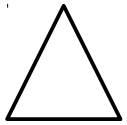
Kernel and user space

Chris Simmonds

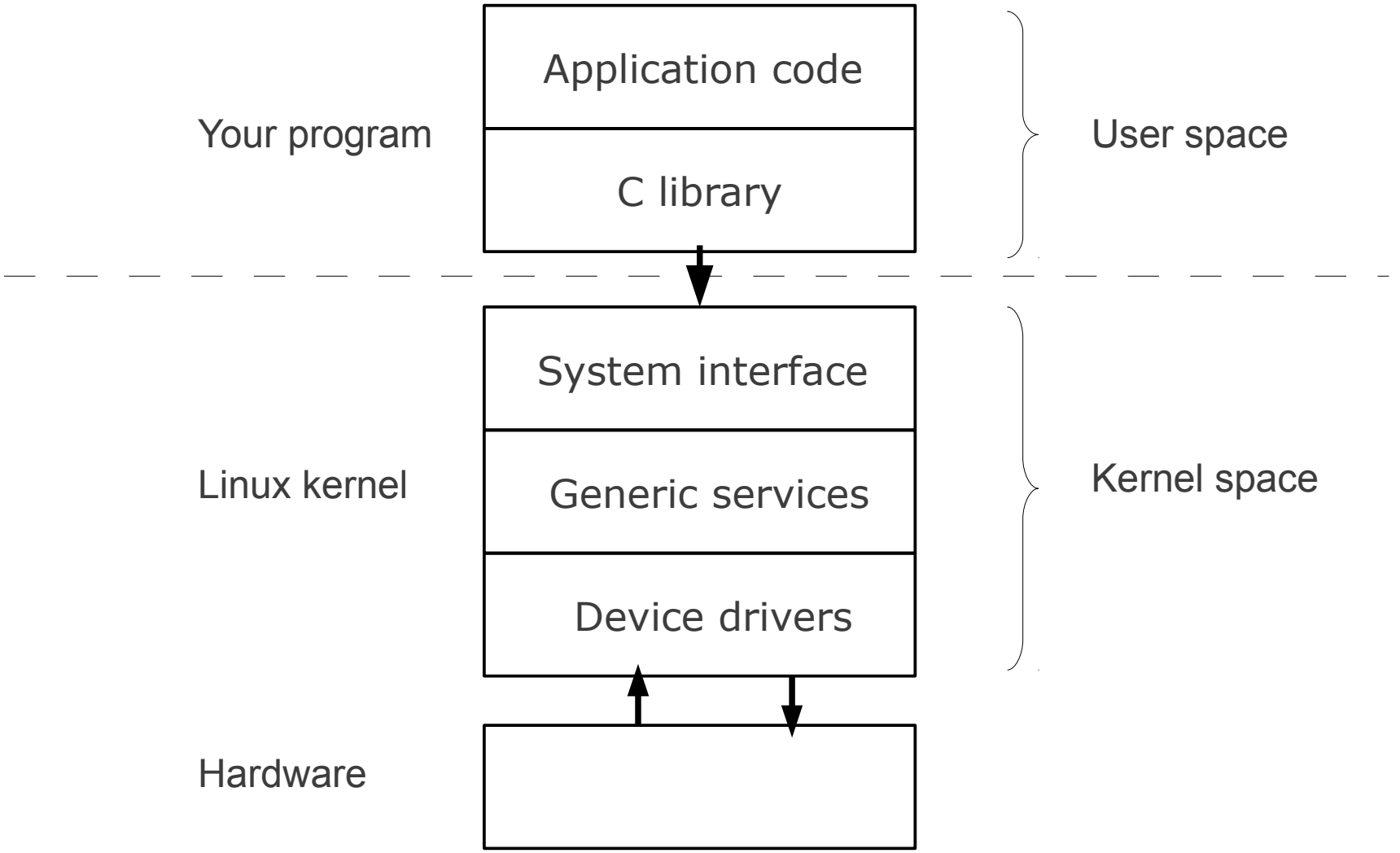
Embedded Linux Conference Europe 2011



Third element: kernel

- Version numbers
- About “BSPs”
- Configuring and cross compiling
- Booting

Kernel vs user space

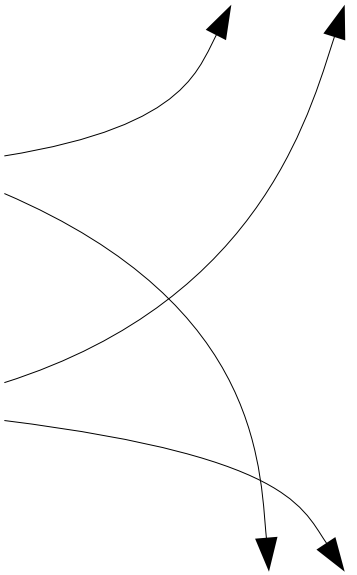


Kernel version numbers

Example 1 (up to May 2011): 2.6.39.1

Release number: changes every 12 weeks or so

Bug fix number: changes every time a bug is fixed, sometimes several times per week



Example 2 (from July 2011): 3.0.1

Bug fix releases

- Maintained by Greg Kroah-Hartman
- Serious bugs are fixed in the current stable version immediately
- Plus certain “stable” and “longterm” releases
  - Current releases (October 2011)
    - 2.6.27.59
    - 2.6.32.46
    - 2.6.33.19
    - 2.6.34.10
    - 2.6.35.14

Board Support Packages

- Mainline kernel works out-of-the-box for a number of development boards
  - e.g. Beagleboard
- But in most cases you will need a BSP from the board or chip vendor
  - Lags mainline by a few versions
  - Levels of support vary between vendors
- For custom boards you will have to write your own BSP

Levels of board support

- Architecture
  - arm,mips, powerpc, x86,...
- Chip (also known as System on Chip, SoC)
  - Atmel 91sam9, Freescale i.MX, TI OMAP, ...
- Board
  - SoC manufacturer evaluation boards
    - Freescale Babbage, TI EVM, ...
  - COTS boards
    - Digi, Eurotech, ...

Levels of board support (cont.)

- Chip level support mostly done by manufacturer
  - often in own kernel tree: e.g. Freescale
- Board level support done by board manufacturer
  - based on SoC kernel

BSP as a patch

- Often the BSP is a patch against a main-line kernel
- Typical procedure is

```
tar xjf linux-2.6.34.tar.bz2
cd linux-2.6.34
patch -p 1 < ../linux-2.6.34-some_bsp.patch
```

Kernel source from git

- Or, you may be directed to a git tree
  - which is more helpful because you will see the commit history and you will get updates as new commits are made
- Typical procedure is

```
