BÀI 8: KIẾN TRÚC LINUX KERNEL

Kernel Architecture & Kconfig

🎯 Mục tiêu bài học

1. **Kiến trúc:** Hiểu mô hình Monolithic Kernel của Linux.
2. **Bản đồ kho báu:** Nắm vững cấu trúc thư mục mã nguồn (`arch` , `drivers` , `fs` ...).
3. **Cơ chế Kbuild:** Hiểu mối quan hệ giữa `Kconfig` , `.config` và `Makefile` .
4. **Thực hành:** Sử dụng thành thạo `make menuconfig` để tùy biến nhân.

1. Linux Kernel là gì?

Là thành phần cốt lõi (Core) nằm giữa Phần cứng (Hardware) và Ứng dụng (User Space).

5 Nhiệm vụ chính:

1. **Quản lý Tiến trình (Process Scheduler):** Chia sẻ CPU cho các app.
2. **Quản lý Bộ nhớ (Memory Management):** Cấp phát RAM, Virtual Memory.
3. **Hệ thống File (VFS):** Đọc/ghi dữ liệu (ext4, fat32).
4. **Mạng (Network Stack):** TCP/IP, WiFi, Bluetooth.
5. **Trình điều khiển thiết bị (Device Drivers):** Giao tiếp phần cứng.

2. Kiến trúc Monolithic vs Modular

Linux là **Monolithic Kernel** (Nhân nguyên khối), nhưng có tính năng **Modular** (Module hóa).

- **Monolithic:** Toàn bộ OS (Scheduler, Driver, FS) chạy chung trong một không gian địa chỉ duy nhất (Kernel Space).
  - *Ưu điểm:* Tốc độ cực nhanh (gọi hàm trực tiếp).
  - *Nhược điểm:* Một driver lỗi có thể làm treo cả hệ thống (Kernel Panic).
- **Modular:** Có thể nạp thêm hoặc gỡ bỏ tính năng (Driver) khi hệ thống đang chạy (`.ko` files) mà không cần khởi động lại.

3. Bản đồ Source Code (Source Tree)

Khi giải nén source kernel (`linux-x.y.z`), bạn sẽ thấy:

- `arch/` : Code riêng cho từng CPU (arm, arm64, x86, riscv).
- `drivers/` : Chiếm 60-70% dung lượng. Chứa code điều khiển phần cứng.
- `fs/` : Các định dạng file (ext4, fat, nfs).
- `include/` : Các file Header (`.h`) dùng chung.
- `init/` : Code khởi động kernel (`main.c`).
- `kernel/` : Phần lõi (Scheduler, Signal).
- `mm/` : Quản lý bộ nhớ (Memory Management).

4. Hệ thống cấu hình Kconfig

Làm sao để chọn driver nào được biên dịch? Đó là nhờ **Kconfig**.

3 Thành phần của Kbuild:

1. **File** `Kconfig` **:** Định nghĩa các menu và lựa chọn (nằm rải rác trong từng thư mục).
2. **File** `.config` **:** (File ẩn ở thư mục gốc) Lưu lại các lựa chọn của người dùng (Kết quả sau khi cấu hình).
3. **File** `Makefile` **:** Dựa vào `.config` để biết cần biên dịch file `.c` nào.

5. Menuconfig & Các trạng thái (Tristate)

Giao diện `make menuconfig` cho phép chọn 3 trạng thái cho một tính năng:

Ký hiệu	Trạng thái	Ý nghĩa	Kết quả biên dịch
`[*]`	**Built-in** (`y`)	Tính năng được tích hợp cứng vào nhân.	Nằm trong file `zImage` . Luôn có mặt khi boot.
`[M]`	**Module** (`m`)	Tính năng nằm trong file riêng.	Tạo ra file `.ko` . Cần lệnh `insmod` để nạp.
`[ ]`	**Exclude** (`n`)	Không sử dụng.	Không biên dịch.

🛠️ PHẦN THỰC HÀNH (LAB 08)

Cấu hình nhân Linux (Kernel Configuration)

Bước 1: Tải Source Kernel

Chúng ta sẽ dùng bản **Mainline** (hoặc bản Vendor nếu dùng board cụ thể).

```
# Tải bản ổn định (Longterm) - Ví dụ 5.10 hoặc 6.1
wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.1.80.tar.xz
# Giải nén
tar -xvf linux-6.1.80.tar.xz
cd linux-6.1.80
```

> **Lưu ý:** Dung lượng giải nén khoảng 1GB++.

Bước 2: Dọn dẹp & Thiết lập biến

```
# 1. Dọn dẹp (nếu đã từng build)
make mrproper

# 2. Thiết lập biến môi trường (Quan trọng!)
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-
```

Nếu quên `export ARCH=arm`, hệ thống sẽ mặc định cấu hình cho x86 (PC của bạn) -> Build sẽ lỗi.

Bước 3: Load cấu hình mặc định (Defconfig)

Không ai ngồi chọn từng cái trong hàng nghìn option. Ta dùng mẫu có sẵn.

- **Với BeagleBone (CPU AM335x):**

```
make omap2plus_defconfig
```

- **Với Raspberry Pi:**

```
make bcm2835_defconfig
```

- **Đa năng (Nhiều chip ARM v7):**

```
make multi_v7_defconfig
```

Bước 4: Tùy biến (Menuconfig)

```
make menuconfig
```

Nhiệm vụ Lab:

1. Bật tính năng: `Device Drivers` -> `LED Support` -> `LED Class Support`.
2. Tìm kiếm: Bấm phím `/` và gõ `USB_GADGET`. Xem nó đang nằm ở đâu.
3. Đổi tên Kernel: `General setup` -> `Local version` -> Thêm tên bạn (vd: `-ngocanh`).
4. Lưu lại (Save) -> Sẽ tạo ra file `.config`.

Bước 5: Kiểm tra kết quả

Mở file `.config` vừa tạo bằng text editor và kiểm tra xem các lựa chọn của bạn đã được lưu chưa.

```
vim .config  # Tìm dòng CONFIG_LOCALVERSION
```

> **Chú ý:** Đừng sửa tay file `.config`! Hãy dùng `make menuconfig` để đảm bảo các phụ thuộc (dependencies) được giải quyết đúng.

📝 Bài tập về nhà

1. Tìm hiểu sự khác biệt giữa `make menuconfig`, `make xconfig` và `make nconfig`.
2. Tìm file `Kconfig` trong thư mục `drivers/leds/` và đọc hiểu cấu trúc của nó (cú pháp `config`, `tristate`, `help`).
3. **Chuẩn bị cho Bài 9:** Tìm hiểu khái niệm "Device Tree" là gì? Tại sao Linus Torvalds từng tức giận vì code ARM lộn xộn trước khi có Device Tree?

Q & A

Hẹn gặp lại ở Bài 9: Device Tree!