1. *Ba mục đích chính của một hệ điều hành là gì?*
   - Thực thi chương trình của người dùng và giúp giải quyết vấn đề của người dùng dễ dàng hơn
   - Làm cho máy tính dễ sử dụng
   - Sử dụng phần cứng một cách hiệu quả

2. *Hệ thống có khả năng tương tác hiệu quả với người dùng là hệ thống nào? (Đơn chương, đa chương, chia thẻ thời gian)*
   - Hệ thống chia sẻ thời gian

3. *Hệ thống được sử dụng trong các thiết bị chuyên dụng điều khiển máy móc là hệ thống nào (Đa chương, đa xử lý, thời gian thực, phân tán)*
   - Hệ thống thời gian thực

4. *Hệ điều hành nào hỗ trợ tính toán hiệu năng cao (chia sẻ thời gian, xử lý song song, xử lý theo lô, thời gian thực)*
   - Hệ điều hành xử lý song song

5. *Hệ điều hành phát triển được chi phối bởi yếu tố gì?*
   - Yêu cầu phần cứng
   - Nhu cầu người dùng
   - Công nghệ và xu hướng mới
   - Bảo mật
   - Hiệu xuất và khả năng mở rộng
   - Tương thích phần mềm
   - Quản lý tài nguyên
   - Thị trường và cạnh tranh

6. *Trong một môi trường đa chương và chia sẻ thời gian, nhiều người dùng chia sẻ hệ thống đồng thời, điều này có thể mang đến những vấn đề gì? Chúng ta có thể đảm bảo mức độ bảo mật giống như trên các máy riêng lẻ không?*
   - Vấn đề có thể gặp phải:
     - Xung đột tài nguyên: Khi nhiều người dùng chia sẻ cùng một tài nguyên (như CPU, bộ nhớ, thiết bị ngoại vi), có thể xảy ra xung đột khi các tài nguyên này không đủ đáp ứng nhu cầu của tất cả người dùng. Điều này có thể dẫn đến việc giảm hiệu suất hệ thống hoặc thậm chí là treo máy.
     - Bảo mật quyền riêng tư: Dữ liệu và tài nguyên của mỗi người có nguy cơ bị truy cập trái phép bởi những người dùng khác. Nếu không có các biện pháp bảo mật thích hợp, thông tin cá nhân hoặc nhạy cảm của người dùng có thể bị rò rỉ.
     - Vấn đề đồng bộ hóa: Trong một môi trường đa chương, việc đồng bộ hóa giữa các tiến trình và tài nguyên là rất quan trọng. Nếu không được quản lý đúng cách, các vấn đề về race conditions hoặc deadlocks có thể xảy ra, gây gián đoạn hoặc làm chậm hệ thống.
     - Hiệu suất hệ thống: Với nhiều người dùng đồng thời sử dụng hệ thống, việc quản lý và điều phối các tiến trình và tác vụ có thể trở nên phức tạp, dẫn đến hiệu suất tổng thể giảm xuống. Các quá trình chuyển đổi giữa các tiến trình có thể gây ra overhead và giảm tốc độ xử lý.
   - Về bảo mật:

Việc đảm bảo mức độ bảo mật giống như trên các máy riêng lẻ trong môi trường chia sẻ thời gian là một thách thức. Tuy nhiên, có thể áp dụng các biện pháp để tăng cường bảo mật:

- Kiểm soát truy cập: Hệ điều hành cần có các cơ chế kiểm soát truy cập mạnh mẽ để đảm bảo rằng chỉ những người dùng được phép mới có thể truy cập và các tài nguyên cụ thể. Các quyền truy cập cần được cấu hình chính xác để ngăn chặn người dùng trái phép truy cập vào dữ liệu hoặc tài nguyên của người dùng khác.
- Cách ly tiến trình: Các tiến trình của từng người dùng cần được cách ly một cách hiệu quả để đảm bảo rằng một tiến trình không thể truy cập hoặc can thiệp vào tiến trình của người dùng khác.
- Mã hóa dữ liệu: Dữ liệu nhạy cảm của người dùng nên được mã hóa, cả khi lưu trữ và khi truyền tải, để giảm nguy cơ bị đánh cắp hoặc rò rỉ thông tin.
- Theo dõi và ghi nhật ký: Việc theo dõi và ghi lại các hoạt động của người dùng có thể giúp phát hiện các hành vi bất thường hoặc các cố gắng truy cập trái phép, từ đó có thể phản ứng kịp thời để bảo vệ hệ thống.
- Cập nhật bảo mật thường xuyên: Hệ điều hành và phần mềm cần được cập nhật thường xuyên để bảo vệ chống lại các lỗ hổng bảo mật mới xuất hiện.

