Nội dung

Chương 1: Giới thiệu về hệ điều hành

1. **Các thành phần của hệ thống máy tính và vai trò của hệ điều hành.**
   - **Phần cứng**: cung cấp các tài nguyên cần thiết cho việc tính toán, xử lý dữ liệu.
   - **Phần mềm**: các chương trình cụ thể (phần mềm hệ thống và ứng dụng).
   - **HĐH**: phần mềm đóng vai trò trung gian giữa phần cứng và người sử dụng chương trình ứng dụng, làm cho việc sử dụng hệ thống máy tính được tiện lợi và hiệu quả.

2. **Trình bày khái niệm hệ điều hành. Phân tích rõ 2 chức năng cơ bản của hệ điều hành.**
   Hệ điều hành là hệ thống phần mềm đóng vai trò trung gian giữa người sử dụng và phần cứng của máy tính nhằm thực hiện 2 chức năng cơ bản:

   - Quản lý tài nguyên:
     - Đảm bảo tài nguyên hệ thống được sử dụng có ích và hiệu quả.
     - Các tài nguyên: bộ xử lý (CPU), bộ nhớ chính, bộ nhớ ngoài (các đĩa), các thiết bị vào ra.
     - Phân phối tài nguyên cho các ứng dụng hiệu quả.
     - Đảm bảo không xâm phạm tài nguyên cấp cho chương trình khác.
   - Quản lý việc thực hiện các chương trình:
     - Nhiệm vụ quan trọng nhất của máy tính là thực hiện các chương trình, 1 chương trình đang trong quá trình chạy gọi là tiến trình (process).
     - Quản lý chương trình để thực hiện thuận lợi, tránh lỗi, đồng thời đảm bảo môi trường cho việc xây dựng và thực hiện chương trình.
     - Để chạy chương trình cần thực hiện một số thao tác nhất định => HĐH giúp việc chạy chương trình dễ dàng hơn, gi người dùng không cần phải thực hiện thao tác.
     - Để tạo môi trường thuận lợi cho chương trình, HĐH tạo ra các máy ảo:
       + Là máy logic với các tài nguyên ảo.
       + Tài nguyên ảo mô phỏng tài nguyên thực được thực hiện bằng phần mềm.
       + Cung cấp các dịch vụ cơ bản như tài nguyên thực.
       + Dễ sử dụng hơn, số lượng tài nguyên ảo có thể lớn hơn số lượng tài nguyên thực.

3. **Dịch vụ của hệ điều hành? Trình bày những dịch vụ điển hình mà hệ điều hành cung cấp. Làm rõ về quá trình tải và chạy hệ điều hành khi mới khởi động.**
   - **Dịch vụ của hệ điều hành** là những công việc mà hệ điều hành thực hiện giúp người dùng hoặc chương trình ứng dụng.
   - **Những dịch vụ điển hình mà hệ điều hành cung cấp**:
     - Tải và chạy chương trình:

+ Để thực hiện, chương trình được tải từ đĩa vào bộ nhớ, sau đó được trao quyền thực hiện các lệnh. Khi thực hiện xong, cần giải phóng bộ nhớ và các tài nguyên.
+ Toàn bộ quá trình này tương đối phức tạp song lại diễn ra thường xuyên. HDH sẽ thực hiện công việc phức tạp và lặp đi lặp lại này.
+ Do HDH là chương trình đầu tiên được thực hiện khi khởi động hệ thống nên HDH tự tải mình vào bộ nhớ.
+ Nhờ có HĐH, lập trình viên, người sử dụng không cần quan tâm chi tiết đến việc tải và chạy chương trình.

- Giao diện với người dùng: cho phép giao tiếp giữa HDH và người dùng dưới dạng dòng lệnh hoặc giao diện đồ họa.
- Thực hiện các thao tác vào/ ra dữ liệu: Hệ điều hành sẽ phụ trách thực hiện yêu cầu vào/ ra dữ liệu với đĩa và các thiết bị ngoại vi thay cho người dùng và chương trình.
- Làm việc với hệ thống file:
  + Nhu cầu đọc, ghi, tạo, xóa, chép file hoặc làm việc với thư mục.
  + Quản lý quyền truy cập, sao lưu.
- Phát hiện và xử lý lỗi: Phát hiện và xử lý kịp thời các lỗi xuất hiện trong phần cứng cũng như phần mềm => Đảm bảo cho hệ thống hoạt động ổn định, an toàn.
- Truyền thông: Cung cấp dịch vụ cho phép thiết lập liên lạc và truyền thông tin dưới dạng thông điệp hoặc qua vùng bộ nhớ dùng chung.
- Cấp phát tài nguyên:
  + Trong các hệ thống cho phép nhiều chương trình thực hiện đồng thời cần có cơ chế cấp phát và phân phối tài nguyên hợp lý.
  + Người dùng và trình ứng dụng không phải tự thực hiện việc cấp phát tài nguyên mà vẫn đảm bảo cấp phát công bằng và hiệu quả.
- Dịch vụ an ninh và bảo mật:
  + Đối với hệ thống nhiều người dùng, người dùng này không được tiếp cận thông tin của người khác nếu không được cho phép.
  + Cần đảm bảo để tiến trình không truy cập trái phép tài nguyên (như vùng nhớ, file mở) của tiến trình khác hay chính HDH sẽ thực hiện bằng cách kiểm soát truy cập tới tài nguyên.

- **Quá trình tải và chạy hệ điều hành khi mới khởi động**

Hệ điều hành là chương trình đầu tiên được thực hiện khi khởi động hệ thống nên hệ điều hành phải tự tải chính mình từ bộ nhớ ngoài vào bộ nhớ trong. Quá trình này gọi là booting, diễn ra như sau:

- Hệ điều hành có một chương trình nhỏ gọi là chương trình tải hay chương trình mồi. Chương trình mồi nằm ở 1 vị trí xác định trên đĩa hoặc thiết bị nhớ ngoài khác.
- Sau khi khởi động hệ thống, một chương trình nằm sẵn trong bộ nhớ ROM sẽ được kích hoạt và đọc chương trình mồi của HĐH vào bộ nhớ.
- Chương trình mồi tải các phần khác của HĐH vào bộ nhớ và trao cho HĐH quyền điều khiển hệ thống. Nếu phần đĩa chứa chương trình mồi bị hỏng, phần cứng sẽ hiển thị thông báo "không tìm thấy HĐH".
- Trong trường hợp máy tính được cài nhiều hệ điều hành, chương trình mồi sẽ cho phép người dùng chọn hệ điều hành để tải vào bộ nhớ.

4. Trình bày về giao diện lập trình của hệ điều hành.

- Để các chương trình có thể sử dụng được những dịch vụ, HDH cung cấp giao diện lập trình.
- Giao diện này bao gồm các lời gọi hệ thống là các lệnh đặc biệt mà chương trình ứng dụng gọi khi cần yêu cầu sử dụng một dịch vụ nào đó từ phía HDH.
- Lời gọi hệ thống được thực hiện qua những thư viện hàm gọi là thư viện hệ thống. Các hàm này sẽ giúp người lập trình gọi lời gọi hệ thống tương ứng của hệ điều hành.

5. Trình bày các thành phần cơ bản của hệ điều hành.

Các thành phần thực hiện nhiệm vụ sau:

- Quản lý tiến trình:
  - Tạo và xoá tiến trình (tiến trình người dùng và hệ thống).
  - Tạm treo và khôi phục các tiến trình bị treo.
  - Điều độ tiến trình.
  - Đồng bộ hoá các tiến trình.
  - Tạo cơ chế liên lạc giữa các tiến trình.
  - Giải quyết các bế tắc.
- Quản lý bộ nhớ:
  - Cung cấp và giải phóng bộ nhớ theo yêu cầu của các tiến trình.
  - Quản lý không gian nhớ đã được cấp và không gian còn trống.
  - Cấp phát, phân phối bộ nhớ cho các tiến trình => đảm bảo việc chạy song song giữa nhiều chương trình.
  - Tạo ra bộ nhớ ảo và ánh xạ địa chỉ bộ nhớ ảo vào bộ nhớ thực. Ngăn chặn các truy cập bộ nhớ không hợp lệ
- Quản lý vào, ra:
  - Quản lý thông qua các chương trình điều khiển.

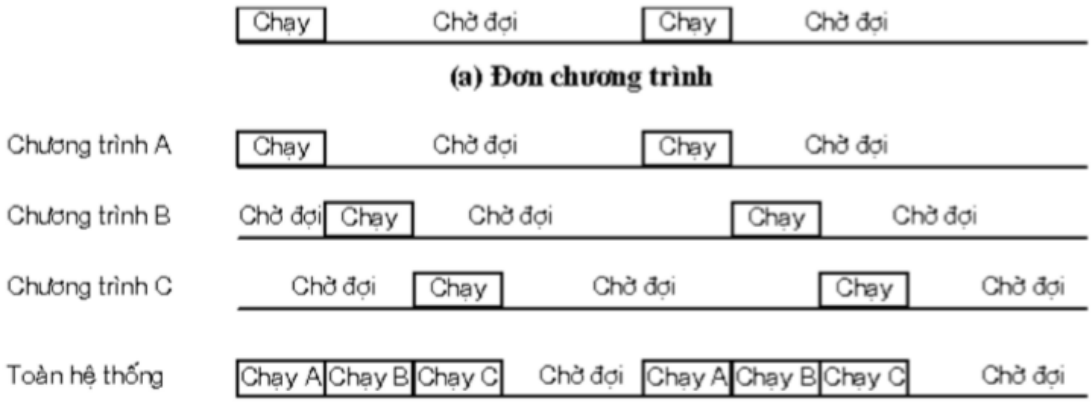
- Đơn giản hoá và tăng hiệu quả quá trình trao đổi thông tin giữa các tiến trình với thiết bị vào ra.

- Quản lý file và thư mục:
  - Tạo, xoá file và thư mục.
  - Cung cấp các chức năng thao tác với file và thư mục, ví dụ đọc/ghi file.
  - Ánh xạ file và thư mục sang bộ nhớ ngoài.
  - Sao lưu dự phòng các files.
- Hỗ trợ mạng và xử lý phân tán: Thông qua các thành phần điều khiển, giao tiếp mạng:
  - Quản lý thiết bị mạng.
  - Hỗ trợ các giao thức truyền thông.
  - Quản lý truyền thông, cân bằng tải.
- Giao diện với người dùng: Là hệ thống thông dịch lệnh giúp cho máy tính hiểu, xử lý được các chỉ thị, các lệnh của người dùng.
- Các chương trình tiện ích và ứng dụng.

6. Trình bày kỹ thuật xử lý theo mẻ (lô) và ưu điểm của kỹ thuật này. Hệ thống xử lý theo mẻ có cần hệ điều hành không ?

- Kỹ thuật xử lý theo mè (lô) là 1 kỹ thuật cho phép tăng hiệu suất sử dụng máy.
- Thay vì làm việc trực tiếp với máy tính, lập trình viên chuẩn bị chương trình trên bìa đục lỗ hoặc trên đĩa từ và giao cho các kỹ thuật viên. Kỹ thuật viên sẽ phân chương trình thành các mẻ, mỗi mẻ gồm những chương trình có yêu cầu giống nhau.
- Mẻ sau đó được nạp vào băng từ và được tải vào máy để thực hiện lần lượt.
- Để tự động hóa xử lý theo mẻ, một chương trình nhỏ gọi là chương trình giám sát (monitor) được giữ thường xuyên trong bộ nhớ. Mỗi khi một chương trình của mẻ kết thúc, chương trình này tự động nạp chương trình tiếp theo của mẻ vào máy và cho phép chương trình này chạy.
  => Giảm đáng kể thời gian chuyển đổi giữa 2 chương trình trong cùng 1 mẻ.
  => Cải thiện đáng kể hiệu suất CPU.
- Sau khi toàn bộ mẻ đã được thực hiện xong, kỹ thuật viên lấy băng từ chứa mẻ ra và nạp tiếp mẻ mới vào để thực hiện.
- Trình giám sát (monitor) chính là **dạng đơn giản nhất của hệ điều hành**.
- Hiệu suất CPU vẫn thấp vì mỗi khi có yêu cầu vào/ra, CPU phải dừng việc xử lý dữ liệu để chờ quá trình vào ra kết thúc. Do tốc độ vào/ra thấp hơn tốc độ CPU rất nhiều nên CPU thường xuyên phải chờ đợi 1 thời gian dài.
- Hệ thống xử lý theo mẻ có cần hệ điều hành.

7. **Trình bày về hệ điều hành đa chương trình. Lý do sử dụng đa chương trình trong máy tính? Lấy VD minh hoạ để phân tích hiệu suất sử dụng CPU của hệ điều hành đa chương trình cao hơn so với hệ điều hành đơn chương trình? Yêu cầu đối với phần cứng khi sử dụng đa chương trình?**
   - **Đa chương trình**:
     - Hệ thống chứa đồng thời nhiều chương trình trong bộ nhớ.
     - Khi 1 chương trình dừng lại để thực hiện vào ra, HĐH sẽ chuyển CPU sang thực hiện một chương trình khác => giảm thời gian chạy không tải của CPU.
   - **Lý do sử dụng đa chương trình**: Để hạn chế nhược điểm của kỹ thuật xử lý theo mẻ. Mặc dù việc xử lý theo mẻ cho phép giảm thời gian chuyển đổi giữa các chương trình ứng dụng song hiệu suất sử dụng CPU vẫn tương đối thấp. Mỗi khi có yêu cầu vào/ra, CPU phải dừng việc xử lý dữ liệu để chờ quá trình vào ra kết thúc. Do tốc độ vào ra luôn thấp hơn tốc độ CPU rất nhiều nên CPU thường xuyên phải chờ đợi trong những khoảng thời gian dài.
   - **VD minh họa hiệu suất** sử dụng thời gian CPU cho trường hợp đơn chương trình và đa chương trình với 3 chương trình cùng được tải vào bộ nhớ một lúc:



   Thời gian thực hiện chương trình xen kẽ với thời gian chờ đợi vào/ra. Dễ dàng nhận thấy, thời gian chờ đợi của CPU trong chế độ đa chương trình giảm đáng kể so với trong trường hợp đơn chương trình.

   - **Việc thực hiện đa chương trình đòi hỏi hỗ trợ từ phần cứng**, đặc biệt là khả năng vào/ra bằng ngắt và cơ chế DMA. Nếu không có cơ chế này, CPU sẽ phải trực tiếp điều khiển quá trình vào/ra thông tin và dữ liệu, hiệu quả của đa chương trình khi đó bằng không.
   - **Nhược điểm đa chương trình:**
     - Mặc dù đa chương trình cho phép sử dụng hiệu quả CPU và các tài nguyên khác của hệ thống song kỹ thuật này không cho phép người dùng tương tác với hệ thống.

- Không đảm bảo được thời gian đáp ứng ngắn như các máy tính thế hệ hiện nay.

