Embedded Linux Software Development

T.Michael Turney

Open Source Ambassador

1.December.2005

e

S.O.A.P.B.O.X.

U R N O A N C

P A D M B R

E L P B E

R O L M

I U I E

O S N N

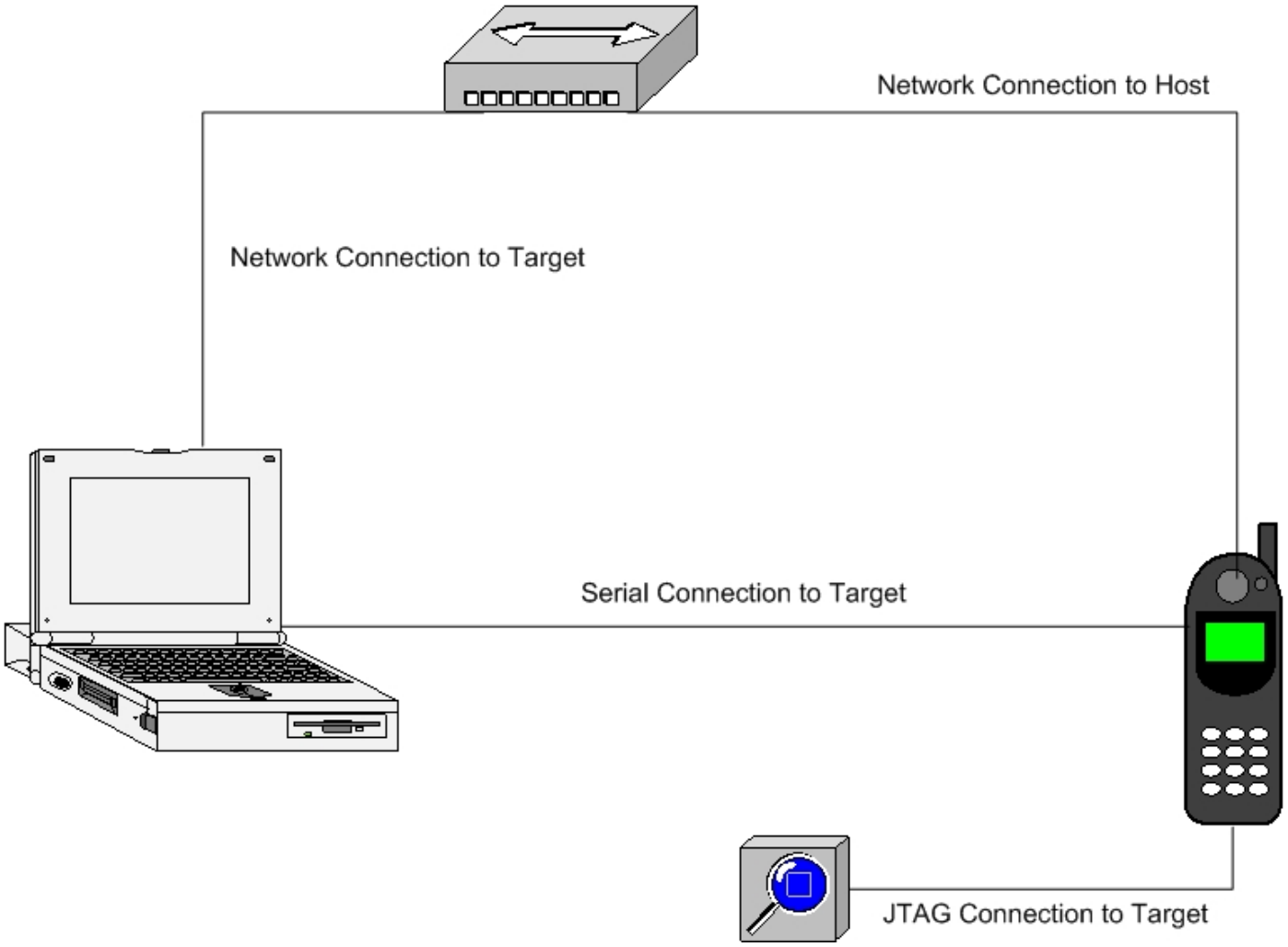
R G T

S

The Embedded Linux Port Process

1. When it all goes right, what you can expect
2. 10,000 foot view zeroes in on each step
   - Step 1 : Squaring away your environment
   - Step 2 : Command & Control of your target
   - Step 3 : Understanding the boot process
1. **G**rossly im**P**olite **L**itigation
2. Resources and Closing Thoughts

