

1. **>>> COURSE MATERIAL <<<**
2. INTRODUCTION TO OPERATING SYSTEM / Course ID 502047

*Hướng dẫn tự học, chỉ sử dụng để tham khảo và có thể thay đổi tùy vào từng học kỳ/nhóm lớp.*

|  |  |  |
| --- | --- | --- |
|  | **Programming Exercises** | --3--------------------- |
|  | **Exam multichoice questions** | 12-456--9-1—-4-6--90---- |
|  | **Discussion questions** | --3---78-0-23-5-78--12-4 |
|  | **Advance discussions** | ----------------------3- |

**BÀI TẬP CHƯƠNG 2 – CẤU TRÚC HỆ ĐIỀU HÀNH**

2.1 Mục đích của các lời gọi hệ thống là gì?

2.2 Mục đích của trình thông dịch lệnh là gì? Tại sao nó thường tách khỏi nhân?

2.3 Những lời gọi hệ thống nào phải được thực thi bởi trình thông dịch lệnh hoặc shell để bắt đầu một tiến trình mới trên hệ thống UNIX?

2.4 Mục đích của các chương trình hệ thống là gì?

2.5 Ưu điểm chính của phương pháp phân lớp trong thiết kế hệ thống là gì? Những nhược điểm của phương pháp này là gì?

2.6 Liệt kê năm dịch vụ được cung cấp bởi hệ điều hành và giải thích rõ mỗi dịch vụ đó đã tạo sự thuận tiện cho người dùng như thế nào. Trong trường hợp nào thì một chương trình cấp người dùng không thể cung cấp các dịch vụ này? Giải thích.

2.7 Tại sao một số hệ thống lưu trữ hệ điều hành trong phần dẻo (firmware), trong khi những hệ thống khác lưu trữ hệ điều hành trên đĩa cứng?

2.8 Làm thế nào một hệ thống có thể được thiết kế để cho phép lựa chọn hệ điều hành khi khởi động? Chương trình khởi động cần phải làm những công việc gì?

2.9 Các dịch vụ và chức năng được cung cấp bởi một hệ điều hành có thể được chia làm hai loại chính. Mô tả ngắn gọn về hai loại đó, và thảo luận chúng khác nhau như thế nào.

2.10 Mô tả ba phương pháp chung để truyền các tham số cho hệ điều hành.

2.11 Mô tả cách thức có được một hồ sơ thống kê về lượng thời gian một chương trình thực thi các phần mã nguồn khác nhau của nó. Thảo luận về tầm quan trọng của việc có được một hồ sơ thống kê như vậy.

2.12 Những lợi thế và bất lợi của việc sử dụng chung giao diện lời gọi hệ thống để thao tác với cả tập tin lẫn thiết bị là gì?

2.13 Người dùng có thể phát triển trình thông dịch lệnh mới bằng cách sử dụng giao diện lời gọi hệ thống do hệ điều hành cung cấp hay không?

2.14 Mô tả lý do Android sử dụng biên dịch trước-thời-hạn (AOT / ahead-of-time) thay vì biên dịch đúng-thời-hạn (JIT / just-in-time).

2.15 Hai mô hình giao tiếp giữa các tiến trình là gì? Điểm mạnh và điểm yếu của từng phương pháp?

2.16 Đối chiếu và so sánh giao diện lập trình ứng dụng (API) và một giao diện nhị phân ứng dụng (ABI[[1]](#footnote-1)).

2.17 Tại sao sự tách biệt giữa cơ chế (mechanism) và chính sách (policy)?

2.18 Đôi khi khó đạt được một cách tiếp cận phân lớp nếu hai thành phần của hệ điều hành phụ thuộc lẫn nhau. Xác định một kịch bản mà khi đó rất khó phân lớp hai thành phần hệ thống yêu cầu liên kết chặt chẽ các chức năng của chúng.

2.19 Ưu điểm chính của phương pháp tiếp cận vi nhân đối với thiết kế hệ thống là gì? Làm thế nào để các chương trình người dùng và dịch vụ hệ thống tương tác trong một mô hình kiến trúc vi nhân? Những bất lợi của việc sử dụng phương pháp tiếp cận vi nhân là gì?

2.20 Những lợi thế của việc sử dụng các mô-đun nhân khả nạp (loadable kernel modules) là gì?

2.21 iOS và Android giống nhau như thế nào? Và chúng khác nhau như thế nào?

2.22 Giải thích tại sao các chương trình Java chạy trên các hệ thống Android không sử dụng máy ảo Java API và máy ảo?

2.23 Hệ điều hành thử nghiệm Synthesis[[2]](#footnote-2) có tích hợp trình biên dịch kết hợp trong nhân. Để tối ưu hóa hiệu suất các lời gọi hệ thống, nhân tập hợp các tác vụ cụ thể trong không gian nhân để tối thiểu đường dẫn mà lời gọi hệ thống cần phải đi qua để đến nhân. Cách tiếp cận này đi ngược lại cách tiếp cận phân lớp, trong đó đường dẫn qua nhân được mở rộng dài hơn để xây dựng hệ điều hành càng tốt hơn. Thảo luận về những ưu và nhược điểm của phương pháp Synthesis đối với thiết kế nhân và tối ưu hóa hiệu năng hệ thống.

2.24 Trong phần 2.3, chúng ta đã mô tả một chương trình sao chép nội dung của một tập tin đến một tập tin đích. Chương trình này hoạt động bằng cách yêu cầu người dùng nhập tên của tập tin nguồn và đích. Hãy viết chương trình này bằng thư viện POSIX, bao gồm kiểm tra tất cả các lỗi cần thiết, và đảm bảo rằng tập tin nguồn tồn tại.

Khi đã hoàn tất, hãy chạy chương trình, đồng thời chạy một tiện ích theo dõi các lời gọi hệ thống. Các hệ thống Linux cung cấp tiện ích strace[[3]](#footnote-3) và các hệ thống macOS sử dụng lệnh dtruss / dtrace (Lệnh dtruss thực chất là mặt trước của lệnh dtrace, cần thực thi ở chế độ đặc quyền, nên cần sudo khi thực thi).

Các công cụ trên có thể sử dụng như sau (giả sử rằng tên chương trình sao chép tập tin đã làm có tên là isFileCopy):

Linux:

strace ./FileCopy

macOS:

sudo dtruss ./FileCopy

Vì các hệ thống Windows không cung cấp một công cụ như vậy, bạn sẽ phải theo dõi phiên bản Windows của chương trình này bằng trình gỡ lỗi.

1. <https://en.wikipedia.org/wiki/Application_binary_interface> [↑](#footnote-ref-1)
2. <https://academiccommons.columbia.edu/doi/10.7916/D80R9XHX> [↑](#footnote-ref-2)
3. <https://www.howtoforge.com/linux-strace-command/> [↑](#footnote-ref-3)