8. Trình bày về hệ điều hành chia sẻ thời gian.

- Chia sẻ thời gian (Đa nhiệm):
  - Chia sẻ thời gian có thể coi như đa chương trình cải tiến.
  - CPU lần lượt thực hiện các công việc khác nhau trong những khoảng thời gian ngắn gọi là lượng tử thời gian.
  - Chuyển đổi giữa các công việc diễn ra với tần số cao và tốc độ CPU lớn.
- Tất cả người dùng đều có cảm giác máy tính chỉ thực hiện chương trình của mình.
- CPU được chia sẻ giữa những người dùng khác nhau tương tác trực tiếp với hệ thống.

9. Nhân của hệ điều hành là ? Trình bày về chế độ nhân và chế độ người dùng ?

- **Nhân của hệ điều hành**:
  - Những thành phần quan trọng không thể thiếu được khi tải vào bộ nhớ tạo thành nhân của hệ điều hành.
  - Nhân (kernel) là phần cốt lõi, là phần thực hiện các chức năng cơ bản nhất, quan trọng nhất của hệ điều hành và thường xuyên được giữ trong bộ nhớ.
  - Nhân có nhiệm vụ quản lý tài nguyên hệ thống (liên lạc giữa các thành phần phần cứng và phần mềm).
- **Chế độ nhân** hay nhân chạy trong chế độ đặc quyền: là chế độ mà chương trình thực hiện trong đó có đầy đủ quyền truy cập và điều khiển phần cứng máy tính.
- **Chế độ người dùng**: chương trình thực hiện trong chế độ người dùng bị hạn chế rất nhiều quyền truy cập và sử dụng phần cứng.

10. Trình bày về cấu trúc nguyên khối, cấu trúc phân lớp và cấu trúc vi nhân của hệ điều hành.

- **Cấu trúc nguyên khối**:
  - Toàn bộ chương trình và dữ liệu của HĐH có chung 1 không gian nhớ. Do vậy có thể coi là 1 khối duy nhất.
  - HĐH trở thành một tập hợp các thủ tục hay các chương trình con.
  - Ưu điểm: nhanh, không mất thời gian giữa các không gian nhớ.
  - Nhược điểm:
    + Không an toàn: khi bất kỳ thành phần nào có sự cố thì toàn bộ hệ thống sẽ không hoạt động được.
    + Không mềm dẻo và khó sửa đổi, thêm bớt thành phần nào sẽ ảnh hưởng tới toàn bộ hệ thống, khi có lỗi khó xác định lỗi do thành phần nào gây ra.
- **Cấu trúc phân lớp**:

- Các thành phần được chia thành các lớp nằm chồng lên nhau.
- Mỗi lớp chỉ có thể liên lạc với lớp nằm kề bên trên và kề bên dưới.
- Mỗi lớp chỉ có thể sử dụng dịch vụ do lớp ngay bên dưới cung cấp.
- Ưu điểm: chia nhỏ chức năng, dễ sử dụng, dễ sửa lỗi.
- Nhược điểm: khó thiết kế (xác định số lớp, phân chia thành phần các chức năng của mỗi lớp), tốc độ chậm hơn cấu trúc nguyên khối.

- **Cấu trúc vi nhân**:
  - Nhân có kích thước nhỏ, chỉ chứa các chức năng quan trọng nhất.
  - Các chức năng còn lại được đặt vào các modul riêng: chạy trong chế độ đặc quyền hoặc người dùng. Khi có yêu cầu từ ứng dụng, nhân sẽ chuyển cho module tương ứng để xử lý và nhận lại kết quả, nhân chủ yếu đóng vai trò trung gian liên lạc.
  - Ưu điểm: mềm dẻo, an toàn hơn so với cấu trúc nguyên khối.
  - Nhược điểm: tốc độ chậm hơn so với cấu trúc nguyên khối

Chương 2: Hệ thống file

1. Định nghĩa khái niệm file. Việc sử dụng khái niệm đem lại ưu điểm gì ?

- File hay tệp tin được định nghĩa như tập hợp các thông tin liên quan đến nhau được đặt tên và lưu trữ trên bộ nhớ ngoài.
- Nhờ có khái niệm file, người dùng có thể quy định cấu trúc, ý nghĩa, cách sử dụng cho thông tin cần lưu trữ và đặt tên cho tập các thông tin này. Người dùng có thể không cần quan tâm tới việc file được lưu trữ cụ thể ở đâu, ra sao.

2. Quy tắc đặt tên File

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

3. Hệ điều hành có cần biết và hỗ trợ các kiểu cấu trúc file không ? Tại sao ?

Việc hỗ trợ cấu trúc file ở mức hệ điều hành có 1 số ưu và nhược điểm.
Ưu điểm:

- Các thao tác với file sẽ dễ dàng hơn đối với người lập trình ứng dụng.
- HDH có thể kiểm soát được các thao tác với file.

Nhược điểm:

- Tăng kích thước hệ thống.
- Tính mềm dẻo của HDH bị giảm.

Vì các nhược điểm nêu trên nên đa số hệ điều hành không hỗ trợ và quản lý kiểu cấu trúc file. Cấu trúc file sẽ do chương trình ứng dụng và người dùng tự tạo ra và quản lý. UNIX, DOS, WINDOWS là các hệ điều hành như vậy.

4. Các thao tác cơ bản khi làm việc với File. Phân tích thao tác "Mở file" và cho biết 1 hệ thống file có nhất thiết phải có thao tác này không ?

- **Những thao tác với file thường gặp**:
  - Tạo file: Tạo file trống chưa có dữ liệu; File được dành 1 chỗ trong thư mục kèm theo một số thuộc tính.
  - Xoá file: Giải phóng không gian mà dữ liệu của file chiếm trên đĩa sau đó giải phóng chỗ của file trong thư mục.
  - Mở file: Được thực hiện trước khi đọc hoặc ghi file. Thực chất của việc mở file là đọc các thuộc tính của file và vị trí file trên đĩa vào bộ nhớ để tăng tốc độ cho các thao tác đọc ghi tiếp theo.
  - Đóng file: Xóa các thông tin về file ra khỏi bảng trong bộ nhớ để nhường chỗ cho các file sắp mở.
  - Đọc file: Thông tin ở vị trí hiện thời sẽ được đọc.
  - Ghi vào file: Dữ liệu được ghi vào vị trí hiện thời của file. Nếu vị trí hiện thời là cuối file, thông tin sẽ được thêm vào và kích thước file tăng lên. Nếu không phải cuối file, thông tin ở vị trí đó sẽ bị ghi đè lên.
  - Định vị (seek): Cho phép xác định vị trí hiện thời để tiến hành đọc hoặc ghi.
  - Đọc thuộc tính của file: Một số chương trình trong quá trình làm việc cần đọc các thuộc tính của file như quyền truy cập cho hệ thống bảo mật.
  - Thay đổi thuộc tính của file: Một số thuộc tính file có thể được đặt lại giá trị.
  - Cắt bỏ nội dung file: Đây là thao tác xoá bỏ nội dung file và giải phóng vùng không gian do file chiếm nhưng vẫn giữ lại các thuộc tính của file trừ độ dài và vị trí.
  - Khóa file: Khi đồng thời nhiều tiến trình thay đổi một nội dung file, có thể dẫn đến kết quả không mong đợi nên một số HĐH cho phép khoá file. Khi một tiến trình mở file thì tiến trình đó có thể yêu cầu khoá file, không cho phép các tiến trình khác truy cập hoặc có thể đọc nhưng không thể ghi vào file đó.
- **Một hệ thống file không nhất thiết phải có thao tác "Mở file"** nhưng thường được thêm vào do nó giúp tăng hiệu quả thao tác vào ra với file.

Quá trình mở file bao gồm tìm kiếm file trong thư mục, đọc thông tin từ khoản mục của file vào bảng các file đang mở trong bộ nhớ. Thông tin đọc thường bao gồm các thuộc tính của file và vị trí các khối chứa nội dung file trên đĩa.
Sau khi mở file, thao tác trả về chỉ số ứng với file trong bảng đó, được sử dụng cho các thao tác đọc ghi sau này.
Nếu file không được mở trước khi đọc ghi, hệ điều hành phải thực hiện lặp lại quá trình trên trước khi thực hiện đọc ghi => tốn thời gian.

5. Khái niệm thư mục. Thông tin trong các khoản mục có nhất thiết phải lưu trữ gần nhau không ?

- Không gian của đĩa được chia thành các phần gọi là đĩa lôgic. Để quản lý các file trên mỗi đĩa lôgic, thông tin về file được lưu trong hệ thống thư mục. Thư mục được tạo thành từ các khoản mục và mỗi khoản mục tương ứng với một file. Thư mục như 1 bảng, mỗi dòng là khoản mục ứng với 1 file, việc tìm ra dòng cần thiết thực hiện theo tên file. Thư mục cho phép ánh xạ tên file vào file đó.
- Thông tin trong các khoản mục không nhất thiết phải lưu trữ gần nhau vì khoản mục chứa thông tin của file: tên, kích thước, vị trí, kiểu file,... hoặc con trỏ tới nơi lưu trữ những thông tin này.

6. Cấu trúc hệ thống thư mục 1 mức và 2 mức

- **Thư mục 1 mức**:
  - Là kiểu tổ chức thư mục đơn giản nhất.
  - Hệ thống chỉ có 1 thư mục duy nhất lưu tất cả các file.
  - Để phân biệt, tên file không được phép trùng nhau => khó chọn tên file.
  - Số lượng file trong thư mục rất lớn => tìm kiếm file tốn thời gian.
- **Thư mục 2 mức**:
  - Phân cho mỗi người dùng 1 thư mục riêng chỉ chứa các file do người đó tạo.
  - Hệ thống vẫn duy trì 1 thư mục gọi là thư mục gốc. Các khoản mục ứng với thư mục người dùng được chứa trong thư mục gốc này.
  - Mỗi khi người dùng truy cập file, file sẽ được tìm kiếm trong thư mục ứng với tên người đó.
  - Các người dùng khác nhau có thể đặt tên file trùng nhau.
  - Tạo tính cô lập người dùng. Hệ thống có thể tồn tại những file thường được nhiều người dùng truy cập tới như file hệ thống, tiện ích,... Để sử dụng được các file như vậy có thể chép chúng vào thư mục của từng người dùng => lãng phí không gian nhớ vì phải lưu nhiều bản sao của file.

7. Cấu trúc thư mục dạng cây và cấu trúc thư mục dạng đồ thị không có chu trình. Ưu điểm của dạng đồ thị không chu trình so với dạng cây là gì ?

- **Thư mục cấu trúc cây**:

- Mỗi thư mục con có thể chứa các thư mục con khác và các file. Hệ thống thư mục khi đó có thể biểu diễn phân cấp như một cây: cành là thư mục còn lá là các file.
- Mỗi thư mục hoặc thư mục con khi đó sẽ chứa tập hợp các file và các thư mục con mức dưới. Để phân biệt khoản mục của file với khoản mục chỉ tới thư mục con mức dưới, người ta thêm 1 bit đặc biệt chứa trong khoản mục.
  + Nếu bit bằng 1 thì đó là khoản mục của thư mục mức dưới
  + Nếu bit bằng 0 thì đó là khoản mục của file.
- Tại mỗi thời điểm, người dùng làm việc với thư mục hiện thời hay thư mục làm việc. Trong quá trình làm việc, người dùng có thể di chuyển sang thư mục khác tức là thay đổi thư mục hiện thời.
- Tổ chức cây thư mục cho từng đĩa:
  + Trong một số hệ thống file như FAT của DOS cây thư mục được xây dựng cho từng đĩa. Hệ thống thư mục được coi như rừng, mỗi cây mọc trên một đĩa.
  + Trong các hệ điều hành khác như Linux, toàn hệ thống chỉ gồm một cây thư mục to, các đĩa là các cành mọc ra từ cây này.

- **Thư mục cấu trúc đồ thị không tuần hoàn/ chu trình:**
  - Cho phép chia sẻ file và thư mục để có thể xuất hiện trong nhiều thư mục riêng khác nhau.
  - Đây là một cấu trúc đồ thị không chứa chu trình, là dạng mở rộng của cấu trúc cây, trong đó lá và cành có thể đồng thời thuộc về những cành khác nhau.
  - Cách triển khai:
    + Sử dụng liên kết (link): con trỏ tới thư mục hoặc file khác.
    + Tạo ra các bản sao của các file và thư mục cần chia sẻ rồi chứa vào những thư mục khác nhau => HĐH phải theo dõi để đảm bảo tính đồng bộ và nhất quán cho các bản sao đó.
  - Cho phép quản lý hệ thống thư mục mềm dẻo hơn cấu trúc cây song cũng phức tạp hơn rất nhiều.
- **Ưu điểm cấu trúc đồ thị không chu trình so với dạng cây**:
  - Cho phép chia sẻ file và thư mục để có thể xuất hiện trong nhiều thư mục khác nhau.
  - Quản lý hệ thống thư mục mềm dẻo hơn cấu trúc cây.

8. Thế nào là đường dẫn tuyệt đối và đường dẫn tương đối ?

- **Đường dẫn tuyệt đối:**
  - Là đường dẫn đi từ gốc của cây thư mục dẫn tới file, bao gồm tất cả các thư mục ở giữa.
  - Các thành phần của đường dẫn ngăn cách với nhau bởi các ký tự đặc biệt.

- Cho phép chỉ ra vị trí của file trong cây thư mục mà không cần biết thư mục hiện thời là thư mục nào.

- **Đường dẫn tương đối** là đường dẫn tính từ thư mục hiện thời. Để có thể sử dụng đường dẫn tương đối, đa số hệ điều hành đưa thêm vào mỗi thư mục hai khoản mục đặc biệt "." Để biểu diễn thư mục hiện tại và ".." để biểu diễn thư mục mức trên.

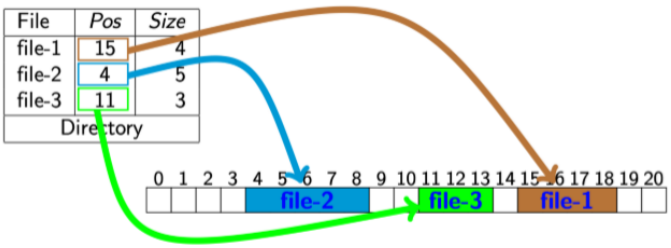
9. Các cấu trúc dữ liệu dùng trong tổ chức bên trong thư mục.