git clone git://igloocommunity.org/git/kernel/igloo-kernel.git
git pull
```

Kernel configuration

- Typical kernel has >> 1000 configuration options
- Default configuration part of the BSP
- Tweak configuration using
  - make menuconfig (ncurses text menu)
  - make xconfig (graphical menus using Qt)
  - make gconfig (graphical menus using Gtk+)
- Files generated
  - .config
  - include/linux/autoconf.h

Building the kernel

- Set CROSS_COMPILE and ARCH

```
export ARCH=arm
export CROSS_COMPILE=arm-angstrom-linux-gnueabi-
```

- Make targets
  - zImage - compressed kernel image
  - uImage - zImage plus U-Boot header
- Files generated
  - vmlinux
  - arch/arm/boot/zImage
  - arch/arm/boot/uImage

Kernel command line

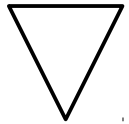
- Kernel behaviour set by “command line”
  - see Documentation/kernel-parameters.txt
- Some examples

  console: device to send kernel messages to, e.g.
  ```
  console=ttyS0,115200
  ```

  root: set device to load root file system from, e.g.
  ```
  root=/dev/sda1
  ```

  quiet: output fewer console messages

  debug: output all console messages



Fourth element: user space

- What is user space?
- Obtaining a root file system
- Busybox
- Two types of init: Busybox and System V
- Managing device nodes: udev
- Mounting a root file system over the network and from flash memory

What is user space?

- A sane (POSIX) environment for applications (unlike the kernel)
- The main components are
  - Programs – e.g. init and a shell
  - Libraries - e.g. libc
  - Configuration files in /etc
  - Device nodes in /dev
  - User data in /home

The root file system

- Mounted by the kernel during boot
  - requires a `root=...` kernel command line
- Loaded from:
  - ram disk (initramfs)
  - storage device: flash, SD, hard disk
  - network: nfs

Creating a root file system

- Roll-Your-Own (RYO)
- Use an integrated build tool
  - Linaro, Yocto, Meego, Open Embedded
- Use a binary distro
  - Ångström
  - Ubuntu or Debian
- Use a commercial embedded Linux distribution
  - Montavista, Wind River, ...

