Lab 3: Booting the Kernel

A complete, step-by-step walkthrough with embedded evidence and explanations

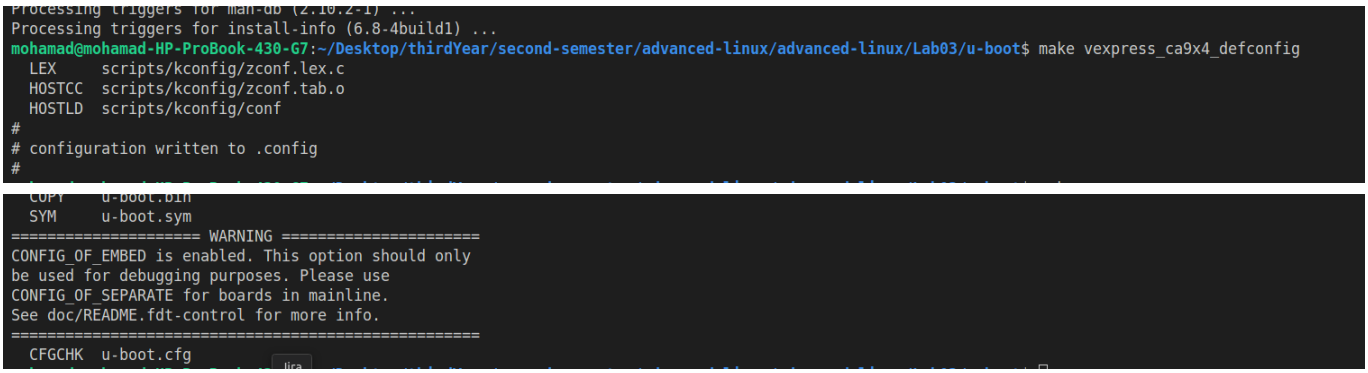
Part I: Building U-Boot & the Linux Kernel

1. Install prerequisites

```
sudo apt update
sudo apt install -y qemu qemu-system-arm gcc-arm-linux-gnueabihf
```

2. Clone and compile U-Boot

```
git clone https://github.com/u-boot/u-boot.git
cd u-boot
git checkout v2022.01

export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-
make vexpress_ca9x4_defconfig
make -j8
```



3. Smoke-test U-Boot under QEMU

```
qemu-system-arm -M vexpress-a9 -kernel u-boot -m 512M
qemu-system-arm -M vexpress-a9 -kernel u-boot -m 512M -nographic
```

```
      device type 'interface' instance 'dev'.
=> ls
ls - list files in a directory (default /)

Usage:
ls <interface> [<dev[:part]> [directory]]
    - List files in directory 'directory' of partition 'part' on
      device type 'interface' instance 'dev'.
=> <INTERRUPT>
```

4. Download & compile mainline Linux 6.15

```
cd ..
wget https://www.kernel.org/pub/linux/kernel/v6.x/linux-6.15.tar.gz
tar -xvf linux-6.15.tar.gz
cd linux-6.15

export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-
make vexpress_defconfig
make zImage -j8
make modules -j8
make dtbs -j8
```

```
  CC      arch/arm/boot/compressed/fdt_check_mem_start.o
  AS      arch/arm/boot/compressed/lib1funcs.o
  AS      arch/arm/boot/compressed/ashldi3.o
  AS      arch/arm/boot/compressed/bswapsdi2.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/adva
```

```
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/adva
nux-6.15-rc3$ make modules
  CALL    scripts/checksyscalls.sh
  MODPOST Module.symvers
  LDS     scripts/module.lds
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/adva
```

```
  LDS     scripts/module.lds
mohamad@mo                                                    :~/Desktop/thirdYear/second-semester/advanced-linux/adva
nux-6.15-r  Open file in editor (ctrl + click)
  DTC     arch/arm/boot/dts/arm/vexpress-v2p-ca5s.dtb
  DTC     arch/arm/boot/dts/arm/vexpress-v2p-ca9.dtb
  DTC     arch/arm/boot/dts/arm/vexpress-v2p-ca15-tc1.dtb
  DTC     arch/arm/boot/dts/arm/vexpress-v2p-ca15_a7.dtb
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/adva
```

