Agenda

Morning Session (2 hours)

- Welcome
- Device Overview
- Tools – Boards & SDK
- What is Linux
- Linux Distributions
- Booting Linux (with Lab)

Afternoon Session (2 hours)

- Introduction to CCSv5.1
- Debugging Linux with CCS
- Quick Introduction to GNU Make
- Lab: Debug Linux App with CCSv5

Find these workshop materials at:
http://processors.wiki.ti.com/index.php/LinuxWorkshopTechDays2011

Outline

- Embedded Processors
- Tools – Boards & SDK
- Intro to Linux
- Linux Distro's
- Booting the Device (Das U-Boot)
- Lab Exercise

TI Embedded Processors Portfolio

Microcontrollers | ARM-Based | DSP

16-bit	32-bit Real-time	32-bit ARM MCU	ARM MPU	ARM + DSP	DSP
MSP430	**C2000™**	**ARM**	**ARM9 Cortex A-8**	**C6000 plus ARM9/Cortex A-8**	**C674x, C55x, C6000 Multicore,**
Ultra-Low Power	Fixed & Floating Point	Industry Std Low Power	Industry-Std Core, High-Perf GPP	Industry-Std Core + DSP for Signal Proc.	Leadership DSP Performance
Up to 25 MHz	Up to 300 MHz	<100 MHz	Accelerators	4800 MMACs/ 1.07 DMIPS/MHz	24,000 MMACS Fix/Float
Flash 1 KB to 256 KB	Flash 32 KB to 512 KB	Flash 64 KB to 1 MB	MMU	MMU, Cache	Up to 3 MB L2 Cache
Analog I/O, ADC LCD, USB, RF	PWM, ADC, CAN, SPI, I^2C	USB, ENET, ADC, PWM, SPI	USB, LCD, MMC, EMAC	VPSS, USB, EMAC, MMC	1G EMAC, SRIO, DDR2/3, PCIe
Measurement, Sensing, General Purpose	Motor Control, Digital Power, Lighting, Sensing	Host Control	Linux/WinCE Android User Apps	Linux/Win/Andr + Video, Imaging, Multimedia	Comm, WiMAX, Industrial/ Medical Imaging
$0.49 to $9.00	$1.50 to $20.00	$2.00 to $8.00	$5.00 to $35.00	$12.00 to $65.00	$4.00 to $99.00+
					

Texas Instruments

Key System Blocks

An integrated solution that reduces System complexity, Power consumption, and Support costs

Low Power

No heat sink or fan required. Ideal for end equipment that require air-tight, sealed enclosures

Graphics Accelerator

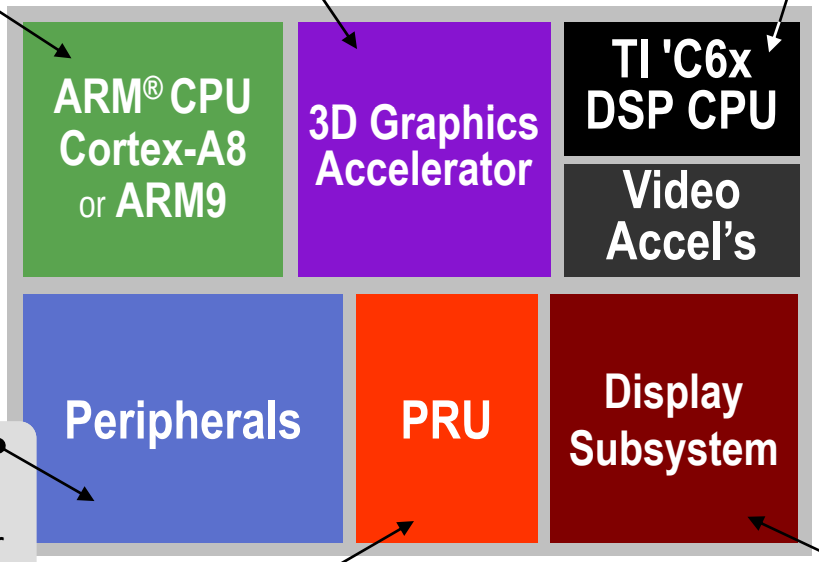
Provides rich image quality, faster graphics performance and flexible image display options for advanced user interfaces

'C6x DSP Core

- Off-load algorithmic tasks from the ARM, freeing it to perform your applications more quickly
- Allows real-time multi-media processing expected by users of today's end-products
- Think of the DSP as the ultimate, programmable hardware accelerator
- **Video Accelerators** – either stand-alone or combined with the DSP provide today's meet today's video demands with the least power req'd

ARM Core

High performance processors (**375MHz - 1GHz**) drive complex applications running on **Linux**, **WinCE** or **Android** systems



Peripherals

Multiplicity of integrated peripheral options tailored for various wired or wireless applications – simplify your design and reduce overall costs

Display Subsystem

Off-loads tasks from the ARM, allowing development of ***rich "iPhone-like" user interfaces*** including graphic overlays and resizing without the need for an extra graphics card

Prog. Real-time Unit (PRU)

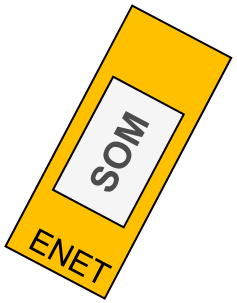
- Use this configurable processor block to extend peripheral count or I/F's
- Tailor for a proprietary interface or build a customized system control unit

NOTE
Features not available on all devices

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
  - Hardware Development Kits
  - Software Development Kits (DVSDK, SDK)
- **Intro to Linux**
- **Linux Distro's**
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**

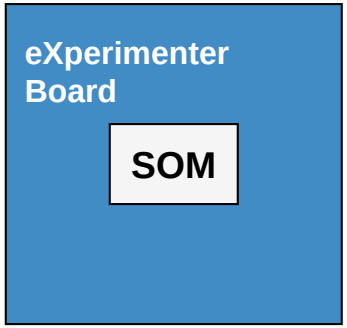
Modular Dev'l Kits – AM3517 Example



SOM Module

AM3517 SOM-M2
Price: < $100
SW Development

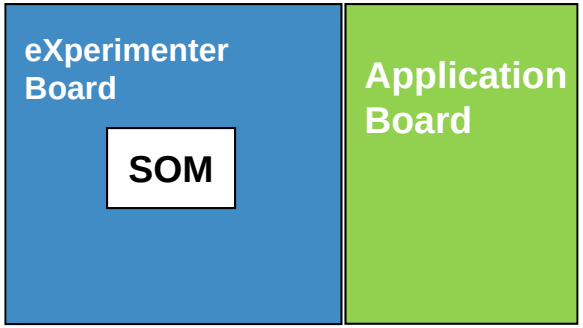
- 1.6" x 2"
- Features:
  - 256 MB DDR2 SDRAM
  - 512 MB NAND flash
  - Wired Ethernet
  - Wireless 802.11b/g/n*
  - Bluetooth 2.1 + EDR IF*
- Self-boot Linux image
- Purchase – Logic via Arrow, Avnet, Digikey
- Support – Logic



eXperimenter Kit

SDK-XAM3517-10-256512R
Price: $199
S/W and H/W Dev't

- 5" x 6"
- Features SOM features +
  - HDMI (video only)
  - MMC/SD card slot
  - Network/USB/Serial/JTAG /Logic-LCD Connectors
  - Built-in XDS100 emulation
- Purchase – Logic via Arrow, Avnet, Digikey
- Support – Logic
- SW: Linux, WinCE



EVM

TMDXEVM3517
Price: $999
Full Development Platform

- EVM additionally includes:
  - LCD
  - Multimedia In/Out
  - KeyPad
  - Connect: CAN, RJ45, USB, UART, stacked SD
- Channel – TI & distribution
- Support – TI & Logic
- Linux and WinCE SDK's (from TI); Android SDK is in development

Hardware Development Environments

4 Types of Hardware Development Tools

Community Board

Use Case

- Evaluation of processor functionality
- Application development with limited peripheral access
- Community-only support

System-on- Module

Use Case

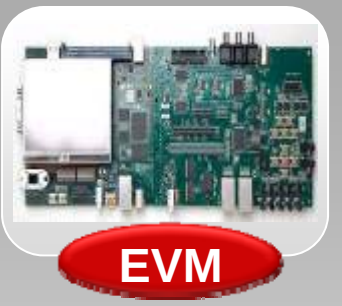
- Simplify system board design
- Medium for Prototype or Production end equipment

eXperimenter Kit

Use Case

- Evaluation of processor functionality
- Application development with limited peripheral access

Evaluation Module



Use Case

- Touch-screen application development with full peripheral access
- Application specific development

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
  - Hardware Development Kits
  - Software Development Kits (DVSDK, SDK)
- **Intro to Linux**
- **Linux Distro's**
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**

Software Development Kits

S/W Dev'l Kit	Description	Processor(s)
Linux PSP SDK	**Small Linux Distro** supporting TI ARM devices	◆ OMAP35, AM35, AM18 ◆ OMAP-L1 ◆ DM644x, DM6467, DM3xx
"DVSDK" (TI Libraries)	**TI provided libraries, examples, demos** Codec Engine (VISA), DSPlink, Codecs/Algos (XDM), BIOS, XDC, Linux utilities, etc.	◆ All TI SOC's: ARM, DSP, ARM+DSP ◆ Obviously, not all devices require all the s/w components
Code Gen Tools (not really "kits" per se)	◆ **Linux GNU Compiler** (CodeSourcery) ◆ **C6000 DSP Compiler** (TI)	◆ All TI ARM and DSP devices where appropriate
Graphics SDK	**Graphix SVSGX development kit** OPENGL ES / VG demos, drivers, targetfs, Getting Started Guide	◆ OMAP3515, OMAP3530 ◆ AM3517, …

- *PSP is a TI specific acronym that represents the name of the group inside of Texas Instruments which "owns" the kernel and driver development activities: Platform Support Package team*
- *Wireless SDK is available independently of these other kits to support the TI WLxxxx Bluetooth/WiFi devices*

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
  - Linux Fundamentals
  - Linux - Basic Commands
- **Linux Distro's**
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**

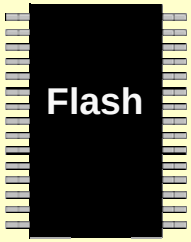
Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
  - Linux Fundamentals
  - Linux - Basic Commands
- **Linux Distro's**
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**

Linux in Three Parts

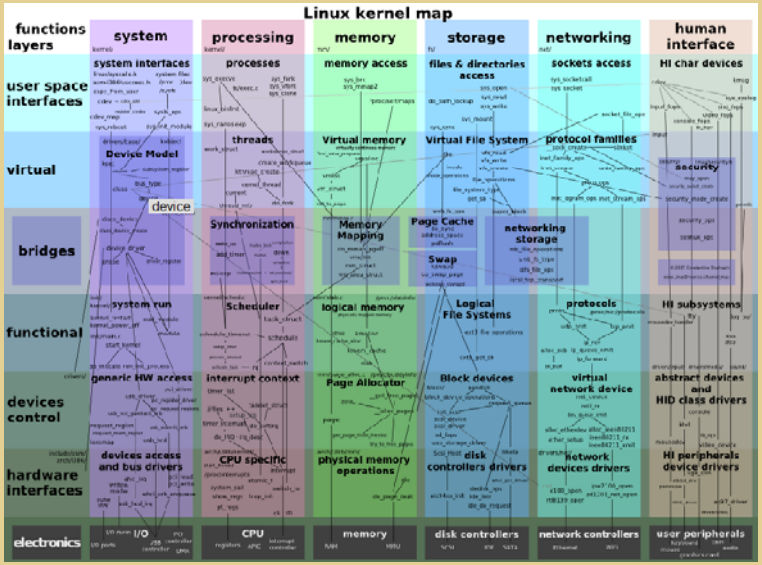
❶ Bootloader

- Provides rudimentary h/w init
- Calls Linux kernel and passes boot arguments



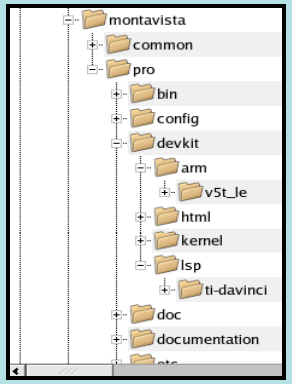
❷ Kernel

- Initializes the system (and device)
- Manages system resources
- Provides services for user programs



❸ Filesystem

- Single filesystem (/ root)
- Stores all system files
- After init, kernel looks to filesystem for “what’s next”
- bootarg tells linux where to find root filesystem



Linux Boot Process

Power On

Boot Loader

U-Boot

- ARM assembly code
- Passes args to Linux (bootargs)

