Cooking up Embedded Linux in Ten Minutes

Joel Isaacson
Ascender Technologies Ltd.

This Talk

- I will attempt to create an embedded Linux system in just 10 minutes (compilation time not included).
- The other 40 minutes of the lecture will be devoted to understanding the relationship of this demonstration to the accepted commercial context.

This Talk

- We will progress in two parallel tracks:
  - A standard presentation.
  - A demonstration of system building.
- I will ping-pong between these two tracks:
  - The light blue background is the presentation
  - The white background is the demonstration

The Recipe's Ingredients

1)An embedded Linux kernel

2)The application (busybox).

3)A file system image.

4)The bootloader

The Linux Kernel

- If we want to finish the recipe within the allocated time, we have to start cooking.
- Start with a standard Linux kernel.
- Configure it to a minimal configuration with no disk support.
- Let's go do it

Summary of Demonstration

- Download the Linux kernel.
- Untar the source.
- Run “make xconfig”
- Run “make”

Embedded Systems

- What exactly is an embedded system?
- I can't define it but “I know it when I see it”
- In general any system without a complex human interface is considered embedded.
- A practical definition is that any system without a keyboard and mouse is embedded.

Why Bother?

- Why not just buy a embedded Linux distribution?
- Sometimes the easy way turns out not to be the best way.
- Using the development techniques of traditional close source RTOS's yields poor results.
- I have seen this again and again.

The Linux Conundrum

- The Linux's design space is order's of magnitude larger than a typical proprietary system.
- Naively going with a “Embedded Linux Vendor” gets you into a very small linear subset of the possibilities that exists.
- I think our kernel should be ready now. Let go on to the application.

Summary of Demonstration

- We used the busybox application.
- Almost all embedded Linux use busybox.
- The functionality/memory ratio of very high.
- You can substitute you application instead.