7. *Hệ điều hành đã trải qua các giai đoạn phát triển nào? Thách thức và vấn đề của từng giai đoạn là gì?*

- Thời kỳ ban đầu (1940s-1950s): Hệ điều hành thủ công
    - Mô tả: Thời kỳ này, các máy tính không có hệ điều hành theo đúng nghĩa hiện đại. Người dùng phải điều khiển máy tính trực tiếp bằng cách nhập lệnh và chương trình bằng tay, thường thông qua bảng điều khiển hoặc thẻ đục lỗ.
    - Thách thức:
        - Tính tiện dụng: Các máy tính không dễ sử dụng và yêu cầu kiến thức kỹ thuật cao.
        - Hiệu quả thấp: Không có cơ chế quản lý tài nguyên tự động, dẫn đến việc sử dụng tài nguyên không hiệu quả.
        - Không có đa nhiệm: Chỉ có thể thực hiện một chương trình tại một thời điểm.
- Hệ điều hành theo lô (1950s-1960s): Batch Processing Systems
    - Mô tả: Hệ điều hành theo lô cho phép nhiều công việc (job) được tập hợp lại và xử lý theo lô, một cách tự động mà không cần sự can thiệp của con người giữa các công việc.
    - Thách thức:
        - Thời gian chờ đợi: Người dùng phải chờ đợi lâu để công việc của họ được thực hiện.
        - Thiếu tương tác người dùng: Không có khả năng tương tác với hệ thống trong quá trình công việc đang được thực hiện.
        - Tài nguyên không được sử dụng tối ưu: Máy tính có thể nhàn rỗi trong khi chờ đợi các thiết bị ngoại vi hoàn thành nhiệm vụ.
- Hệ điều hành chia sẻ thời gian (1960s-1970s): Time-Sharing Systems
    - Mô tả: Hệ điều hành chia sẻ thời gian cho phép nhiều người dùng truy cập và sử dụng máy tính cùng một lúc, mỗi người dùng có ảo giác rằng họ đang sử dụng máy tính một cách độc lập.
    - Thách thức:

        - Quản lý tài nguyên: Yêu cầu phải quản lý tốt CPU, bộ nhớ, và các thiết bị ngoại vi để đảm bảo mỗi người dùng có một trải nghiệm mượt mà.
        - Bảo mật: Nhiều người dùng cùng chia sẻ một hệ thống, dẫn đến nguy cơ bảo mật và quyền riêng tư.
        - Hiệu suất: Phải đảm bảo rằng hiệu suất không giảm quá nhiều khi số lượng người dùng tăng lên.
- Hệ điều hành đa chương và đa nhiệm (1970s-1980s): Multitasking and Multiprogrmming
    - Mô tả: Hệ điều hành đa chương cho phép nhiều chương trình chạy đồng thời, trong khi đa nhiệm cho phép một CPU thực hiện nhiều tác vụ bằng cách chuyển đổi nhanh chóng giữa các tác vụ.
    - Thách thức:
        - Đồng bộ hóa tiến trình: Xử lý các vấn đề về đồng bộ hóa giữa các tiến trình để tránh xung đột và deadlock.
        - Quản lý bộ nhớ: Đảm bảo rằng mỗi chương trình có đủ bộ nhớ và ngăn chặc tình trạng tràn bộ nhớ.
        - Phức tạp hóa hệ điều hành: Hệ điều hành trở nên phức tạp hơn, đòi hỏi các thuật toán quản lý tài nguyên phức tạp hơn.
- Hệ điều hành phân tán (1980s-nay): Distributed Operating Systems
    - Mô tả: Hệ điều hành phân tán quản lý nhiều máy tính kết nối với nhau trong một mạng, và tài nguyên được chia sẻ giữa các máy tính này.
    - Thách thức:
        - Giao tiếp và phối hợp: Phải đảm bảo rằng các hệ thống trong mạng có thể giao tiếp và phối hợp hiệu quả.
        - Bảo mật và tin cậy: Với nhiều hệ thống kết nối, vấn đề bảo mật và độ tin cậy trở nên phức tạp hơn.
        - Quản lý lỗi: Khả năng xử lý lỗi trong hệ thống phân tán là một thách thức lớn do tính phức tạp của nó.
- Hệ điều hành thời gian thực (RTOS) (1980s-nay): Real-Time Operating Systems
    - Mô tả: Hệ điều hành thời gian thực được sử dụng trong các hệ thống yêu cầu phản hồi ngay lập tức, chẳng hạn như trong các thiết bị nhúng, điều khiển tự động hóa, hoặc thiết bị y tế.
    - Thách thức:
        - Độ trễ thấp: Phải đảm bảo rằng hệ thống có thể đáp ứng trong khoảng thời gian rất ngắn, thường là ngay lập tức.
        - Độ tin cậy cao: Hệ thống phải hoạt động một cách đáng tin cậy vì bất kỳ sự chậm trễ nào có thể gây hậu quả nghiêm trọng.
        - Quản lý tài nguyên thiên nhiên: Tài nguyên phải được quản lý rất chặt chẽ để đảm bảo không có sự trì hoãn trong các tác vụ quan trọng.
- Hệ điều hành hiện đại (1990s-nay): Modern Operating Systems
    - Mô tả: Hệ điều hành hiện đại bao gồm các hệ thống hỗ trợ đa nhiệm, đa người dung, bảo mật nâng cao, quản lý tài nguyên tốt, và hỗ trợ các công nghệ mới như điện toán đám mây, ảo hóa, và trí tuệ nhân tạo.
    - Thách thức:
        - Bảo mật: Phải đối mặt với các mối đe dọa an ninh mạng ngày càng phức tạp.
        - Khả năng mở rộng: Phải hỗ trợ khả năng mở rộng để đáp ứng nhu cầu ngày càng tăng của các ứng dụng và dịch vụ.

