nhật bảo mật để cải thiện khả năng bảo vệ của hệ thống của bạn.

CHƯƠNG 9. HỖ TRỢ HỆ THỐNG VÀ CẬP NHẬT



Hình 9.1: Thông báo cập nhật macOS



Hình 9.2: Thông báo cập nhật iOS

Có nhiều lợi ích khi cập nhật hệ thống macOS và iOS như:

- **Sửa lỗi và cải thiện hiệu suất:** Các bản cập nhật thường bao gồm các bản vá lỗi để khắc phục các vấn đề đã biết, điều này có thể giúp cải thiện hiệu suất và độ ổn định của hệ thống của bạn.
- **Thêm các tính năng mới:** Các bản cập nhật tính năng thường bao gồm các tính năng mới và cải tiến, điều này có thể giúp bạn tận dụng tối đa thiết bị của mình.
- **Tăng cường bảo mật:** Các bản cập nhật bảo mật thường bao gồm các bản vá để khắc phục các lỗ hổng bảo mật, điều này có thể giúp bảo vệ hệ thống của bạn khỏi các mối đe dọa bảo mật.

Tuy nhiên với một số thiết bị thế hệ cũ khi cập nhật phiên bản phần mềm mới thường gặp một số tình trạng giật, lag. Nguyên nhân có thể do Apple cung cấp một số tính năng mới khiến cho phần cứng của các máy thế hệ cũ này không còn tương thích, phần cứng của các thiết bị này không còn đủ yêu cầu đáp ứng vào thời điểm hiện tại nữa. Trường hợp trên chỉ xảy ra với một số thiết bị đã được ra mắt từ rất lâu.

macOS và iOS được Apple cập nhật thông qua một quy trình được gọi là Software Update Service. Quy trình này bao gồm các bước sau:

Bước 1: Apple phát triển các bản cập nhật hệ thống trên các thiết bị dùng hệ điều hành macOS, iOS.

Bước 2: Apple tạo các bản cập nhật hệ thống dưới dạng các tệp có thể tải xuống.

Bước 3: Apple phân phối các bản cập nhật hệ thống qua mạng phân phối của mình.

Bước 4: Thiết bị của người dùng tải xuống các bản cập nhật hệ thống từ mạng phân phối của Apple.

Bước 5: Thiết bị của người dùng cài đặt các bản cập nhật hệ thống.

Để cập nhật hệ thống trên macOS và iOS, người dùng có thể thực hiện theo các bước:

- **Trên macOS**

Bước 1: Mở ứng dụng System Preferences.

Bước 2: Nhấp vào Software Update.

Bước 3: Nhấp vào Check for updates.

Bước 4: Nếu có bản cập nhật khả dụng, hãy nhấp vào Install now.

- **Trên iOS**

Bước 1: Mở ứng dụng Settings.

Bước 2: Nhấp vào General.

Bước 3: Nhấp vào Software Update.

Bước 4: Nếu có bản cập nhật khả dụng, hãy nhấp vào Download and install.

Mặc dù Apple không đưa ra bất kỳ đảm bảo rõ ràng nào về hỗ trợ phần mềm, nhưng có thể dễ nhận thấy Apple cung cấp hỗ trợ cho macOS và iOS trong một thời gian dài khoảng chừng 5-6 năm(iPhone 8 được hỗ trợ cập nhật lên iOS 16 sau 5 năm thiết bị được ra mắt, hay như MacBook Pro 2017 được hỗ trợ cập nhật lên macOS Ventura sau 5 năm thiết bị được ra mắt). Việc này đảm bảo rằng thiết bị của họ sẽ nhận được các bản cập nhật bảo mật và tính năng trong thời gian dài giúp các thiết bị sử dụng hệ điều hành macOS và iOS có thể dùng các thiết bị mà vẫn đảm bảo về hiệu suất và an toàn bảo mật.