Typical Embedded Linux Diagram



The Perfect World

- Development Host is setup
  - minicom is configured and working
  - tftp is configured and working
  - NFS server is configured and working
  - DHCPD is configured and working
  - Cross-Development tools are installed
  - Root filesystem for target is installed
  - Kernel for target is built

The Perfect World

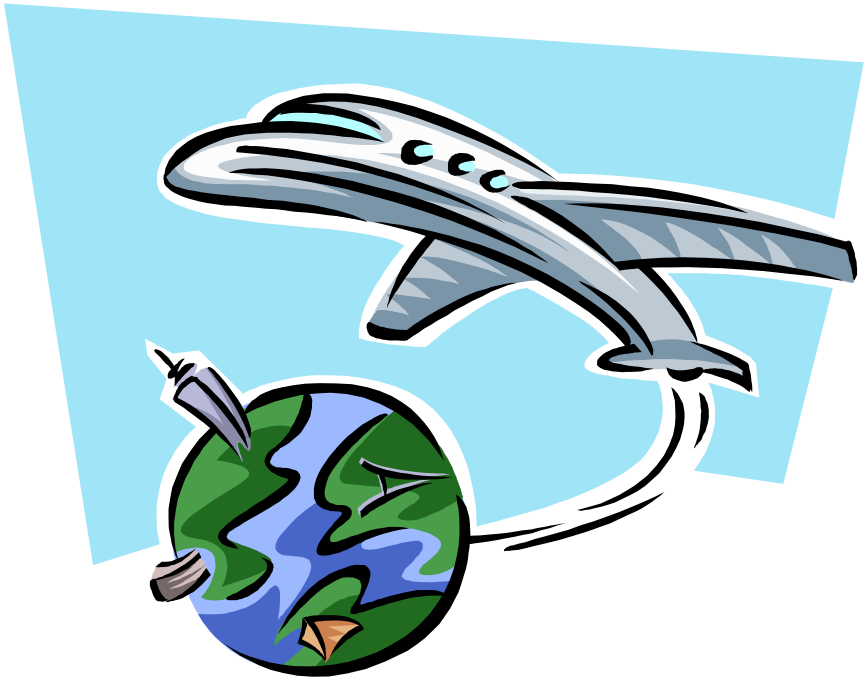
- Target has boot firmware that…
  - Can initialize target hardware
  - Can do network boot of kernel
  - Can pass board parameters to kernel
  - Has serial port support and command interpreter
  - U-boot comes to mind
    - http://sourceforge.net/projects/u-boot

The Perfect World

- Development host boots and starts all necessary services
- Target boots to firmware prompt
- Target downloads kernel through network
- Kernel starts booting
- Kernel requests NFS-mount root filesystem
- Target displays login prompt on serial console and responds to telnet

10,000 Feet Doesn't Help With Details

- The devil is in the details, so lets grab the two-horned beast by the tail and take a ride



Step 1 : Squaring Away Your Environment

minicom Setup

- Make sure package is installed
  - FedoraCore2 : minicom-2.00.0-18.1
- As root user, run
  - *minicom –s*
- Things to specify include
  - Port (e.g. /dev/ttyS0)
  - Baud rate (should agree with firmware)
  - Clear out some of the modem init strings
  - Turn off Hardware flow control
- chmod 666 /dev/ttyS0

tftp Setup

- Make sure package is installed
  - FedoraCore2 : tftp-server-0.33.3
- Create /tftpboot directory and open it up
  - chmod 777 /tftpboot
- Create / modify configuration file
  - /etc/xinetd.d/tftp
- Restart xinetd daemon
  - */etc/init.d/xinetd restart*

NFS Server Setup

- Make sure package is installed, FC2
  - system-config-nfs-1.2.3-2, nfs-utils-1.0.6-20
- Create /etc/exports file
- Start NFS server
  - /etc/init.d/nfs start
- Should NFS server start at boot-time?
  - *chkconfig nfs on*

DHCPD Setup

- Make sure package is installed
  - FedoraCore2 : dhcp-3.0.1-rcl2-4
- Create /etc/dhcpd.conf file
- Start DHCPD
  - /etc/init.d/dhcpd start
- Should DHCPD start at boot-time?
  - chkconfig dhcpd on

Cross-Development Toolchain

- You have three choices here:
  - Purchase a distribution
  - Download a distribution from the internet
  - Roll your own

Purchase Toolchain

- Pros
  - Benefits of a commercial product
    - Support
    - Quality (more testing than other options)
  - Many companies testing the Linux waters want the *safety-net* of a vendor-provided solution
- Cons
  - $$$
  - Beware of vendor-specific extensions