    - Tương thích ngược: Đảm bảo rằng hệ điều hành mới vẫn tương thích với phần mềm và phần cứng cũ.
    - Hiệu suất cao: Cung cấp hiệu suất cao trong khi hỗ trợ nhiều tính năng phức tạp.

8. *Hệ điều hành phải làm gì để khắc phục vấn đề về phần cứng, màn hình nhỏ… trên thiết bị di động*
    - Tối ưu hóa giao diện người dùng (UI/UX):
        - Thiết kế giao diện đơn giản và trực quan: Giao diện người dùng cần được thiết kế đơn giản, với các biểu tượng lớn, dễ nhận biết và dễ sử dụng. Các chức năng chính nên được đặt ở vị trí dễ tiếp cận trên màn hình.
        - Hỗ trợ cảm ứng: Giao diện nên tối ưu hóa cho cảm ứng, với cử chỉ như vuốt, chạm và kéo thả, thay vì sử dụng các yếu tố điều khiển nhỏ như trên máy tính để bàn.
        - Điều chỉnh theo kích thước màn hình: Hệ điều hành cần hỗ trợ khả năng hiển thị linh hoạt, tự động điều chỉnh giao diện và nội dung theo kích thước và độ phân giải của màn hình.
    - Quản lý tài nguyên hiệu quả:
        - Tối ưu hóa sử dụng bộ nhớ: Hệ điều hành cần quản lý bộ nhớ hiệu quả để các ứng dụng có thể chạy mượt mà trên các thiết bị có dung lượng bộ nhớ hạn chế.
        - Tiết kiệm năng lượng: Quản lý việc sử dụng CPU, GPU và các tài nguyên khác một cách tối ưu để kéo dài thời lượng pin, Hệ điều hành nên giảm thiểu các tác vụ nền không cần thiết và tối ưu hóa việc sử dụng pin trong khi các ứng dụng đang chạy.
        - Quản lý kết nối mạng: Tối ưu hóa việc sử dụng băng thông mạng và hỗ trợ kết nối mạng không ổn định, đặc biệt là khi di chuyển hoặc trong các khu vực có tín hiệu yếu.
    - Tương thích phần cứng:
        - Hỗ trợ đa dạng phần cứng: Hệ điều hành cần tương thích với nhiều loại phần cứng khác nhau, từ các thiết bị có cấu hình cao đến các thiết bị giá rẻ với cấu hình thấp.
        - Tối ưu hóa cho vi xử lý di động: Hệ điều hành phải tận dụng tối đa khả năng của vi xử lý di động, bao gồm các tính năng tiết kiệm năng lượng và tăng hiệu suất.
    - Quản lý ứng dụng:
        - ứng dụng nhẹ và tối ưu: Khuyến khích các nhà phát triển tạo ra các ứng dụng nhẹ, tối ưu hóa cho các thiết bị có tài nguyên hạn chế. Điều này có thể bao gồm việc sử dụng các phiên bản rút gọn của ứng dụng hoặc các ứng dụng dựa trên web.
        - Quarnl ý đa nhiệm hiệu quả: Hệ điều hành cần cung cập khả năng quản lý đa nhiệm thông minh, như tạm dừng hoặc đóng các ứng dụng không cần thiết khi người dùng chuyển đổi giữa các ứng dụng, để tiết kiệm bộ nhớ và tài nguyên.
    - Bảo mật và quyền riêng tư:
        - Bảo mật dữ liệu người dùng: Tích hợp các tính năng bảo mật mạnh mẽ để bảo vệ dữ liệu cá nhân và đảm bảo quyền riêng tư của người dùng trên thiết bị di động, ngay cả khi thiết bị bị mất hoặc bị đánh cắp.
        - Quản lý quyền truy cập: Cung cấp cơ chế quản lý quyền truy cập mạnh mẽ, cho phép người dùng kiểm soát các quyền mà ứng dụng yêu cầu, từ đó giảm thiểu nguy cơ lạm dụng tài nguyên hệ thống hoặc thông tin cá nhân.