CHƯƠNG 10. KẾT LUẬN

Bài tập lớn nghiên cứu về hệ điều hành macOS và iOS đã đánh dấu một chặng đường quan trọng trong hành trình học tập của chúng em. Chúng em đã nghiên cứu và tìm hiểu về cấu trúc hệ điều hành macOS và iOS. Việc này đã giúp chúng em hiểu rõ về cách các thành phần hệ thống tương tác với nhau, từ kernel, framework đến các tiến trình quan trọng. Chúng em đã thấu hiểu cơ bản về cách hệ thống hoạt động và cách phát triển ứng dụng cho các nền tảng này. Chúng em đã học được cách sử dụng các công cụ phát triển, xây dựng giao diện, và triển khai ứng dụng trên các thiết bị Apple. Việc này đã giúp chúng em có trải nghiệm thực tế về việc phát triển phần mềm cho các hệ điều hành trong tương lai. Bên cạnh đó, việc tìm hiểu về an ninh và bảo mật cũng là một phần quan trọng của bài tập lớn. Chúng em đã phải nghiên cứu về các vấn đề an ninh và bảo mật liên quan đến hệ điều hành macOS và iOS, đặc biệt là trong việc phát triển ứng dụng an toàn và bảo vệ dữ liệu người dùng. Điều này đã giúp chúng em nhận thức sâu hơn về tầm quan trọng của an ninh trong phát triển ứng dụng. Tuy nhiên, bài tập lớn vẫn còn một số vấn đề còn tồn đọng. Chúng em nhận thấy rằng còn rất nhiều để học về phát triển ứng dụng trên các nền tảng này, bao gồm vấn đề về hiệu suất, tối ưu hóa ứng dụng, và tích hợp các tính năng đặc biệt. Ngoài ra, chúng em còn nhiều việc để nghiên cứu về cách bảo vệ ứng dụng và dữ liệu của người dùng trước các mối đe dọa tiềm ẩn. Để hiểu sâu hơn, chúng em cần kết hợp với các dự án thực tế hơn và tương tác với các doanh nghiệp phát triển ứng dụng trên nền tảng Apple.

Tóm lại, Bài tập lớn này đã giúp chúng em có kiến thức cơ bản về hệ điều hành macOS và iOS, nhưng còn rất nhiều vấn đề phức tạp và thú vị để khám phá trong tương lai khi chúng em tiếp tục nghiên cứu về lĩnh vực này.

TÀI LIỆU THAM KHẢO

1. Chương 1: Giới thiệu về macOS và iOS

- https://www.itorelease.com/2019/10/what-are-advantages-and-disadvantages-of-macos/#google_vignette
- <https://www.itorelease.com/2020/09/advantages-and-disadvantages-of-ios-operating-system/>

2. Chương 2: Cấu trúc hệ thống

- <https://www.tutorialspoint.com/mac-os-x-structure>
- <https://www.tutorialspoint.com/apple-ios-architecture>
- <https://developer.apple.com/documentation/quartz>
- https://developer.apple.com/library/archive/documentation/GraphicsImaging/Conceptual/OpenGL-MacProgGuide/opengl_intro/opengl_intro.html

3. Chương 3: Hàm Shell và lời gọi hệ thống

- <https://support.apple.com/vi-vn/guide/terminal/apd5265185d-f365-44cb-8b09-71a064a42125/mac>
- <https://ish.app/>
- <https://termius.com/>
- <https://apps.apple.com/us/app/prompt-2/id917437289>

4. Chương 4: Quản lý bộ nhớ

- <https://support.apple.com/vi-vn/guide/mac-help/mh11852/mac>
- <https://developer.arm.com/documentation/101811/0103/The-Memory-Management-Unit--MMU-->
- https://developer.apple.com/documentation/uikit/app_and_environment/managing_your_app_s_life_cycle

- <https://developer.apple.com/documentation/DISPATCH>
- <https://docs.swift.org/swift-book/documentation/the-swift-programming-language/automaticreferencetracking/>