Linux Kernel

Linux Kernel

Initialize hardware

File System

Init Process

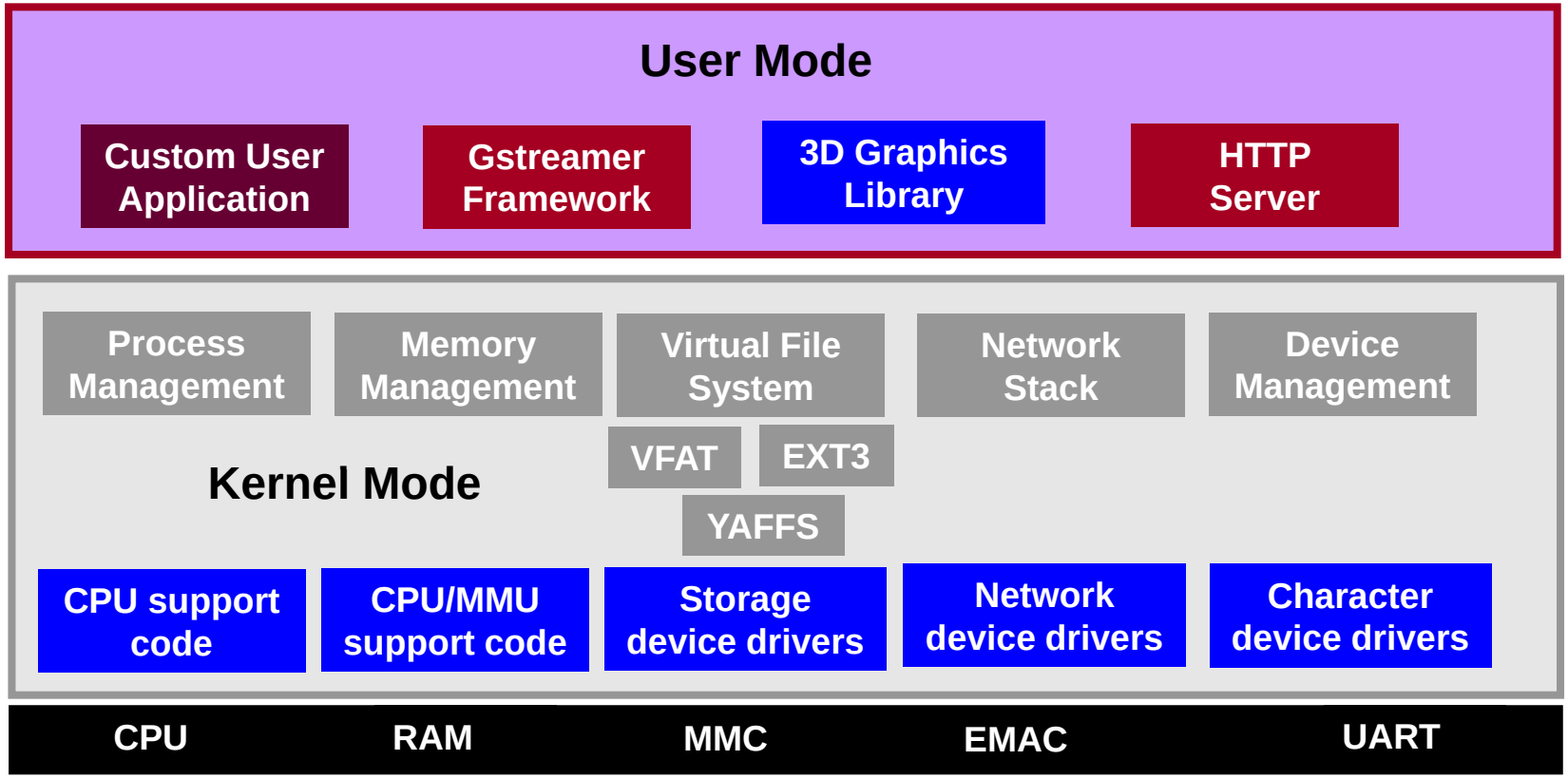
/sbin/init – 1st process exe by kernel

Login Prompt

- Login console
- Usually one of first prog's to run

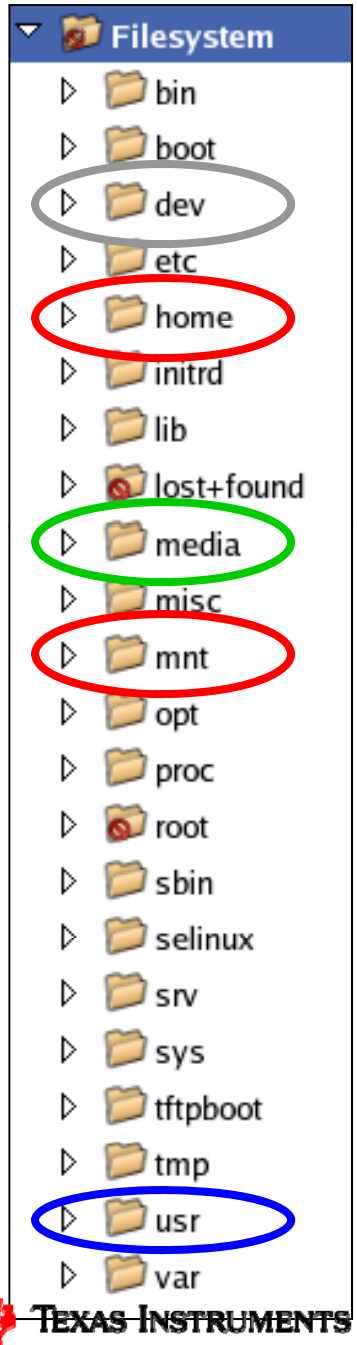
Looking more closely at the Kernel ...

What's in the Linux Kernel



A few more details about the filesystem...

Red Hat / Ubuntu : Root File System



Some folders common to Linux:

/dev – Common location to list all device drivers

/home - Storage for user's files

- Each user gets their own folder (e.g. /home/user)
- Similar to "My Documents" in Windows
- DVSDK GSG directory for TI tools, examples, working directory
- "root" user is different, that user's folder is at /root

/media – Usually find CDROM drive(s) mounted here

/mnt – Common location to mount other file systems

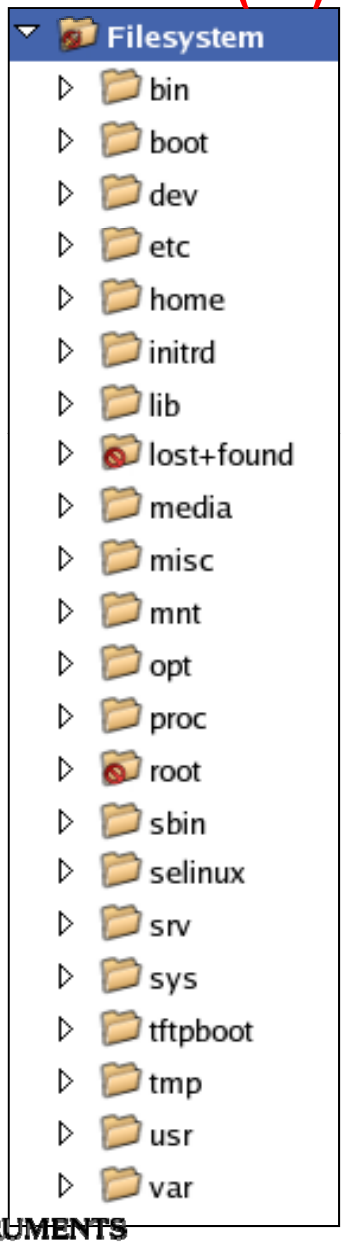
- Linux only allows one filesystem
- Add other disks (physical, network, etc) by mounting them to an empty directory in the root filesystem
- Windows adds new filesystems (C:, D:, etc.) rather than using a single one

/usr – Storage for user binaries

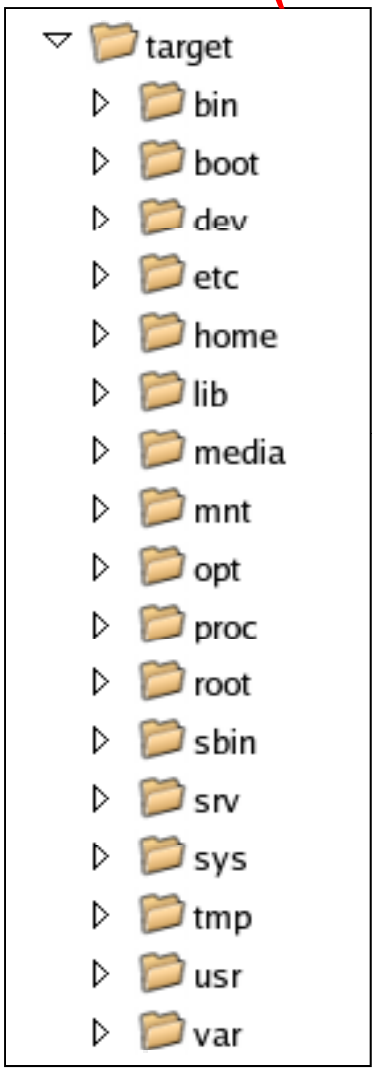
- X86 Compiler for Ubuntu programs (gcc) is stored in here

Filesystems: Red Hat vs. Montavista

Red Hat (PC)

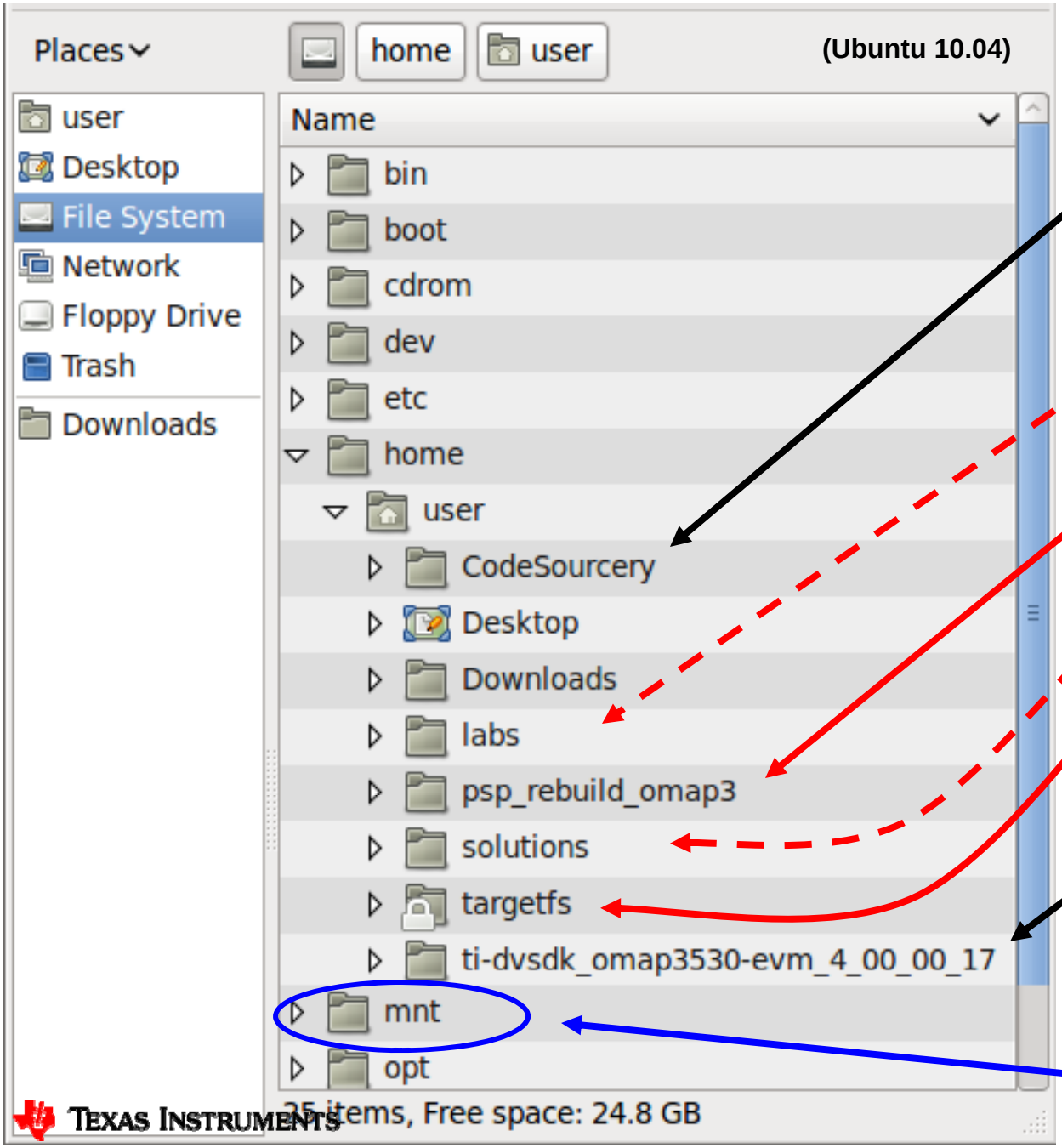


MontaVista (ARM)



- Tools/Host filesystem: location our dev'l tools
- Target filesystem: filesystem to run on TI processors
- Notice the similarities between the two different Linux filesystems
- When releasing to production, it's common to further reduce the target filesystem to eliminate cost

Workshop Files (SDK 4.x)



Code Gen Tools

Lab exercises working directory:

- **Lab files** – where you'll do your work
- **Linux Kernel and ARM rootfile system***
- **Lab solutions**
- **targetfs*** (linked to: ~/psp_rebuild_omap3/linux_filesys)

DVSDK (TI libraries):

- BIOS
- Codec Engine
- XDC tools
- Etc.