- Cập nhật và hỗ trợ liên tục:
    - Cập nhật thường xuyên: Hệ điều hành cần cung cấp các bản cập nhật thường xuyên để cải thiện hiệu suất, bảo mật và tương thích với các công nghệ mới nhất, mà không gây ảnh hưởng đáng kể đến hiệu suất của thiết bị.
    - Hỗ trợ lâu dài cho thiết bị cũ: Đảm bảo rằng các thiết bị cũ vẫn nhận được các bản cập nhật phần mềm và bảo mật, ngay cả khi không thể tận dụng hết các tính năng mới nhất.
- Hỗ trợ đám mây:
    - Đồng bộ hóa và lưu trữ đám mây: Tích hợp các dịch vụ đám mây để hỗ trợ đồng bộ hóa dữ liệu, giảm thiểu nhu cầu lưu trữ trên thiết bị và cho phép người dùng truy cập dữ liệu từ bất kỳ thiết bị nào.
    - Ứng dụng web và PWA (Progressive Web Apps): Khuyến khích sử dụng các ứng dụng web và PWA để giảm thiểu dung lượng cài đặt trên thiết bị và cung cấp trải nghiệm người dùng nhất quán trên nhiều nền tảng.
- Hỗ trợ điều khiển bằng giọng nói:
    - Trợ lý ảo và điều khiển giọng nói: Cung cấp khả năng điều khiển thiết bị bằng giọng nói, giúp người dùng thực hiện các tác vụ mà không cần tương tác trực tiếp với màn hình, từ đó khắc phục hạn chế của màn hình nhỏ.

9. *Hệ điều hành trên máy tính MainFrame có các đặc tính gì?*
    - Khả năng xử lý song song và đa nhiệm (Parallel Processing and Multitasking): được thiết kế đề xử lý hàng nghìn đến hàng triệu giao dịch mỗi giây, tận dụng được nhiều CPU
    - Khả năng mở rộng (Scalability): khả năng mở rộng cũng áp dụng cho phần cứng, cho phép thêm tài nguyên như CPU, bộ nhớ, và ổ đĩa mà không cần dừng hệ thống.
    - Độ tin cậy và khả năng phục hồi cao (High Reliability and Fault Tolerance): hệ thống có thể tiếp tục hoạt động ngay cả khi một phần của nó gặp lỗi. Hệ điều hành hỗ trợ các tính năng như dự phòng (redundancy) và chuyển đổi dự phòng (failover) để duy trì hoạt động liên tục. Ngoài ra, hệ điều hành có các cơ chế tự động phát hiện và sửa chữa các lỗi mà không cần sự can thiệp của người dùng.
    - Bảo mật (Security): có các tính năng bảo mật rất mạnh mẽ, bao gồm mã hóa, quản lý truy cập, và kiểm tra bảo mật để bảo vệ dữ liệu và ứng dụng. Hệ điều hành cung cấp các cơ chế phức tạp để quản lý quyền truy cập, cho phép thiết lập quyền cho từng người dùng, ứng dụng, và tiến trình một cách chi tiết.
    - Quản lý tài nguyên hiệu quả (Efficient Resource Management): tối ưu hóa việc sử dụng CPU, bộ nhớ, và các thiết bị ngoại vi, đảm bảo tài nguyên được phân bổ và sử dụng hiệu quả. Hệ điều hành có khả năng quản lý và cân bằng tải công việc (workload balancing) để đảm bảo hiệu suất ổn định ngay cả khi hệ thống chịu tải nặng.
    - Hỗ trợ giao dịch và dữ liệu lớn (Transaction Processing and Big Data): hỗ trợ các hệ thống quản lý cơ sở dữ liệu lớn, đảm bảo truy cập nhanh chóng và quản lý hiệu quả các kho dữ liệu khổng lồ.
    - Khả năng hỗ trợ đa người dùng (Multi-user Support)
    - Khả năng chạy các ứng dụng kế thừa (Legacy Application Support): Mainframe vẫn thường được sử dụng để chạy các ứng dụng quan trọng đã tồn tại nhiều năm, do đó, hệ điều hành phải tương thích với các ứng dụng này.
    - Khả năng ảo hóa (Virtualization): cho phép chạy nhiều hệ điều hành và môi trường ảo trên cùng một máy tính vật lý.

