Custom Embedded Linux OS Development for Raspberry Pi 2B

1. Project Objective

The objective of this project is to **build a minimal custom embedded Linux operating system** for the **Raspberry Pi 2B** from scratch, in order to understand:

- Linux boot process
- Kernel configuration and compilation
- Root filesystem creation
- BusyBox integration
- System bring-up using QEMU (hardware-independent testing)

This project focuses on **system-level understanding**, not just application development.

2. Target Platform Selection

Hardware

- **Board:** Raspberry Pi 2 Model B
- **CPU:** ARM Cortex-A7 (ARMv7)
- **Architecture:** 32-bit ARM

3. Development Environment Setup

Host System

- Linux PC (Ubuntu/Debian)

Required Tools

- ARM cross compiler (arm-linux-gnueabihf-gcc)
- GNU Make
- QEMU (ARM system emulator)
- Git

Why Cross Compilation?

The target hardware (ARM) is different from the host machine (x86_64) or (AMD). Cross compilation allows building ARM binaries on a faster x86 system.

4. Understanding the Linux Boot Flow

Before implementation, the Linux boot flow was studied:

```
Bootloader (optional in QEMU)
            ↓
Linux Kernel (zImage)
       ↓
Device Tree Blob (DTB)
 ↓
Initramfs / Root Filesystem
 ↓
/init → Shell
```

For early bring-up and learning, **U-Boot was skipped initially** to reduce complexity.

5. Linux Kernel Configuration and Build

5.1 Kernel Source Selection

- Source: Raspberry Pi Linux kernel
- Reason: Includes Raspberry Pi-specific drivers and device trees

5.2 Kernel Configuration

Command used:

```
make bcm2709_defconfig
```

Why bcm2709_defconfig?

- BCM2709 is the SoC used in Raspberry Pi 2B
- Loads a known-working base configuration
- Avoids manual configuration of thousands of options

5.3 Required Kernel Options Enabled

Initramfs Support

```
General Setup → Initial RAM filesystem support
```

Why:
To allow booting a root filesystem passed via QEMU using -initrd.

devtmpfs

```
Device Drivers → Generic Driver Options
           - Maintain devtmpfs
           - Automount devtmpfs
```

Why:
Automatically creates /dev nodes (required for shell and devices).

Serial Console (PL011)

```
Device Drivers → Character Devices → Serial Drivers
- ARM AMBA PL011
```

5.4 Kernel Build

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage dtbs
```

Output Files

- zImage → compressed ARM kernel
- bcm2709-rpi-2-b.dtb → hardware description

6. Root Filesystem (RFS) Design

6.1 Why BusyBox?

BusyBox provides:

- Shell (/bin/sh)
- Core utilities (ls, mount, echo)
- Minimal size
- Ideal for embedded systems

6.2 Static BusyBox Build

Configuration:

```
Build static binary (no shared libs)
```

Why static build?

- No dependency on glibc or shared libraries
- Smaller and simpler root filesystem
- Reliable for early bring-up

6.3 Root Filesystem Structure

Directories created:

```
/bin
/sbin
/etc
/proc
/sys
/dev
/init
```

This follows standard Linux filesystem hierarchy.

6.4 Init Script (/init)

```
#!/bin/sh
mount -t proc none /proc
mount -t sysfs none /sys
mount -t devtmpfs none /dev

echo "Welcome MoniOS"
exec /bin/sh
```

Why /init?

- First user-space program executed by kernel
- Responsible for mounting virtual filesystems
- Starts the user shell

7. Initramfs Creation

Command:

```
find . | cpio -o -H newc > rootfs.cpio
```

Why cpio newc?

- Required format for Linux initramfs
- Kernel can unpack it directly into RAM
- Faster than block-based rootfs

8. Booting Using QEMU

Machine Selection

```
-M raspi2b
```

Why QEMU?

- Hardware-independent testing
- Faster debug cycle
- No risk of SD card corruption

Final Boot Command

```
qemu-system-arm \
-M raspi2b \
-kernel zImage \
-dtb bcm2709-rpi-2-b.dtb \
-initrd rootfs.cpio \
-append "console=ttyAMA0 rdinit=/init" \
-nographic
```

9. Common Issues Encountered & Fixes

Issue: Kernel Panic – Unable to mount root fs

Cause: Incorrect use of root=/dev/ram0

Fix:
Removed root= and used rdinit=/init for initramfs boot.

Issue: BusyBox SHA hardware acceleration error

Cause: x86-specific crypto optimizations enabled

Fix:
Disabled SHA hardware acceleration in BusyBox configuration.

Issue: Missing .config

Cause: Kernel not configured or config lost

Fix:
Re-ran make bcm2709_defconfig and saved configuration properly.

10. Final Result

The system successfully boots into a minimal Linux shell:

```
Welcome MoniOS
/ #
```