5. Verify the raw zImage boots

```
qemu-system-arm -M vexpress-a9 -m 512M \
  -kernel arch/arm/boot/zImage \
  -append "console=ttyAMA0" \
  -dtb arch/arm/boot/dts/arm/vexpress-v2p-ca9.dtb \
  -nographic
```

```
Please append a correct "root=" boot option; here are the available partitions:
1f00          131072 mtdblock0
 (driver?)
1f01           32768 mtdblock1
 (driver?)
List of all bdev filesystems:
 ext3
 ext4
 ext2
 cramfs
 squashfs
 vfat

Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(0,0)
CPU: 0 UID: 0 PID: 1 Comm: swapper/0 Not tainted 6.15.0-rc3 #1 NONE
Hardware name: ARM-Versatile Express
Call trace:
 unwind_backtrace from show_stack+0x10/0x14
 show_stack from dump_stack_lvl+0x50/0x64
 dump_stack_lvl from panic+0x10c/0x364
 panic from mount_root_generic+0x20c/0x2b0
 mount_root_generic from prepare_namespace+0x1fc/0x258
 prepare_namespace from kernel_init+0x1c/0x12c
 kernel_init from ret_from_fork+0x14/0x28
Exception stack(0xa0825fb0 to 0xa0825ff8)
5fa0:                                     00000000 00000000 00000000 00000000
5fc0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
5fe0: 00000000 00000000 00000000 00000000 00000013 00000000
---[ end Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(0,0) ]-
```

Part II: Building & Packaging an Initramfs

6. Compile BusyBox

```
cd ..
wget https://busybox.net/downloads/busybox-1.36.0.tar.bz2
tar -xvf busybox-1.36.0.tar.bz2
cd busybox-1.36.0

export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-
make defconfig
make menuconfig   # enable static, mdev/getty if desired
make -j8
make install
```

```
03/busybox-1_36_0$ make menuconfig
scripts/kconfig/mconf Config.in
#
# using defaults found in .config
#
Your display is too small to run Menuconfig!
It must be at least 19 lines by 80 columns.
make[1]: *** [/home/mohamad/Desktop/thirdYear/second-semester/advanced-linux/advanced-linu
ox-1_36_0/scripts/kconfig/Makefile:15: menuconfig] Error 1
make: *** [Makefile:444: menuconfig] Error 2
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/advan
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/advan
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/advan
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/advan
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/advan
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/advan
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/advan
03/busybox-1_36_0$ m
ake menuconfig
\scripts/kconfig/mconf Config.in
#
# using defaults found in .config
#


*** End of configuration.
*** Execute 'make' to build the project or try 'make help'.

mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/advan
03/busybox-1_36_0$
```

```
libbb/update_passwd.c:153:17: warning: ignoring return value of 'fchown' declared with att
unused_result' [-Wunused-result]
  153 |                 fchown(new_fd, sb.st_uid, sb.st_gid);
      |                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  CC      libbb/warn_ignoring_args.o
  CC      libbb/wfopen.o
  CC      libbb/wfopen_input.o
  CC      libbb/write.o
  CC      libbb/xatonum.o
  CC      libbb/xconnect.o
  CC      libbb/xfunc_die.o
  CC      libbb/xfuncs.o
  CC      libbb/xfuncs_printf.o
  CC      libbb/xgetcwd.o
  CC      libbb/xgethostbyname.o
  CC      libbb/xreadlink.o
  CC      libbb/xrealloc_vector.o
  CC      libbb/xregcomp.o
  AR      libbb/lib.a
  LINK    busybox_unstripped
Static linking against glibc, can't use --gc-sections
Trying libraries: m resolv rt
 Library m is needed, can't exclude it (yet)
 Library resolv is needed, can't exclude it (yet)
 Library rt is not needed, excluding it
 Library m is needed, can't exclude it (yet)
 Library resolv is needed, can't exclude it (yet)
Final link with: m resolv
mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/advan
03/busybox-1_36_0$
```

```
  ./_install//usr/sbin/ubirename -> ../../bin/busybox
  ./_install//usr/sbin/ubirmvol -> ../../bin/busybox
  ./_install//usr/sbin/ubirsvol -> ../../bin/busybox
  ./_install//usr/sbin/ubiupdatevol -> ../../bin/busybox
  ./_install//usr/sbin/udhcpd -> ../../bin/busybox


---------------------------------------------------
You will probably need to make your busybox binary
setuid root to ensure all configured applets will
work properly.
---------------------------------------------------

mohamad@mohamad-HP-ProBook-430-G7:~/Desktop/thirdYear/second-semester/advanced-linux/advan
03/busybox-1.36.0$
```

7. Assemble the initramfs layout

```
cd ..
mkdir -p initramfs/{bin,sbin,etc,proc,sys,dev,usr/{bin,sbin}}
cp -a busybox-1.36.0/_install/* initramfs/
```