`/mnt/hgfs/shared`

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
  - Linux Fundamentals
  - Linux - Basic Commands
- **Linux Distro's**
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**

Linux Command Summary

Some commands used in this workshop:

File Management
- ls and ls –la
- cd
- cp
- ln and ln –s
- mv
- rm
- pwd
- tar (create, extract .tar and tar.gz files)
- chmod
- chown
- mkdir
- mount, umount (in general, what is "mounting" and how do you do it?)
- alias
- touch

Network
- /sbin/ifconfig, ifup, ifdown
- ping
- nfs (What is it? How to share a folder via NFS. Mounting via NFS.)

VMware Shared Folders
- /mnt/hgfs/<*shared name*>

Program Control
- <ctrl>-c
- ps, top
- kill
- renice

Kernel
- insmod, rmmod

Linux Users
- root
- user
- su (… exit)

BASH
- What is BASH scripting
- What are environment variables
- How to set the PATH environment variable
- What is .bashrc? (like DOS autoexec.bat)
- man pages
- change command line prompt

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
- **Linux Distro's**
  - What are Distributions?
  - O/S Choices
  - Community Options
  - Commercial Options
  - Commercial vs Community
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
- **Linux Distro's**
  - What are Distributions?
  - O/S Choices
  - Community Options
  - Commercial Options
  - Commercial vs Community
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**

Build It Yourself ?

Quote from kernel.org:

If you're new to Linux, you don't want to download the kernel, which is just a component in a working Linux system. Instead, you want what is called a *distribution* of Linux, which is a complete Linux system.

There are numerous distributions available for download on the Internet as well as for purchase from various vendors; some are general-purpose, and some are optimized for specific uses.

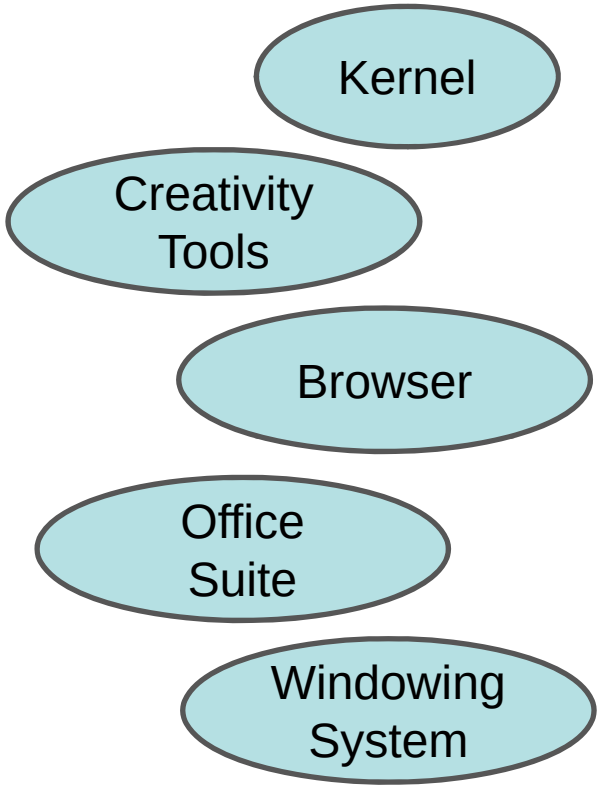
- **This may be a bit of an understatement – even experienced users usually use a distribution**
- **Creating a distribution takes a lot of effort**
- **Maintaining a distribution … takes even more effort**
- **In fact, using a distribution even takes quite a bit of effort**

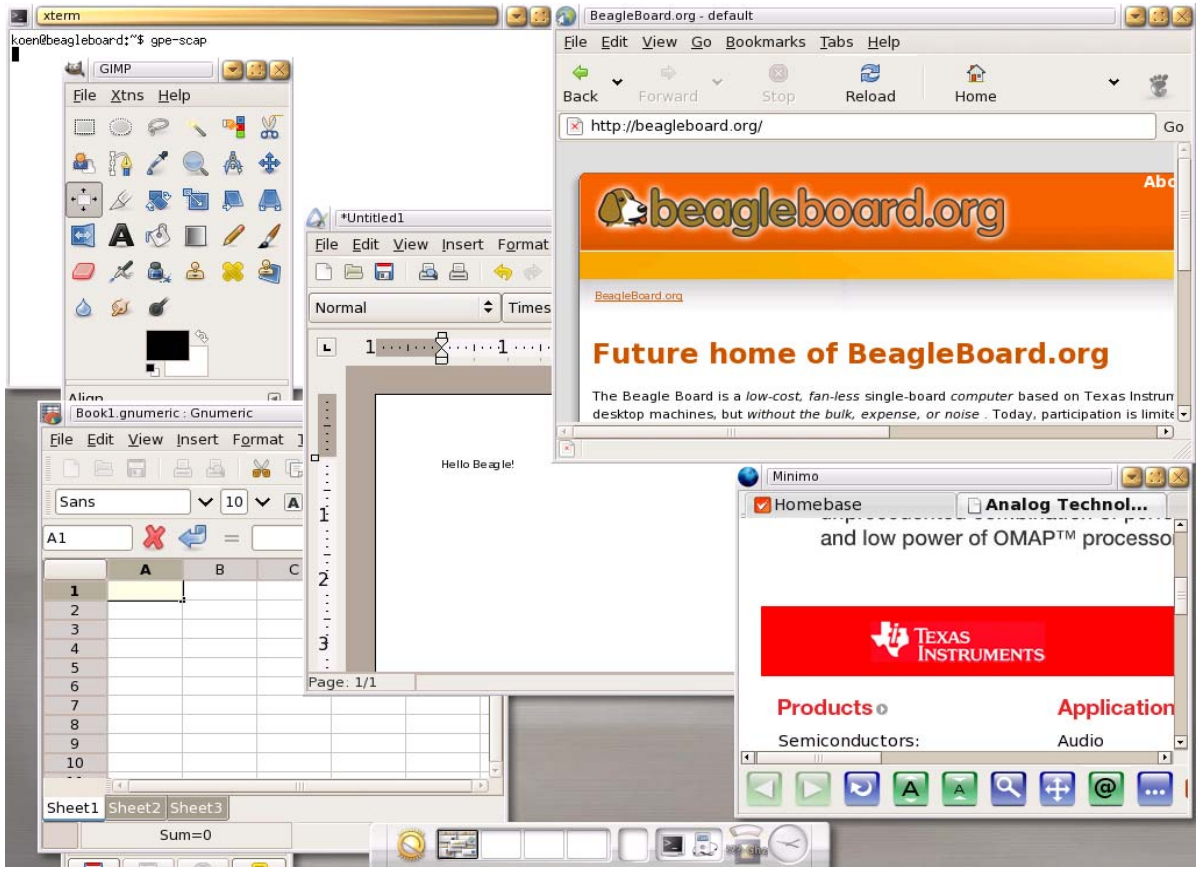
What Is a 'Linux Distribution'

A 'Linux distribution' is a combination of the components required to provide a working Linux environment for a particular platform:

1. **Linux kernel port**
   - A TI LSP or Linux PSP is a Linux kernel port to a device, not just a set of device drivers
2. **Bootloader**
   - Uboot is the standard bootloader for ARM Linux
3. **Linux 'file system'**
   - This does NOT mean a specific type of file system like FAT file system or flash file system … rather, it more like the "C:\" drive in Windows
   - It refers to all the 'user mode' software that an application needs such as graphics libraries, network applications, C run-time library (glibc, uclibc), codec engine, dynamically-loaded kernel modules (CMEM, DSPLINK)
4. **Development tools**
   - CodeSourcery - GCC, GDB
   - MV DevRocket, CCSv5 (beta), GHS Multi, etc.

Linux Distributions



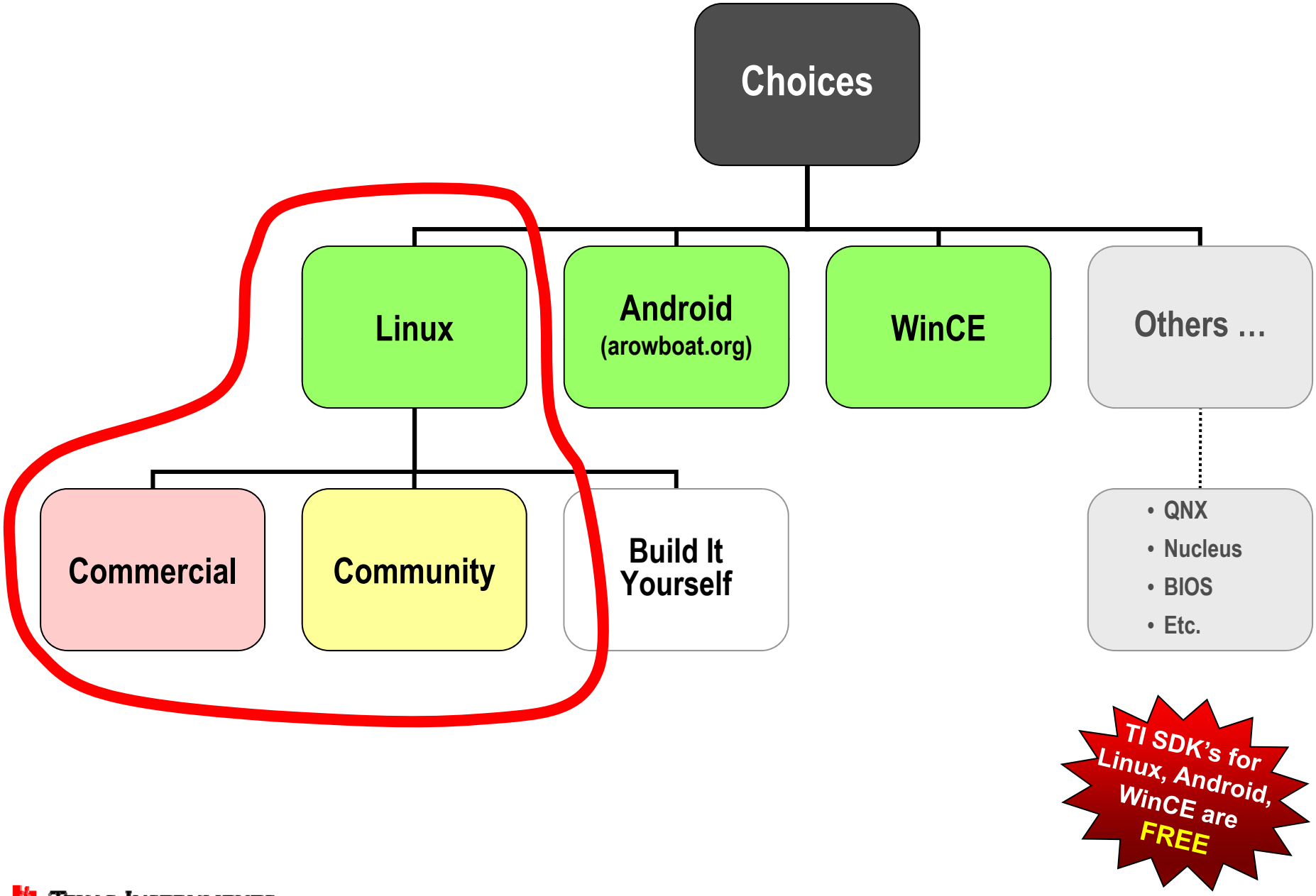


- **Linux isn't complete without a distribution**
- **MontaVista and Timesys, for example, provide commercial (i.e. production) distribution for TI's DaVinci / OMAP processors**
- **A few distributions supporting the open-source BeagleBoard (OMAP35x-based) include: OpenEmbedded, Ubuntu, Fedora, Android, Gentoo, ARMedslack and ALIP**

