Linux Root Filesystems

Advanced Embedded Linux Development

with **Dan Walkes**

University of Colorado **Boulder**

Learning objectives:

Minimal Root Filesystems

Filesystem Hierarchy Standard

Busybox overview

Booting Our QEMU Kernel

- QEMU_AUDIO_DRV=none qemu-system-arm -m 256M -nographic -M versatilepb -kernel zImage
  - What is missing?

```
[ 1.886379] Kernel panic - not syncing: VFS: Unable to mount root fs on
unknown-block(0,0)
[ 1.895105] ---[ end Kernel panic - not syncing: VFS: Unable to mount root
fs on unknown-block(0, 0)
```

- Panic is because we need a root filesystem!

Root Filesystem - initramfs

- Option 1 - initramfs
  - set of files extracted into RAM, typically used to setup your real root filesystem
  - May load modules to support hardware needed to access the root filesystem.
  - We will use it for simplicity with QEMU for Assignment 3.

Root Filesystem

- Option 2 - block device
  - block device supported by the kernel, specified in a kernel command line with a root= parameter

Minimal Root Filesystem Contents

- init
  - starts the system - usually scripts
- shell
  - Runs shell scripts, handles command prompt
- daemons
  - background programs that provide services

Minimal Root Filesystem Contents

- shared libraries
- configuration files (/etc)
- device nodes (/dev)
- /proc and /sys pseudo filesystems
- kernel modules (/lib/modules/[kernel version])

Filesystem Hierarchy Standard

- Where does the kernel expect files to be located in the root filesystem?
  - It doesn't!
  - Only expects the init program, specified in command line
- What about programs?
  - Expectations Defined in the Filesystem Hierarchy Standard (FHS)

Filesystem Hierarchy Standard

- /bin - programs for all users, used at boot
- /dev - device nodes and other files
- /etc - system configuration files
- /lib - shared libraries
- /proc, /sys - proc and sysfs filesystem
- /sbin - programs for the system administrator, used at boot

Mastering Embedded Linux Programming Chapter 5
http://refspecs.linuxfoundation.org/fhs.shtml

Filesystem Hierarchy Standard

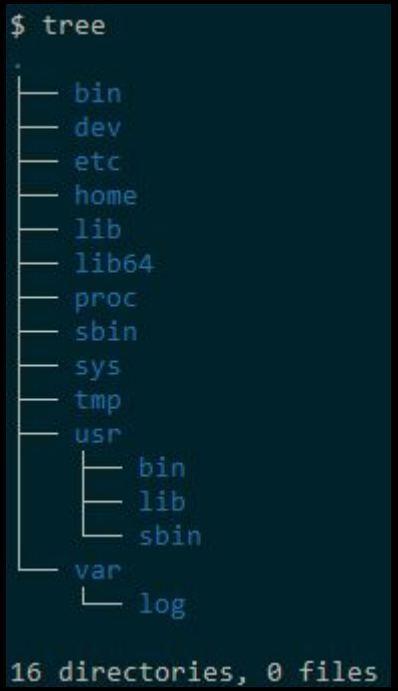
- /tmp - temporary files - can be deleted on boot
- /usr /usr/bin, /usr/sbin - additional programs libraries, utilities
- /var - files modified at runtime (/var/log) which need to be retained after boot

Mastering Embedded Linux Programming Chapter 5
http://refspecs.linuxfoundation.org/fhs.shtml

How do we create a rootfs?

- Start by creating a folder tree

```
mkdir -p bin dev etc home lib lib64 proc sbin sys tmp usr var
mkdir -p usr/bin usr/lib usr/sbin
mkdir -p var/log
```



What about the files?

- Now we've got the filesystem structure, where do we get the files we need to fill it?
- Use Busybox
- Single binary that implements essential Linux programs
- Symbolic links are used to tell which program is being requested

BusyBox Overview

- Single binary that implements essential Linux programs
- Symbolic links are used to tell which program is being requested

```
$ ls -l bin/cat bin/busybox
-rwxr-xr-x 1 root root 1025472 May 30 20:13 bin/busybox
lrwxrwxrwx 1 root root       7 May 30 20:13 bin/cat -> busybox
```

Creating Root Filesystem

- Make and install busybox:
  - make distclean
  - make defconfig
  - make ARCH=${ARCH} CROSS_COMPILE=${CROSS_COMPILE}
  - make CONFIG_PREFIX=/path/to/rootdir ARCH=${ARCH} CROSS_COMPILE=${CROSS_COMPILE} install

Creating Root Filesystem

- Add needed shared libraries from toolchain sysroot

```
$ aarch64-none-linux-gnu-readelf -a bin/busybox | grep "program interpreter"
      [Requesting program interpreter: /lib/ld-linux-aarch64.so.1]
$ aarch64-none-linux-gnu-readelf -a bin/busybox | grep "Shared library"
 0x0000000000000001 (NEEDED)             Shared library: [libm.so.6]
 0x0000000000000001 (NEEDED)             Shared library: [libresolv.so.2]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
```

- Program interpreter placed in “lib” directory
- Libraries placed in lib64 directory (since arch is 64 bit)

Devices

- Created with mknod (make node)
- mknod <name> <type> <major> <minor>
  - Null device is a known major 1 minor 3
  - Console device is known major 5 minor 1

```
sudo mknod -m 666 dev/null c 1 3
sudo mknod -m 600 dev/console c 5 1
```

Mastering Embedded Linux Programming Chapter 5
https://github.com/torvalds/linux/blob/master/Documentation/admin-guide/devices.txt

Make the contents owned by root

```
$ cd rootfs/
$ sudo chown -R root:root *
```

Why use root as owner?

- Your user account doesn't exist on the system we are creating
  - Note: you need to be sudo root to run this command

Using your rootfs with the target

- Ramdisk
  - Disk image loaded in RAM by bootloader
- Disk Image
  - Disk image for use with memory (like sdcard)
- Network File System (NFS)
- We will use a Ramdisk for assignment 3.

Use CPIO to create initramfs

```
cd "$OUTDIR/rootfs"
find . | cpio -H newc -ov --owner root:root > ${OUTDIR}/initramfs.cpio
```

```
gzip -f initramfs.cpio
```

Running in qemu

- Run the start-qemu-terminal script
  - References your kernel image and rootfs

You should see a root shell prompt after a boot-up delay