- **Danh sách:**
  - Tổ chức thư mục dưới dạng danh sách các khoản mục.
  - Việc tìm kiếm khoản mục khi đó được thực hiện bằng cách duyệt lần lượt danh sách.
  - Để thêm file mới vào thư mục, trước tiên phải duyệt cả thư mục để kiểm tra xem khoản mục với tên file như vậy đã có chưa. Khoản mục mới sau đó được thêm vào cuối danh sách hoặc một ô trong bảng nếu ô này chưa được sử dụng.
  - Để mở file hoặc xóa file, ta cũng tiến hành tìm tuần tự khoản mục tương ứng.
  - Việc tìm kiếm trong danh sách rất chậm do phải duyệt lần lượt.
  - Lưu trữ thư mục trong bộ nhớ.
- **Cây nhị phân:**
  - Xây dựng thư mục bằng cấu trúc dữ liệu có hỗ trợ sắp xếp như cây nhị phân (B Tree) hoặc cây nhị phân cân bằng, giúp tăng tốc độ tìm kiếm.
  - Hệ thống file NTFS của Windows NT là một ví dụ sử dụng thư mục kiểu này.
- **Bảng băm:**
  - Dùng hàm băm để tính vị trí của khoản mục trong thư mục theo tên file.
  - Thời gian tìm kiếm nhanh.
  - Việc tạo và xoá file cũng được thực hiện đơn giản bằng cách tính vị trí của khoản mục cần tạo hay xoá trong bảng băm.
  - Nhược điểm lớn nhất của cấu trúc này là hàm băm phụ thuộc vào kích thước của bảng băm do đó bảng phải có kích thước cố định. Nếu số lượng khoản mục vượt quá kích thước chọn cho bảng băm thì phải chọn lại hàm băm.
- **Tổ chức thư mục của DOS (FAT):**
  - Mỗi đĩa lôgic có cây thư mục riêng của mình, bắt đầu từ thư mục gốc ROOT.
  - Thư mục gốc được đặt ở phần đầu của đĩa, ngay sau sector khởi động BOOT và bảng FAT.
  - Thư mục gốc chứa các file và thư mục con.
  - Thư mục con có thể chứa file và thư mục mức dưới nữa.
  - Thư mục được tổ chức dưới dạng bảng. Mỗi khoản mục chiếm 1 dòng trong bảng này và có độ dài cố định là 32bytes.
- **Tổ chức thư mục của Linux:**

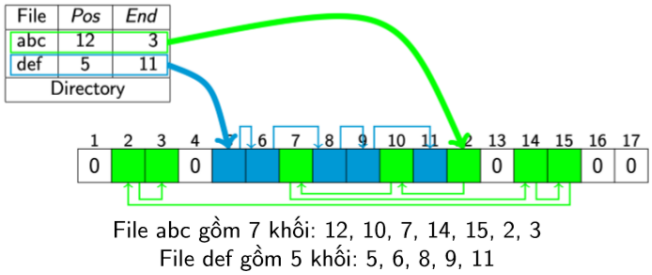
- Thư mục hệ thống file Ext2 của Linux có cách tổ chức bên trong đơn giản.
- Mỗi khoản mục chứa tên file và địa chỉ i-node.
- Toàn bộ thông tin còn lại về thuộc tính và vị trí các khối dữ liệu của file được lưu trong i-node chứ không phải trong thư mục.
- Kích thước khoản mục phụ thuộc vào độ dài tên file. Do vậy, phần đầu mỗi khoản mục có một trường cho biết kích thước của khoản mục này.

10. Trình bày các phương pháp cấp phát không gian cho file: Ưu/ Nhược điểm và nêu VD, so sánh điểm giống và khác.

- **Cấp phát các khối liên tiếp:**
  - File được cấp phát các khối liên tiếp trên đĩa.
  - Bảng cấp phát xác định vị trí file gồm 1 khoản mục cho 1 file, khối bắt đầu, và số khối (độ dài) của file.



  - HĐH cấp phát trước và biết kích thước file khi tạo file.
  - Ưu điểm:
    + Cho phép truy cập trực tiếp và truy cập tuần tự.
    + Truy cập file đơn giản, tốc độ cao do tiết kiệm thời gian di chuyển đầu từ khi đọc các khối.
  - Nhược điểm:
    + Khó tìm ra khoảng không gian trống đủ lớn trên đĩa để cấp phát cho file. Khi có yêu cầu cấp phát, hệ điều hành cần kiểm tra các vùng trống để tìm ra vùng có kích thước thích hợp.
    + Gây phân mảnh ngoài. (hiện tượng các vùng trống còn lại trên đĩa có kích thước quá nhỏ và do vậy không thể cấp phát cho file có kích thước lớn hơn)
    + Phải biết kích thước file khi tạo file.
- **Sử dụng danh sách kết nối:**
  - File được phân phối các khối nhớ kết nối với nhau thành danh sách kết nối, đầu khối chứa con trỏ trỏ tới khối tiếp theo.
  - Mỗi khi file được cấp thêm khối mới, khối vừa cấp được thêm vào cuối danh sách.

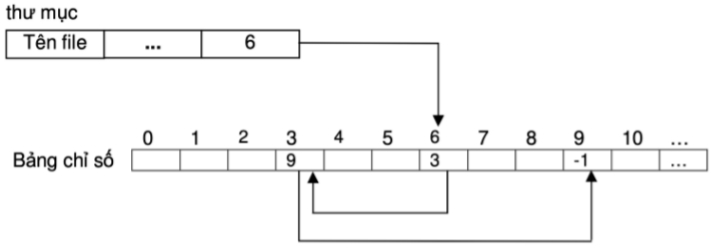


File abc gồm 7 khối: 12, 10, 7, 14, 15, 2, 3
File def gồm 5 khối: 5, 6, 8, 9, 11

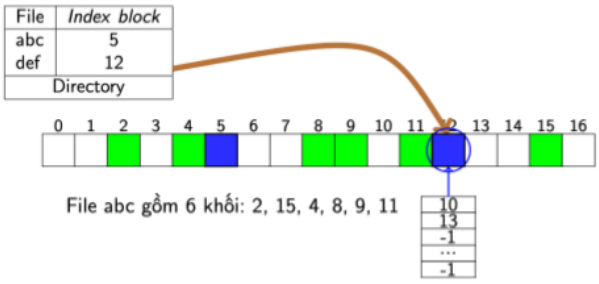
  - Các khối thuộc về một file có thể nằm bất cứ đâu trên đĩa.

- Khoản mục của file trong bảng cấp phát sẽ chứa con trỏ tới khối đầu tiên của file. Để truy cập file, hệ điều hành đọc lần lượt từng khối và sử dụng con trỏ để xác định khối tiếp theo.
- Ưu điểm:
  + Không bị phân mảnh ngoài do các khối không nhất thiết nằm cạnh nhau.
  + Không yêu cầu biết trước kích thước khi tạo file.
  + Dễ tìm vị trí cho file, khoản mục đơn giản.
- Nhược điểm:
  + Chỉ hỗ trợ truy cập tuần tự mà không cho phép truy cập file trực tiếp. Để đọc một khối nào đó ta phải theo tất cả các con trỏ từ khối đầu tiên cho tới khối cần đọc.
  + Tốc độ truy cập file không cao.
  + Việc liên kết bằng con trỏ cũng làm giảm độ tin cậy và tính toàn vẹn của hệ thống file. Trong trường hợp giá trị các con trỏ bị thay đổi không đúng do lỗi, việc xác định khối nào thuộc file nào sẽ không chính xác.

- **Sử dụng danh sách kết nối trên bảng chỉ số:**
  - Bảng chỉ số là một bảng trong đó mỗi ô ứng với một khối của đĩa. Con trỏ tới khối tiếp theo của file được chứa trong ô tương ứng của bảng này.



  - Mỗi đĩa logic có 1 bảng chỉ số được lưu ở vị trí xác định, thường là đầu đĩa.
  - Kích thước mỗi ô trong bảng phụ thuộc vào số lượng khối trên đĩa.
  - Cho phép truy cập file trực tiếp: Thay vì đọc tất cả các khối nằm trước khối cần truy cập, ta chỉ cần đi theo chuỗi con trỏ chứa trong bảng chỉ mục.
- **Sử dụng khối chỉ số:**
  - Tất cả con trỏ tới các khối thuộc về một file được tập trung một chỗ.
  - Mỗi file có 1 mảng riêng chứa trong một khối gọi là khối chỉ số (I-node). Mảng này chứa thuộc tính của file và vị trí các khối của file trên đĩa.



  - Ô thứ i của mảng này chứa con trỏ tới khối thứ i của file.
  - Vấn đề chọn kích thước cho i-node:
    + Nhỏ: tiết kiệm không gian nhưng có thể không đủ để chứa các con trỏ tới các khối nếu file lớn.

+ Lớn: với file nhỏ chỉ chiếm 1 hoặc 2 ô thì các ô còn lại không được sử dụng sẽ gây lãng phí bộ nhớ.

- Giải pháp:
  + Cách 1: Cho phép thay đổi kích thước i-node bằng cách sử dụng danh sách kết nối.
  + Cách 2: Sử dụng i-node có cấu trúc nhiều mức.
- Ưu điểm:
  + Cho phép truy cập trực tiếp các khối trong file.
  + Các khối thuộc về 1 file không cần nằm gần nhau => tránh phân mảnh ngoài
- Nhược điểm:
  + Giống dùng danh sách kết nối, đó là do các khối thuộc về 1 file không nằm gần nhau, tốc độ truy cập file bị giảm vì phải di chuyển đầu đọc nhiều lần.

11. Các phương pháp quản lý không gian trống trên đĩa

- **Bảng bit:**
  - Bảng bit là một mảng một chiều. Mỗi ô của mảng có kích thước 1 bit và ứng với 1 khối nhớ trên đĩa.
  - Khối đã được cấp phát có bit tương ứng là 0, khối chưa được cấp phát có bit tương ứng là 1 (hoặc ngược lại).
  - Dễ tìm ra 1 hoặc nhóm các khối trống liên tiếp.
  - Với những đĩa kích thước lớn, việc đọc toàn bộ bảng bit vào bộ nhớ có thể đòi hỏi khá nhiều bộ nhớ.
- **Danh sách kết nối:**
  - Các khối trống được liên kết với nhau thành danh sách.
  - Mỗi khối trống chứa con trỏ chỉ tới khối trống tiếp theo.
  - Địa chỉ khối trống đầu tiên trong danh sách được lưu trữ ở một vị trí đặc biệt trên đĩa và được HĐH giữ trong bộ nhớ khi cần làm việc với các file.
  - Đòi hỏi truy cập lần lượt các khối trống khi cần duyệt danh sách. Tuy nhiên, trong đa số trường hợp, hệ điều hành ít khi phải duyệt danh sách mà có thể cấp phát ngay các khối ở đầu danh sách.
- **Danh sách vùng trống:**
  - Các khối nằm liền nhau thường được cấp phát và giải phóng đồng thời
    => HĐH lưu vị trí khối trống đầu tiên của vùng các khối trống liên tiếp và số lượng các khối trống nằm liền ngay sau đó.
  - Hai thông tin này sau đó sẽ được lưu trong một danh sách riêng.

12. Các yêu cầu và phương pháp để đảm bảo tính toàn vẹn của hệ thống file

- **Yêu cầu phải đảm bảo tính toàn vẹn:**
  - Hệ thống file chứa nhiều cấu trúc dữ liệu có mối liên kết:

+ Nếu thông tin về các liên kết bị hư hại thì tính toàn vẹn của hệ thống file bị phá vỡ.
+ Liên kết đứt đoạn, không biết lưu ở khối nào tiếp theo.

- Một số trường hợp phá vỡ tính toàn vẹn:
+ Các khối không có mặt trong danh sách các khối trống, đồng thời cũng không có mặt trong một file nào.
+ Một khối vừa thuộc 1 file nào đó vừa có mặt trong danh sách khối trống.
+ Một khối đồng thời thuộc về 2 file cùng 1 lúc.
+ File bị xoá trong khi khoản mục ứng với file trong thư mục vẫn còn hoặc trường hợp ngược lại.
- Mặc dù có thể kiểm tra tính toàn vẹn của các liên kết trong hệ thống file bằng các chương trình có trên 1 số HĐH. Tuy nhiên, kiểm tra toàn bộ hệ thống file đòi hỏi nhiều thời gian. Việc kiểm tra chỉ cho phép phát hiện lỗi sau khi đã xảy ra và không đảm bảo khôi phục dữ liệu đối với một số lỗi.

- **Phương pháp đảm bảo tính toàn vẹn:**
  - Sử dụng 1 kỹ thuật dựa trên khái niệm giao tác (transaction). Giao tác là một tập hợp các thao tác cần phải được thực hiện trọn vẹn cùng với nhau.
  - Với hệ thống file: mỗi giao tác sẽ bao gồm những thao tác thay đổi liên kết cần thực hiện cùng nhau (Ví dụ các thao tác cập nhật liên kết liên quan với một khối trên đĩa).
  - Toàn bộ trạng thái hệ thống file được hệ thống ghi lại trong một file log (một dạng file nhật ký ghi lại thông tin về hệ thống theo thời gian). Mỗi khi thực hiện giao tác, hệ thống kiểm tra xem giao tác có được thực hiện trọn vẹn không. Nếu giao tác không được thực hiện trọn vẹn, HDH sử dụng thông tin từ log để khôi phục hệ thống file về trạng thái trước khi thực hiện giao tác.

13. Cách kiểm soát truy cập file sử dụng mật khẩu và sử dụng danh sách quản lý truy cập.

- **Sử dụng mật khẩu:**
  - Mỗi file sẽ có một mật khẩu gắn với một số quyền nào đó.
  - Người dùng phải nhớ nhiều mật khẩu nếu số lượng file/ thư mục lớn.
  - Mỗi thao tác truy cập đều đòi hỏi cung cấp mật khẩu nên rất mất thời gian và không tiện lợi.
- **Sử dụng danh sách quản lý truy cập ACL (Access Control List):**
  - Mỗi file sẽ được gắn một danh sách đi kèm gọi là danh sách quản lý truy cập ACL chứa thông tin định danh người dùng và các quyền mà người dùng đó được thực hiện với file.
  - ACL thường được lưu trữ như một thuộc tính của file hoặc thư mục.
  - Phương pháp sử dụng ACL thường được sử dụng cùng với cơ chế đăng nhập.

- Các quyền truy cập cơ bản: đọc; ghi, thay đổi; xoá; thay đổi chủ file.

14. Các nội dung về hệ thống file FAT

- **Trình bày các bước để đọc bảng FAT từ thẻ nhớ USB (FAT16)**
  - Bước 1: Xác định đường dẫn thẻ nhớ USB trên hệ điều hành. Mở ổ đĩa tương ứng vị trí thẻ nhớ USB (thường là ổ E:)
  - Bước 2: Tìm và đọc bộ boot sector của thẻ nhớ USB bằng hàm đọc sector mức thấp absread.
  - Bước 3: Sử dụng thông tin từ bộ boot sector:
    + Cấp phát bộ nhớ để đọc FAT:
    unsigned int *fat = (unsigned int *)malloc(kích thước bảng FAT tính bằng sector * kích thước sector tính bằng byte)
    + Vị trí sector bắt đầu đọc chính là số lượng sector dành cho vùng đầu đĩa đến trước FAT.
  - Bước 4: Tiếp tục dùng hàm đọc sector mức thấp absread để đọc bảng FAT.
- **Trình bày các bước để đọc thư mục gốc từ thẻ nhớ USB vào bộ nhớ trong (hệ thống file là các thẻ nhớ FAT16)**
  - Bước 1: Xác định đường dẫn thẻ nhớ USB trên hệ điều hành. Mở ổ đĩa tương ứng vị trí thẻ nhớ USB (thường là ổ E:)
  - Bước 2: Tìm và đọc bộ boot sector của thẻ nhớ USB bằng hàm đọc sector mức thấp absread.
  - Bước 3: Sử dụng thông tin từ bộ boot sector xác định được:
    + Bộ nhớ cần cấp phát để đọc ROOT: 32 * số khoản mục tối đa trong thư mục gốc ROOT
    + Số sector cần đọc = bộ nhớ cần cấp phát / kích thước sector tính bằng byte
    + Ví trí sector bắt đầu đọc = số lượng sector dành cho vùng đầu đĩa đến trước FAT + kích thước bảng FAT tính bằng sector * số lượng bảng FAT
  - Bước 4: Tiếp tục dùng hàm đọc sector mức thấp absread để đọc thư mục gốc.