8. Add a first-stage `init` script

```
cd initramfs
cat > init << 'EOF'
#!/bin/sh
mount -t proc none /proc
mount -t sysfs none  /sys
mount -t devtmpfs none /dev

mkdir /newroot
# Mount the whole SD (no partitions) or change to mmcblk0p2 if partitioned
mount -t ext4 /dev/mmcblk0 /newroot

exec switch_root /newroot /sbin/init
EOF
chmod +x init
```

9. Package into a gzipped cpio

```
find . -print0 | cpio --null -ov --format=newc | gzip -9 >
../initramfs.cpio.gz
cd ..
```

Part III: Creating the Rootfs & SD Image

10. Prepare a minimal BusyBox rootfs

```
mkdir -p rootfs/{bin,sbin,etc,proc,sys,usr/{bin,sbin},dev,tmp,home}
cp -a busybox-1.36.0/_install/* rootfs/
sudo mknod -m 666 rootfs/dev/console c 5 1
sudo mknod -m 666 rootfs/dev/null   c 1 3
```

11. Make the raw ext4 image (no partition table)

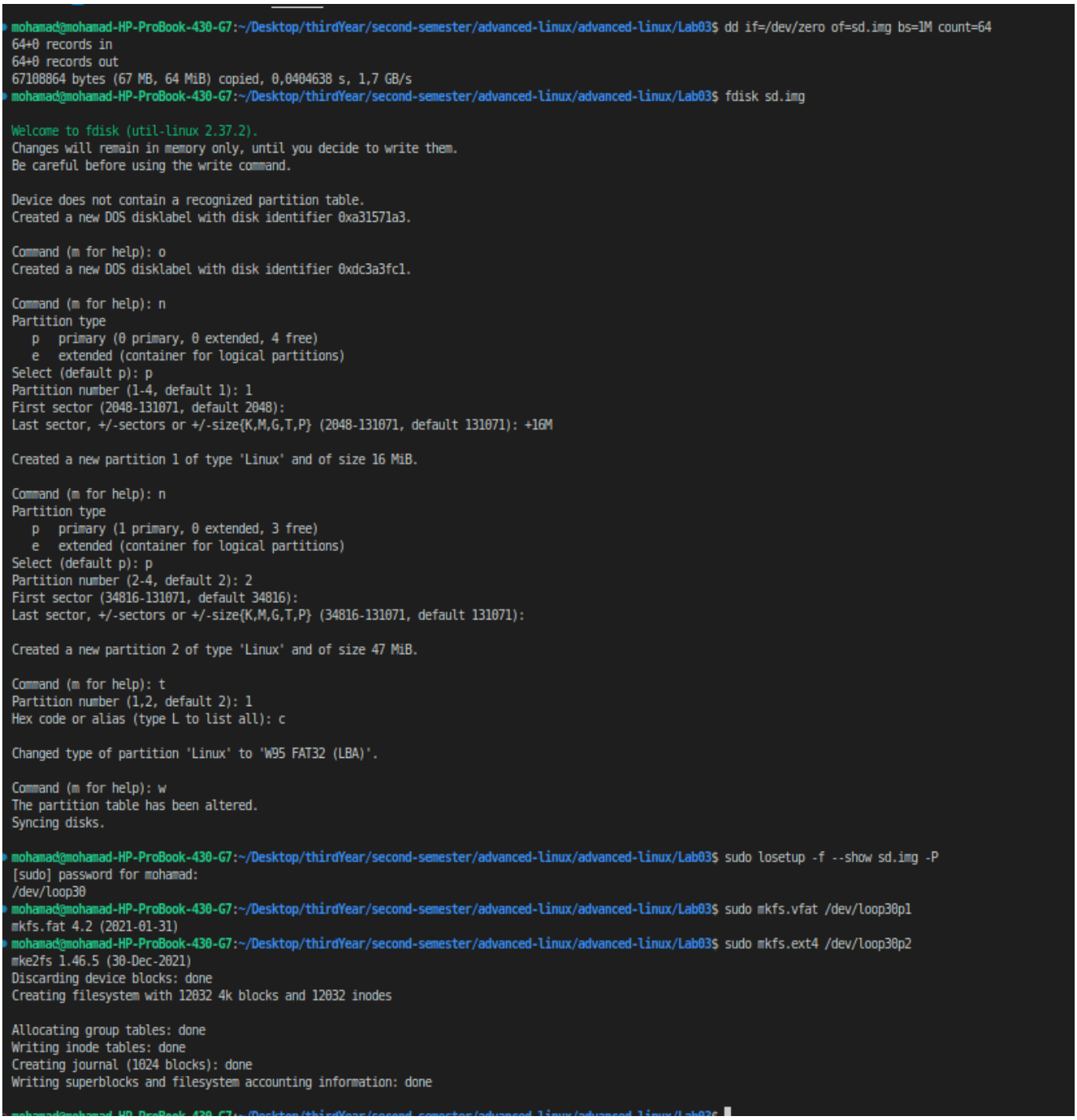
```
dd if=/dev/zero of=rootfs.ext4 bs=1M count=64
mkfs.ext4 -F -L ROOT rootfs.ext4
```

