

# **BÀI 2: DÒNG LỆNH & SHELL SCRIPTING**

## **Command Line Mastery & Automation**

---

# Mục tiêu bài học

1. **Thành thạo CLI:** Sử dụng mượt mà các lệnh quản lý file và quyền hạn.
2. **Tư duy tự động hóa:** Hiểu Shell Script là gì.
3. **Thực chiến:** Phân tích và viết script tự động build ( `auto_build.sh` ).
4. **Triển khai:** Tự động nạp code xuống board qua mạng (SCP/SSH).

# 1. Ôn tập: Các lệnh sinh tồn (CLI Survival)

Nhóm lệnh	Lệnh cơ bản	Tác dụng
Di chuyển	<code>pwd</code> , <code>cd</code> , <code>ls</code>	Xem vị trí, đổi thư mục, liệt kê file.
File	<code>cp</code> , <code>mv</code> , <code>rm</code>	Copy, Di chuyển, Xóa.
Thư mục	<code>mkdir</code> , <code>rmdir</code>	Tạo/Xóa thư mục.
Nội dung	<code>cat</code> , <code>grep</code>	Xem và tìm kiếm nội dung file.

“ **Thực hành:** Mở Terminal và kiểm tra phiên bản trình biên dịch:

`arm-linux-gnueabihf-gcc --version`

”

## 2. Quyền hạn (Permissions) & Thực thi

Trong Linux, một file muốn chạy được phải có quyền **Execute (x)**.

- Kiểm tra quyền: `ls -l auto_build.sh`
- Cấp quyền chạy: `chmod +x auto_build.sh`
- Chạy script: `./auto_build.sh`

“ Nếu không cấp quyền, bạn sẽ gặp lỗi: **Permission denied** ”

### 3. Shell Scripting: Code thực tế

Chúng ta sẽ phân tích file `code_examples/Lab_02_Scripting/auto_build.sh` có trong Project.

Cấu trúc cơ bản:

```
#!/bin/bash
# Shebang: Báo cho hệ thống biết dùng bash để chạy

set -e
# Quan trọng: Dừng ngay nếu có lỗi (Exit on error)

# Khai báo biến (Cấu hình)
CROSS_COMPILE="arm-linux-gnueabihf-"
TARGET_IP="192.168.1.100"
```

## 4. Hàm kiểm tra Toolchain (Function)

Trích đoạn từ `auto_build.sh` :

```
check_toolchain() {  
    echo "Checking cross-compiler..."  
    # command -v: Kiểm tra lệnh có tồn tại không  
    if ! command -v ${CROSS_COMPILE}gcc &> /dev/null; then  
        echo "Error: Cross-compiler not found!"  
        exit 1  
    fi  
    echo "Toolchain OK"  
}
```

“ **Bài học:** Luôn kiểm tra công cụ trước khi bắt đầu build.

”

# 5. Quy trình Build & Deploy tự động

Logic chính của script:

```
build_project() {  
    cd ../Lab_01_Hello  
    make clean  
    # Gọi makefile với tham số CROSS_COMPILE  
    make CROSS_COMPILE=${CROSS_COMPILE}  
}  
  
deploy_to_target() {  
    # Copy file xuống board qua SCP  
    scp hello root@${TARGET_IP}:/home/root/  
  
    # Chạy thử trên board qua SSH  
    ssh root@${TARGET_IP} "chmod +x /home/root/hello && /home/root/hello"  
}
```

# PHẦN THỰC HÀNH (LAB 02)

**Yêu cầu:** Chạy thử hệ thống build tự động.

1. Di chuyển vào thư mục Lab 2: `cd code_examples/Lab_02_Scripting`
2. Cấp quyền thực thi cho script: `chmod +x auto_build.sh`
3. Chạy script: `./auto_build.sh`

“ **Lưu ý:** Nếu bạn chưa có board thật, script sẽ dừng ở bước deploy (do không ping thấy `192.168.1.100` ). Hãy quan sát thông báo lỗi màu đỏ/vàng. ”





## Bài tập mở rộng

Sửa file `auto_build.sh` để thêm các tính năng sau:

1. Thêm biến `BUILD_DIR` để gom các file rác ( `.o` , `.bin` ) vào một chỗ cho gọn.
2. Tự động tạo file nén `release.tar.gz` sau khi build xong.
3. Thêm timestamp (ngày giờ) vào thông báo log.

# Q & A

## Hẹn gặp lại ở Bài 3: Advanced C & Makefile!