Chương 3: Quản lý bộ nhớ

1. Địa chỉ logic và địa chỉ vật lý

- **Địa chỉ logic**
  - Là địa chỉ được gán cho các lệnh và dữ liệu không phụ thuộc vào vị trí cụ thể của tiến trình trong bộ nhớ, được sinh ra trong tiến trình (CPU đưa ra).
  - Chương trình ứng dụng chỉ quan tâm tới địa chỉ logic còn địa chỉ vật lý là do hệ điều hành.
  - Khi thực hiện chương trình, CPU "nhìn thấy" và sử dụng địa chỉ lôgic này để trỏ đến các phần khác nhau của lệnh, dữ liệu.
- **Địa chỉ vật lý**

- Là địa chỉ chính xác trong bộ nhớ của máy tính.
- Các mạch nhớ sử dụng địa chỉ vật lý để truy nhập tới chương trình và dữ liệu.
- Để truy cập tới đối tượng trong bộ nhớ, địa chỉ lôgic cần được ánh xạ thành địa chỉ vật lý bởi khối quản lý bộ nhớ hay khối ánh xạ địa chỉ.

2. Kĩ thuật "Tải trong quá trình chạy" và "Liên kết động và thư viện dùng chung" trong tổ chức chương trình. Phân tích ưu điểm từng kỹ thuật.

- **Tải trong quá trình chạy**
  - Trên thực tế, không phải chương trình nào cũng cần tải toàn bộ vào bộ nhớ để thực hiện trong 1 phiên làm việc.
  - Thay vì tải toàn bộ chương trình vào bộ nhớ, ta chỉ:
    + Tải chương trình chính vào bộ nhớ và chạy.
    + Hàm/ chương trình con nào chưa được gọi đến thì chưa được tải vào bộ nhớ.
    + Mỗi khi có một lời gọi hàm, chương trình gọi sẽ kiểm tra xem hàm được gọi đã nằm trong bộ nhớ chưa. Nếu chưa, chương trình tải sẽ được gọi để tải hàm vào bộ nhớ, ánh xạ địa chỉ hàm vào không gian nhớ chung của chương trình và thay đổi bảng địa chỉ để ghi lại các ánh xạ đó.
    + Việc kiểm tra và tải các hàm do chương trình người dùng đảm nhiệm. Hệ điều hành chỉ cung cấp các hàm phục vụ việc tải các module.
  - Ưu điểm:
    + Tiết kiệm và tối ưu hoá bộ nhớ.
    + Tăng tốc độ khởi động chương trình: Chỉ chương trình chính được tải vào bộ nhớ.
    + Tăng độ linh hoạt: Chỉ những hàm/ chương trình con được dùng mới phải tải vào bộ nhớ, giúp tối ưu hoá hiệu suất chương trình.
- **Liên kết động và thư viện dùng chung**
  - Trong quá trình liên kết tĩnh, các hàm thư viện được liên kết luôn vào chương trình chính.
    => Kích thước chương trình = kích thước chương trình vừa được dịch + kích thước các hàm thư viện.
  - Trên thực tế, có các hàm thư viện được dùng thường xuyên sẽ khiến các hàm xuất hiện lặp lại trong các chương trình, làm tăng không gian mà các chương trình chiếm, bao gồm cả không gian trên đĩa và bộ nhớ chính.
  - Sử dụng kỹ thuật liên kết động:
    + Trong giai đoạn liên kết, không kết nối các hàm thư viện vào chương trình mà chỉ chèn các thông tin về hàm thư viện đó.
    + Trong thời gian chạy, khi đoạn mã chèn vào được thực hiện, đoạn này sẽ kiểm tra xem mô đun thư viện đã có trong bộ nhớ chưa. Nếu chưa, mô đun thư viện sẽ được đọc vào bộ nhớ, sau đó chương trình sẽ thay địa chỉ đoạn mã

chèn thành địa chỉ mô đun thư viện. Lần tiếp theo cần sử dụng, modul thư viện sẽ được chạy trực tiếp.

+ Mỗi mô đun thư viện chỉ có 1 bản sao duy nhất chứa trong bộ nhớ, các tiến trình có thể dùng chung phần mã này => thư viện dùng chung.

+ Cần hỗ trợ từ HĐH.

- Ưu điểm:

+ Tiết kiệm và tối ưu hoá bộ nhớ.

+ Tăng tốc độ khởi động chương trình.

+ Tăng độ linh hoạt.

+ Sử dụng thư viện dùng chung giúp việc cập nhật và sửa lỗi dễ dàng hơn.

3. Trình bày kĩ thuật "Phân chương cố định"

- Bộ nhớ phân thành những chương có kích thước cố định ở những vị trí cố định.
- Mỗi chương chứa được đúng một tiến trình.
- Khi được tải vào, tiến trình được cấp phát một chương. Sau khi tiến trình kết thúc, hệ điều hành giải phóng chương và chương có thể được cấp phát tiếp cho tiến trình mới.
- Lựa chọn kích thước các chương:
  - Bằng nhau:

    + Đơn giản nhưng không mềm dẻo.

    + Tiến trình có kích thước lớn hơn kích thước chương sẽ không thể tải vào chương và chạy được.

    + Gây ra phân mảnh trong.
  - Khác nhau: Có 2 cách lựa chọn chương nhớ để cấp cho tiến trình đang chờ bằng cách chọn chương có kích thước nhỏ nhất:

    + Mỗi chương có 1 hàng đợi riêng: Tiến trình có kích thước phù hợp với chương nào sẽ nằm trong hàng đợi của chương đó.

    ~ Ưu điểm: Tiết kiệm bộ nhớ, giảm phân mảnh trong.

    ~ Nhược điểm: Hệ thống không tối ưu, sẽ có thời điểm hàng đợi của chương lớn hơn thì rỗng, trong khi hàng đợi của chương nhỏ hơn thì có nhiều tiến trình. Các tiến trình nhỏ này buộc phải đợi được cấp chương nhỏ trong khi có thể tải vào chương lớn hơn và chạy.

    + Một hàng đợi chung cho tất cả các chương: Mỗi khi có một chương trống, tiến trình nằm gần đầu hàng đợi nhất và có kích thước phù hợp với chương nhất sẽ được tải vào để thực hiện.

    ~ Ưu điểm: Tiết kiệm bộ nhớ, giảm phân mảnh trong và tối ưu hệ thống.
- Ưu điểm:
  - Đơn giản và ít xử lý.
  - Giảm thời gian tìm kiếm.

- Nhược điểm:
  - Số lượng tiến trình trong bộ nhớ bị hạn chế bởi số lượng chương.
  - Bị phân mảnh trong.

4. Trình bày kĩ thuật "Phân chương động". Phân tích ưu nhược của phương pháp này so với "Phân chương cố định". VD minh hoạ cho các chiến lược cấp phát chương động mà hệ điều hành thường sử dụng (First Fit, Best Fit và Worst Fit).

- **Phân chương động**
  - Kích thước và số lượng chương đều có thể thay đổi.
  - Các vùng trống bộ nhớ có thể được liên kết thành một danh sách kết nối.
  - Khi tiến trình yêu cầu bộ nhớ, hệ điều hành sẽ cố gắng cung cấp cho tiến trình 1 chương có kích thước đúng bằng kích thước tiến trình đó đang cần:
    + Nếu tìm thấy vùng trống tương ứng, vùng trống được chia thành 2 phần. Một phần cấp cho tiến trình theo kích thước yêu cầu, phần còn lại được bổ sung vào danh sách vùng trống mà HĐH quản lý.
    + Nếu không tìm thấy, hệ điều hành phải chờ tới khi có được 1 vùng trống thoả mãn hoặc cho phép tiến trình khác trong hàng đợi ưu tiên thoả mãn được thực hiện trước (nếu độ ưu tiên được đảm bảo).
  - Khi tiến trình kết thúc sẽ tạo vùng trống trong bộ nhớ và các vùng trống nằm cạnh nhau được nhập lại thành vùng lớn hơn.
  - Kỹ thuật này tránh việc phân mảnh trong của phân chương cố định nhưng lại gây ra phân mảnh ngoài. Để giải quyết vấn đề này thường có 2 cách:
    + Sử dụng kỹ thuật dồn bộ nhớ:
      ~ Các chương thuộc về các tiến trình được dịch chuyển lại nằm kề nhau.
      ~ Các vùng trống giữa các tiến trình dồn thành một vùng trống duy nhất.
      ~ Đòi hỏi một thời gian nhất định để di chuyển các chương nhớ và làm nảy sinh vấn đề bố trí lại địa chỉ trong các tiến trình.
      => không thể thực hiện quá thường xuyên.
    + Sử dụng các chiến lược lựa chọn vùng trống để cấp phát:
      ~ Vùng thích hợp đầu tiên (first fit): chọn vùng trống đầu tiên có kích thước >= kích thước cần cấp phát.
      ~ Vùng thích hợp nhất (best fit): chọn vùng trống nhỏ nhất có kích thước >= kích thước cần cấp phát.
      ~ Vùng không thích hợp nhất (worst fit): chọn vùng trống lớn nhất.
- **VD minh hoạ cho các chiến lược cấp phát động**
  Trong bộ nhớ có 4 vùng trống kích thước lần lượt là 3MB, 8MB, 7MB, và 10MB; yêu cầu cấp phát vùng nhớ kích thước 6MB. Ba chiến lược cấp phát ở trên sẽ cho kết quả như sau:
  - Chiến lược first fit sẽ chọn vùng trống 8MB.

- Chiến lược best fit sẽ chọn vùng trống 7MB.
- Chiến lược worst fit sẽ chọn vùng trống 10MB.

5. **Trình bày kỹ thuật phân chương sử dụng phương pháp kề cận (buddy). Phân tích rõ các điểm giống/ khác và ưu nhược của phương pháp kề cận so với phân chương cố định và phân chương động.**

- **Phương pháp kề cận - Buddy system:**
  - Nguyên tắc: Chia đôi liên tiếp vùng trống tự do cho tới khi thu được vùng trống nhỏ nhất thỏa mãn
  - Các chương và khối trống có kích thước là 2^k ($L \leq k \leq H$) với 2^L là kích thước khối nhớ nhỏ nhất và 2^H là kích thước khối nhớ lớn nhất có thể được cấp phát (2^H thường bằng kích thước toàn bộ không gian nhớ)
  - Khi yêu cầu cấp vùng nhớ S:
    + Nếu $2^{(H-1)} < S \leq 2^H$: cấp cả 2^H
    + Nếu $S \leq 2^{(H-1)}$ thì chia vùng trống tìm được thành 2 khối bằng nhau kích thước 2^(H-1) và so sánh tiếp, cứ tiếp tục chia đôi đến khi tìm được khối nhỏ nhất thỏa mãn $2^{(k-1)} < S \leq 2^k$ hoặc đạt giới hạn dưới 2^L.
  - Sau 1 thời gian xuất hiện các khối trống kích thước 2^k. HĐH quản lý các khối bằng cách tạo các danh sách kết nối các khối có kích thước bằng nhau.
  - Khi có 2 khối trống kề cận cùng kích thước sẽ được hợp lại thành 1.
  - Khi có yêu cầu cấp phát, hệ thống xem xét danh sách các khối có kích thước phù hợp nhất để chọn ra, nếu không chọn được thì tìm trong danh sách các khối lớn hơn và tiến hành chia đôi đến khi tìm được khối phù hợp.
- **Giống và khác so với 2 phương pháp phân chương cố định và phân chương động:**
  - Kích thước và số lượng chương trong 2 phương pháp kề cận và động có thể thay đổi, cố định thì không thể.
  - Cách lựa chọn chương để cấp phát của 3 phương pháp này cũng khác nhau.
- **Ưu điểm và nhược điểm của phương pháp kề cận là dung hoà của 2 phương pháp phân chương cố định và phân chương động:**
  - Khả năng tìm kiếm và cung cấp bộ nhớ nhanh hơn phương pháp phân chương động.
  - Vẫn có thể gây phân mảnh ngoài và phân mảnh trong nhưng hạn chế hơn 2 phương pháp trên.
  - Khá phức tạp trong việc quản lý vùng trống.

6. **Cơ chế ánh xạ địa chỉ khi sử dụng kĩ thuật phân chương bộ nhớ. Khi di chuyển chương sang vị trí khác cần thay đổi thông tin gì trong khối quản lý ánh xạ (ánh xạ địa chỉ) ?**
   - Nhiệm vụ ánh xã địa chỉ logic sang địa chỉ vật lý do khối ánh xạ địa chỉ của phần cứng thực hiện như sau:
     - Khi HĐH tải tiến trình vào và thực hiện, CPU sẽ dùng 2 thanh ghi đặc biệt:
       + Thanh ghi cơ sở chứa địa chỉ bắt đầu của tiến trình trong bộ nhớ.
       + Thanh ghi giới hạn chứa độ dài chương chứa tiến trình.
     - Địa chỉ lôgic được so sánh với nội dung thanh ghi giới hạn:
       + Nếu lớn hơn: lỗi truy cập.
       + Nếu nhỏ hơn: đưa tới bộ cộng với thanh ghi cơ sở thành địa chỉ vật lý.
   - Khi di chuyển chương sang vị trí khác, nội dung thanh ghi cơ sở sẽ được thay đổi thành địa chỉ mới của chương. Các phép ánh xạ sau đó vẫn diễn ra như cũ.

7. **Phân trang bộ nhớ và cơ chế ánh xạ địa chỉ của kỹ thuật này ? Ưu/ nhược điểm của phương pháp**
   - **Kỹ thuật phân trang bộ nhớ:**
     - Bộ nhớ vật lý được chia thành các khối nhỏ kích thước cố định và bằng nhau gọi là khung trang hoặc đơn giản là khung:
       + Khung trang vật lý được đánh số 0,1,2,... là địa chỉ vật lý của khung trang.
       + Khung trang được dùng làm đơn vị phân phối nhớ.
     - Không gian địa chỉ lôgic của tiến trình được chia thành những khối gọi là trang có kích thước bằng kích thước khung trang.
     - Mỗi tiến trình sẽ được cấp các khung để chứa các trang nhớ của mình.
     - Trang có thể chứa trong các khung nằm rải rác trong bộ nhớ.
   - **Cơ chế ánh xạ địa chỉ:**
     - HĐH quản lý việc cấp phát khung cho mỗi tiến trình bằng bảng trang: mỗi ô tương ứng với 1 trang và chứa số của khung cấp cho trang đó. Mỗi tiến trình có bảng trang riêng.
     - Ánh xạ địa chỉ gồm 2 phần là số thứ tự trang (p) và độ dịch (địa chỉ lệch) của địa chỉ so với đầu trang (o).
     - Biểu diễn địa chỉ logic dưới dạng địa chỉ có độ dài (m + n) bit với:
       + m bit cao biểu diễn số thứ tự trang (p).
       + n bit thấp biểu diễn độ dịch trong trang nhớ (o).
     - Việc tách phần p và o trong địa chỉ logic được thực hiện dễ dàng bằng phép dịch bit.
     - Chuyển địa chỉ logic sang địa chỉ vật lý:
       + Lấy m bit cao của địa chỉ => được số thứ tự trang.
       + Dựa vào bảng trang, tìm được số thứ tự khung vật lý (k).