The rootfs during development

- Advantages of mounting rootfs over NFS
  - easy to access and modify the rootfs
  - No limit on size

Step 1. Export a directory on the development host with a line like this in /etc/exports

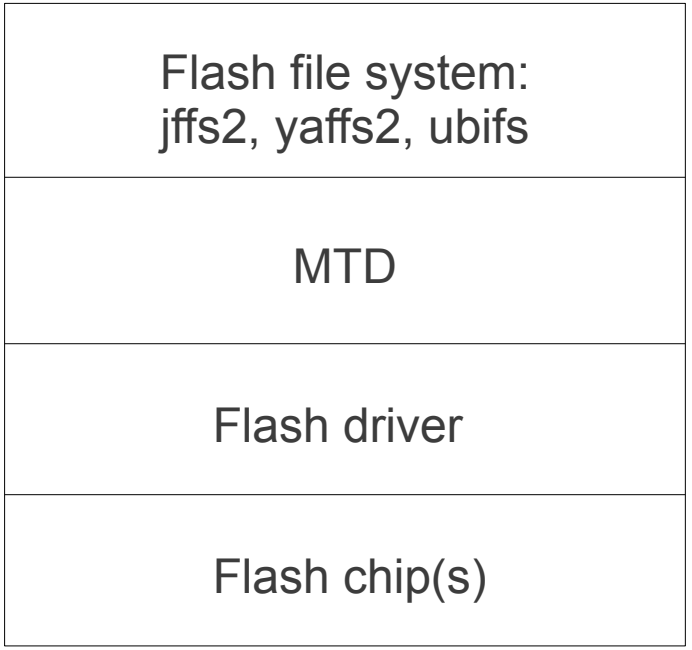
```
/home/chris/rootdir *(rw,sync,no_subtree_check,no_root_squash)
```

Step 2. Set kernel parameters

```
root=/dev/nfs rw nfsroot=192.168.1.1:/home/chris/rootdir ip=192.168.1.101
```

The rootfs in nand flash

Raw NAND flash memory is accessed via Memory Technology Device layer (MTD)

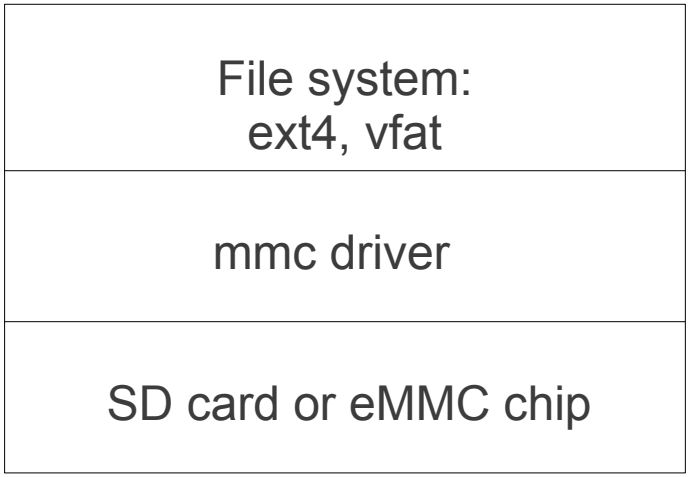


Typical kernel parameters:

```
root=/dev/mtdblock1 rootfstype=jffs2
```

The rootfs in mmc flash

- Either SD or Micro-SD card, or eMMC
  - eMMC is just like an SD card that is soldered onto the board



Typical kernel parameters:

```
root=/dev/mmcblk0p2
```

Summary

- Kernel
  - Your choice of kernel is limited by BSP
  - Many build-time kernel configuration options
  - Boot-time configuration via command line
- User space
  - Starts when kernel mounts rootfs
  - First program to run is (default) /sbin/init
  - For small projects: RYO
  - Otherwise use a framework
    - Linaro, Yocto, Meego, LTIB, etc
    - A commercial environment
    - Android!