- Quản lý mạng và giao tiếp (Network and Communication Management): có khả năng quản lý mạng phức tap và tích hợp chặt chẽ với các hệ thống mạng khác, đảm bảo kết nối liên tục và bảo mật cao giữa các thành phần của hệ thống.

10. Hệ điều hành thời gian thực có những loại gì? Đặc tính của từng loại?

- Hệ điều hành thời gian thực cứng (Hard Real-Time Operating System)\
    - Đặc điểm chính:
        - Hạn chót cứng (Hard Deadlines): Trong RTOS cứng, việc đáp ứng các hạn chót là bắt buộc và không thể trì hoãn. Mọi sự chậm trễ, dù là rất nhỏ, đều có thể dẫn đến thất bại của hệ thống, và hậu quả có thể rất nghiêm trọng.
        - Độ tin cậy cao: Được thiết kế để hoạt động với độ tin cậy rất cao, đảm bảo rằng hệ thống luôn đáp ứng được các yêu cầu thời gian thực.
        - Ứng dụng: RTOS cứng thường được sử dụng trong các hệ thống nhúng yêu cầu độ chính xác và tin cậy tuyệt đối như hệ thống điều khiển máy bay, điều khiển tên lửa, hoặc các thiết bị y tế quan trọng như máy điều hòa nhịp tim.
    - Ví dụ: VxWorks (sử dụng trong hệ thống không gian, thiết bị quân sự), INTEGRITY (được sử dụng trong các hệ thống nhúng với yêu cầu an toàn cao)
- Hệ điều hành thời gian thực mềm (Soft Real-Time Operating System)
    - Đặc điểm chính:
        - Hạn chót mềm (Soft Deadliné): Trong RTOS mềm, việc không đáp ứng được hạn chót không dẫn đến thất bại của hệ thống, nhưng có thể làm giảm chất lượng dịch vụ hoặc hiệu suất của ứng dụng. Hệ thống có thể chịu đựng một số chậm trễ nhất định mà không ảnh hưởng nghiêm trọng.
        - Tối ưu hóa hiệu suất: Tập trung vào việc tối ưu hóa hiệu suất để đáp ứng hầu hết các yêu cầu thời gian thực, nhưng có thể linh hoạt trong việc xử lý các tác vụ không quá quan trọng.
        - Ứng dụng: RTOS mềm thường được sử dụng trong các ứng dụng đa phương tiện, hệ thống viễn thông, hoặc các hệ thống mà thời gian thực là quan trọng nhưng không phải yếu tố sống còn.
    - Ví dụ: Linux với bản vá thời gian thực (Linux with Real-Time patches)- Được sử dụng trong các hệ thống điều khiển công nghiệp hoặc hệ thống viễn thông, Windows CE-được sử dụng trong các thiết bị nhúng, điện thoại di động, hoặc các thiết bị điện tử tiêu dùng.
- Hệ điều hành thời gian thực chắc chắn (Firm Real-Time Operating System)
    - Đặc điểm chính:
        - Hạn chót chắc chắn (Firm Deadlines): Hạn chót trong RTOS chắc chắn là quan trọng, và không đáp ứng được hạn chót có thể gây thiệt hại, nhưng không phải là thất bại của hệ thống. Hiệu suất giảm nếu hạn chót không được đáp ứng, nhưng hệ thống vẫn có thể tiếp tục hoạt động.
        - Cân bằng giữa cứng và mềm: RTOS chắc chắn cân bằng giữa yêu cầu của RTOS cứng và RTOS mềm, cho phép một số linh hoạt nhưng vẫn giữ độ tin cậy cao.
        - Ứng dụng: Thường được sử dụng trong các hệ thống kiểm soat giao thông, quản lý tài nguyên năng lượng, hoặc các hệ thống tự động hóa nhà máy.