+ Địa chỉ vật lý bắt đầu của khung là k * (kích thước trang).
+ Địa chỉ vật lý của byte được tham chiếu = địa chỉ bắt đầu của khung + địa chỉ lệch (độ dịch).

- **Ưu điểm của phân trang**:
  - Không tồn tại hiện tượng phân mảnh ngoài.
  - Hệ số song song cao và cho phép sử dụng chung trang.
  - Cơ chế ánh xạ địa chỉ hoàn toàn trong suốt đối với chương trình.
- **Nhược điểm của phân trang**:
  - Tồn tại hiện tượng phân mảnh trong:
    + Phân mảnh trong khi phân trang có giá trị trung bình bằng nửa trang.
    + Luôn xuất hiện ở trang cuối cùng.
    + Giảm kích thước trang cho phép tiết kiệm bộ nhớ. Tuy nhiên, kích thước trang nhỏ làm số lượng trang lớn, bảng trang to, khó quản lý và không tiện cho việc trao đổi với đĩa do thực hiện theo từng khối lớn.
  - Đòi hỏi hỗ trợ của phần cứng cho chiến lược phân trang lớn.
  - Khi chương trình lớn, bảng quản lý trang nhiều phần tử.

8. Kỹ thuật giúp tăng tốc độ truy cập bảng trang và bảng trang nhiều mức (đọc thêm)

Mỗi thao tác truy cập bộ nhớ đều đòi hỏi truy cập bảng phân trang => tổ chức bảng phân trang sao cho tốc độ truy cập là cao nhất.

- **Một số kỹ thuật liên quan tới việc tổ chức bảng trang**
  - Sử dụng tập hợp các thanh ghi làm bảng phân trang:
    + Tốc độ truy cập bảng phân trang rất cao.
    + Kích thước và số lượng bảng phân trang bị hạn chế do số lượng thanh ghi của CPU không nhiều.
  - Lưu bảng trang trong bộ nhớ trong:
    + Vị trí mỗi bảng được trỏ bởi thanh ghi cơ sở của bảng trang PTBR.
    + Mỗi thao tác truy cập bộ nhớ trong đòi hỏi 2 thao tác, một để đọc ô nhớ tương ứng trong bảng trang và một để thực hiện truy cập cần thiết. Do tốc độ bộ nhớ tương đối chậm => Sử dụng thêm bộ nhớ cache tốc độ cao.
- **Bảng trang nhiều mức**
  - Các hệ thống tính toán hiện đại cho phép sử dụng không gian địa chỉ lôgic rất lớn (2^32 đến 2^64) => Số lượng trang cần quản lý tăng dẫn đến kích thước bảng trang tăng.
  - Do đó cần chia bảng trang thành các phần nhỏ hơn.
  - Việc chia nhỏ được thực hiện bằng cách tổ chức bảng trang nhiều mức. Mỗi khoản mục của bảng mức trên chỉ tới bảng trang khác.

9. Khái niệm phân đoạn bộ nhớ và ưu/ nhược điểm của phương pháp phân đoạn bộ nhớ, so sánh với phân trang ?

- **Khái niệm:**
  - Là cách tổ chức cho phép chương trình được chia thành những phần kích thước khác nhau gọi là đoạn (segment) tuỳ theo ý nghĩa của chúng. Chẳng hạn, ta có thể phân biệt:
    + Đoạn chương trình: chứa mã toàn bộ chương trình, hay một số hàm hoặc thủ tục của chương trình.
    + Đoạn dữ liệu: chứa các biến toàn cục, các mảng.
    + Đoạn ngăn xếp: chứa ngăn xếp của tiến trình trong quá trình thực hiện.
  - Mỗi đoạn chiếm một vùng liên tục:
    + Có vị trí bắt đầu và kích thước.
    + Có thể nằm tại bất cứ đâu trong bộ nhớ.
    + Đối tượng, phần tử trong từng đoạn được xác định bởi vị trí tương đối so với đầu đoạn.
- **Ưu điểm:**
  - Tránh hiện tượng phân mảnh trong, dễ sắp xếp bộ nhớ.
  - Dễ chia sẻ các đoạn giữa các tiến trình khác nhau.
  - Kích thước mỗi đoạn có thể thay đổi không ảnh hưởng đoạn khác.
- **Nhược điểm:**
  - Gây phân mảnh ngoài.
- **So sánh với phân trang:**
  - Phân trang đơn giản hơn do các trang có kích thước bằng nhau trong khi đó phân đoạn cho phép tiến tới cấu trúc của tiến trình.
  - Phân đoạn khắc phục được phân mảng trong của phân trang nhưng lại gây ra phân mảnh ngoài.

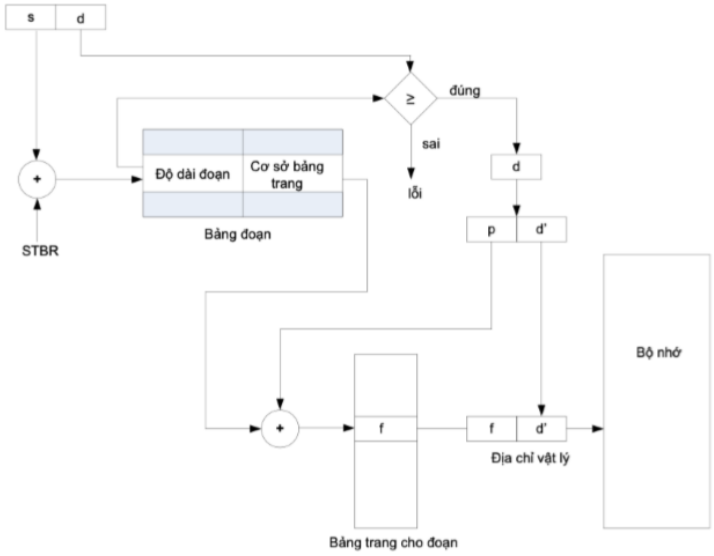
10. Cơ chế ánh xạ địa chỉ khi sử dụng kĩ thuật phân đoạn bộ nhớ.

- Địa chỉ lôgic bao gồm hai thành phần có cấu trúc như sau: <số thứ tự (tên) đoạn (s), độ dịch trong đoạn (o)>
- Ánh xạ địa chỉ: Sử dụng bảng đoạn (segment table) cho mỗi tiến trình. Mỗi phần tử của bảng tương ứng 1 đoạn, chứa:
  - Dấu hiệu (0/1): Đoạn đã tồn tại trong bộ nhớ.
  - Địa chỉ cơ sở: Vị trí bắt đầu của đoạn trong bộ nhớ.
  - Giới hạn đoạn: Độ dài đoạn, được sử dụng để chống truy cập trái phép ra ngoài đoạn.
- Cơ chế ánh xạ địa chỉ:
  - Từ chỉ số đoạn s, vào bảng đoạn, tìm địa chỉ vật lý bắt đầu của đoạn.
  - So sánh độ dịch o với chiều dài đoạn, nếu lớn hơn => địa chỉ sai - Lỗi truy cập

- Địa chỉ vật lý là tổng của địa chỉ vật lý bắt đầu đoạn và địa chỉ lệch.

11. Trình bày phương pháp kết hợp phân trang và phân đoạn. Vẽ sơ đồ và giải thích cơ chế ánh xạ địa chỉ.

- Tiến trình bao gồm nhiều đoạn, mỗi đoạn lại được phân trang. => Mỗi tiến trình có một bảng đoạn riêng và mỗi đoạn lại có bảng trang riêng của mình.

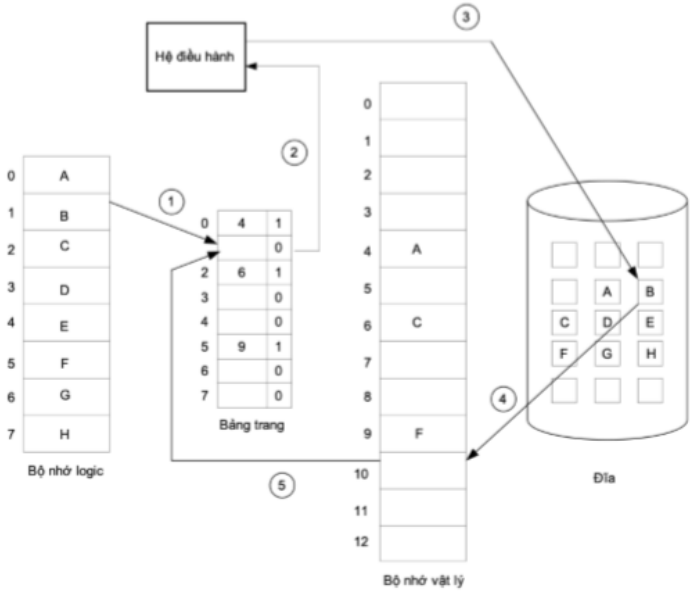


- Cơ chế ánh xạ địa chỉ: Địa chỉ gồm 3 phần <s, p, o>:
  - s là số thứ tự đoạn. s cho phép xác định khoản mục ứng với đoạn trong bảng đoạn. Khoản mục chứa con trỏ tới bảng trang cho đoạn đó.
  - Số thứ tự trang p cho phép xác định khung trang tương ứng.
  - Độ dịch o sẽ được cộng vào địa chỉ khung để tính ra địa chỉ vật lý.

12. Trình bày kỹ thuật "Nạp trang theo nhu cầu" cho bộ nhớ ảo. Nêu các bước, cho VD minh hoạ quá trình thực hiện ngắt thiếu trang? Nạp trang hoàn toàn theo nhu cầu khác nạp trang trước như thế nào.

- **Nạp trang theo nhu cầu**
  - Nạp trang theo nhu cầu dựa trên phân trang kết hợp trao đổi bộ nhớ đĩa. Tiến trình được phân trang và các trang chứa trên đĩa.
  - Khi cần thực hiện tiến trình, ta nạp tiến trình vào bộ nhớ. Chỉ những trang cần dùng mới được nạp vào, không nhất thiết nạp hết cùng 1 lúc.
  - Tại 1 thời điểm, mỗi tiến trình gồm các trang đang ở trong bộ nhớ và các trang còn trên đĩa. Để nhận biết, người ta thêm một bit (bit P) vào khoản mục trong bảng phân trang. Giá trị của bit bằng 1(0) cho thấy trang tương ứng đã ở trong (ngoài) bộ nhớ hoặc ngược lại.
  - Khi tiến trình truy cập tới một trang sẽ kiểm tra bit P. Nếu trang đã ở trong bộ nhớ, việc truy cập diễn ra bình thường. Nếu trang chưa được nạp vào, một sự kiện gọi là thiếu trang hay lỗi trang sẽ xảy ra. Phần cứng làm nhiệm vụ ánh xạ địa chỉ trang sinh ra một ngắt và được chuyển cho HĐH xử lý.

- **Thủ tục xử lý ngắt sinh ra do thiếu trang gồm các bước sau:**
  - Bước 1: HĐH tìm một khung trống trong bộ nhớ vật lý.
  - Bước 2: Đọc trang bị thiếu từ đĩa vào khung trang vừa tìm được.
  - Bước 3: Sửa lại khoản mục tương ứng trong bảng trang: đổi bit P = 1 và số khung đã cấp cho trang.
  - Bước 4: Khôi phục lại trạng thái của tiến trình và thực hiện tiếp lệnh bị ngắt.



- **Chiến lược nạp trang hoàn toàn theo nhu cầu:**
  - Bắt đầu một tiến trình mà không nạp bất kỳ trang nào vào bộ nhớ.
  - Khi con trỏ lệnh được HĐH chuyển tới lệnh đầu tiên của tiến trình để thực hiện, sự kiện thiếu trang sẽ sinh ra và trang tương ứng được nạp vào.
  - Tiến trình sau đó thực hiện bình thường cho tới lần thiếu trang tiếp theo.
- **Chiến lược nạp trang trước:** các trang chưa cần đến cũng được nạp vào bộ nhớ

13. Phân tích rõ cùng 1 lệnh có thể xảy ra nhiều sự kiện lỗi trang không ? (đọc thêm)

Có. Một lệnh có thể sinh ra các yêu cầu truy cập bộ nhớ khác nhau (để đọc lệnh và các toán hạng). Nếu các địa chỉ truy cập thuộc về những trang mới khác nhau thì lệnh đó sẽ gây ra nhiều sự kiện thiếu trang và do vậy làm giảm nghiêm trọng tốc độ thực hiện. Tuy nhiên trong thực tế, hiện tượng này ít xảy ra do tính cục bộ của lệnh và dữ liệu.

14. Lý do phải "Đổi trang" và các bước tiến hành quá trình "Đổi trang"

- **Lý do phải đổi trang**
  - Khi xảy ra thiếu trang, hệ điều hành tìm một khung trống trong bộ nhớ, đọc trang thiếu vào khung và tiến trình sau đó hoạt động bình thường.
  - Tuy nhiên, do kích thước của các tiến trình có thể lớn hơn kích thước bộ nhớ thực rất nhiều nên có lúc sẽ xảy ra tình trạng không còn khung nào trống để nạp trang mới nào.
  - Khi đó có 3 giải pháp: kết thúc tiến trình đó, tạm trao đổi tiến trình này ra đĩa hoặc sử dụng kỹ thuật đổi trang. Kỹ thuật đổi trang được sử dụng trong đa số trường hợp.

- **Quá trình đổi trang**
  - Bước 1: Xác định trang trên đĩa cần nạp vào bộ nhớ.
  - Bước 2: Nếu có khung trống thì chuyển sang bước 4.
  - Bước 3: Chọn 1 khung trên bộ nhớ theo 1 thuật toán nào đó để giải phóng, ghi nội dung khung bị đổi ra đĩa (nếu cần); cập nhật bảng trang và bảng khung.
  - Bước 4: Đọc trang cần nạp vào khung trống hoặc khung vừa giải phóng; cập nhật bảng trang và bảng khung.
  - Bước 5: Thực hiện tiếp tiến trình từ điểm bị dừng trước khi đổi trang.

15. Trình bày các chiến lược đổi trang "Đổi trang tối ưu" (OPT/MIN), "Vào trước ra trước" (FIFO), "Trang có lần sử dụng cuối cách lâu nhất" (LRU) và "Thuật toán đồng hồ" (CLOCK).