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
- **Linux Distro's**
  - What are Distributions?
  - O/S Choices
  - Community Options
  - Commercial Options
  - Commercial vs Community
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**

O/S Choices


Choices
Linux
Android
(arowboat.org)
WinCE
Others …
Commercial
Community
Build It
Yourself
• QNX
• Nucleus
• BIOS
• Etc.
TI SDK's for
Linux, Android,
WinCE are
FREE

Linux Distributions Options for TI

Custom (Build it Yourself)		Community		Commercial
Custom from Sources	**Open Embedded (OE)**	**TI SDK (PSP)**	**Ångström**	◆ **Timesys** ◆ **MontaVista** ◆ **Mentor** ◆ **RidgeRun**
◆ "GIT" from kernel.org, and others	◆ Bit-Bake ◆ Recipies	◆ OE / GIT ◆ Binary Updated for each SDK release	◆ Binary ◆ Narcissus (online tool) ◆ OE	◆ Source ◆ Binary (Update patches)

Ease of Use

- *Expert User (only)*
- Latest

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
- **Linux Distro's**
  - What are Distributions?
  - O/S Choices
  - Community Options
  - Commercial Options
  - Commercial vs Community
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**

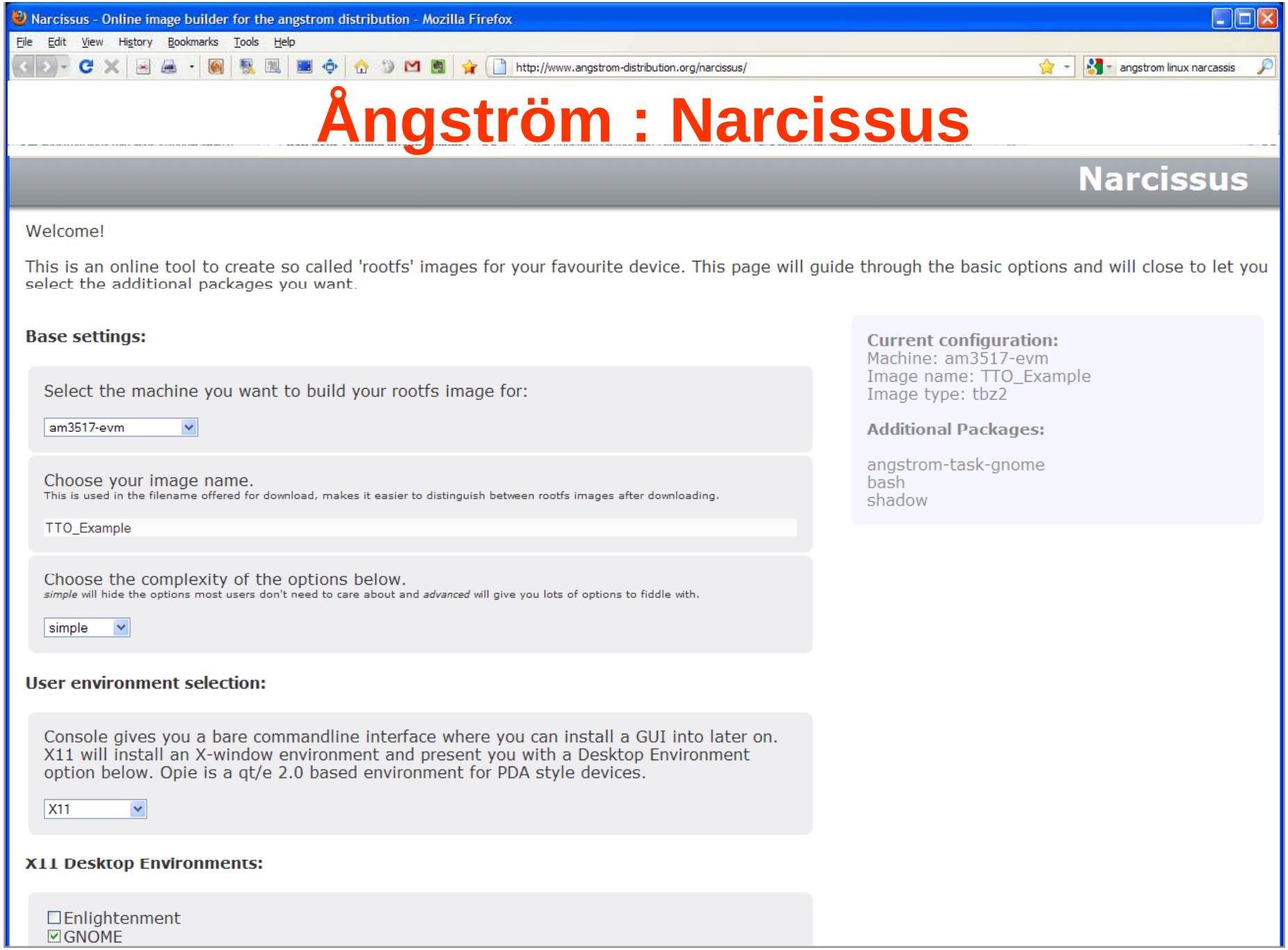
Community Options

- **TI Linux SDK (PSP)**
  - Pre-built snapshot of Linux tested against specific version of TI Software Development Kits
  - Updated at each new SDK/DVSDK release
  - PSP = Platform Support Package (name of TI team)
  - Currently, a “BusyBox-based” bare-bones distro (“lean/mean”)
  - Arago open-source OE project
    - Advantage of OE – recipies can be reused by Angstrom (or custom OE) users
    - In general, users shouldn’t (re)build using OE; no reason to, because if you want more features, we recommend you go with Angstrom (also built using OE)
- **Ångström**

…

Community Options

- **TI Linux SDK (PSP)**
  - Pre-built snapshot of Linux tested against specific version of TI Software Development Kits
  - Updated at each new SDK/DVSDK release
  - PSP = Platform Support Package (name of TI team)
  - Currently, a “BusyBox-based” bare-bones distro (“lean/mean”)
  - Arago open-source OE project
    - Advantage of OE – recipies can be reused by Angstrom (or custom OE) users
    - In general, users shouldn’t (re)build using OE; no reason to, because if you want more features, we recommend you go with Angstrom (also built using OE)
- **Ångström**
  - Open-source, full-featured Linux distro targeted for embedded systems
  - Get it from:
    - User-compiled binaries widely available for many targets
    - **Narcissus** (http://www.angstrom-distribution.org/narcissus) Web-based tool creates binary vers (w/ your own package sel’n)
  - Built using OE (user community can re-use TI OE recipies)



(http://www.angstrom-distribution.org/narcissus)

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
- **Linux Distro's**
  - What are Distributions?
  - O/S Choices
  - Community Options
  - Commercial Options
  - Commercial vs Community
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**

Commercial O/S Vendors

- **Linux**
  - **TimeSys**
  - **MontaVista**
  - **Wind River**
  - **Mentor**
  - **Ridgerun**
- **WinCE**
  - Adeneo
  - Mistral
  - MPC Data
  - BSQUARE
- **RTOS**
  - Green Hills
  - Wind River (VxWorks)
  - ELogic (ThreadX)
  - QNX
  - Mentor (Nucleus)

Linux Partner Strategy

- **Commercial**: provide support, off-the-shelf Linux distributions or GNU tools
- **Consultants:** provide training, general embedded Linux development expertise, or specific expertise for developing drivers or particular embedded applications
- http://www.tiexpressdsp.com/index.php/Linux_Consultants_and_Commercial_Linux_Providers

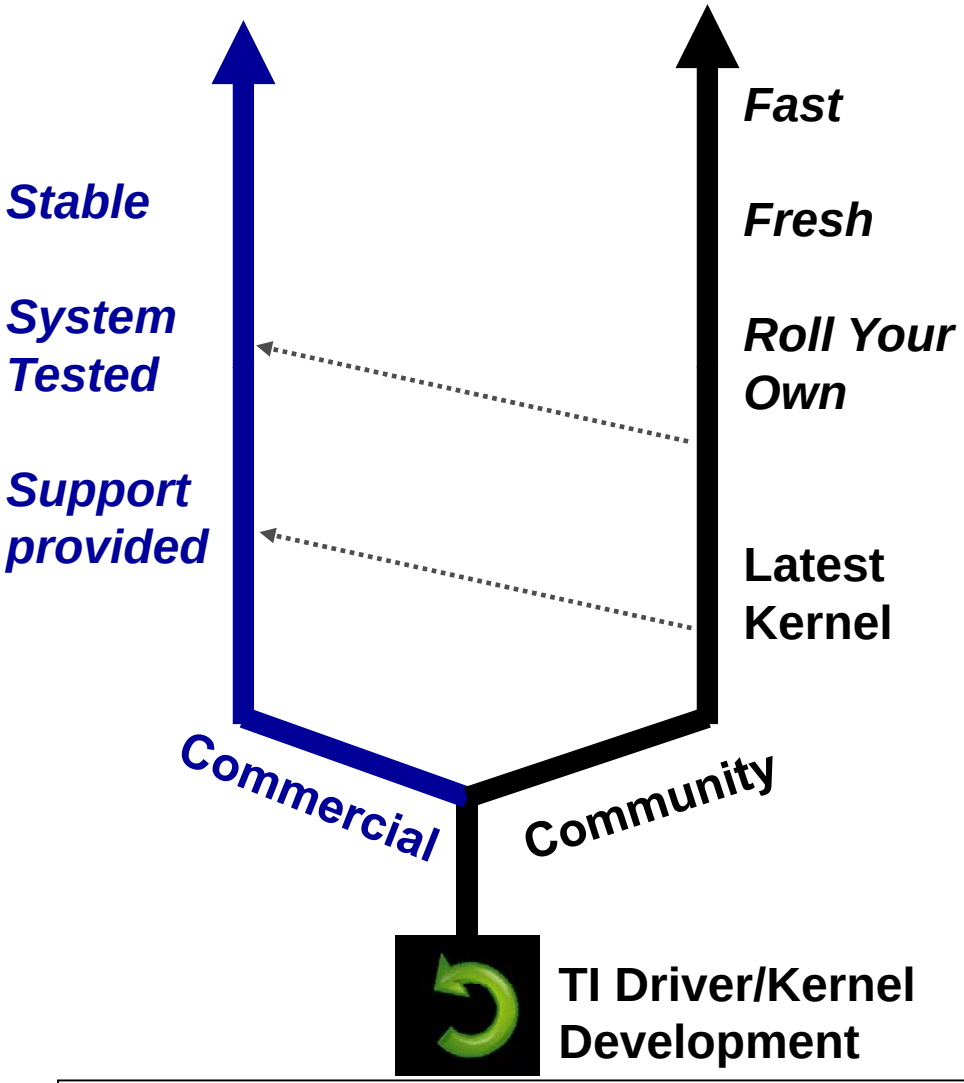
Commercial Linux Product Partners

Vendor	Business Model	Cost	Devices
Timesys	Web-based sales/support + distributors. Standard product, some services. Free version for development boards.	$	AM3517, OMAP35xx, OMAP-L137, OMAP-L138, DM365, DM644x (Linux and Android services)
MontaVista	Worldwide direct sales, standard products, services	$$	MVL 5.0 products are nearly obsolete. MVL 6.0 releases for OMAP3530 available, OMAP-L138, AM3517 in process.
Mentor	Worldwide direct sales, products only available through services engagement	$$	OMAP35xx only today; more coming. Mentor is current Android partner.
WindRiver	Worldwide direct sales, standard products, services, only vendor to offer multi-year support for a fixed Linux version	$$	WR Linux releases for OMAP35xx and OMAP-L138
Ridgerun	Web-based sales/support. Standard SDKs and services.	$	OMAP35xx, DM355/335, DM365, DM6446, OMAP-L138. Has good gstreamer experience.