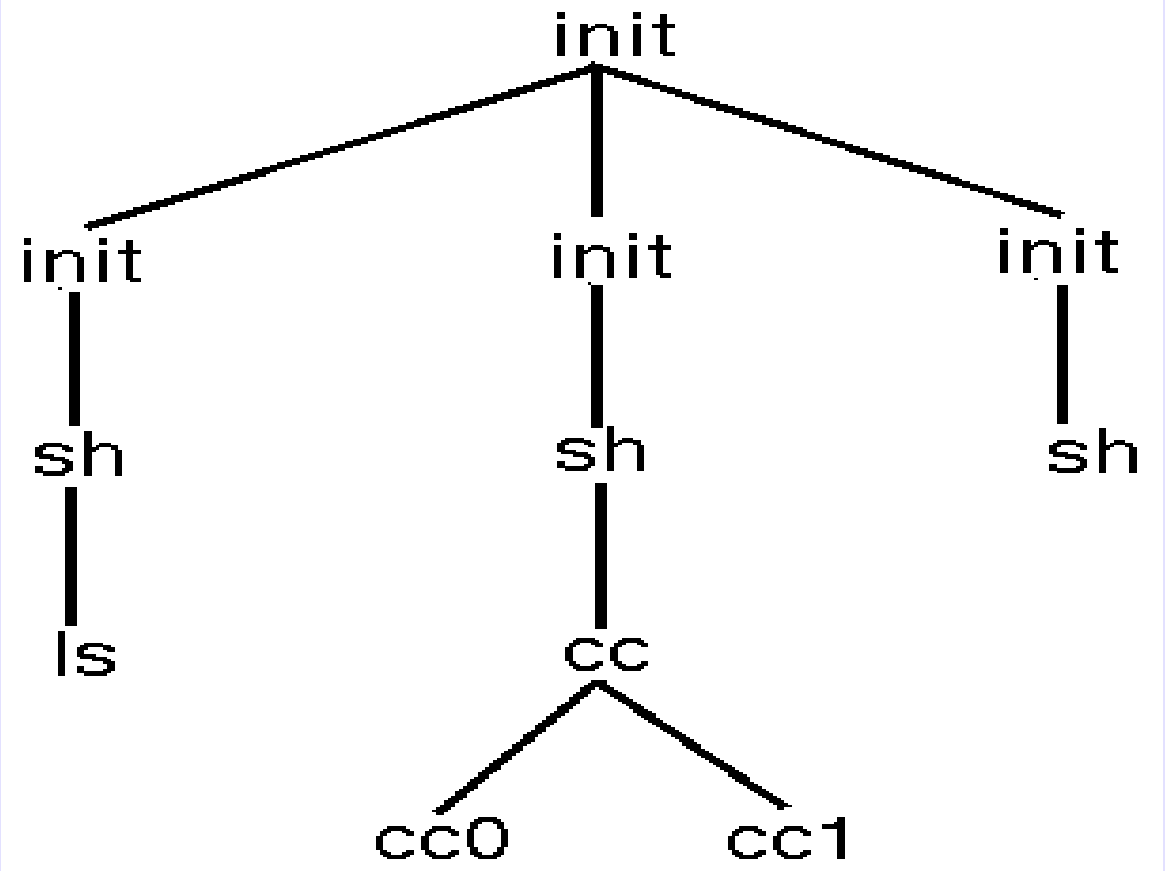
The Linux Filesystem

- At first it might seem strange that you need a filesystem for a system that doesn't even have a disk.
- The only way to exec code under Linux is as a file.
- We therefore have to create a filesytem without using actual physical hardware.

The Linux Process

- The only way to create a process in Unix/Linux is via another Linux process via “fork”.
- Of course on initialization the Linux kernel creates one initial process. This process will be the progenitor of a subsequent processes.
- The fork/exec system calls create the process hierarchy

The Linux Process Tree



The Linux Filesystem

- We make a Linux file system with the file /linuxrc being the initial program executed.
- We now will make the filesystem image

Summary of Demonstration

- We have created a compressed filesystem image that contains our application and some other files needed to run Linux.
- We will use this compressed filesystem image to create an initfd.
- This is the image that is loaded into memory with Linux.

Booting

- Every system need a way to initialize memory.
- Every computer architecture has different ways of doing this.
- Intel architecture has grub or lilo.
- Other architectures have u-boot (Das U-Boot)

Putting it all Together

- We will edit the /boot/grub/menu.lst file.
- We will add the following entries to this file:

```
title embedded2
        root (hd0,5)
        kernel  /elinux2 ro root=/dev/ram0
        initrd /embed2.gz
```

Rebooting

- We will now reboot the system and boot the embedded system.

Demonstration
Compiling Kernel

```
441  cd /usr/src
442  ls
443  tar xfj linux-2.6.18.3.tar.bz2    # untar linux
444  cd linux-2.6.18.3
445  ls
446  cp ../config.embed .config        # copy prepared config
447  make xconfig
448  time make
449  cd arch/i386/boot/
450  ls -ltr
451  sudo cp bzImage /boot/embed2
```

Demonstration
Compiling Busybox

```
453  tar xfj busybox-1.2.2.tar.bz2 # Untar busybox
454  cd busybox-1.2.2
455  make defconfig     # Use the default configuration
456  make menuconfig # Make static linked and default shell
457  time make
458  make install
460  mkdir TMp
461  cd Tmp
       # Create 4 Megabyte zero'ed file
462  dd if=/dev/zero of=embed2 bs=1k count=4k
464  /sbin/mkfs.ext2 embed2 4096 # Format as ext2
      # Mount file embed2 using loopback device
466  sudo mount embed2 /mnt/embed -o loop
468  pushd ../_install/
471  sudo cp -a . /mnt/embed # Copy Busybox to /mnt/embed
```

Demonstration
Compiling Busybox II

```
472  cd /mnt/embed
476  sudo tar xvf ~joel/Dev/dev.tar  # Make /dev directory
478  popd
479  sudo umount /mnt/embed         # Unmount /mnt/embed
483  gzip embed2                    # Compress embed2
485  sudo cp embed2.gz /boot        # Copy to /boot
```

Demonstration
Editing Grub Bootloader

```
488  sudo vi /boot/grub/menu.lst
489  sudo reboot
```

Added to menu.lst

```
title embedded2
        root (hd0,5)
        kernel  /elinux2 ro root=/dev/ram0
        initrd /embed2.gz
```

Demonstration
Contents of /dev

```
crw-r--r-- 1 root root 5, 1 2004-10-30 18:59 console
crw-r--r-- 1 root root 1, 3 2004-10-30 18:59 null
brw-r--r-- 1 root root 1, 1 2004-10-30 18:59 ram
crw-r--r-- 1 root root 4, 0 2004-10-30 18:59 systty
crw-r--r-- 1 root root 4, 0 2006-11-26 15:09 tty0
crw-r--r-- 1 root root 4, 1 2004-10-30 18:59 tty1
crw-r--r-- 1 root root 4, 2 2004-10-30 18:59 tty2
crw-r--r-- 1 root root 4, 3 2004-10-30 18:59 tty3
crw-r--r-- 1 root root 4, 4 2004-10-30 18:59 tty4
crw-r--r-- 1 root root 4, 5 2006-11-26 15:08 tty5
crw-r--r-- 1 root root 4, 6 2006-11-26 15:08 tty6
crw-r--r-- 1 root root 4, 7 2006-11-26 15:08 tty7
```