- **Đổi trang tối ưu**
  - HĐH chọn trang nhớ sẽ không được dùng tới trong khoảng thời gian lâu nhất trong tương lai để đổi hay trang có lần sử dụng tiếp theo xa nhất.
  - Giảm tối thiểu sự kiện thiếu trang và tối ưu theo tiêu chuẩn này.
  - Hệ điều hành cần đoán trước nhu cầu sử dụng các trang trong tương lai. Điều này rất khó trong thực tế do thứ tự truy cập trang không cố định và không biết trước nên chỉ dùng để so sánh với các chiến lược đổi trang khác.
- **Vào trước ra trước**
  - Trang được nạp vào bộ nhớ trước sẽ bị đổi ra trước.
  - Chiến lược đơn giản nhất, có thể triển khai bằng cách sử dụng hàng đợi FIFO.
  - Trang bị trao đổi là trang nằm trong bộ nhớ lâu nhất.
- **Trang có lần sử dụng cuối cách lâu nhất (LRU)**
  - Trang bị đổi là trang mà thời gian từ lần truy cập cuối cùng đến thời điểm hiện tại là lâu nhất (lâu rồi không truy cập thì thay thế).
  - Theo nguyên tắc cục bộ về thời gian, đó chính là trang ít có khả năng sử dụng tới nhất trong tương lai.
  - Thực tế LRU cho kết quả tốt gần như phương pháp đổi trang tối ưu.
  - Có 2 cách để xác định được trang có lần truy cập cuối diễn ra cách thời điểm hiện tại lâu nhất:

    + Sử dụng biến đếm:

      ~ Mỗi khoản mục của bảng phân trang sẽ có thêm một trường chứa thời gian truy cập trang lần cuối - thời gian logic.

      ~ CPU cũng được thêm một bộ đếm thời gian lôgic này.

      ~ Chỉ số của bộ đếm tăng mỗi khi xảy ra truy cập bộ nhớ.

      ~ Mỗi khi một trang nhớ được truy cập, chỉ số của bộ đếm sẽ được ghi vào trường thời gian truy cập trong khoản mục của trang đó.

      => Trường thời gian luôn chứa thời gian truy cập trang lần cuối.

=> Trang bị đổi là trang có giá trị thời gian nhỏ nhất.

+ Sử dụng ngăn xếp:

~ Ngăn xếp đặc biệt được sử dụng để chứa các số trang.

~ Mỗi khi một trang nhớ được truy cập, số trang sẽ được chuyển lên đỉnh ngăn xếp.

~ Đỉnh ngăn xếp sẽ chứa trang được truy cập gần đây nhất. Đáy ngăn xếp chính là trang LRU, tức là trang cần trao đổi.

~ Cách này tránh phải tìm kiếm trong bảng phân trang và thích hợp thực hiện bằng phần mềm.

- **Thuật toán đồng hồ**
  - Cải tiến FIFO nhằm tránh thay những trang được nạp vào lâu vẫn có khả năng sử dụng.
  - Mỗi trang được gắn thêm 1 bit sử dụng U. U = 1 ngay khi trang được truy cập đọc/ ghi hoặc được đọc vào bộ nhớ.
  - Các trang có thể bị đổi được liên kết vào 1 danh sách vòng.
  - Khi một trang nào đó bị đổi, con trỏ được dịch chuyển để trỏ vào trang tiếp theo trong danh sách.
  - Khi có nhu cầu đổi trang, HDH kiểm tra trang đang bị trỏ tới:
    + Nếu U = 0: Trang sẽ bị đổi ngay.
    + Nếu U = 1: Đặt U = 0 và trỏ sang trang tiếp theo, lặp lại quá trình.
  - Như vậy, thuật toán này căn cứ vào 2 thông tin để đưa ra quyết định đổi trang:
    + Thời gian trang được tải vào, thể hiện qua vị trí trang trong danh sách giống như FIFO.
    + Gần đây trang có được sử dụng hay không, thể hiện qua bit U.
  - Việc kiểm tra thêm bit U tương tự việc cho trang thêm khả năng được giữ trong bộ nhớ.

16. Kỹ thuật "Đệm trang" và ưu điểm.

- HĐH dành ra một số khung trống được kết nối thành danh sách liên kết gọi là các trang đệm.
- Khi có yêu cầu đổi trang:
  + Trang bị đổi như bình thường nhưng nội dung trang này không bị xóa ngay khỏi bộ nhớ.
  + Khung chứa trang bị đổi được đánh dấu là khung trống và thêm vào cuối danh sách trang đệm.
- Trang mới sẽ được nạp vào khung đứng đầu trong danh sách trang đệm.
- Tới thời điểm thích hợp, hệ thống sẽ ghi nội dung các trang trong danh sách đệm ra đĩa.
- Kỹ thuật đệm trang cho phép cải tiến tốc độ do một số lý do sau:

- Nếu trang bị đổi có nội dung cần ghi ra đĩa, hệ điều hành vẫn có thể nạp trang mới vào ngay:
  + Việc ghi ra đĩa sẽ được lùi lại tới một thời điểm muộn hơn.
  + Thao tác ghi ra đĩa có thể thực hiện đồng thời với nhiều trang nằm trong danh sách được đánh dấu trống => tiết kiệm thời gian hơn do thao tác ghi đĩa được tiến hành theo khối lớn.
- Trang bị đổi vẫn được giữ trong bộ nhớ một thời gian:
  + Trong thời gian này, nếu có yêu cầu truy cập, trang sẽ được lấy ra từ danh sách đệm và sử dụng ngay mà không cần nạp lại từ đĩa.
  + Vùng đệm khi đó đóng vai trò như bộ nhớ cache.

17. Các phương pháp xác định số lượng khung trang tối đa cấp cho mỗi tiến trình và xác định phạm vi cấp phát ?

- **Phương pháp cấp phát số lượng khung cố định**
  - HĐH cấp cho tiến trình một số lượng cố định khung để chứa trang nhớ của mình.
  - Số lượng này được xác định vào thời điểm tạo mới tiến trình và không thay đổi trong quá trình tiến trình tồn tại.
  - Đến đây lại có hai cách xác định số lượng khung tối đa:
    + Cấp phát bằng nhau:
      ~ Mỗi tiến trình được cấp số khung tối đa giống nhau.
      ~ Số lượng khi đó được xác định dựa trên kích thước bộ nhớ và mức độ đa chương trình mong muốn.
    + Cấp phát không bằng nhau:
      ~ Số lượng khung tối đa cấp cho tiến trình có thể khác nhau và được tính toán dựa trên đặc điểm tiến trình.
      ~ Cách đơn giản nhất là cấp cho mỗi tiến trình số lượng khung tỷ lệ thuận với kích thước tiến trình.
      ~ Có thể cấp cho tiến trình có mức ưu tiên cao hơn nhiều khung hơn trong những hệ thống có quy định mức ưu tiên cho tiến trình.
- **Phương pháp cấp phát số lượng khung thay đổi**
  - Số lượng khung tối đa cấp cho mỗi tiến trình có thể thay đổi trong quá trình thực hiện.
  - Việc thay đổi số khung tối đa phụ thuộc vào tình hình thực hiện của tiến trình.
  - Sử dụng bộ nhớ hiệu quả hơn phương pháp cố định.
  - Để thay đổi số lượng khung tối đa một cách hợp lý, HĐH cần theo dõi và xử lý thông tin về tình hình sử dụng bộ nhớ của tiến trình.
- **Phạm vi cấp phát khung**
  - Phạm vi cấp phát được phân thành:

+ Cấp phát toàn thể: Cho phép tiến trình đổi trang mới vào bất cứ khung nào (không bị khóa), kể cả khung đã được cấp phát cho tiến trình khác.
+ Cấp phát cục bộ: Trang chỉ được đổi vào khung đang được cấp cho chính tiến trình đó.

- Phạm vi cấp phát có quan hệ mật thiết với số lượng khung tối đa:
+ Số lượng khung cố định tương ứng với phạm vi cấp phát cục bộ.
+ Số lượng khung thay đổi tương ứng với phạm vi cấp phát toàn thể.

18. Tình trạng trì trệ (thrashing) trong quản lý bộ nhớ là gì ?

- Trì trệ là tình trạng phải đổi trang liên tục do thiếu bộ nhớ. Một tiến trình rơi vào tình trạng trì trệ khi thời gian đổi trang của tiến trình lớn hơn thời gian thực hiện.
- Xảy ra khi:
  - Bộ nhớ máy tính có kích thước hạn chế.
  - Tiến trình đòi hỏi truy cập đồng thời nhiều trang nhớ.
  - Hệ thống có mức độ đa chương trình cao.
- Khi tiến trình rơi vào tình trạng trì trệ, tần suất thiếu trang tăng đáng kể => Dấu hiệu để phát hiện và giải quyết vấn đề trì trệ.
- Hệ thống sẽ theo dõi và ghi lại tần suất thiếu trang. Hệ điều hành có thể đặt ra giới hạn trên và giới hạn dưới cho tần suất thiếu trang của tiến trình:
  - Khi tần suất vượt giới hạn trên, hệ thống cấp cho tiến trình thêm khung mới, nếu không thể tìm khung mới để cấp thêm thì tiến trình có thể bị treo hoặc bị kết thúc.
  - Khi tần suất thiếu trang thấp hơn giới hạn dưới, hệ thống thu hồi một số khung của tiến trình.

Chương 4: Quản lý tiến trình

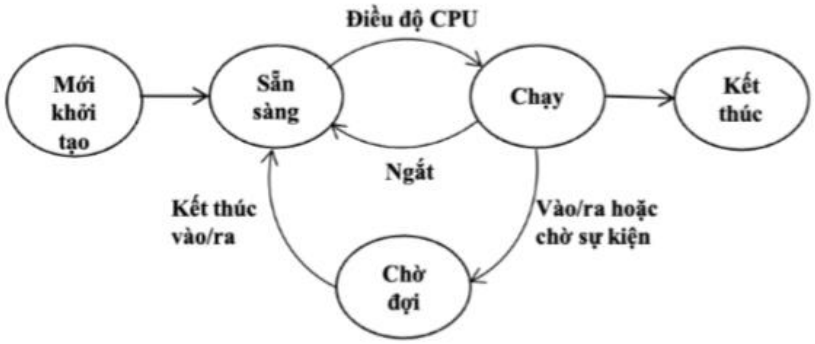
1. Khái niệm tiến trình. Điểm khác nhau giữa tiến trình và chương trình. Nêu tên các thao tác khi làm việc với tiến trình.

- **Tiến trình** là một chương trình đang trong quá trình thực hiện. Tiến trình sinh ra khi chương trình được tải vào bộ nhớ để thực hiện tiến trình người dùng hoặc tiến trình hệ thống.
- **Điểm khác nhau giữa tiến trình và chương trình**:
  - Chương trình là một thực thể tĩnh, không thay đổi theo thời gian, trong khi tiến trình là thực thể động.
  - Chương trình không sở hữu tài nguyên cụ thể, trong khi mỗi tiến trình được cấp một số tài nguyên như bộ nhớ để chứa tiến trình, các cổng và thiết bị vào/ra, các file mở, thời gian CPU để thực hiện lệnh
- **Các thao tác liên quan tới hoạt động quản lý tiến trình**:
  - Tạo mới tiến trình

- Kết thúc tiến trình
- Chuyển đổi giữa các tiến trình
- Điều độ tiến trình
- Đồng bộ hoá tiến trình
- Đảm bảo liên lạc giữa các tiến trình

2. Năm trạng thái của tiến trình, sơ đồ chuyển đổi giữa 5 trạng thái này.

- Khi thực hiện, tiến trình thay đổi trạng thái. Trạng thái của tiến trình là một phần trong hoạt động hiện tại của tiến trình bao gồm 5 trạng thái:
  - Mới khởi tạo (New): tiến trình đang được tạo ra.
  - Sẵn sàng (Ready): tiến trình chờ được cấp CPU để thực hiện lệnh.
  - Chạy (Running): lệnh của tiến trình được CPU thực hiện.
  - Chờ đợi (Waiting): tiến trình chờ đợi một sự kiện nào đó xảy ra (blocked).
  - Kết thúc (Terminated): tiến trình đã kết thúc việc thực hiện nhưng chưa bị xóa.
- **Sơ đồ chuyển đổi giữa các trạng thái**:



  - Khởi tạo -> Sẵn sàng: tiến trình khởi tạo xong và đã được tải vào bộ nhớ, chỉ chờ được cấp CPU để chạy.
  - Sẵn sàng -> Chạy: do kết quả điều độ CPU của HĐH, tiến trình được HĐH cấp phát CPU và chuyển sang trạng thái chạy
  - Chạy -> Sẵn sàng: do kết quả điều độ hoặc do ngắt xảy ra, HĐH cấp phát CPU cho tiến trình khác, tiến trình hiện thời chuyển sang trạng thái sẵn sàng và chờ được cấp CPU để chạy tiếp.
  - Chạy -> Chờ đợi: Tiến trình chỉ chạy khi có 1 sự kiện nào đó xảy ra, chuyển sang trạng thái chờ được phân phối CPU để chạy tiếp.
  - Chạy -> Kết thúc: khi tiến trình đã thực hiện xong.

3. Các thông tin được lưu trong khối quản lý tiến trình – PCB (Process Control Block)

Các thông tin chính trong PCB:

- Số định danh của tiến trình (PID): cho phép phân biệt với tiến trình khác.
- Trạng thái tiến trình: một trong năm trạng thái.

- Nội dung một số thanh ghi CPU:
  - Thanh ghi con trỏ lệnh: trỏ tới lệnh tiếp theo.
  - Thanh ghi con trỏ ngăn xếp : lưu tham số / tình trạng hàm khi thực hiện lời gọi hàm / thủ tục của chương trình.
  - Các thanh ghi điều kiện và trạng thái.
  - Các thanh ghi đa năng.
- Thông tin phục vụ việc điều độ tiến trình.
- Thông tin về bộ nhớ của tiến trình.
- Danh sách các tài nguyên khác.
- Thông tin thống kê phục vụ quản lý.

4. Thao tác "tạo mới tiến trình", "kết thúc tiến trình" và "chuyển đổi giữa các tiến trình".

- **Tạo mới tiến trình**

  Để tạo ra một tiến trình mới, OS thực hiện một số bước như sau:
  - Gán số định danh cho tiến trình được tạo mới và tạo một ô trong bảng tiến trình.
  - Tạo không gian nhớ cho tiến trình và PCB.
  - Khởi tạo PCB: HĐH gán giá trị các thành phần của PCB.
  - Liên kết PCB của tiến trình vào các danh sách quản lý, ví dụ vào danh sách tiến trình đang chạy.
- **Kết thúc tiến trình**

  Kết thúc bình thường: yêu cầu HDH kết thúc mình bằng cách gọi lời gọi hệ thống exit().