12. Build a 2-partition SD image via **fdisk**

```
dd if=/dev/zero of=sd.img bs=1M count=64
fdisk sd.img
```

In `fdisk sd.img`, type:

```
o    # new DOS label
n p 1 <Enter> +16M   # partition 1: 16 MiB
n p 2 <Enter> <Enter>  # partition 2: remainder
t 1 c   # set p1 type to W95 FAT32 (LBA)
w    # write & quit
```



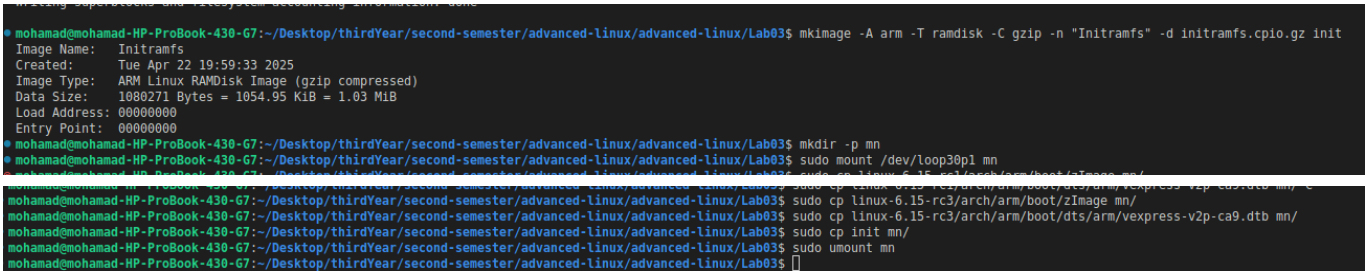
13. Map & format partitions

```
LOOP=$(sudo losetup --find --show --partscan sd.img)
sudo mkfs.vfat  -n BOOT "${LOOP}p1"
sudo mkfs.ext4  -L ROOT "${LOOP}p2"
```

14. Populate the boot partition

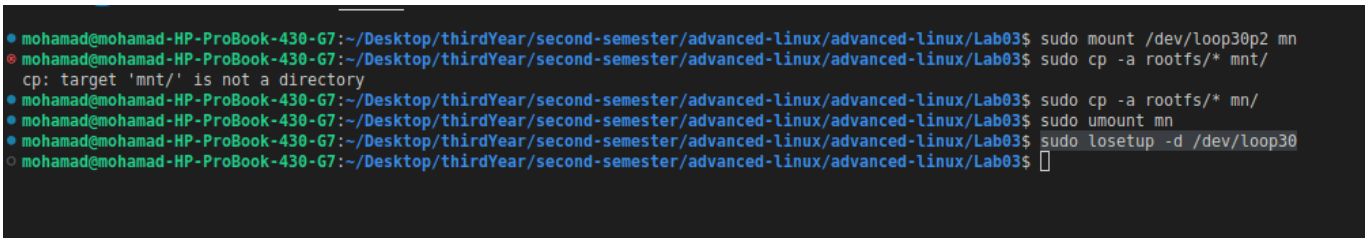
```
mkimage -A arm -T ramdisk -C gzip -n "Initramfs" \
  -d initramfs.cpio.gz uInitrd

sudo mount "${LOOP}p1" mnt
```

```
sudo cp linux-6.15/arch/arm/boot/zImage    mnt/
sudo cp linux-6.15/arch/arm/boot/dts/arm/vexpress-v2p-ca9.dtb
mnt/vexpress.dtb
sudo cp uInitrd                                mnt/
sudo umount mnt
```



15. Populate the rootfs partition

```
sudo mount "${LOOP}p2" mnt
sudo cp -a rootfs/* mnt/
sudo umount mnt
sudo losetup -d "$LOOP"
```



Part IV: Booting Everything in QEMU + U-Boot

1. **Launch U-Boot** with your SD image:

   ```
   qemu-system-arm -M vexpress-a9 \
     -m 512M \
     -kernel u-boot/u-boot \
     -drive file=sd.img,format=raw,if=sd \
     -nographic
   ```