    - Ví dụ: QNX-sử dụng trong hệ thống viễn thông và điều khiern công nghiệp, FreeRTOS-phổ biến tỏng các hệ thống nhúng yêu cầu thời gian thực chắc chắn nhưng không quá cứng nhắc.
- Hệ điều hành thời gian thực phân tán (Distributed Real-Time Operating System)
    - Đặc điểm chính:
        - Phân tán nhiều nút: Hệ thống RTOS này được phân tán trên nhiều nút mạng khác nhau, nơi các tác vụ thời gian thực được xử lý đồng thời ở nhiều vị trí khác nhau.
        - Đồng bộ hóa thời gian: Đòi hỏi sự đồng bộ hóa thời gian chặt chẽ giữa các nút để đảm bảo rằng các tác vụ thời gian thực được hoàn thành đúng hạn.
        - Ứng dung: Sử dụng trong hệ thống điều khiển giao thông, hệ thống giám sát và điều khiển quy mô lớn, hoặc các ứng dụng đám mây thời gian thực.
    - Ví dụ: ROS 2 (Robot Operating System 2)-sử dụng trong các hệ thống robot phân tán, LynxOS-được sử dụng trong các hệ thống điều khiển công nghiệp và các ứng dụng viễn thông phân tán.
- Hệ điều hành thời gian thực nhúng (Embedded Real-Time Operating System)
    - Đặc điểm chính:
        - Nhúng vào thiết bị: RTOS nhúng được thiết kế để hoạt động trong các thiết bị với tài nguyên hạn chế, chẳng hạn như bộ nhớ và năng lượng.
        - Tiết kiệm năng lượng và tối ưu hóa tài nguyên: Được tối ưu hóa để sử dụng tài nguyên một cách hiệu quả, với khả năng tiết kiệm năng lượng cao.
        - Ứng dụng: Được sử dụng trong các thiết bị nhúng như cảm biến, thiết bị IoT, thiết bị điện tử tiêu dùng, và các hệ thống nhúng nhỏ gọn khác.
    - Ví dụ: FreeRTOS-phổ biến trong các thiết bị nhúng nhỏ gọn, ThreadX-sử dụng trong các thiết bị IoT và hệ thống nhúng với tài nguyên hạn chế.

11. *Hệ điều hành có thể phân loại theo những tiêu chí nào? Mỗi cách phân loại có những loại hệ điều hành nào?*
    - Theo giao diện người dùng:
        - Hệ điều hành dòng lệnh (CLI – Command line interface): Người dùng tương tác với hệ điều hành thông qua các lệnh bằng văn bản. VD: MS-DOC, Unix, FreeDOS.
        - Hệ điều hành đồ họa (GUI-Graphical user interface): Người dùng tương tác với hệ điều hành thông qua giao diện đồ họa. VD: Windows, macOS, Ubuntu với giao diện đồ họa GNOME hoặc KDE.
    - Theo mục đích sử dụng:
        - Hệ điều hành đa nhiệm (Multitasking OS): Có khả năng chạy nhiều tác vụ cùng lúc. VD: Windows, macOS, Linux.
        - Hệ điều hành thời gian thực (Real-Time OS): Được thiết kế để xử lý các tác vụ với thời gian phản hồi nhanh, dùng trong các hệ thống điều khiển hoặc nhúng. VD: QNX, VxWork, FreeRTOS.
        - Hệ điều hành nhúng (Embedded OS): Được thiết kế để chạy trên các thiết bị nhúng, với yêu cầu tài nguyên thấp. VD: Android, FreeRTOS.