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
- **Linux Distro's**
  - What are Distributions?
  - O/S Choices
  - Community Options
  - Commercial Options
  - Commercial vs Community
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**

TI Customers Can CHOOSE a Linux Path: Community or Commercial



Community first path

- TI delivers LSP/DVSDK to community
- Smaller set of applications
- Customer builds up solution
- Open source assets
- Customer assets
- Faster access, newer kernels
- More customer responsibility
  - Invest own resources vs. $$

Commercial complement path

- Commercial Linux partner pulls from community
- Partner adds value: production testing, tools integration, support, application bundles, etc. for customers
- Service and subscription sales
- Executing with MontaVista, Timesys…
- Opportunities for other commercial vendors

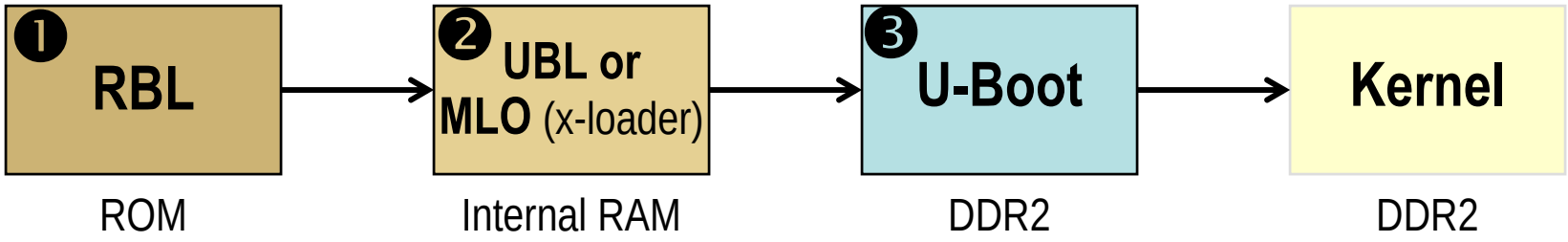
Outline

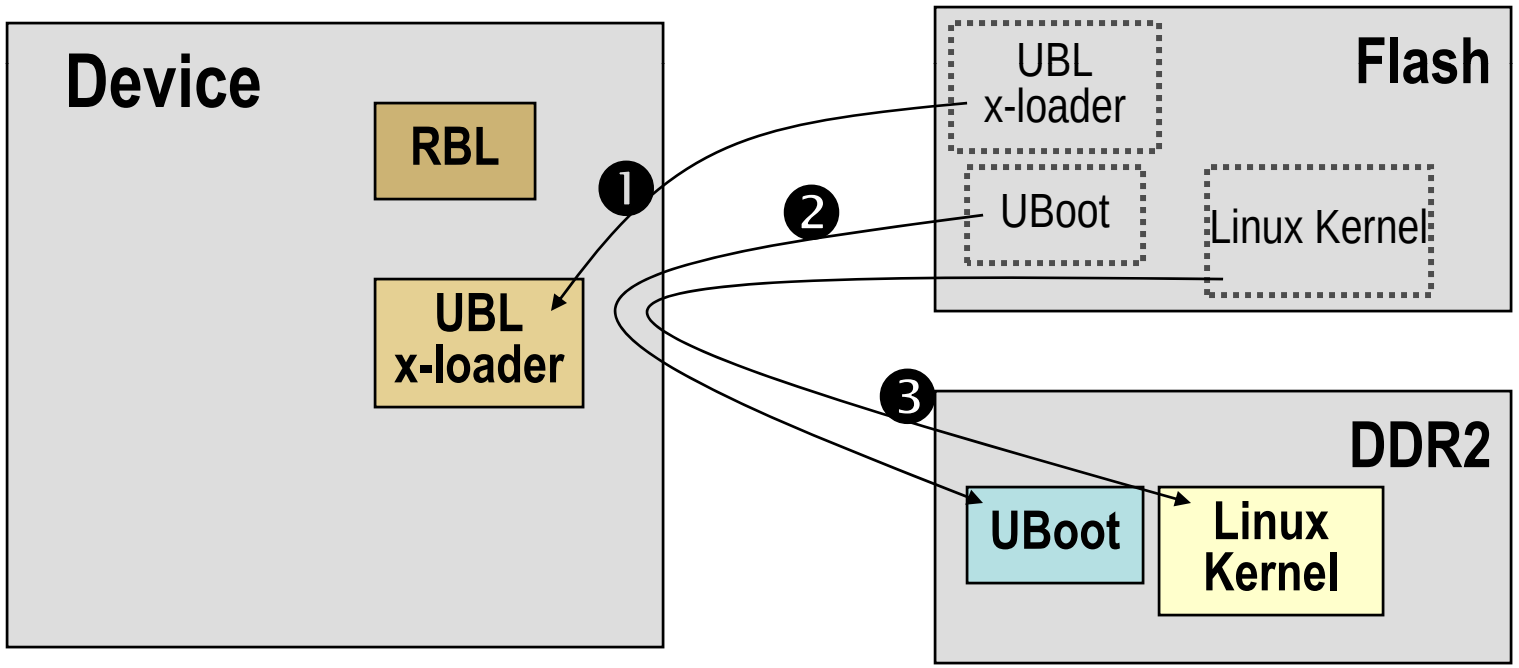
- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
- **Linux Distro's**
- **Booting the Device (Das U-Boot)**
  - Boot Sequence
  - Boot Image Location(s)
  - Configuring Uboot
- **Lab Exercise**

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
- **Linux Distro's**
- **Booting the Device (Das U-Boot)**
  - Boot Sequence
  - Boot Image Location(s)
  - Configuring Uboot
- **Lab Exercise**

Booting Linux – ROM to Kernel





Bootloader Components

Boot stage	Operations	User Config'd	DaVinci	OMAP3x
First-level	This is ROM'd code for detecting desired boot type (NAND, UART, …) and loading executable code of second-level bootloader from selected peripheral/interface	**No**	**RBL**	**RBL**
Second-level	The primary function of this boot loader is to initialize external memory and system clocks so that a larger, more advanced boot loader (in this case U-boot) can be loaded.	**Board Designer**	**UBL**	**XLDR** (MLO)
Linux boot	"Das U-boot" is the standard open-source Linux boot loader for ARM. It supports networking for TFTP/NFS booting. It is used to locate, load and execute the Linux kernel in uImage format and is also responsible for passing arguments to the kernel	**Yes**	**U-boot**	**U-Boot**

Customizing UBL / XLDR

1. Configure system clocks
2. Setup memory interfaces

** In this workshop we will only configure the 3rd level bootloader (Das U-boot).*

U-boot...

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
- **Linux Distro's**
- **Booting the Device (Das U-Boot)**
  - Boot Sequence
  - Boot Image Location(s)
  - Configuring Uboot
- **Lab Exercise**

To Boot Linux, You Need…

1. Bootloader (U-Boot)
2. Linux Kernel
3. Filesystem

- **At reset, U-Boot bootloader is executed**
- **U-Boot loads O/S kernel into DDR2 memory; then,**
- **Connects to the root filesystem**
  If you don't know what this is, think of it as the 'c:\' drive of in Windows PC

Where are these 3 located?

Where Do You Find …

Where located:	DM6446 EVM Default		
1a. UBL or Xloader/MLO	Flash		
1b. Bootloader (U-Boot)	Flash		
2. Linux Kernel	Flash		
3. Filesystem	Hard Drive		

"HDD boot"

- By default, the DM6446 DVEVM ships in "HDD boot" mode; this allows the demo applications to run "out-of-the-box"
- OMAP3530 & AM3517 ship with boot code in NAND. An MMC card demo also ships with the EVM's. Also, the SDK provides an MMC image

Another option is ...

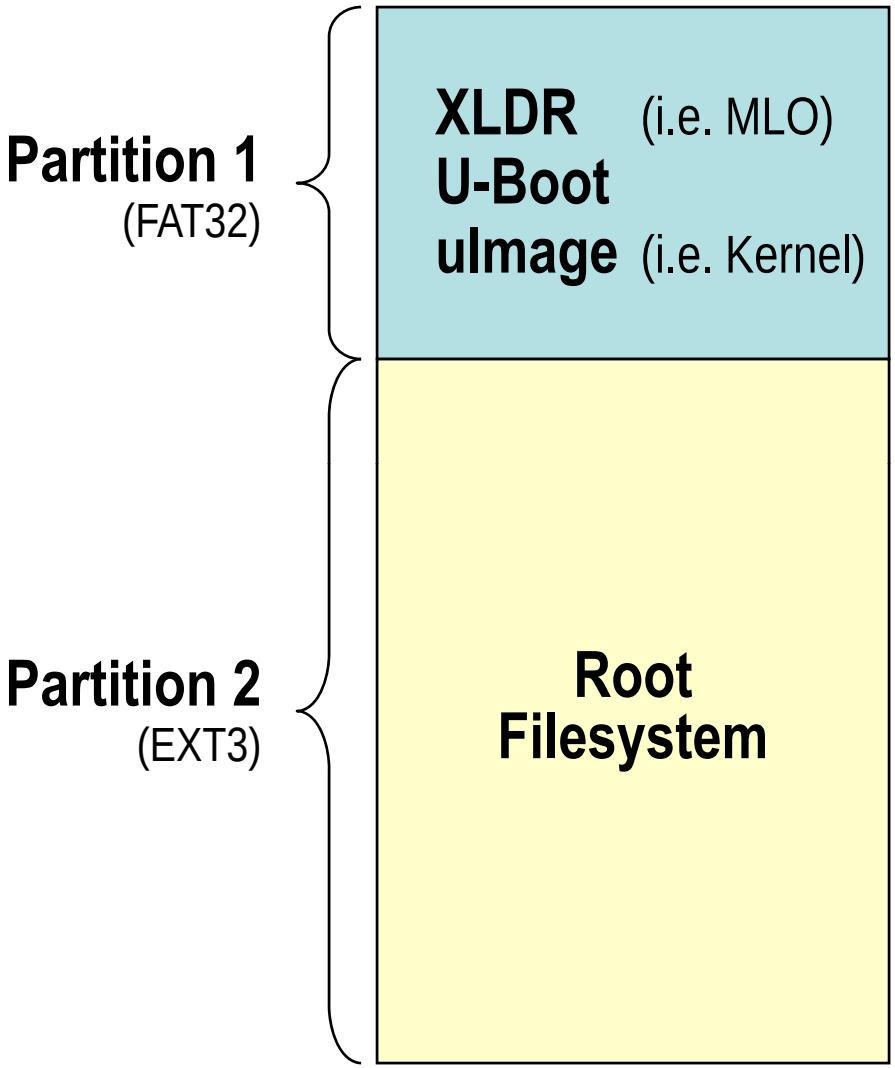
Where Do You Find …

Where located:	DM6446 EVM Default	AM3517 1-day Wkshp	
1a. UBL or Xloader/MLO	Flash	MMC	
1b. Bootloader (U-Boot)	Flash	MMC	
2. Linux Kernel	Flash	MMC	
3. Filesystem	Hard Drive	MMC	
	"HDD boot"	"MMC boot"	

- By default, the DM6446 DVEVM ships in "HDD boot" mode; this allows the demo applications to run "out-of-the-box"
- OMAP3530 & AM3517 ship with boot code in NAND Flash. Also, the SDK provides an MMC image you can burn to a card.

The MMC card would look like …

SD / MMC Boot



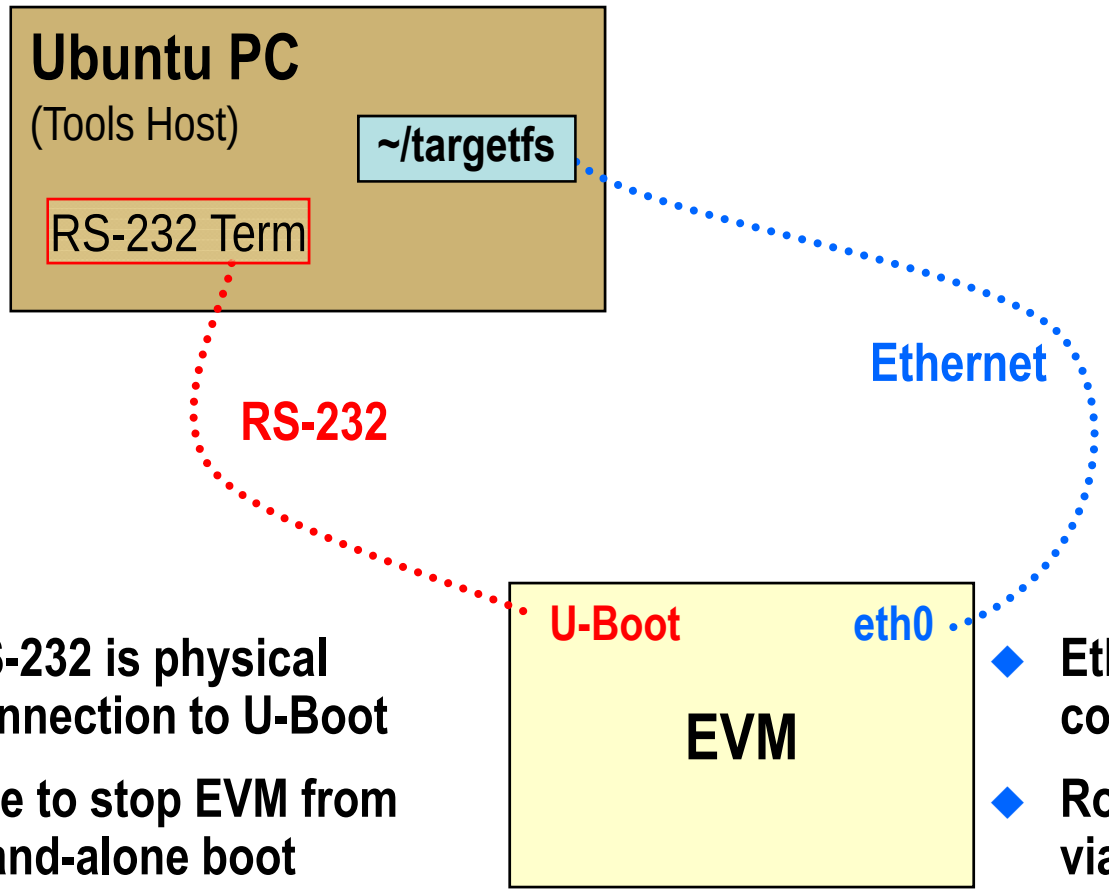
A 3rd boot option is ...

