

BÀI 7: THỰC HÀNH U-BOOT

U-Boot Porting, Compilation & Commands

Mục tiêu bài học

1. **Source Code:** Tải và hiểu cấu trúc thư mục của U-Boot.
2. **Configuration:** Biết cách chọn cấu hình cho Board (`make _defconfig`).
3. **Compilation:** Biên dịch ra file `MLO` và `u-boot.img` .
4. **Deployment:** Phân vùng thẻ nhớ và nạp Bootloader.
5. **Usage:** Sử dụng thành thạo các lệnh U-Boot (`setenv` , `tftp` , `bootz`).

1. Chuẩn bị Source Code

U-Boot là dự án mã nguồn mở, được quản lý tại denx.de.

Tải source code (Mainline):

```
git clone https://github.com/u-boot/u-boot.git
cd u-boot
# Checkout về một phiên bản ổn định (VD: v2021.01)
git checkout v2021.01
```

Cấu trúc thư mục quan trọng:

- **configs/** : Chứa file cấu hình mặc định cho từng board (***_defconfig**).
- **board/** : Code khởi tạo riêng cho từng board (RAM, Pinmux).
- **include/configs/** : File **.h** định nghĩa biến môi trường mặc định.

2. Quy trình Build (3 Bước thần thánh)

Để build U-Boot cho **BeagleBone Black** (CPU AM335x):

Bước 1: Dọn dẹp (Clean)

```
make distclean
```

Bước 2: Cấu hình (Configure)

Tìm file config trong **configs/** folder. Với BBB, nó là **am335x_evm_defconfig**.

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- am335x_evm_defconfig
```

Bước 3: Biên dịch (Compile)

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j4
```

3. Kết quả đầu ra (Output Artifacts)

Sau khi build xong, bạn cần quan tâm 2 file:

1. **MLO** (Multimedia LOader): Chính là SPL.
 - File này phải nhỏ (dưới 100KB) để vừa SRAM.
 - ROM Code sẽ tìm file tên là **MLO** đầu tiên.
2. **u-boot.img** : Chính là U-Boot Stage 3.
 - Chứa đầy đủ tính năng (CLI, Net, USB...).

“ **Lưu ý:** Với Raspberry Pi, file output thường là **u-boot.bin** và cách nạp sẽ khác (copy đè lên **kernel7.img**). ”

4. Chuẩn bị Thẻ nhớ (SD Card)

ROM Code của chip OMAP/Sitara yêu cầu thẻ nhớ phải chuẩn:

- **Partition 1:** Định dạng **FAT32**, cờ **Bootable**.
- **Partition 2:** Định dạng **EXT4** (để chứa RootFS sau này).

Công cụ: Dùng **GParted** (GUI) hoặc **fdisk** (CLI).

“ **Quan trọng:** File **MLO** phải được copy vào thẻ nhớ **ĐẦU TIÊN** để đảm bảo nó nằm ở các sector đầu tiên (ROM Code mới đọc được). ”

5. Các lệnh U-Boot cơ bản (CLI)

Khi board khởi động, nhấn phím bất kỳ để vào chế độ lệnh => .

Nhóm lệnh	Lệnh	Tác dụng
Biến môi trường	<code>printenv</code>	In danh sách biến.
	<code>setenv name value</code>	Đặt giá trị biến (RAM).
	<code>saveenv</code>	Lưu biến xuống Flash/SD (Vĩnh viễn).
Bộ nhớ/File	<code>fatls mmc 0:1</code>	Liệt kê file trong thẻ nhớ.
	<code>fatload mmc 0:1 <addr> <file></code>	Load file từ thẻ nhớ vào RAM.
Boot	<code>bootz <addr> - <dtb_addr></code>	Boot Kernel Linux.

6. Biến môi trường quan trọng

- **bootdelay** : Thời gian đếm ngược (giây) trước khi tự boot.
- **bootcmd** : "Kịch bản" tự động chạy khi hết giờ đếm ngược.
 - Ví dụ: **fatload mmc 0 zImage; bootz ...**
- **bootargs** : Tham số truyền cho Kernel (Quan trọng!).
 - Ví dụ: **console=ttyS0,115200 root=/dev/mmcblk0p2 rw**
 - Ý nghĩa: "Kernel ơi, hãy in log ra cổng UART0, và tìm RootFS ở phân vùng 2 thẻ nhớ".

PHẦN THỰC HÀNH (LAB 07)

Build & Run U-Boot trên Board thật

Lab 07: Các bước thực hiện

1. **Cài đặt:** `sudo apt install libssl-dev bison flex swig python3-dev` .
2. **Build:** Thực hiện 3 bước build cho board của bạn (BBB/RPi).
3. **Format thẻ:** Tạo 1 phân vùng FAT32 (100MB).
4. **Nạp:**
 - Copy `MLO` vào thẻ nhớ.
 - Copy `u-boot.img` vào thẻ nhớ.
5. **Test:** Cắm thẻ vào Board, kết nối UART, cấp nguồn.
 - *Kết quả mong đợi:* Thấy log U-Boot in ra màn hình Console.

Thử thách: Boot qua mạng (TFTP)

Thay vì copy thẻ nhớ, hãy load file từ máy tính qua dây mạng.

1. Cài TFTP Server trên Ubuntu: `sudo apt install tftpd-hpa` .
2. Trên U-Boot, cấu hình IP:

```
setenv ipaddr 192.168.1.10 # IP của Board  
setenv serverip 192.168.1.2 # IP của Ubuntu
```

3. Load file thử:

```
tftp 0x82000000 zImage_test
```



Bài tập về nhà

1. Tạo một **Boot Script** (`boot.scr`):

- Viết file text chứa các lệnh U-Boot.
- Dùng tool `mkimage` để biên dịch thành `boot.scr` .
- Cấu hình để U-Boot tự chạy script này.

2. Tìm hiểu lệnh `md` (Memory Display) và `mw` (Memory Write) để thao tác trực tiếp RAM.

3. Chuẩn bị cho **Bài 8**: Tìm hiểu trước về `make menuconfig` của Kernel.

Q & A

Hẹn gặp lại ở Bài 8: Linux Kernel Architecture!