Download Toolchain

- Some architectures are better supported than others
- No *safety-net* of a vendor-provided solution
- caveat emptor (no money, but same principle)
- Easy way to jumpstart a *skunk-works* project
- May not scale as well as a commercial product
- The entity that provides the toolchain is making money somehow, their responsiveness to your input and queries may not be satisfactory

Build Toolchain

- Not for the *faint-of-heart*
- Know what you are doing
- At a minimum you will be building…
  - gcc-4.0.2.tar.bz2
  - gdb-6.3.tar.gz
  - glibc-2.3.6.tar.gz
  - Binutils-2.14.tar.gz
- You will become intimate with this site
  - http://kegel.com/crosstool/

Root Filesystem for Target

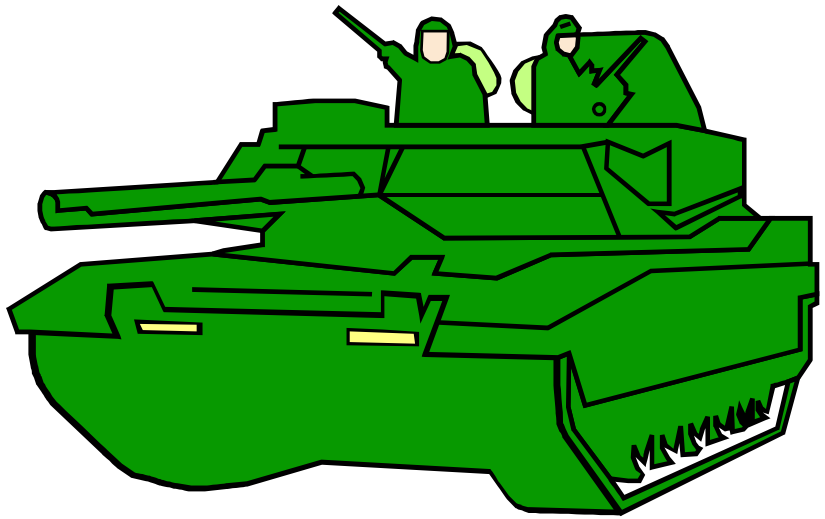
- Same comments as the toolchain
- Most distributions start with debian packages
  - http://www.debian.org/distrib/packages
- You will become intimate with busybox
  - http://www.busybox.net

Kernel for Target

- Same comments as the toolchain

- All distributions start from kernel.org
  - http://www.kernel.org/
- The equivalent of a BSP is required, the Linux kernel must be personalized for each target board that it runs on
  - (more on this later)

Step 2 : Command & Control of Your Target



Firmware - 101

- POST (Power On Self Test)
- Initialize board hardware, e.g.
  - SDRAM controller
  - Interface ports (serial, network)
- Boot system

- This gives you the basics, but a good development environment provides more…

Firmware - Advanced

- Provides a command interpreter
- Supports network boot of kernel
- Supports FW variables
- Supports scripting capability

- This gives you a nice development environment for an embedded Linux project

Step 3 : Understanding the Boot Process

Firmware Brings up Hardware

- The BIOS or Firmware must…
  - Initialize SDRAM controller
  - Initialize other hardware, e.g.,
    - Timer channel
    - Serial channel
    - Network interface
- Development version should provide rudimentary commands to the engineer

Firmware hands-off to the Linux Loader

1. FW loads Linux kernel
   - From Flash
   - From network
   - From other device
1. FW jumps to Linux Loader entry point

Time-Out!!!

We need some OSS tools…

http://cscope.sourceforge.net/

http://cbrowser.sourceforge.net/

Linux Loader Prepares the Kernel for Boot

…/arch/ppc/boot/mbx/head.S (**start**)

- Calls **start_ldr**
  - **embed_config (initialize board info struct)**
    - **typedef struct bd_info {**
      - **unsigned int bi_memstart;**
      - **unsigned int bi_memsize;**
      - **unsigned int bi_intfreq;**
      - **unsigned int bi_busfreq;**
      - **unsigned char bi_enetaddr[6];**
      - **unsigned int bi_baudrate;**
    - **} bd_t;**

Linux Loader cont…

- **serial_init** (initialize serial port)
- **decompress_kernel**
  - Does the heavy lifting
  - Determine size of vmlinuz kernel
  - Determine size (presence) of initrd
  - Print load data on serial console
  - Display command line
  - Allow user to modify command line
  - Relocate and decompress kernel

Setup registers for kernel

Jump to kernel entry point (0x00)

Linux Kernel Boots

…/arch/ppc/kernel/head_8xx.S (**_start**)

