Embedded Linux

Olivier Gruber

Professeur

Université Joseph Fourier

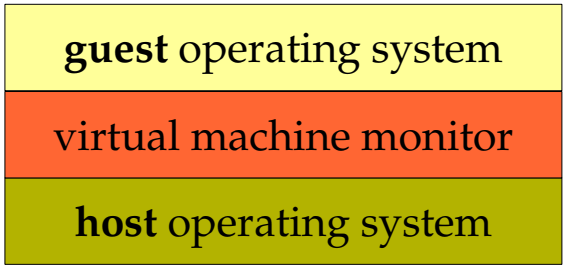
Projet SARDES

(INRIA et IMAG-LSR)

Embedded Linux

- Pre-requisite
  - Be root on your machine
  - Virtual Machine Monitor
    - Download VirtualBox from www.virtualbox.org
  - Linux kernel sources
    - Download Linux kernel sources, suggested version 2.6.23.9
    - From http://www.kernel.org/
    - Or ftp://ftp.free.fr/mirrors/ftp.kernel.org/linux/kernel
  - Grub loader
    - Download Grub loader, version 0.97
    - From ftp://alpha.gnu.org/gnu/grub/

- VirtualBox
  - Virtual Machine Monitor
    - Advanced technology for hosting several guest operating systems
    - Within a process, virtualize a bare hardware
  - Typical use
    - Installing a different operating system
      - Windows on Linux, Linux on Mac-OS or Windows on Mac-OS
    - Simplifies operating system work
      - Safer and faster
      - Ability to show virtual devices
        - Hard disks
        - CD-ROM



- VirtualBox
  - Create a virtual machine
    - With 16MB of RAM
    - With a hard disk of 32MB
  - Next steps
    - Create a bootable CD-ROM image
    - Make it visible to VirtualBox as a boot device
    - Boot from it

- BIOS boot sequence
  - Boot devices setup in the BIOS setup
    - Usually floppy, CD and hard disk
    - Could be also USB devices (not always supported)
  - Hardware boot process
    - Loads first sector (512bytes) of a boot device
    - Jumps in it
- Boot loader
  - Linux kernel is just too large to be loaded directly by the BIOS
  - We need a staged loading process...

- GRUB = GRand Unified Boot-loader
  - From GNU (GNU is Not Unix)
    - Read the README and INSTALL (as always)
  - Configuring
    - ./configure –prefix=PATH
    - **Do** give a PATH to a local directory in your home
    - Otherwise it installs on /boot/grub
  - Building
    - make
  - Installing
    - make install

- GRUB = GRand Unified Boot-loader
  - Two stages
    - Stage1
      - 512byte boot sector
      - Will be installed on the first boot sector of a boot device
      - Will load stage2
    - Stage2
      - 100KB loader
      - Understands certain file system formats
        - MSDOS FAT16 and FAT32
        - Minix fs, Linux ext2, ReiserFS,
        - JFS, XFS, BSD ufs
      - Will load and uncompress the Linux kernel

- Bootable CD-ROM
  - Make an ISO 9660 image
    - CD-ROM data disks use a different file system than hard disks
    - Look at the manual of *mkisofs*
  - Make it bootable
    - GRUB is compatible with booting CD-ROM
    - Through the stage2_eltorito

DO NOT BURN A CD
Make an image (iso file)

- GRUB How-Tos
  - http://www.gnu.org/software/grub/manual/html_node/Installation.html#Installation
    - Installing GRUB natively
    - Making a bootable CD-ROM

- Bootable CD-ROM
  - Make the ISO image visible to your virtual machine
    - Using the disk manager in VirtualBox
  - Boot from it
    - You should see the grub prompt
    - There is not much we can do...
      - We need a bootable Linux CD-ROM
      - So that we can boot from it
      - Partition and format the hard drive
      - Install GRUB on it

Embedded Linux

- Linux kernel
  - Look under /boot
    - vmlinuz-2.6.23.9-xyz
    - System.map-2.6.23.9-xyz
  - Look under /boot/grub
    - You see the GRUB files
  - Kernel itself
    - One executable, fairly large 500KB to 1.5MB compressed
    - System map is about kernel symbols

- Kernel Modules
  - Not all functionality are statically linked in the Linux kernel
    - New device drivers such a network cards or disk controllers
    - New bus supports such as PCI, PCMIA, USB, etc.
    - Higher functions such as IP tables or SCSI support
  - Propose the concept of modules
    - Can be dynamically loaded and unloaded
      - Better usage of kernel memory
      - Supports PnP devices without rebooting
    - Suited for embedded systems?
      - Kernel is a tad larger with module support enabled
      - Need more static footprint since modules are in the file system
        - Under /lib/modules/
      - No single answer...

- Kernel Modules
  - Under /lib/modules/
    - **One hierarchy per version of the kernel**
    - Per kernel version
      - Hierarchy of modules organized by functional themes
      - Look under /lib/modules/x.y.z/kernel

```
# ls /lib/modules/2.6.18.8-0.7-default/kernel
arch  crypto  drivers  fs  kernel  lib  net  security  sound
#
```

- Kernel Modules
  - Modules have dependencies between them
    - Generated at each Linux boot by the command

```
# depmod -a
```

  - Remembered in a modules.dep

```
# ls /lib/modules/2.6.18.8-0.7-default/
CiscoVPN  build  kernel  misc  source  weak-updates
modules.ccwmap      modules.ofmap  modules.usbmap
modules.dep         modules.pcimap
modules.ieee1394map  modules.seriomap
modules.inputmap    modules.symbols
modules.alias  modules.isapnpmap
modules.unsupported
```

- Kernel Modules
  - Manipulating modules
    - Listing modules: **lsmod**
    - Inserting module: **insmod**
    - Removing a module: **rmmod**
  - Dealing with dependencies
    - Use **modprobe** if modules may have dependencies
  - Look at the man pages...

- Kernel file system (/proc)
  - Virtual file system representing the state of the machine
    - A way for the kernel to communicate with user space
  - Example:
    - The command lsmod is in fact reading the information from /proc

```
# cat /proc/modules
```

    - Look at numerical directories under /proc
      - Information about processes
  - Full documentation under
    - /usr/src/linux/Documentation/filesystems/proc.txt
    - Peek around your /proc hierarchy

- Making a bootable Linux CD-ROM
  - Booting
    - We need a GRUB-enabled ISO image
    - We need a kernel and its modules
  - Boot sequence
    - GRUB loads itself
    - Loads and uncompress the Linux kernel image
    - Starts executing the kernel
  - Then what?

- Making a bootable Linux CD-ROM
  - We need an initial process...
    - The **init** process
  - From where?
    - From what file system?
      - The root file system...
      - Parameter to the Linux kernel startup
      - GRUB root command
    - How does the kernel read that file system?

- Making a bootable Linux CD-ROM
  - Initial RamDisk (initrd)
    - The **init** process
    - A minimal file system image
  - Look at /boot/initrd
    - It is a cpio archive compressed by gzip
      - Uncompress and unarchive
    - Or it is a compressed ext2 file system image
      - Uncompress and then mount through the loop driver
  - Mounted as the root file system
    - Mounted on /

- Making a bootable Linux CD-ROM
  - We need to tailor the init process
    - To make minimal so that it works and ends in a shell
  - We need minimal commands
    - Such as ls, cat, mkdir, etc.
    - We need a shell interpreter
  - So we need the necessary libraries...
    - Use ldd to see what the dependencies are
    - The manual approach is tedious
    - Write a shell script to gather the necessary libraries