Where Do You Find …

Where located:	DM6446 EVM Default	AM3517 1-day Wkshp	Good for Development
1a. UBL or Xloader/MLO	Flash	MMC	Flash or MMC
1b. Bootloader (U-Boot)	Flash	MMC	Flash or MMC
2. Linux Kernel	Flash	MMC	TFTP (from Ubuntu)
3. Filesystem	Hard Drive	MMC	NFS (from Ubuntu)
	"HDD boot"	"MMC boot"	"NFS boot"

- By default, the DM6446 DVEVM ships in "HDD boot" mode; this allows the demo applications to run "out-of-the-box"
- OMAP3530 & AM3517 ship with boot code in NAND. An MMC card demo also ships with the EVM's. Also, the SDK provides an MMC image
- "NFS boot" (network boot) is good for application development

NFS Boot



- RS-232 is physical connection to U-Boot
- Use to stop EVM from stand-alone boot
- Configure U-Boot's modes by setting/saving environment variables

- Ethernet provides physical connection to Linux PC
- Root filesystem is accessed via NFS protocol
- Don't need to 'flash' EVM after compiling new program

Note: `~/targetfs` = `/home/user/targetfs`

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
- **Linux Distro's**
- **Booting the Device (Das U-Boot)**
  - Boot Sequence
  - Boot Image Location(s)
  - Configuring Uboot
- **Lab Exercise**

Das U-Boot

- The Linux PSP SDK board is delivered with the open-source boot loader: Das U-Boot (U-Boot)
- At runtime, U-Boot is usually loaded in on-board Flash or an SD/MMC card
- In general, U-Boot performs the functions:
  1. Initializes the DaVinci EVM hardware
  2. Provides boot parameters to the Linux kernel
  3. Starts the Linux kernel

Configuring U-Boot and Starting Linux (5 Steps)

1. **Connect an RS232 serial cable and start a Tera Term**
2. **Power on the DVEVM and press any key in TeraTerm to abort the boot sequence**
3. **Set U-Boot variables to select how Linux will boot (save changes to flash to retain settings after power cycle)**
4. **Boot Linux using either:**
   - **the U-Boot "boot" command**
   - **power-cycle the DVEVM**
5. **After Linux boots, log in to the DVEVM target as "root"**

- Note, login with: "user" for the Tools Linux PC
  "root" for the DVEVM target
- You can use any RS-232 comm application (Linux or Win), we use Tera Term for its macro capability

What U-Boot param's do I need?

Configuring U-Boot

Common Uboot Commands:

- printenv - prints one or more uboot variables
- setenv - sets a uboot variable
- saveenv - save uboot variable(s)
- run - evaluate a uboot variable expression
- ping - (debug) use to see if Uboot can access NFS server

Common Uboot Variables:

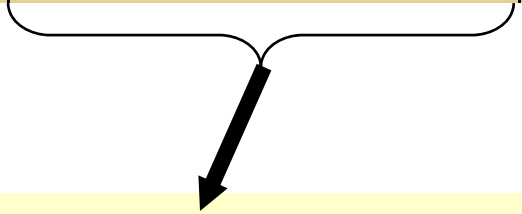
- You can create whatever variables you want, though some are defined either by Linux or common style
  - bootcmd - where Linux kernel should boot from
  - bootargs - string passed when booting Linux kernel
    e.g. tells Linux where to find the root filesystem
  - serverip - IP address of root file system for NFS boot
  - nfspath - Location on serverip for root filesystem

Boot Variations

Mode	IP	Linux Kernel	Root Filesystem
1.	dhcp	Flash	HDD
2.	dhcp	Flash	NFS
3.	dhcp	TFTP	HDD
4.	dhcp	TFTP	NFS
5.	dhcp	MMC	NFS
6.	dhcp	MMC	MMC

Boot Variations (kernel)

Mode	IP	Linux Kernel	Root Filesystem
1.	dhcp	Flash	HDD
2.	dhcp	Flash	NFS
3.	dhcp	TFTP	HDD
4.	dhcp	TFTP	NFS
6.	dhcp	MMC	MMC



U-Boot's bootcmd variable specifies the root filesystem

Flash	`setenv bootcmd bootm 0x2050000`
MMC	`setenv bootcmd "mmc init; fatload mmc 0 ${loadaddr} uImage; run mmcargs; bootm ${loadaddr}"`
TFTP	`setenv bootcmd 'dhcp;bootm'`

Boot Variations (filesystem)

Mode	IP	Linux Kernel	Root Filesystem
1.	dhcp	Flash	HDD
2.	dhcp	Flash	NFS
3.	dhcp	TFTP	HDD
4.	dhcp	TFTP	NFS
5.	dhcp	MMC	MMC

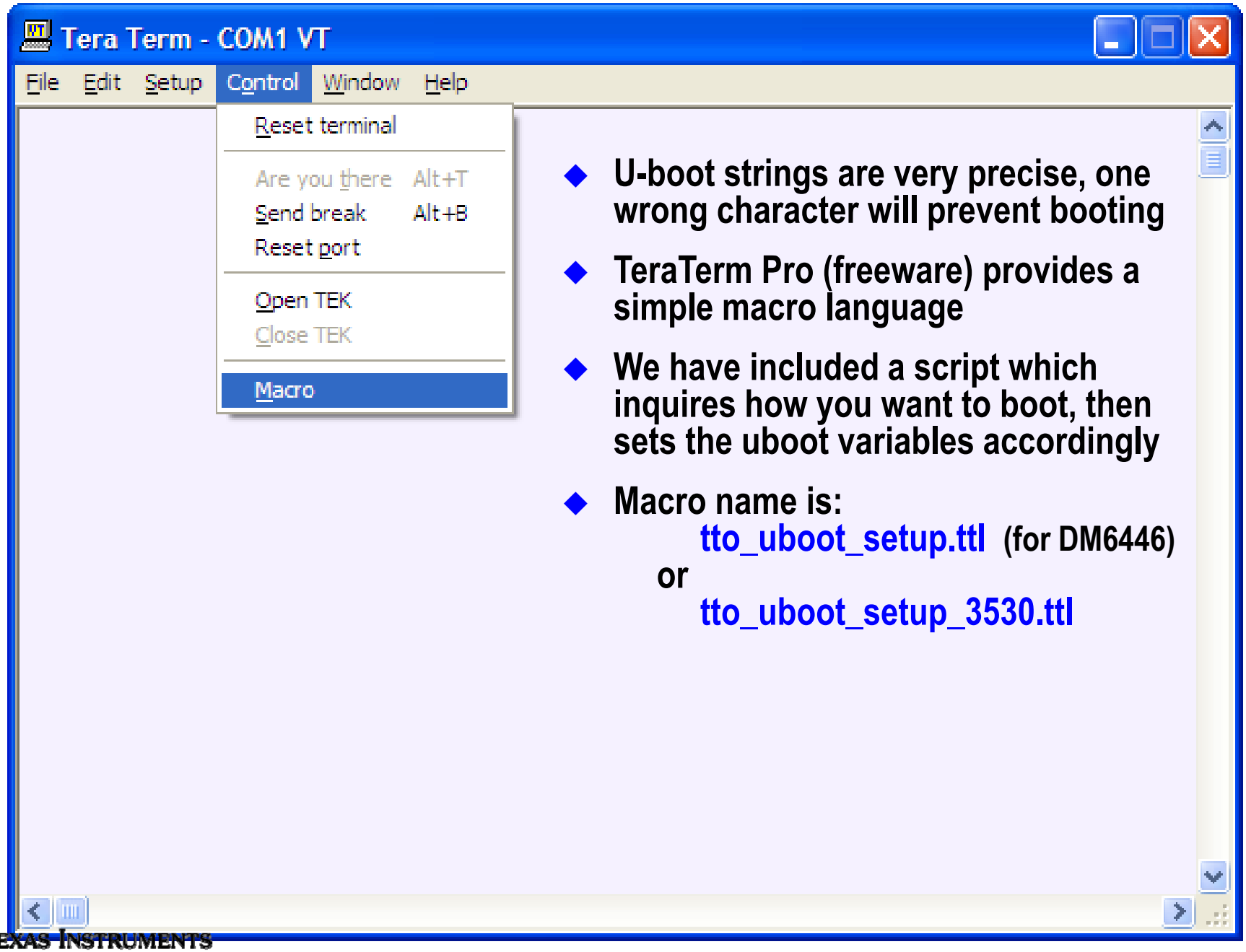
U-Boot's bootargs variable specifies the root filesystem

HDD	`setenv bootargs console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/hda1, nolock mem=120M`
MMC	`setenv bootargs console=ttyS0,115200n8 noinitrd root=/dev/mmcblk0p2 rootfstype=ext3 rootwait nolock mem=120M`
NFS	`setenv bootargs console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/nfs nfsroot=$(serverip):$(nfspath),nolock mem=120M`

Configuring U-Boot Kernel via TFTP, Filesystem from NFS (network)

```
[rs232]# baudrate 115200
[rs232]# setenv stdin serial
[rs232]# setenv stdout serial
[rs232]# setenv stderr serial
[rs232]# setenv bootdelay 3
[rs232]# setenv bootfile uImage
[rs232]# setenv serverip 192.168.2.101
[rs232]# setenv nfspath /home/user/workdir/filesys
[rs232]# setenv bootcmd 'dhcp;bootm'
[rs232]# setenv bootargs console=ttyS0,115200n8
     noinitrd rw ip=dhcp root=/dev/nfs
     nfsroot=$(serverip):$(nfspath),nolock
     mem=120M
[rs232]# saveenv
```

Using Tera Term Macros



- U-boot strings are very precise, one wrong character will prevent booting
- TeraTerm Pro (freeware) provides a simple macro language
- We have included a script which inquires how you want to boot, then sets the uboot variables accordingly
- Macro name is:
  tto_uboot_setup.ttl (for DM6446)
  or
  tto_uboot_setup_3530.ttl

