Development Environment

Embedded Linux Primer Ch 1&2
Topics

1) Systems: Host and Target
2) Host setup
3) Host-Target communication
Host and Target
Host & Target

- **Host**
  - Development PC
- **Native Compiler:**
  - Run compiler on host to build for host:
    
    ```
    $ gcc hello.c -o hello
    ```
- **Cross Compiler:**
  - Many "cross" tools: Run on host, work with target:
    Ex: `arm-linux-gnueabihf-gdb`

- **Target**
  - Embedded device

 monetary  $ means Linux prompt on host.
Host & Target

- Tool naming:
  arm-none-linux-gnueabihf-gcc

  **target architecture**
  **vendor (company)**
  Optional

  GNU EABI
  (embedded ABI)
  hf = Hardware Floatingpoint

- ABI:
  - Standard specifying how the program will:
    - layout data types in memory
    - .. (passing arguments, returning values).
    - perform system calls

  gcc = Tool
## Host vs Target Resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Host</th>
<th>Target (BeagleBone Green)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Ubuntu 18.04</td>
<td>Debian Linux</td>
</tr>
<tr>
<td>CPU</td>
<td>~3Ghz 12-core x64</td>
<td>1Ghz ARM Cortex-A8, 32 bit</td>
</tr>
<tr>
<td>RAM</td>
<td>16,000 Meg</td>
<td>512 MB</td>
</tr>
<tr>
<td>Storage</td>
<td>4,000 GB harddrive</td>
<td>4GB eMMC</td>
</tr>
<tr>
<td>Screen</td>
<td>23&quot; LCD, multi-monitor</td>
<td>None; could use a cape.</td>
</tr>
<tr>
<td>Input</td>
<td>Keyboard, mouse</td>
<td>1 button, USB Cape for lots!</td>
</tr>
<tr>
<td>Audio</td>
<td>In/out</td>
<td>via cape</td>
</tr>
<tr>
<td>Ethernet</td>
<td>1,000 BaseT</td>
<td>100 BaseT</td>
</tr>
<tr>
<td>Other</td>
<td>DVD, Card reader</td>
<td>uSD Card, GPIO</td>
</tr>
<tr>
<td>Terminal</td>
<td>Screen &amp; Keyboard</td>
<td>TTL serial &amp; SSH</td>
</tr>
<tr>
<td>Cost</td>
<td>~$1,000</td>
<td>~$50-$100</td>
</tr>
</tbody>
</table>

- **eMMC**: Embedded (on a chip) flash storage (MultiMedia Card)
- **TTL 3.0V**: Transistor to transistor logic
Working with Hardware

• Many embedded systems run on custom hardware.
• Interact with the world using:
  – GPIO:...
    Set a pin to be on (3.3V) / off (0V), or read it.
  – I²C:...
    Communicate with chips like an accelerometer.
  – A2D:...
    Read analog voltages (Ex: battery voltage).
  – PWM:...
    Generate a sort-of analog voltage.
• For us, the Zen cape allows us to use all of these!
Host Setup
Host Setup

• Run Linux.
  – A definition of "Crazy": Developing for embedded Linux on non-Linux host.

• Run Linux as main OS, or in virtual machine (VM).
  – VMWare Player: lets you run Linux inside Windows in a VM.
  – Selectively configure resources the VM gets.
  – Able to run multiple VM's on one machine.

Confusion:
Host PC: Computer you code on.
Host OS: In VM context means “real” OS on computer.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
</table>
| **ls**  | Directory listing. Arguments: -a for all, -l for long (all info) | ls  
ls -l |
| **pwd** | Show current directory name | pwd |
| **mkdir** | Make a directory | mkdir myNewPlace |
| **cd**  | Change directory | cd myDir  
cd \myDir  
cd ..  
cd \ |
| **chmod** | Change file permissions | chmod a+r hello.a |
| **chown** | Change file owner | chown bfraser hello.a |
| **sudo** | Execute as administrator | sudo chown bfraser hello.a |
| **apt-get** | Install a program | sudo apt-get install somepackage |
| **gedit** | Edit a file (new window) | gedit hello.cpp & |
| **ifconfig** | Configure networking | ifconfig eth0 192.168.0.1 |
| **mount** | Mount a file-system | mount -t nfs 
192.168.0.103:/opt/img /mnt/img |
| **nano** | Edit a file in the terminal | nano hello.cpp |
## Basic Linux Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>Dump to screen</td>
<td>cat hello.cpp</td>
</tr>
<tr>
<td>less</td>
<td>Show on screen with &quot;more..&quot; prompt. ('q' to quit)</td>
<td>less hello.cpp</td>
</tr>
<tr>
<td>tar</td>
<td>Archive management (unzip)</td>
<td>tar xvfj hello.tar.jz2</td>
</tr>
<tr>
<td>find</td>
<td>List all files in sub-folders</td>
<td>find</td>
</tr>
<tr>
<td>grep</td>
<td>Search for a string</td>
<td>grep &quot;Hello world&quot; *.cpp</td>
</tr>
<tr>
<td>`</td>
<td>`</td>
<td>Pipe: redirect output to second program's input</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Redirect output to a file</td>
<td>ls &gt; listing.txt</td>
</tr>
<tr>
<td>rm</td>
<td>Remove a file (delete)</td>
<td>rm listing.txt</td>
</tr>
<tr>
<td>echo</td>
<td>Print some text</td>
<td>echo hello</td>
</tr>
<tr>
<td>dmesg</td>
<td>Show kernel boot messages</td>
<td>dmesg</td>
</tr>
</tbody>
</table>

