

**Tên: Trần Viết Gia Huy**

**MSSV: 31231027056**

**Lớp: CS0001**

**Mã LHP: 26D1INF50915501**

---

**Câu 1: Hãy trình bày khái niệm lập trình hệ thống và nêu vai trò của lập trình hệ thống trong phát triển phần mềm. Theo bạn, vì sao lập trình hệ thống lại quan trọng đối với phát triển ứng dụng?**

*Khái niệm:*

Lập trình hệ thống là "lập trình của các hệ thống". Cụ thể hơn là việc xây dựng các phần mềm thuộc loại "Software-facing software". Nó tập trung vào việc tạo ra các phần mềm nền tảng, không (hoặc rất ít) tương tác trực tiếp với người dùng cuối mà tồn tại để hỗ trợ, vận hành cho các phần mềm khác.

*Vai trò:*

Phần mềm hệ thống đóng vai trò là "unsung heroes" và là nền móng cho toàn bộ hệ sinh thái phần mềm hiện đại. Nó lấp đầy khoảng cách giữa các phần cứng/hệ điều hành và phần mềm ứng dụng.

*Tầm quan trọng đối với phát triển ứng dụng:*

Lập trình hệ thống cực kỳ quan trọng vì sự ổn định của ứng dụng phụ thuộc vào nó. Nếu hệ thống ngầm chạy chậm, lỗi hoặc không ổn định thì toàn bộ ứng dụng phía trên (User-facing software) sẽ thất bại.

**Câu 2: So sánh lập trình hệ thống và lập trình ứng dụng theo các tiêu chí:**

- **Mức độ tương tác với hệ điều hành**
- **Quản lý tài nguyên, dữ liệu**
- **Yêu cầu về hiệu năng và độ an toàn**
- **Nêu ví dụ minh họa cho mỗi loại.**

Tiêu chí	Lập trình hệ thống (Systems Programming)	Lập trình ứng dụng (App Programming)
Mức độ tương tác HĐH	Thấp (Low-level): Kiểm soát tài nguyên hệ thống trực tiếp	Cao (High-level): Ít can thiệp hệ thống, tập trung vào logic nghiệp vụ
Quản lý tài nguyên	Quản lý trực tiếp (tiến trình, luồng, bộ nhớ).	Chủ yếu do hệ điều hành hoặc Hệ quản trị CSDL quản lý thay.
Hiệu năng & An toàn	<p>Hiệu năng: Yêu cầu cao, tối ưu chặt chẽ.</p> <p>An toàn: Cần xử lý bảo mật và ổn định nghiêm ngặt vì lỗi có thể sập hệ thống.</p>	<p>Hiệu năng: Ít yêu cầu tối ưu hệ thống hơn .</p> <p>An toàn: Ít rủi ro ảnh hưởng toàn bộ hệ thống.</p>
Ví dụ minh họa	Windows Service, Process Manager, Driver.	Ứng dụng Desktop, Web, Mobile, phần mềm quản lý.

**Câu 3: Giải thích mối quan hệ giữa hệ điều hành và chương trình ứng dụng.**

**Theo bạn, tại sao ứng dụng không thể làm việc trực tiếp với phần cứng mà phải thông qua hệ điều hành?**

*Mối quan hệ giữa hệ điều hành và chương trình ứng dụng:*

- Hệ điều hành đóng vai trò trung gian giữa phần cứng và các lớp phần mềm phía trên.
- Theo sơ đồ phân cấp: Người dùng → Ứng dụng → Lập trình ứng dụng → Lập trình hệ thống → Hệ điều hành → Phần cứng.
- Ứng dụng (App) chạy trong User Mode, trong khi nhân hệ điều hành chạy trong Kernel Mode.

*Ứng dụng không làm trực tiếp với phần cứng:*

Hệ điều hành chia môi trường thực thi thành 2 chế độ để cân bằng giữa Độ an toàn và Sức mạnh:

- Vấn đề an toàn: Nếu cho phép ứng dụng truy cập trực tiếp phần cứng, một lỗi nhỏ trong ứng dụng có thể làm sập toàn bộ hệ thống.
- Sự giới hạn cần thiết: Bằng cách bắt buộc ứng dụng chạy ở User Mode, quyền hạn bị giới hạn và không thể truy cập trực tiếp phần cứng. Khi ứng dụng cần tài nguyên, nó phải gửi yêu cầu (System Call) xuống HĐH để đảm bảo an toàn.