        - Hệ điều hành mạng (Network OS): Được thiết kế để hỗ trợ quản lý và điều hành mạng. VD: Windows Server, Novell NetWare, Linux Server.
    - Theo số lượng người dùng:
        - Hệ điều hành đơn người dùng (Single-User OS): Chỉ hỗ trợ một người dùng tại một thời điểm. VD: MS-DOS.
        - Hệ điều hành đa người dùng (Multi-User OS): Hỗ trợ nhiều người dùng truy cập và làm việc cùng lúc. VD: Unix, Linux, Windows Server.
    - Theo kiến trúc hệ thống:
        - Hệ điều hành 32-bit: Hỗ trợ bộ xử lý 32-bit và chỉ có thể truy cập tối đa 4GB bộ nhớ RAM. VD: Windows XP 32-bit, Linux 32-bit.
        - Hệ điều hành 64-bit: Hỗ trợ bộ xử lý 64-bit và có thể truy cập lượng RAM lớn hơn 4GB. VD: Windows 10 64-bit, macOS, hầu hết các bản phân phối Linux hiện đại.
    - Theo hệ thống quản lý tiến trình:
        - Hệ điều hành đơn nhiệm (Single-tasking OS): Chỉ thực hiện một tác vụ tại một thời điểm. VD: MS-DOS.
        - Hệ điều hành đa nhiệm (Multi-tasking OS): Có khả năng xử lý nhiều tác vụ cùng lúc. VD: Linux, Windows, macOS.
    - Theo khả năng phân tán và quản lý tài nguyên:
        - Hệ điều hành phân tán (Distributed OS): Điều khiển một mạng máy tính và xử lý như một hệ điều hành duy nhất, phân phối tài nguyên và tác vụ trên nhiều máy. VD: Plan 9, Amoeba.
        - Hệ điều hành cục bộ (Local OS): Chỉ quản lý tài nguyên trên một máy tính duy nhất. VD: Windows, macOS.
    - Theo quyền truy cập và phát triển:
        - Hệ điều hành mã nguồn mở (Open-source OS): Người dùng có thể truy cập và chỉnh sửa mã nguồn của hệ điều hành. VD: Linux, FreeBSD, Android (phần lõi).
        - Hệ điều hành mã nguồn đóng (Closed-source OS): Mã nguồn không được công khai và chỉ có nhà phát triển chính thức được phép chỉnh sửa. VD: Windows, macOS, iOS.

12. Hệ thống đa chương là gì (multiprograming system)

- Hệ thống đa chương (Multiprogramming System) là một phương pháp quản lý tài nguyên trong các hệ điều hành cho phép nhiều chương trình được thực thi cùng lúc trên cùng một bộ xử lý. Trong hệ thống này, khi một chương trình đang chờ đợi (ví dụ, chờ nhập xuất dữ liệu), CPU có thể chuyển sang thực thi chương trình khác. Điều này giúp tăng hiệu quả sử dụng CPU và giảm thời gian chờ đợi của các chương trình.

13. Hệ thống chia sẻ thời gian là gì? (Timesharing System)

- Hệ thống chia sẻ thời gian thực là một phương pháp quản lý tài nguyên của hệ điều hành, cho phép nhiều người dùng hoặc chương trình cùng truy cập và sử dụng CPU trong khoảng thời gian rất ngắn. Với hệ thống này, CPU sẽ luân phiên chuyển đổi giữa các tác vụ khác nhau trong những khoảng thời gian ngắn (thường gọi là "quantum"), tạo cảm giác cho người dùng rằng các tác vụ đang được thực thi đồng thời, mặc dù thực tế chỉ có một CPU xử lý.

14. Lập lịch ưu tiên là gì?

- Lập lịch ưu tiên là một thuật toán lập lịch trong hệ điều hành, trong đó các tác vụ (hay tiến trình) được sắp xếp và thực thi dựa trên độ ưu tiên. Mỗi tác vụ sẽ được gán một mức độ ưu tiên và hệ điều hành sẽ quyết định thứ tự thực thi dựa trên mức

độ này. Các tác vụ có độ ưu tiên cao hơn sẽ được thực thi trước các tác vụ có độ ưu tiên thấp hơn.

15. *Chế độ giám sát và chế độ người sử dụng giúp bảo vệ hệ thống như thế nào?*
    - Chế độ giám sát (Supervisor mode): còn được gọi là chế độ hạt nhân hoặc chế độ đặc quyền, chế độ này cho phép hệ điều hành có quyền truy cập đầy đủ vào tất cả các tài nguyên hệ thống, bao gồm bộ nhớ, CPU, và các thiết bị phần cứng. Chế độ này chỉ dành cho các phần của hệ điều hành và các quy trình quan trọng cần trực tiếp quản lý tài nguyên hệ thống. (Quyền hạn cao nhất, kiểm soát các tiến trình người dùng, bảo vệ tài nguyên)
    - Chế độ người sử dụng: đây là chế độ dành cho các chương trình người dùng và các ứng dụng không cần truy cập trực tiếp vào tài nguyên hệ thống hoặc các phần cứng nhạy cảm. (Quyền truy cập hạn chế, cách ly tài nguyên, ngăn chặn lỗi và sự cố).
16. *Sự khác biệt giữa bẫy và ngắt là gì? Chức năng của chúng là gì?*