  Tiến trình bị kết thúc trong các trường hợp:
  - Bị tiến trình cha kết thúc.
  - Do các lỗi.
  - Yêu cầu nhiều bộ nhớ hơn so với số lương hệ thống có thể cung cấp.
  - Thực hiện lâu hơn thời gian giới hạn.
  - Do quản trị hệ thống hoặc hệ điều hành kết thúc.
- **Chuyển đổi giữa các tiến trình – chuyển đổi ngữ cảnh**:
  - Trong quá trình thực hiện, CPU có thể được chuyển từ tiến trình hiện thời sang thực hiện tiến trình khác.
  - Thông tin về tiến trình hiện thời trong PCB được gọi là ngữ cảnh (context) của tiến trình.
  - Việc chuyển đổi tiến trình xảy ra khi:

    + Có ngắt: ngắt do đồng hồ/ngắt vào/ra.

    + Tiến trình gọi lời gọi hệ thống.
  - Khi chuyển sang thực hiện tiến trình khác, ngữ cảnh lưu vào PCB.

- Khi được cấp phát CPU thực hiện trở lại, ngữ cảnh được khôi phục từ PCB vào các thanh ghi và bảng tương ứng.
- Trong trường hợp đơn giản như hệ thống chuyển sang thực hiện ngắt vào/ra, sau đó quay lại thực hiện tiếp tiến trình hiện thời. Ngữ cảnh cần lưu khi đó sẽ bao gồm những thông tin có thể bị hàm xử lý ngắt làm thay đổi, cụ thể là nội dung các thanh ghi và trạng thái CPU.
- Trong trường hợp phức tạp hơn như sau khi thực hiện ngắt, hệ thống chuyển sang thực hiện tiến trình khác. Việc chuyển đổi ngữ cảnh sẽ bao gồm thêm một số thao tác khác như:
  + Thay đổi trạng thái tiến trình.
  + Cập nhật thông tin thống kê trong PCB.
  + Chuyển liên kết PCB của tiến trình vào danh sách ứng với trạng thái mới.
  + Cập nhật PCB của tiến trình mới được chọn.
  + Cập nhật nội dung thanh ghi và trạng thái CPU.
- Việc chuyển đổi tiến trình đòi hỏi thời gian cùng với tài nguyên hệ thống => ảnh hưởng tới tốc độ nếu diễn ra quá thường xuyên.

5. Khái niệm về luồng và mô hình đa luồng. Ưu điểm của mô hình đa luồng

- Mỗi đơn vị thực hiện của tiến trình, tức là một chuỗi lệnh được cấp phát CPU để thực hiện độc lập được gọi là một **luồng** thực hiện.
- Hệ điều hành hiện nay thường hỗ trợ **đa luồng**, tức là cho phép nhiều chuỗi lệnh được thực hiện cùng lúc trong phạm vi một tiến trình:
  - Trong hệ thống cho phép đa luồng, tiến trình vẫn là 1 đơn vị để HDH phân phối tài nguyên.
  - Mỗi tiến trình sở hữu chung một số tài nguyên:
    + Không gian nhớ của tiến trình (logic): chứa chương trình (các lệnh), phần dữ liệu của tiến trình.
    + Các tài nguyên khác: các file đang mở, thiết bị hoặc cổng vào/ra.
  - Mỗi luồng cần có khả năng quản lý con trỏ lệnh, nội dung thanh ghi.
  - Luồng cũng có trạng thái riêng chứa trong khối quản lý luồng.
  - Luồng có ngăn xếp riêng.
  - Tất cả các luồng của tiến trình chia sẻ không gian nhớ và tài nguyên.
- **Ưu điểm của mô hình đa luồng**:
  - Tăng hiệu năng và tiết kiệm thời gian.
  - Dễ dàng chia sẻ tài nguyên và thông tin.
  - Tăng tính đáp ứng.
  - Tận dụng được kiến trúc xử lý với nhiều CPU.
  - Thuận lợi cho việc tổ chức chương trình.

6. **Luồng mức nhân và mức người dùng. Ưu điểm và nhược điểm.**
   - **Luồng mức người dùng**
     - Do trình ứng dụng tự tạo ra và quản lý mà không có sự hỗ trợ từ hệ điều hành.
     - Hệ điều hành vẫn coi tiến trình như một đơn vị duy nhất với một trạng thái tiến trình duy nhất.
     - Việc phân phối CPU được thực hiện cho cả tiến trình thay vì cho từng luồng cụ thể
     - Ưu điểm:
       + Tiết kiệm thời gian do việc chuyển đổi luồng không đòi hỏi chuyển sang chế độ nhân.
       + Trình ứng dụng có thể điều độ riêng, không phụ thuộc vào cách điều độ của hệ điều hành.
       + Có thể sử dụng khi hệ điều hành không hỗ trợ đa luồng.
     - Nhược điểm:
       + Không tận dụng ưu điểm về tính đáp ứng của mô hình đa luồng.
       + Không cho phép tận dụng kiến trúc nhiều CPU.
   - **Luồng mức nhân**
     - Được tạo ra và quản lý bởi hệ điều hành.
     - Hệ điều hành cung cấp giao diện lập trình bao gồm các lời gọi hệ thống mà trình ứng dụng có thể gọi để yêu cầu tạo, xóa luồng và thay đổi tham số liên quan quản lý tới quản lý luồng.
     - HĐH Windows và Linux là 2 ví dụ trong việc hỗ trợ luồng mức nhân.
     - Ưu điểm: Tăng tính đáp ứng và khả năng thực hiện đồng thời của các luồng trong cùng tiến trình.
     - Nhược điểm: Tốc độ chậm do tạo và chuyển đổi luồng thực hiện trong chế độ nhân.

7. **Khái niệm điều độ tiến trình. Có khác gì so với điều độ luồng không ?**
   - **Điều độ tiến trình** hay lập lịch là quyết định tiến trình nào được sử dụng tài nguyên phần cứng khi nào, trong thời gian bao lâu. Tập trung vào vấn đề điều độ đối với CPU là quyết định thứ tự và thời gian sử dụng CPU.
   - **Điều độ tiến trình và điều độ luồng**:
     - Trong những hệ thống trước đây, tiến trình là đơn vị thực hiện chính, là đối tượng được cấp CPU, và việc điều độ được thực hiện đối với tiến trình.
     - Hệ thống hiện nay thường hỗ trợ luồng. Trong trường hợp này, luồng mức nhân là đơn vị thực hiện được HĐH cấp phát CPU chứ không phải tiến trình, và do vậy việc điều độ được hệ điều hành thực hiện trực tiếp với luồng.
     - Tuy nhiên, thuật ngữ điều độ tiến trình vẫn được sử dụng rộng rãi và được hiểu tương đương với điều độ luồng, trừ khi có giải thích cụ thể.

8. Điều độ có phân phối lại và không phân phối lại.

- **Điều độ có phân phối lại**
  - HĐH có thể sử dụng cơ chế ngắt để thu hồi CPU của 1 tiến trình đang trong trạng thái chạy.
  - HĐH có thể phân phối lại CPU một cách chủ động, không phụ thuộc vào hoạt động của tiến trình hiện thời.
  - Đảm bảo chia sẻ thời gian thực sự.
  - Đòi hỏi phần cứng có bộ định thời gian và một số hỗ trợ khác.
  - Vấn đề quản lý tiến trình phức tạp hơn.
- **Điều độ không phân phối lại**
  - Tiến trình đang ở trạng thái chạy sẽ được sử dụng CPU cho đến khi xảy ra 1 trong các tình huống sau:
    + Tiến trình kết thúc.
    + Tiến trình phải chuyển sang trạng thái chờ đợi do thực hiện yêu cầu vào/ra hoặc lời gọi hệ thống.
    + Chờ đợi tín hiệu đồng bộ từ tiến trình khác.
  - Điều độ không phân phối lại còn được gọi là điều độ hợp tác do chỉ thực hiện được khi tiến trình hợp tác và nhường CPU. Nếu tiến trình không hợp tác và dùng CPU vô hạn, các tiến trình khác sẽ không bao giờ được cấp CPU.

9. Các tiêu chí đánh giá thuật toán điều độ tiến trình

- Lượng tiến trình được thực hiện xong:
  - Số lượng tiến trình thực hiện xong trong 1 đơn vị thời gian.
  - Tiêu chí này mang tính trung bình và là một độ đo tính hiệu quả của hệ thống.
- Hiệu suất sử dụng CPU: Cố gắng để CPU càng ít phải nghỉ càng tốt.
- Thời gian vòng đời trung bình của tiến trình: Từ lúc có yêu cầu tạo tiến trình đến khi kết thúc.
- Thời gian chờ đợi:
  - Tổng thời gian tiến trình nằm trong trạng thái sẵn sàng và chờ cấp CPU.
  - Chịu ảnh hưởng trực tiếp của thuật toán điều độ tiến trình.
- Thời gian đáp ứng: Đây là tiêu chí hướng tới người dùng và thường được sử dụng trong hệ thống tương tác trực tiếp.
- Tính dự đoán được: Vòng đời, thời gian chờ đợi, thời gian đáp ứng phải ổn định, không phụ thuộc vào tải của hệ thống.
- Tính công bằng: Các tiến trình cùng độ ưu tiên phải được đối xử như nhau.

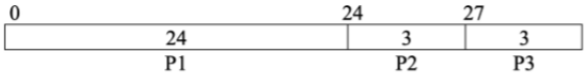
10. Các thuật toán điều độ tiến trình. Cho VD minh hoạ, tính được thời gian chờ đợi trung bình, thời gian vòng đời trung bình.

- **Thuật toán đến trước phục vụ trước** (First Come First Served)

- Tiến trình được quyền sử dụng CPU theo trình tự xuất hiện.
- Tiến trình sở hữu CPU tới khi kết thúc hoặc chờ đợi vào ra => thuật toán này thường là thuật toán điều độ không phân phối lại.

Ví dụ: Cho 3 tiến trình với thứ tự xuất hiện và độ dài chu kỳ CPU như sau:

Tiến trình	Thời gian
P1	24
P2	3
P3	3



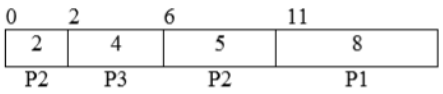
Thời gian chờ đợi của P1, P2, P3 lần lượt là: 0, 24, 27.
Thời gian chờ đợi trung bình $(0 + 24 + 27)/3 = 17$

- Thuật toán đơn giản và dễ thực hiện bằng cách dùng hàng đợi FIFO nhưng có thời gian chờ đợi trung bình của tiến trình lớn do tiến trình ngắn phải chờ đợi như tiến trình dài.

- **Thuật toán điều độ quay vòng** (Round Robin)
  - Mỗi tiến trình được cấp một lượng tử thời gian t để sử dụng.
  - Khi hết thời gian, tiến trình bị trưng dụng vi xử lý và được đưa vào cuối hàng đợi sẵn sàng => có cơ chế phân phối lại, sử dụng ngắt đồng hồ.
  - Nếu có n tiến trình, thời gian chờ đợi nhiều nhất là (n-1) * t.
  - Cải thiện thời gian đáp ứng của tiến trình so với FCFS nhưng vẫn có thời gian chờ đợi trung bình tương đối dài.
- **Thuật toán điều độ ưu tiên tiến trình ngắn nhất** (Shortest Job First)
  - Chọn trong hàng đợi tiến trình có chu kỳ sử dụng CPU tiếp theo ngắn nhất để phân phối CPU.
  - Nếu có nhiều tiến trình với chu kỳ CPU tiếp theo bằng nhau, chọn tiến trình đứng trước.
  - Thời gian chờ đợi trung bình nhỏ hơn nhiều so với FCFS.
  - Thuật toán này là điều độ không phân phối lại.
  - Khó sử dụng trên thực tế do đòi hỏi phải biết trước độ dài chu kỳ sử dụng CPU tiếp theo của tiến trình.
- **Thuật toán điều độ ưu tiên thời gian còn lại ngắn nhất** (Shortest Remaining Time First)
  - Là thuật toán ưu tiên tiến trình ngắn nhất có thêm cơ chế phân phối lại.
  - Khi 1 tiến trình mới xuất hiện trong hàng đợi, HDH so sánh thời gian còn lại của tiến trình đang chạy với thời gian còn lại của tiến trình mới xuất hiện.
  - Nếu tiến trình mới xuất hiện có thời gian còn lại ngắn hơn, HDH thu hồi CPU của tiến trình đang chạy, phân phối cho tiến trình mới.

Tiến trình	Thời điểm xuất hiện	Độ dài chu kỳ CPU
P1	0	8
P2	0	7
P3	2	4

Kết quả điều độ sử dụng SRTF được thể hiện trên biểu đồ sau:



- Có thời gian chờ đợi trung bình nhỏ nhưng đòi hỏi HĐH phải dự đoán được độ dài chu kỳ sử dụng CPU của tiến trình.
- So với điều độ quay vòng, việc chuyển đổi tiến trình diễn ra ít hơn.

- **Thuật toán điều độ có mức ưu tiên** (Priority Scheduling)
  - Mỗi tiến trình có 1 mức ưu tiên.
  - Tiến trình có mức ưu tiên cao hơn sẽ được cấp CPU trước.
  - Các tiến trình có mức ưu tiên bằng nhau được điều độ theo FCFS.
  - Mức ưu tiên được xác định theo nhiều tiêu chí khác nhau.
  - Là điều độ không phân phối lại. Tuy nhiên có thể thêm cơ chế phân phối lại bằng cách: nếu tiến trình mới xuất hiện có mức ưu tiên cao hơn tiến trình đang chạy, HĐH sẽ thu hồi CPU và phân phối cho tiến trình mới.
  - VD sau đây là điều độ có mức ưu tiên không phân phối lại (số ưu tiên nhỏ ứng với độ ưu tiên cao):

Tiến trình	Mức ưu tiên
P1	4
P2	1
P3	3

11. Các vấn đề trong sử dụng và quản lý các tiến trình đồng thời.

- **Tiến trình cạnh tranh tài nguyên với nhau**
  - Cạnh tranh trực tiếp sử dụng chung 1 số tài nguyên như bộ nhớ, đĩa, thiết bị ngoại vi hoặc kênh vào ra:
    + Tiến trình cạnh tranh sẽ phải chờ đợi do chỉ 1 tiến trình được cấp tài nguyên.
    + Ảnh hưởng thời gian thực hiện của tiến trình.
  - Đối với các tiến trình cạnh tranh tài nguyên với nhau cần giải quyết một số vấn đề sau:
    + Vấn đề đoạn nguy hiểm và đảm bảo loại trừ tương hỗ:
      ~ Loại trừ tương hỗ là đảm bảo tiến trình này đang truy cập tài nguyên thì tiến trình khác không được truy cập tài nguyên đó.
      ~ Tài nguyên như vậy gọi là tài nguyên nguy hiểm. Đoạn chương trình có yêu cầu sử dụng tài nguyên nguy hiểm gọi là đoạn nguy hiểm.
      ~ Hai tiến trình không được phép thực hiện đồng thời trong đoạn nguy hiểm của mình.
    + Không để xảy ra bế tắc: là tình trạng 2 hoặc nhiều tiến trình không thể thực hiện tiếp do chờ đợi lẫn nhau tài nguyên mà không được cấp.
    + Không để đói / thiếu tài nguyên (starvation): Đói tài nguyên là chờ đợi quá lâu mà không đến lượt sử dụng một tài nguyên nào đó.
- **Tiến trình hợp tác với nhau qua tài nguyên chung**

- Trao đổi thông tin giữa các tiến trình hợp tác là chia sẻ vùng nhớ dùng chung (biến toàn thể), hay các file.
- Việc các tiến trình đồng thời truy cập dữ liệu dùng chung làm nảy sinh một số vấn đề:
  + Đòi hỏi đảm bảo loại trừ tương hỗ.
  + Xuất hiện tình trạng bế tắc và đói.
  + Yêu cầu đảm bảo tính nhất quán dữ liệu.
- Điều kiện chạy đua (race condition): tình huống mà một số luồng /tiến trình đọc, ghi dữ liệu sử dụng chung và kết quả phụ thuộc vào thứ tự các thao tác đọc, ghi:
  + Đặt thao tác truy cập và cập nhật dữ liệu dùng chung vào đoạn nguy hiểm.
  + Loại trừ tương hỗ để các thao tác này không bị tiến trình khác xen ngang.