- R3 : pointer to board info struct
- R4 : pointer to start of initrd
- R5 : pointer to end of initrd
- R6 : pointer to start of command line
- R7 : pointer to end of command line
- Map first 8M of ram as single page
- Map internal I/O space as single page
- Jump to **start_here**

Linux Kernel continues to boot…

- …/arch/ppc/kernel/head_8xx.S (**start_here**)
  - Call **identify_machine** (…/arch/ppc/kernel/setup.c)
    - Call **m8xx_init**
      - Initialize struct machdep_calls ppc_md
      - Initialize struct ide_machdep_calls ppc_ide_md
  - Call **MMU_init** (…/arch/ppc/mm/init.c)
    - Setup kernel memory mappings
    - Setup I/O memory mappings
  - Enable MMU
  - Jump to **start_kernel**

Linux Kernel hands-off to the Init Process

- …/init/main.c (**start_kernel**)
  - Initializes all kernel sub-systems
  - **kernel_thread**(init, NULL, CLONE_FS | CLONE_FILES | CLONE_SIGNAL);
- …/init/main.c (**init**)
  - In this order…
    - Command line
    - /sbin/init
    - /etc/init
    - /bin/init
    - /bin/sh

Grossly imPolite Litigation

- Any engineer working on a project using OSS (open source software) has to be aware of the GNU GPL.
- The General Public License is the main license used by most OSS.
- http://www.opensource.org/
  - Defines OSS
  - Lists approved OSS licenses

- **Q8: If I embed GNU/Linux in an application, what are my obligations to the Open Source community?**

A: The GNU General Public License **(GPL version 2)** is very specific about the obligations imposed
on developers leveraging Open Source. If you deploy/redistribute program binaries derived
from source code licensed under the GPL, you must

Supply the source code to derived GPL code or Make an offer (good for 3 years) to supply the source code

Retain all licensing / header information, copyright notices, etc. in those sources

Redistribute the text of the GPL with the binaries and/or source code

Note that your obligation is strictly to the recipients of binaries (e.g.: your customers). As long as you supply source code together with your GPL binaries, you have no responsibility to the "community" at large. Moreover, the source code for those portions of your application that are unique, not derived from GPL code, are not at all affected by the GPL -- you may license them as you choose.

T.Mike's 3-Step Program

- Do you distribute or sell…
  - A Service
  - Hardware
  - Software

Step 1 : Service

- The best position, vis-à-vis GPL
- Your use of GPL software is probably locked away in a server room
- In essence, you aren't distributing product with GPL, so your GPL issues are kept internal.
- Very similar to using GCC as your toolchain

Step 2 : Hardware

- The second best position, vis-à-vis GPL
- Your use of GPL is hidden within a hardware device
- Unless you are a chip vendor, changes you make to the kernel *probably* don't include *intellectual property* important to your company
- There is always a spoiler…
  - http://www.gpl-violations.org/

http://www.gpl-violations.org/

- Products listed here include…
  - TomTom
  - Numerous notebooks
- **Why gpl-violations.org?**
- The project was started to raise the awareness about past and present violations of the GNU General Public License. Its main purpose is therefore gathering, maintaining and distributing information about people who use and distribute GPL licensed free software without adhering to the license terms.

Step 3 : Software

- The worst position, vis-à-vis GPL
- Think long and hard about how a business model to package/distribute OSS is going to make money
- In order to survive, you have to have value-add , and if you do have value-add, how are you protecting it?

Resources

- Web Resources
  - Repositories of OSS
    - http://directory.fsf.org/
    - http://sourceforge.net/
    - http://freshmeat.net/
  - Too many others to list…
  - Can’t go wrong with google to start with…

Resources

- Book Resources
  - Understanding the Linux Kernel (3rd Edition)
    - Copyright November 2005
    - ISBN 0-596-00565-2
  - Linux Device Drivers (3rd Edition)
    - Copyright February 2005
    - ISBN 0-596-00590-3
  - Building Embedded Linux Systems (1st Edition)
    - Copyright April 2003
    - ISBN 0-596-00222-X
  - Running Linux (5th Edition)
    - Copyright December 2005
    - ISBN 0-596-00760-4

Closing Thoughts

- Linux is here to stay
- Embedded Linux is *icing on the cake*
- Linux makes it easier to *roll your own*
  - But does your company make $$ selling OS's?
- It still comes down to the same simple decision…
  - Partner with a vendor who provides you the best tools to get your project to product…

e

S.O.A.P.B.O.X.

o u u r u n c
m t r o i i
e s a p l t
t l o t e
a s m
n i e
d t n
i i t
n o
g n
s