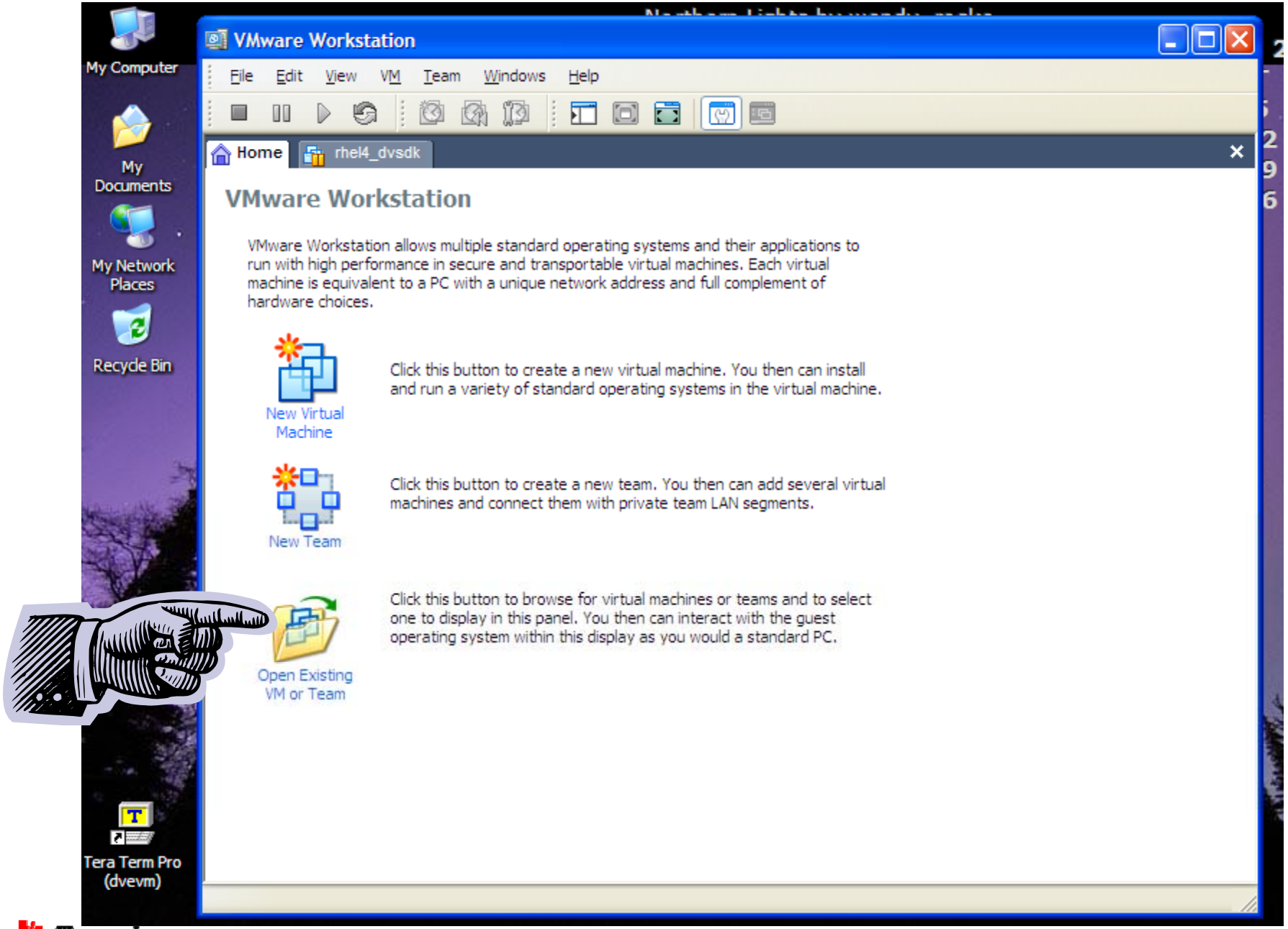
Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
- **Linux Distro's**
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**
  - VMware
  - Lab Setup/Procedure

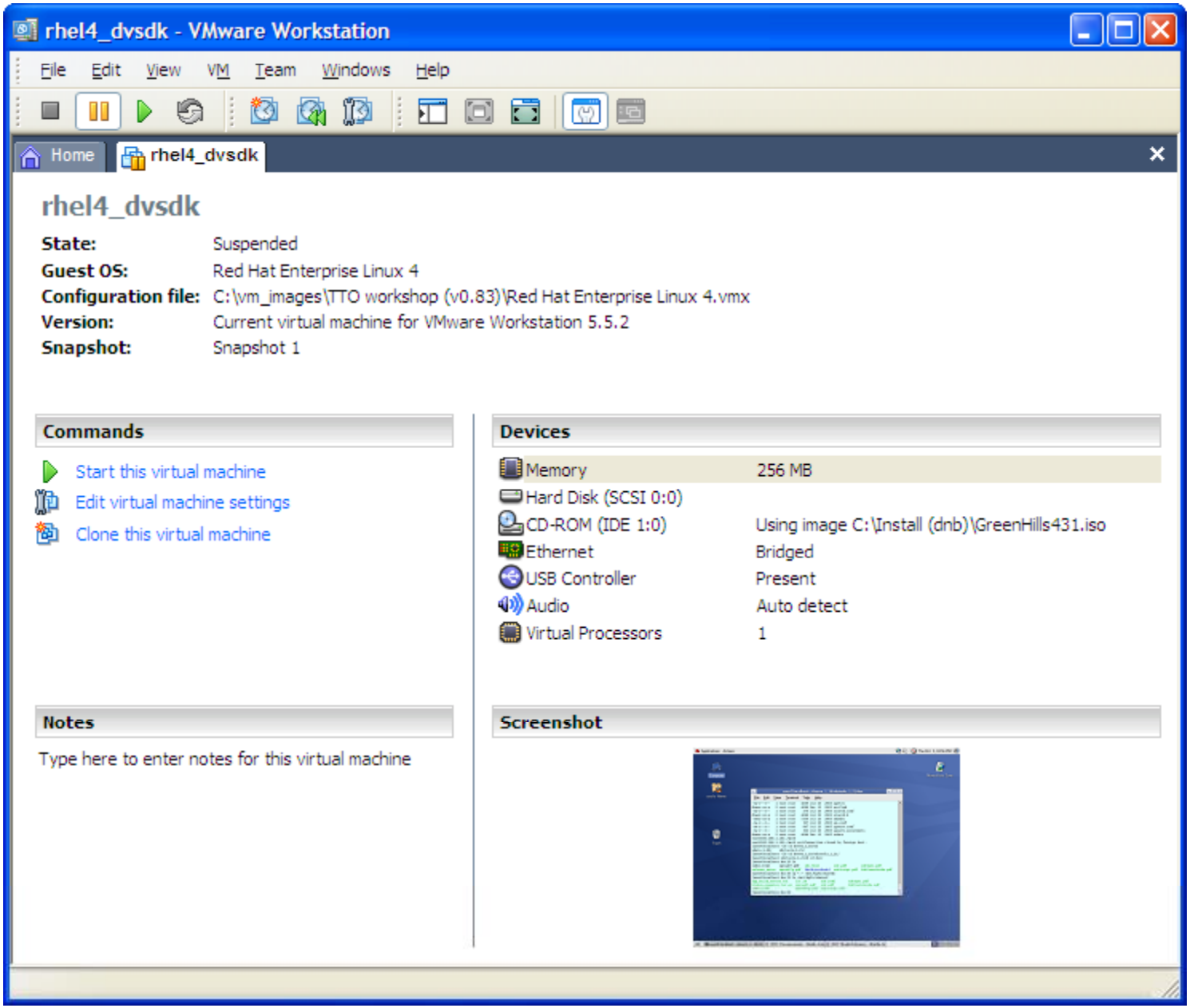
Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
- **Linux Distro's**
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**
  - VMware
  - Lab Setup/Procedure

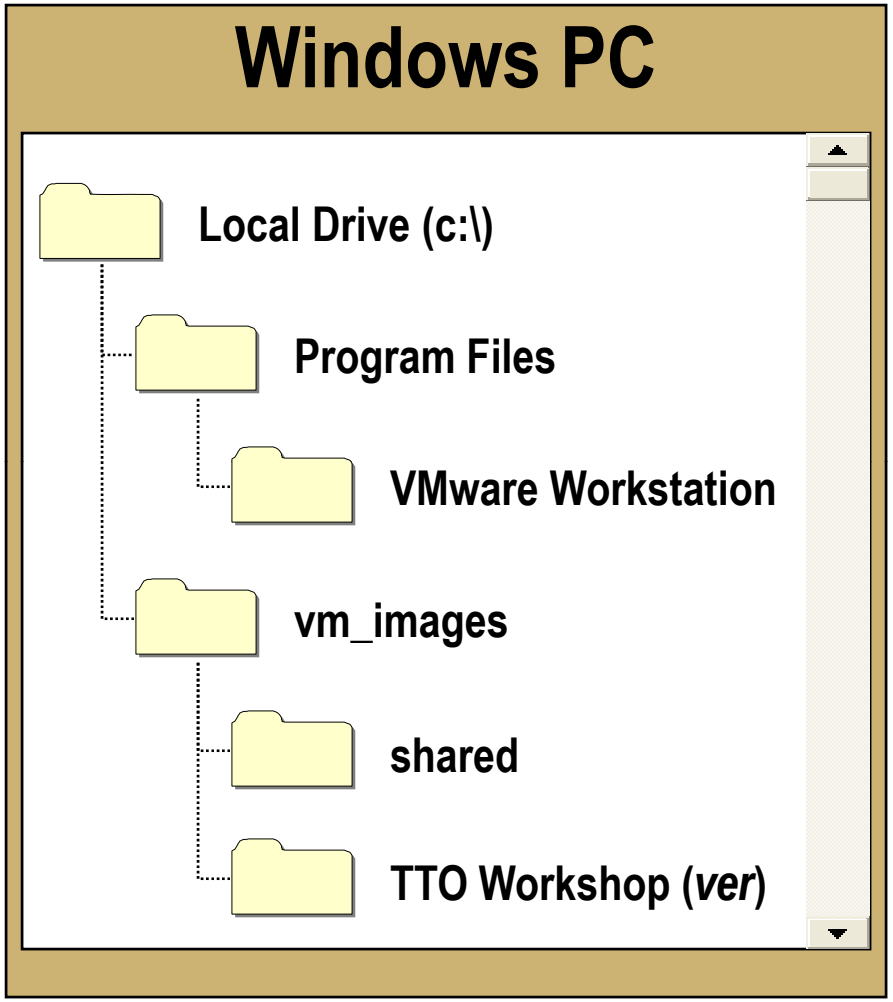
VMware – Run Linux inside Windows Application



VMware


rhel4_dvsdk - VMware Workstation
File Edit View VM Team Windows Help
Home
rhel4_dvsdk
rhel4_dvsdk
State: Suspended
Guest OS: Red Hat Enterprise Linux 4
Configuration file: C:\vm_images\TTO workshop (v0.83)\Red Hat Enterprise Linux 4.vmx
Version: Current virtual machine for VMware Workstation 5.5.2
Snapshot: Snapshot 1
Commands
Start this virtual machine
Edit virtual machine settings
Clone this virtual machine
Devices
Memory 256 MB
Hard Disk (SCSI 0:0)
CD-ROM (IDE 1:0) Using image C:\Install (dnb)\GreenHills431.iso
Ethernet Bridged
USB Controller Present
Audio Auto detect
Virtual Processors 1
Notes
Type here to enter notes for this virtual machine
Screenshot

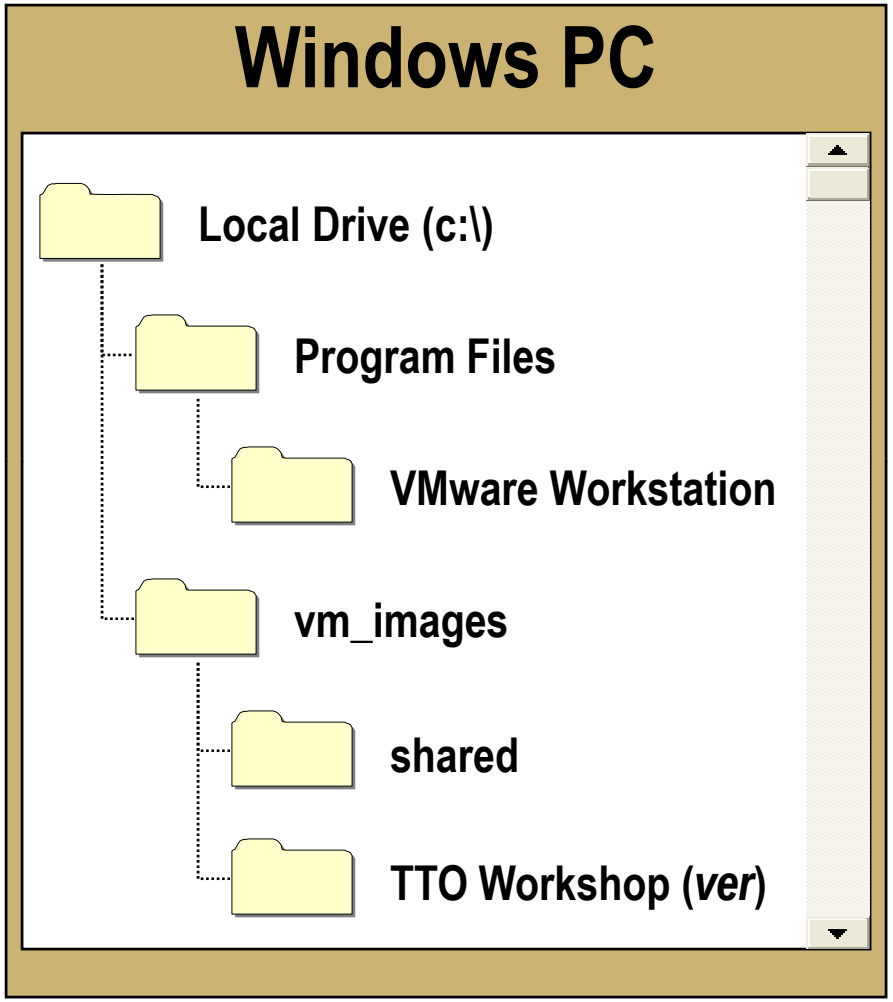
VMware – Virtual Machine



Why VMware?