Recommended Linux Shell Tutorial:  
Communication

How can we access the target?
We need a Linux terminal, but how?

Communications Overview

- **Serial Port:**
  - Access target's Linux terminal via the serial port (Need serial port when can't use SSH: booting or errors)
    - Serial protocol for +/-12V
    - 0-3V (or 0-5V) serial protocol.
    - Zen cape has TTL over USB (micro USB port)
  - Host uses “Screen” program show serial data.

- **Ethernet Network:**
  - 

![Diagram of Ethernet and TTL over USB connections]
RS232 Serial Protocol

- RS232 voltages
  - $-12V = 1$; $+12 = 0$
- Each bit has fixed time dependant on bitrate
- Starts with start bit(s) ($+12v$); ends with stop bit(s) ($-12v$)
- Diagram shows a ‘K’ character (0x4B)

RS = “Recommended Standard”
RS232 Real World View

- Oscilloscope trace of ‘K’ (0x4B), 1 start bit, 8 bits, 2 stop bit.
- HW has to be sync’d to know where to sample each bit.
- Timing errors lead to garbage characters.
RS-232 & TTL Settings

- RS-232 is a "Serial Port"
  - Connector is often a DB9 / DB25 with bi-directional data: can Tx and Rx at the same time.

- Zen cape has FTDI chip to convert TTL serial data from the microprocessor to USB; host has drivers to access it like a “normal” serial port. (Called TTL-232)

- BeagleBone Serial Port Settings
  - Speed (bps), #bits/byte, parity check, # stop bits:
    .. < 14kBytes / second!
  - Optional handshaking to control data transmission.
  - We'll always use no handshaking
Screen Program

• Run Screen on host to view target's serial port.

• Screen Usage on host
  – Install:   $ sudo apt-get install screen
  – Run:      $ sudo screen /dev/ttyUSB0 115200

• Screen Operations
  – Show help: Control a + (no control) ?
  – Quit:     Control a + (no control) \\

• Linux Ports
  – /dev/ttyUSB0 how Linux supports TTL over USB

Demo: Screen, dmesg.
Network

• BeagleBone can network in two ways:
  - Ethernet
    • Normal “RJ45” Ethernet connection.
    • BBG uses DHCP to get an IP address: DHCP =..
    • DHCP client program: dhclient is running by default.
  - Ethernet over USB
    • Micro USB cable allows BeagleBone to mount on host PC as a network connection.
    • Host has IP: 192.168.7.1
    • Target has IP: 192.168.7.2
Networking Basics

• Find out IP settings:
  $ ifconfig
  # ifconfig

• ssh to open a terminal to the target
  $ ssh debian@192.168.7.2

• ping to test TCP/IP connection to board:
  $ ping 192.168.7.2
  # ping 192.168.7.1

$: means host PC command
# means target command
Files over the Network

- Mounting directory over NFS
  - NFS:...
  - Use NFS to make application testing MUCH faster:
    - Transferring ~50 meg takes ~1min vs ~1hr.
    - On the target, mount the host's directory and..

- “Pro” Tip:
  Always look for ways to make development faster.
Review

1. What is cross compiling?

2. What does sudo do?

3. What will we use TTL over USB for?

4. Explain why NFS is useful.
Summary

- Develop on host, deploy on target.
  - Cross compile on host for target.

- Target has limited resources, but custom hardware:
  - GPIO, I2C, A2D, PWM.

- Host running Linux in VM or native

- Communicate to target using TTL-232 and Ethernet:
  - DHCP, Ping, TFTP, NFS.