Yếu tố	Bẫy(Trap)	Ngắt(Interrupt)
Nguồn gốc	Do phần mềm hoặc chương trình người dùng sinh ra	Do phần cứng hoặc thiết bị ngoại vi sinh ra
Nguyên nhân	Lỗi phần mềm (chia cho 0,…) hoặc lệnh gọi hệ thống	Sự kiện phần cứng(như bàn phím nhấn, kết thúc một phép tính)
Thời điểm	Xảy ra tại thời điểm xác định trong quá trình thực thi mã	Xảy ra ngẫu nhiên, không phụ thuộc vào mã của chương trình
Mục đích	Thực hiện các thao tác đặc biệt như xử lý lỗi hoặc gọi hệ thống	Đáp ứng các sự kiện phần cứng yêu cầu hệ điều hành can thiệp

17. *Chế độ đặc quyền là gì? Cho một ví dụ của lệnh đặc quyền.*
    - Chế độ đặc quyền là một chế độ hoạt động của CPU, trong đó hệ điều hành có toàn quyền truy cập vào tất cả tài nguyên của hệ thống, bao gồm bộ nhớ, thiết bị phần cứng, và các vùng tài nguyên nhạy cảm khác. Trong chế độ này, các lệnh có khả năng thay đổi cấu trúc hệ thống, quản lý tài nguyên hoặc điều khiển thiết bị đều có thể được thực thi mà không bị giới hạn.
    - Ví dụ: trong hệ điều hành Linux, các lệnh để điều khiển việc quản lý bộ nhớ hoặc thiết bị đều thuộc loại lệnh đặc quyền. Cụ thể, lệnh cli (Clear interrupt flag) trong hệ thống x86 là một lệnh đặc quyền. Lệnh này tắt các ngắt phần cứng tạm thời, cho phép CPU thực hiện các tác vụ quan trọng mà không bị ngắt giữa chừng.
18. *Đưa ra 2 lý do tại sao cache hữu ích. Nêu những vấn đề cache giải quyết. Nêu những vấn đề do cache gây ra?*
    - Lý do cache hữu ích:
        - Tăng tốc độ truy cập dữ liệu: cache lưu trữ các dữ liệu mà CPU hoặc chương trình thường xuyên sử dụng. Khi dữ liệu này nằm trong cache, hệ thống có thể truy cập nhanh hơn so với truy cập vào bộ nhớ chính hoặc ổ đĩa, do đó giảm thời gian chờ đợi và tăng hiệu suất.
        - Giảm tải cho bộ nhớ chính cache giúp giảm số lần truy cập vào bộ nhớ chính (RAM) hoặc ổ đĩa, giảm bớt tài nguyên trên các thành phần này, giúp kéo dài tuổi thọ và tiết kiệm tài nguyên hệ thống.
    - Những vấn đề mà cache giải quyết:
        - Giảm độ trễ: cache giúp giảm độ trễ trong việc truy cập dữ liệu, đặc biệt trong các tác vụ cần xử lý nhanh như các ứng dụng thời gian thực, trò chơi, hoặc các ứng dụng đồ họa.

    - Tối ưu hóa hiệu suất hệ thống: cache làm giảm tần suất truy cập vào bộ nhớ chính và các thiết bị lưu trữ ngoài, giúp hệ thống vận hành mượt mà và ổn định hơn.
- Những vấn đề mà cache gây ra:
    - Vấn đề nhất quán dữ liệu: Khi dữ liệu được lưu trữ trong cache và cũng tồn tại trong bộ nhớ chính, có thể xảy ra sự không đồng bộ giữa các phiên bản dữ liệu. Điều này gọi là vấn đề nhất quán dữ liệu (cache coherence), đặc biệt là trong các hệ thống đa lõi hoặc đa bộ xử lý.
    - Tốn tài nguyên và phức tạp trong quản lý: cache yêu cầu thêm dung lượng bộ nhớ và phức tạp trong quản lý, như xác định dữ liệu nào cần lưu trữ trong cache và dữ liệu nào cần loại bỏ khi cache đầy. Cách chính sách thay thế cũng cần được triển khai để quản lý cache hiệu quả, điều này làm tăng độ phức tạp của hệ thống.