- Allows simultaneous use of Linux and Windows with one PC
- Virtual Linux PC within a Windows application
- VMware provides free “player” version of their software
- Virtual PC settings and hard disc image are stored inside any Windows folder
- Easily distribute virtual Linux PC with all DaVinci tools pre-installed
- By keeping known “good” copy of VMware image, you can easily reset Linux PC

Workshop VMware Image



Workshop VMware Images

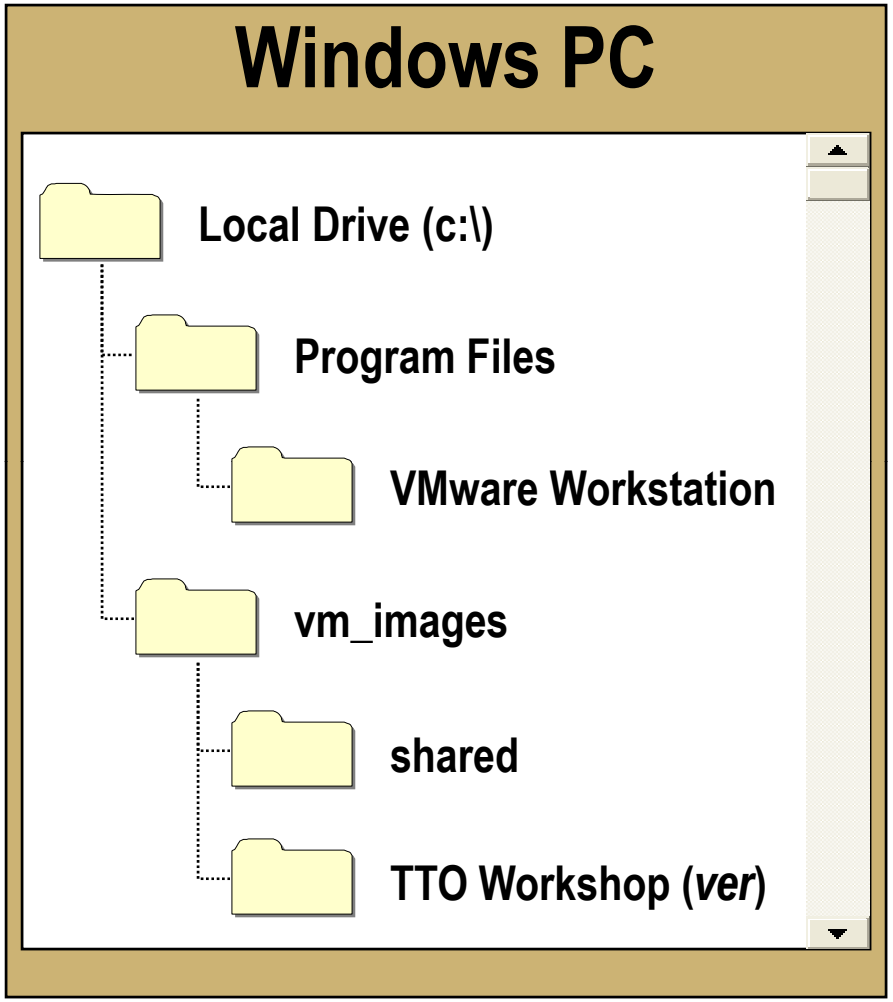
Notes:

- Screensaver & Firewall off
- NFS, TFTP, GIMP installed
- VMware toolbox installed

OMAP3530/AM3517 Labs:

- **Ubuntu 10.04**
- **id = user, psw = *none***
- **DVSDK/SDK Tools:**
  - Community Linux (Arago)
  - CodeSourcery Toolset

VMware – Free Player vs. Full Workstation



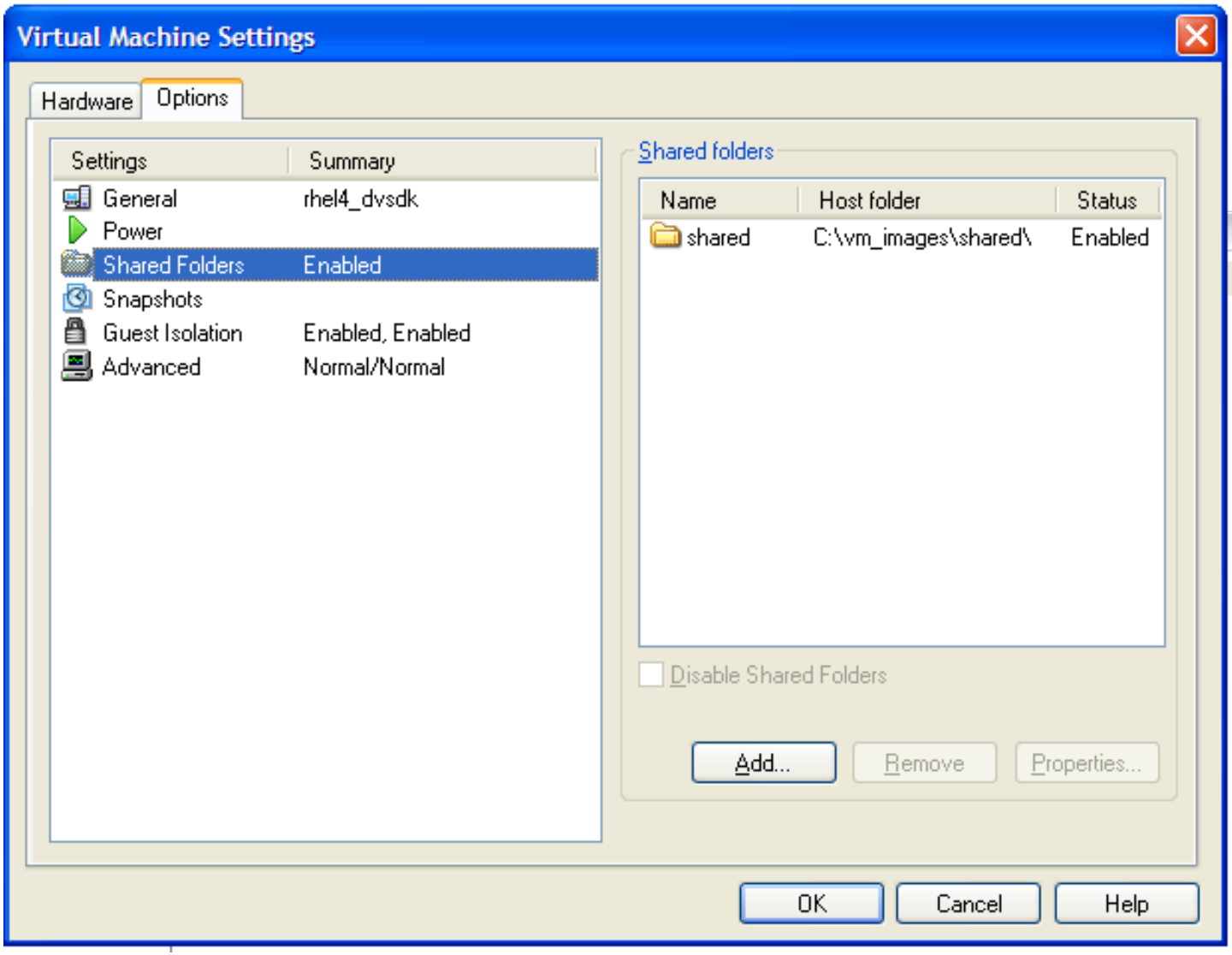
Full Workstation

- Can build VMware PC image from scratch
- "Snapshot" feature allows you to save & restore previous machine states (handy!)
- "Shared Folders" feature makes it easy to share files between Linux and Windows
- Not free, but small discount with current users referral code
- Workstation users get both full/free

Free Player

- Free
- Someone else has to create original VMware image (and do h/w mods)
- No snapshot feature

VMware : Shared Folders



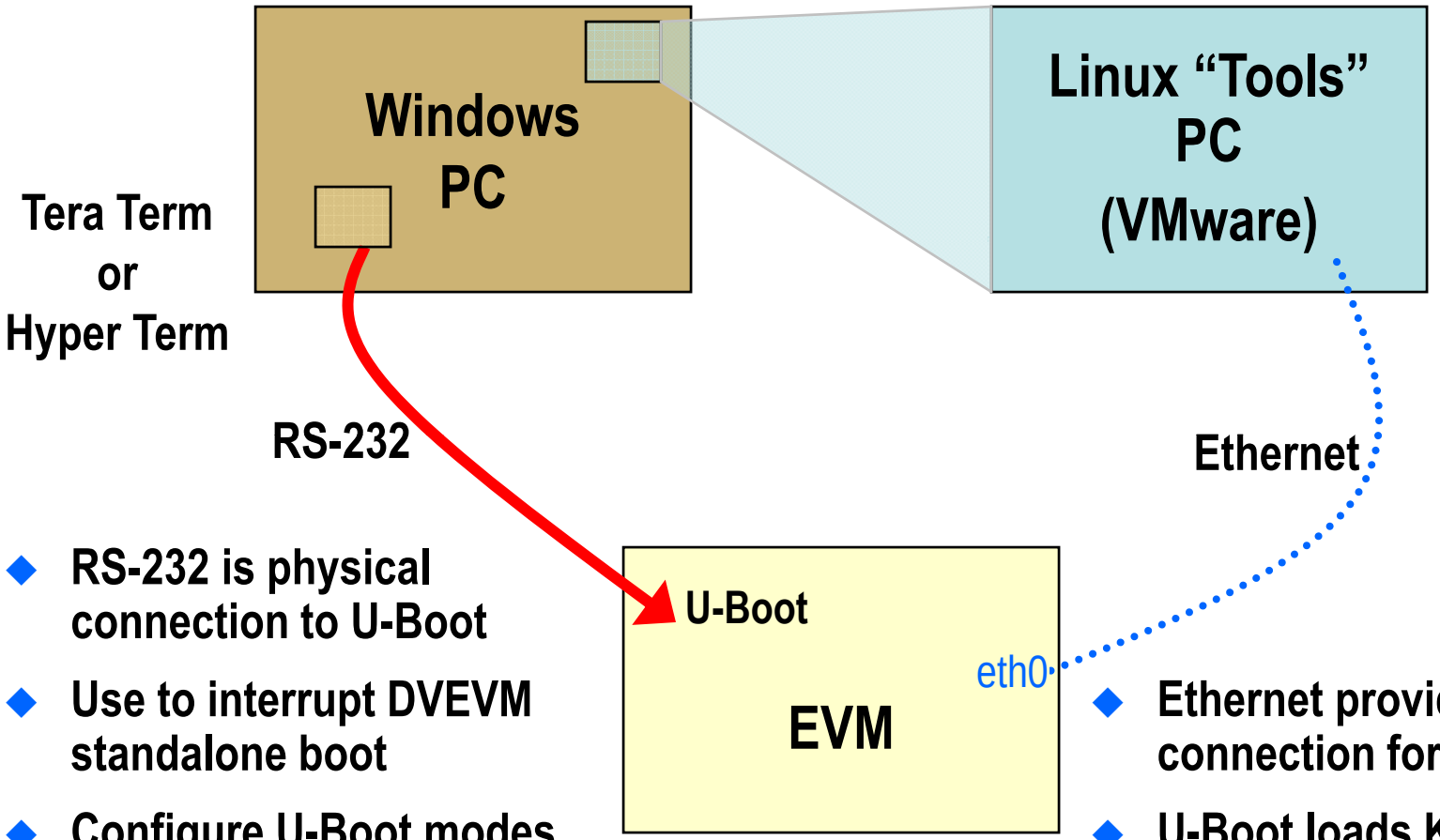
Sharing folders

- VMware shared folders
- NFS
- Samba

VMware Shared Folders

- Easiest method
- Access from: /mnt/hgfs/shared

NFS Boot



- RS-232 is physical connection to U-Boot
- Use to interrupt DVEVM standalone boot
- Configure U-Boot modes by setting/saving environment variables

- Ethernet provides physical connection for booting
- U-Boot loads Kernel from "Tools" PC into DDR2 memory using TFTP
- Filesystem is accessed via NFS protocol

Outline

- **Embedded Processors**
- **Tools – Boards & SDK**
- **Intro to Linux**
- **Linux Distro's**
- **Booting the Device (Das U-Boot)**
- **Lab Exercise**
  - VMware
  - Lab Setup/Procedure

Lab

a) **Start VMware & configure Ubuntu image**

b) **Install workshop lab files to Ubuntu**

c) **Create SD/MMC card image** (so you can boot EVM)

d) **Talk to the EVM** (RS232 and networking)

e) **Configure U-boot and start EVM**

Location:	DM6446	OMAP3530	AM3517
UBL / MLO	NOR Flash	MMC	MMC
U-boot	NOR Flash	MMC	MMC
Kernel (uImage)	TFTP	TFTP	MMC
Filesystem	NFS ~/workdir/filesys	NFS ~/targetfs	MMC


Texas
Instruments