**Câu 4: Sử dụng C#/.NET, hãy viết một chương trình đơn giản để:**

**In ra thông tin cơ bản về môi trường hệ thống (phiên bản hệ điều hành, thư mục hiện tại, thời gian hệ thống). Sinh viên giải thích chương trình đang sử dụng dịch vụ nào của hệ điều hành.**

**Chương trình:** [https://github.com/Tommyhuy1705/System\\_Programming.git](https://github.com/Tommyhuy1705/System_Programming.git)

#### *1. Dịch vụ Quản lý Tài nguyên Phần cứng*

Dịch vụ Quản lý Tiến trình & Luồng (Process & Thread Management):

- Lệnh: `Environment.ProcessorCount`

→ Cho biết số lượng bộ xử lý logic để quyết định số lượng luồng (Threads) cần tạo nhằm tối ưu hóa hiệu năng xử lý song song, tránh lãng phí tài nguyên hoặc gây tắc nghẽn.

Dịch vụ Quản lý Bộ nhớ (Memory Management):

- Lệnh: `Environment.SystemPageSize`

→ Truy vấn kích thước của một "trang nhớ" (page) mà HĐH đang sử dụng để quản lý bộ nhớ ảo (Virtual Memory).

Dịch vụ Thời gian hệ thống (System Timer):

- Lệnh: `Environment.TickCount64`

→ Sử dụng bộ đếm thời gian của phần cứng (hardware timer) do HĐH quản lý để biết thời gian uptime của hệ thống.

## *2. Dịch vụ Hệ thống Tập tin & I/O (File System & I/O Services)*

- Lệnh: `DriveInfo.GetDrives()`, `drive.TotalSize`, `drive.AvailableFreeSpace`

→ Gọi API của hệ điều hành để yêu cầu trình điều khiển thiết bị (Device Driver) báo cáo danh sách các phân vùng, định dạng (NTFS/FAT32) và dung lượng trống. Đây là sự tương tác mức thấp với thiết bị lưu trữ.

- Lệnh: `Environment.CurrentDirectory`

→ Truy xuất thông tin đó từ khối quản lý tiến trình (Process Control Block - PCB).

## *3. Dịch vụ Thông tin Môi trường (System Configuration Services)*

- Lệnh: `Environment.OSVersion`, `Environment.MachineName`,  
`Environment.UserName`

→ Truy xuất các biến môi trường và thông tin cấu hình mà HĐH cung cấp khi khởi tạo phiên làm việc (Session).

- Lệnh: `Environment.SystemDirectory`

→ Xác định vị trí các file hệ thống quan trọng (thường là `System32`).

**Câu 5: Hãy nêu 3 ví dụ bạn cần sử dụng lập trình hệ thống trong quá trình phát triển ứng dụng hoặc phần mềm. Giải thích vì sao các ứng dụng đó cần làm việc gần với hệ điều hành và quản lý tài nguyên hệ thống.**

### **1. Windows Service**

- Một chương trình tự động sao lưu dữ liệu (Backup Service) chạy ngầm mỗi đêm.

- Đây là loại "Software-facing software" chạy background (không có UI), yêu cầu phải chạy 24/7 và cần độ ổn định cực cao. Nó cần tương tác sâu với File System và lập lịch của HĐH.

## **2. Ứng dụng giám sát hiệu năng (Task Manager/Process Monitor)**

- Phần mềm hiển thị CPU, RAM đang được sử dụng.
- Cần truy cập vào thông tin quản lý Tiến trình (Process) và Bộ nhớ (Memory) mà HĐH đang quản lý. Ứng dụng web/mobile thông thường không thể đọc được dữ liệu này.

## **3. Trình điều khiển thiết bị (Device Driver)**

- Phần mềm giúp máy tính nhận diện một loại máy in chuyên dụng mới.
- Cần làm việc ở mức thấp nhất, gần như trực tiếp với phần cứng và Kernel Mode để chuyển đổi tín hiệu từ phần mềm sang tín hiệu phần cứng.