- **Tiến trình có liên lạc nhờ gửi thông điệp**
  - Các tiến trình hợp tác có thể trao đổi thông tin trực tiếp với nhau bằng cách gửi thông điệp (message passing).
  - Cơ chế liên lạc được hỗ trợ bởi thư viện của ngôn ngữ lập trình hoặc HĐH.
  - Không có yêu cầu loại trừ tương hỗ.
  - Có thể xuất hiện bế tắc và đói.

12. Giải thuật peterson cho đoạn nguy hiểm. Ưu và nhược điểm của phương pháp. Phân tích giải thuật này có thoả mãn loại trừ tương hỗ, tiến triển hay chờ đợi có giới hạn hay không ?

- **Giải thuật Peterson** do Gary Peterson để xuất năm 1981 cho bài toán đoạn nguy hiểm là giải pháp thuộc nhóm phần mềm.
  Giải thuật Peterson đề xuất cho bài toán đồng bộ hai tiến trình P0 và P1:
  - P0 và P1 thực hiện đồng thời với một tài nguyên chung và một đoạn nguy hiểm chung.
  - Mỗi tiến trình thực hiện vô hạn và xen kẽ giữa đoạn nguy hiểm với phần còn lại của tiến trình.
  - Yêu cầu 2 tiến trình trao đổi thông tin qua 2 biến chung:
    + int turn: xác định đến lượt tiến trình nào được vào đoạn nguy hiểm.
    + Cờ cho mỗi tiến trình: flag[i]=true nếu tiến trình thứ i yêu cầu được vào đoạn nguy hiểm.
- **Ưu điểm giải thuật Peterson**:
  - Thoả mãn điều kiện loại trừ tương hỗ:
    + P0 chỉ có thể vào đoạn nguy hiểm nếu flag[1]=false hoặc turn=0; P1 cũng tương tự nếu flag[0]=false hoặc turn=1.
    + Nếu P0 và P1 có thể vào đoạn nguy hiểm cùng 1 lúc, tức flag[0]=flag[1]=true.

+ Dễ thấy rằng P0 và P1 không thể thực hiện thành công các câu lệnh while của chúng cùng một lúc, vì giá trị của turn chỉ có thể là 0 hoặc 1 => P0 và P1 không thể vào đoạn nguy hiểm cùng 1 lúc.

- Thoả mãn điều kiện tiến triển và chờ đợi có giới hạn:
  + P0 chỉ có thể bị P1 ngăn cản vào đoạn nguy hiểm nếu flag[1]=true và turn=1
  + Có 2 khả năng với P1 ở ngoài đoạn nguy hiểm:
  ~ P1 chưa sẵn sàng vào đoạn nguy hiểm => flag[1]=false, P0 có thể vào ngay đoạn nguy hiểm
  ~ P1 đã đặt flag[1]=true và đang trong vòng lặp while => turn=1 hoặc turn=0. P0 vào đoạn nguy hiểm ngay nếu turn=0. P1 vào đoạn nguy hiểm, sau đó đặt flag[1]=false nếu turn=1 => quay lại khả năng 1.

- Nhược điểm:
  - Sử dụng trên thực tế tương đối phức tạp.
  - Đòi hỏi tiến trình đang yêu cầu vào đoạn nguy hiểm phải nằm trong trạng thái chờ đợi tích cực.
  - Chờ đợi tích cực là tình trạng chờ đợi trong đó tiến trình vẫn phải sử dụng CPU để kiểm tra xem có thể vào đoạn nguy hiểm hay chưa. Đối với giải thuật này, tiến trình phải lặp đi lặp lại thao tác kiểm tra trong while trước khi vào được đoạn nguy hiểm => gây lãng phí thời gian CPU.

13. Các giải pháp phần cứng cho vấn đề loại trừ tương hỗ và đoạn nguy hiểm ? Ưu nhược điểm của Test_and_Set. Sử dụng Test_and_Set thực hiện loại trừ tương hỗ cho bài toán "Triết gia ăn cơm". Sử dụng Test_and_Set có gây ra bế tắc và đói hay không ?

- **Các giải pháp phần cứng cho vấn đề loại trừ tương hỗ và đoạn nguy hiểm:**
  - Cấm các ngắt:
    + Cấm không để xảy ra ngắt trong thời gian tiến trình đang ở trong đoạn nguy hiểm để truy cập tài nguyên.
    + Đơn giản nhưng làm giảm tính mềm dẻo của HĐH, có thể ảnh hưởng tới khả năng đáp ứng các sự kiện cần ngắt.
    + Không thể sử dụng đối với máy tính nhiều CPU.
  - Sử dụng lệnh máy đặc biệt:
    + Phần cứng được thiết kế có một số lệnh máy đặc biệt.
    + Hai thao tác kiểm tra và thay đổi giá trị cho một biến (ô nhớ) kiểm tra và xác lập Test_and_Set, hoặc các thao tác so sánh và hoán đổi giá trị hai biến, được thực hiện trong cùng một lệnh máy.
    + Đơn vị thực hiện không bị xen vào giữa như vậy được gọi là thao tác nguyên tử (atomic).
    + Ưu điểm:

~ Việc sử dụng tương đối đơn giản và trực quan.
~ Có thể dùng để đồng bộ nhiều tiến trình.
~ Có thể sử dụng cho trường hợp đa xử lý với nhiều CPU có bộ nhớ chung.

+ Nhược điểm:

~ Đòi hỏi tiến trình đang yêu cầu vào đoạn nguy hiểm phải nằm trong trạng thái chờ đợi tích cực. Chờ đợi tích cực là tình trạng chờ đợi trong đó tiến trình vẫn phải sử dụng CPU để kiểm tra xem có thể vào đoạn nguy hiểm hay chưa.
~ Có thể xuất hiện bế tắc và đói.

- **Sử dụng Test_and_Set thực hiện loại trừ tương hỗ và đoạn nguy hiểm cho bài toán "Triết gia ăn cơm"**

5 triết gia như 5 tiến trình đồng thời với tài nguyên nguy hiểm là đũa và đoạn nguy hiểm là đoạn dùng đũa để ăn.
Sử dụng Test_and_Set:

```
bool chopstick[6] = {false, false, false, false, false, false};
void P(int i){
        for(;;) {
                while(Test_and_Set(chopstick[i]));
                while(Test_and_Set(chopstick[(i + 1) % 5]));
                <Ăn cơm>
                chopstick[i] = false;
                chopstick[(i + 1) % 5] = false;
                <Suy nghĩ>
        }
}
void main() {
        StartProcess(P(1));  ...  StartProcess(P(5));
}
```

- **Sử dụng Test_and_Set:**
  - Có thể gây ra bế tắc: VD trong bài toán "Triết gia ăn cơm", cả năm người cùng lấy được đũa bên trái nhưng sau đó không lấy được đũa bên phải do đũa đã bị người bên phải giữ, không ai chịu nhường đũa xuống => Xảy ra chờ đợi vòng tròn.
  - Có thể gây ra đói: VD trong bài toán "Triết gia ăn cơm", khi 1 triết gia thực hiện việc lấy đũa chậm hơn so với triết gia khác do việc lấy đũa không theo quy luật nào, khiến triết gia đó không bao giờ đủ 2 chiếc đũa để ăn.

14. Phương pháp cờ hiệu (semaphore). Sử dụng phương pháp này cho bài toán "Triết gia ăn cơm" và "Người sản xuất và người tiêu dùng với kho (bộ đệm) hạn chế".

- **Phương pháp cờ hiệu:**
  - Cờ hiệu S là 1 biến nguyên được khởi tạo bằng khả năng phục vụ đồng thời của tài nguyên.
  - Giá trị của S chỉ có thể thay đổi nhờ gọi 2 thao tác là wait và signal và để tránh tình trạng chờ đợi tích cực, sử dụng 2 thao tác phong toả (block) và đánh thức (wakeup):
    + wait(S):
      ~ Giảm S đi 1 đơn vị
      ~ Nếu giá trị của S < 0 thì tiến trình gọi wait(S) sẽ bị phong toả (bởi thao tác block) và thêm vào hàng đợi của cờ hiệu, nếu không thì tiến trình sẽ được thực hiện tiếp.
    + signal(S):
      ~ Tăng S lên 1 đơn vị
      ~ Nếu giá trị của S <= 0 thì 1 trong các tiến trình đang bị phong toả được giải phóng nhờ thao tác đánh thức (wakeup) trong signal và có thể thực hiện tiếp.
  - Khi tiến trình cần truy cập tài nguyên, thực hiện thao tác wait của cờ hiệu tương ứng.
  - Sau khi dùng xong tài nguyên, tiến trình thực hiện thao tác signal trên cùng cờ hiệu: tăng giá trị cờ hiệu và cho phép một tiến trình đang phong tỏa được thực hiện tiếp.
- **Sử dụng cờ hiệu cho bài toán "Triết gia ăn cơm"**
  5 triết gia như 5 tiến trình đồng thời với tài nguyên nguy hiểm là đũa và đoạn nguy hiểm là đoạn dùng đũa để ăn. Cờ hiệu cho phép giải quyết bài toán như sau:
  - Mỗi đũa được biểu diễn bằng 1 cờ hiệu
  - Nhặt đũa: wait()
  - Đặt đũa xuống: signal()

```
semaphore chopstick[6] = {1, 1, 1, 1, 1, 1};
void P(int i){
        for(;;) {
                wait(chopstick[i]);
                wait(chopstick[(i + 1) % 5]);
                <Ăn cơm>
                signal(chopstick[(i + 1) % 5]);
```

```
            signal(chopstick[i]);
            <Suy nghĩ>
        }
    }
    void main() {
        StartProcess(P(1));  ...  StartProcess(P(5));
    }
```

- **Sử dụng cờ hiệu cho bài toán "Người sản xuất và người tiêu dùng với bộ đệm hạn chế"**

  Tương ứng với 3 yêu cầu đồng bộ của bài toán sử dụng các cờ hiệu tương ứng:
  - Người sản xuất và tiêu dùng không được sử dụng bộ đệm cùng lúc: sử dụng cờ hiệu lock khởi tạo bằng 1.
  - Khi bộ đệm rỗng, người tiêu dùng không nên cố lấy sản phẩm: sử dụng cờ hiệu empty khởi tạo bằng 0.
  - Khi bộ đệm đầy, người sản xuất không được thêm sản phẩm: sử dụng cờ hiệu full, khởi tạo bằng N.

  Sử dụng cờ hiệu:

```
    const int N;
    semaphore lock = 1;
    semaphore empty = 0;
    semaphore full = N;
    void producer(){
        for(;;) {
            <Sản xuất>
            wait(full);
            wait(lock);
            <Thêm 1 sản phẩm vào bộ đệm>
            signal(lock);
            signal(empty);
        }
    }
    void consumer(){
        for(;;) {
            wait(empty);
            wait(lock);
            <Lấy 1 sản phẩm từ bộ đệm>
            signal(lock);
            signal(full);
            <Tiêu dùng>
```

```
        }
    }

    void main() {
        StartProcess(producer);
        StartProcess(consumer);
    }
```

15. Phương pháp Monitor loại trừ tương hỗ và đoạn nguy hiểm.

Monitor được định nghĩa dưới dạng một kiểu dữ liệu trừu tượng của ngôn ngữ lập trình bậc cao, chẳng hạn như một class của C++ hoặc Java. Mỗi monitor gồm một dữ liệu riêng, hàm khởi tạo và một số hàm hoặc phương thức để truy cập dữ liệu với các đặc điểm sau:

- Tiến trình/dòng chỉ có thể truy cập dữ liệu của monitor thông qua các hàm hoặc phương thức của monitor.
- Tại mỗi thời điểm:
  - Chỉ một tiến trình được thực hiện trong monitor.
  - Tiến trình khác gọi hàm của monitor sẽ bị phong tỏa, xếp vào hàng đợi của monitor để chờ cho đến khi monitor được giải phóng.
    => Đảm bảo loại trừ tương hỗ đối với đoạn nguy hiểm, đặt tài nguyên nguy hiểm vào trong monitor.
- Khi tiến trình đang thực hiện trong monitor và bị dừng lại để đợi sự kiện hay một điều kiện nào đó được thỏa mãn, tiến trình cần trả lại monitor để tiến trình khác sử dụng. Tiến trình chờ đợi sẽ được khôi phục lại từ điểm dừng sau khi điều kiện đang chờ đợi được thỏa mãn => Sử dụng các biến điều kiện.
- Các biến điều kiện được khai báo và sử dụng trong monitor với 2 thao tác cwait() và csignal():
  - x.cwait():
    + Tiến trình đang ở trong monitor và gọi cwait bị phong tỏa cho tới khi điều kiện x xảy ra.
    + Tiến trình bị xếp vào hàng đợi của biến điều kiện x.
    + Monitor được giải phóng và một tiến trình khác sẽ được vào monitor.
  - x.csignal():
    + Tiến trình gọi csignal để thông báo điều kiện x đã thỏa mãn.
    + Nếu có tiến trình đang bị phong tỏa và nằm trong hàng đợi của x do gọi x.cwait() trước đó thì tiến trình đó sẽ được giải phóng.
    + Nếu không có tiến trình bị phong tỏa thì thao tác csignal sẽ không có tác dụng gì cả.

16. Trình bày giải pháp giúp không xảy ra bế tắc khi sử dụng cờ hiệu cho bài toán "Triết gia ăn cơm" (đọc thêm) (tỉ lệ ra thấp)

- Giải pháp có thể dùng: Tại mỗi thời điểm chỉ cho tối đa bốn người ngồi vào bàn.
- Thực hiện giải pháp "Tại mỗi thời điểm chỉ cho tối đa bốn người ngồi vào bàn":

```
semaphore chopstick[6] = {1,1,1,1,1,1};
semaphore table = 4;
void Philosopher(int i){
        for(;;){
                wait(table);
                wait(chopstick[i]);  //lấy đũa bên trái
                wait(chopstick[(i+1)%5]); //lấy đũa bên phải
                <Ăn cơm>
                signal(chopstick[(i+1)%5]);
                signal(chopstick[i]);
                signal(table);
                <Suy nghĩ>
        }
}
void main(){
        StartProcess(Philosopher(1));  ...  StartProcess(Philosopher(5));
}
```