5. Chương 5: Quản lý file

- <https://developer.apple.com/design/human-interface-guidelines/file-management>
- <https://www.makeuseof.com/tag/new-macos-filesystem-how-apfs-works/>
- https://hetmanrecovery.com/recovery_news/hfs-file-system-data-recovery-algorithm.htm
- <https://www.bartleby.com/subject/engineering/computer-science/concepts/macos>
- <https://support.apple.com/vi-vn/guide/mac-help/mchlp1203/mac>
- <https://support.apple.com/vi-vn/guide/mac-help/mchl211c911f/14.0/mac/14.0>
- <https://support.apple.com/vi-vn/guide/mac-help/mchld5a35146/mac>
- <https://support.apple.com/vi-vn/guide/mac-help/mh35868/mac>
- <https://support.apple.com/en-us/HT204144>

6. Chương 6: So sánh và đặc điểm riêng

- <https://www.bartleby.com/subject/engineering/computer-science/concepts/macos>
- <https://www.hexnode.com/blogs/the-ultimate-guide-to-mac-shell-scripting/#:~:text=Mac's%20default%20shell%20is%20either,the%20default%20shell%20is%20zsh.>
- <https://support.apple.com/en-vn/guide/activity-monitor/welcome/mac>
- <https://developer.apple.com/design/human-interface>

ce-guidelines/file-managemen

7. Chương 7: Ứng dụng và hiệu suất

- https://developer.apple.com/library/archive/documentation/Performance/Conceptual/power_efficiency_guidelines_osx/AppNap.html
- <https://developer.apple.com/tutorials/swiftui/creating-a-macos-app>
- <https://browser.geekbench.com/mac-benchmarks>
- <https://browser.geekbench.com/ios-benchmarks>

8. Chương 8: Bảo mật và quản lý thiết bị

- <https://support.apple.com/vi-vn/guide/mac-help/mh11785/mac>
- <https://support.apple.com/vi-vn/guide/keychain-access/kyca1083/mac>
- <https://support.apple.com/vi-vn/guide/apple-configuration/apsd8b1c65bd/mac>
- <https://support.apple.com/vi-vn/guide/remote-desktop/apd8b1c65bd/mac>
- <https://support.apple.com/en-ph/guide/deployment/depcoaadd3fe/web>

9. Chương 9: Hỗ trợ hệ thống và cập nhật

- <https://support.apple.com/vi-vn/HT204204>
- <https://support.apple.com/vi-vn/HT201541>
- <https://support.apple.com/vi-vn/HT213264>
- <https://support.apple.com/en-us/103267>

ĐÓNG GÓP CỦA CÁC THÀNH VIÊN

1. Phạm Đức Chính - B21DCCN181 (Trưởng nhóm - 18%)

- Lên kế hoạch, xây dựng khung nội dung bài tập lớn và phân chia công việc cho các thành viên.
- Tìm hiểu nội dung và chỉnh sửa LaTeX chương 2 "Cấu trúc hệ thống"
- Đánh giá lại toàn bộ báo cáo bản cũ để sửa đổi.

2. Nguyễn Văn Hòa - B21DCCN380 (17%)

- Tìm hiểu nội dung chương 1 "Giới thiệu về macOS và iOS".
- Dựng khung LaTeX và gộp báo cáo trước khi các thành viên tự chỉnh sửa.

3. Nguyễn Đức An - B21DCCN001 (16%)

- Tìm hiểu nội dung và chỉnh sửa LaTeX chương 3 và 4 "Hàm Shell và lời gọi hệ thống" và "Quản lí bộ nhớ".

4. Phạm Hữu Đoàn - B21DCCN229 (17%)

- Tìm hiểu nội dung và chỉnh sửa LaTeX chương 5 và 6 "Quản lí file" và "So sánh và đặc điểm riêng".
- Rà soát báo cáo bản mới.

5. Bùi Thanh Tùng - B21DCAT214 (16%)

- Tìm hiểu nội dung và chỉnh sửa LaTeX chương 7 "Ứng dụng và hiệu suất".

6. Đỗ Đắc Huy - B21DCCN431 (16%)

- Tìm hiểu nội dung và chỉnh sửa LaTeX chương 8 và 9 "Bảo mật và quản lí thiết bị" và "Hỗ trợ hệ thống và cập nhật".