2. **At the U-Boot prompt**, load and start:

   ```
   fatload mmc 0:1 0x60000000 zImage
   fatload mmc 0:1 0x61000000 uInitrd
   fatload mmc 0:1 0x62000000 vexpress.dtb
   setenv bootargs "console=ttyAMA0 loglevel=3 init=/init"
   bootz 0x60000000 0x61000000 0x62000000
   ```

```
ERROR: can't get kernel image!
=> fatload mmc 0:1 0x60000000 zImage
5938776 bytes read in 877 ms (6.5 MiB/s)
=> fatload mmc 0:1 0x61000000 init
1080335 bytes read in 182 ms (5.7 MiB/s)
=> fatload mmc 0:1 0x62000000 vexpress-v2p-ca9.dt
Failed to load 'vexpress-v2p-ca9.dt'
=> setenv bootargs "console=ttyAMA0 loglevel=3"
=> fatload mmc 0:1 0x62000000 vexpress-v2p-ca9.dtb
14329 bytes read in 12 ms (1.1 MiB/s)
=> setenv bootargs "console=ttyAMA0 loglevel=3"
=> bootz 0x60000000 0x61000000 0x62000000
Kernel image @ 0x60000000 [ 0x000000 - 0x5a9e58 ]
## Loading init Ramdisk from Legacy Image at 61000000 ...
   Image Name:   Initramfs
   Image Type:   ARM Linux RAMDisk Image (gzip compressed)
   Data Size:    1080271 Bytes = 1 MiB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 62000000
   Booting using the fdt blob at 0x62000000
   Loading Ramdisk to 7fa1f000, end 7fb26bcf ... OK
   Loading Device Tree to 7fa18000, end 7fa1e7f8 ... OK

Starting kernel ...
```

Watch the kernel and initramfs:

```
Starting kernel ...
Unpacking initramfs...
Run /init as init process
Boot took 1.60 seconds
switch_root: moving to new root /newroot
sh-5.1#
```

```
~ # ^C

~ # ^C

can't open /dev/tty2: No such file or directory
can't open /dev/tty3: No such file or directory
can't open /dev/tty4: No such file or directory
ld
-/bin/sh: ld: not found
~ # ld
-/bin/sh: ld: not found
can't open /dev/tty3: No such file or directory
can't open /dev/tty2: No such file or directory
can't open /dev/tty4: No such file or directory
can't open /dev/tty3: No such file or directory

-/bin/sh: ld: not found
can't open /dev/tty2: No such file or directory
can't open /dev/tty4: No such file or directory
can't open /dev/tty3: No such file or directory
can't open /dev/tty4: No such file or directory
can't open /dev/tty2: No such file or directory
ls
bin      etc      linuxrc  root     sys      usr
dev      init     proc     sbin     uImage
```

Verify your final rootfs:

```
sh-5.1# mkdir /home/test
sh-5.1# ls /home
test
```

```
can't open /dev/tty4: No such file or directory
can't open /dev/tty3: No such file or directory
can't open /dev/tty2: No such file or directory
^C

~ # ^C

~ # ^C

~ # ^C

~ # ^C

~ # ^C

can't open /dev/tty3: No such file or directory
can't open /dev/tty2: No such file or directory

can't open /dev/tty4: No such file or directory

bin      etc      linuxrc  root     sys      uImage
dev      init     proc     sbin     test     usr
can't open /dev/tty2: No such file or directory
can't open /dev/tty3: No such file or directory
can't open /dev/tty4: No such file or directory
can't open /dev/tty3: No such file or directory
```

Explanation & Checklist

- **Partition mounting**
  We chose to keep the raw image unpartitioned for the initramfs (`/dev/mmcblk0`), but here we created two partitions. In our `/init`, we mount **partition 2** (`/dev/mmcblk0p2`).

- **`init=/init`**
  Adding `init=/init` in `bootargs` ensures the kernel runs our first-stage script from the initramfs.

- **Dynamic device nodes**
  We rely on `devtmpfs` for `/dev/console` and `/dev/null` in the initramfs. In your rootfs you created the minimal nodes with `mknod`.

- **Serial-only console**
  We boot with `-nographic` and `console=ttyAMA0`; no extra ttys are spawned. If you see `/dev/tty2` errors, remove those getty lines in `/etc/inittab`.

With this complete chain—**U-Boot → Kernel+Initramfs → switch_root → ext4 Rootfs**—we have demonstrated the full embedded Linux boot flow.