

EmbeddedCraft
crafting intelligent systems

IMBUENT
Tanner's Technologies

EMBEDDED LINUX Session -2

1

IMBUENT

Topics Covered

- >> Introduction to Embedded Linux
- >> Linux Features
- >> Embedded Linux Features
- >> Kernel Types
- >> Embedded Linux Setup
- >> QEMU
- >> Setting up Host

2

IMBUENT

EMBEDDED LINUX

Embedded Linux means Linux OS running in an Embedded System

wikipedia says...
Embedded Linux is the use of a Linux operating system in an embedded computer systems such as a mobile phones, personal digital assistants, media players and other consumer electronics devices, networking equipment, machine control, industrial automation, navigation equipment and medical instruments.

According to survey conducted by Venture Development Corporation, Linux was used by 18% of embedded engineers

http://en.wikipedia.org/wiki/Embedded_Linux

3

IMBUENT

LINUX Features 1/2

- Linux is **Monolith** kernel with support of Modular architecture.
- Protected mode so programs or user's can't access unauthorized areas.
- Networking with TCP/IP and other protocols.
- Multiple user capability.
- Shared libraries
- True multitasking
- X - A graphical user interface similar to windows, but supports remote sessions over a network.

4

IMBUENT

LINUX Features 2/2

- Advanced server functionality:
 - FTP server
 - Telnet server
 - BOOTP server
 - DHCP server
 - Samba server
 - DNS server
 - SNMP services
 - Mail services
 - Network file sharing
 - much, much more...
- File System Support:

Ext2	ROMFS	RAMFS
JFFS2	PROCF	DEVFS
DOS (FAT),	Windows95,98 (FAT32),	Windows NT,
2000 (NTFS),	Apple,	minix,
and others		

5

IMBUENT

Embedded Linux Features

Configurable kernel
 Configurable features, Configurable size, Configurable functionality

Device Support
 wide range of device are supported like USB, Ethernet etc.

Royalty Free
 No need to pay royalty to for any type of product.

Growing availability of embedded applications
 Database (SQL Lite, Metalite), webserver (Boa, thttpd) Graphics (PEG, Nano)

Open Source
 Source code can be customized for specific need of embedded system

6

GNU Linux

Main components of GNU/Linux operating system

GUI: X.org, XFCE, Gnome, KDE

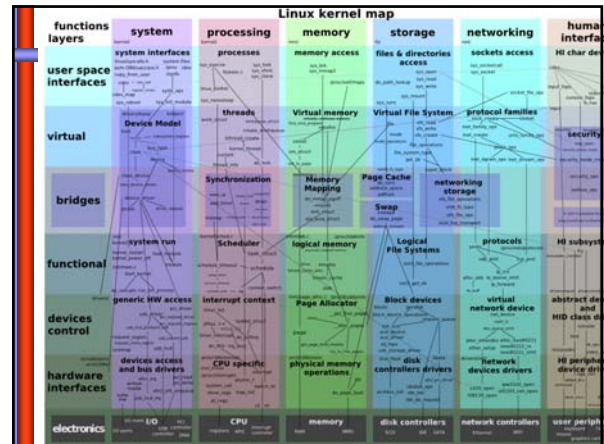
sshd, PHP, MySQL, Apache

gcc, GNU C Library, coreutils, bash, other libraries

Linux kernel

processes, memory management, file systems, sockets, drivers and modules, protocols, computer hardware

7

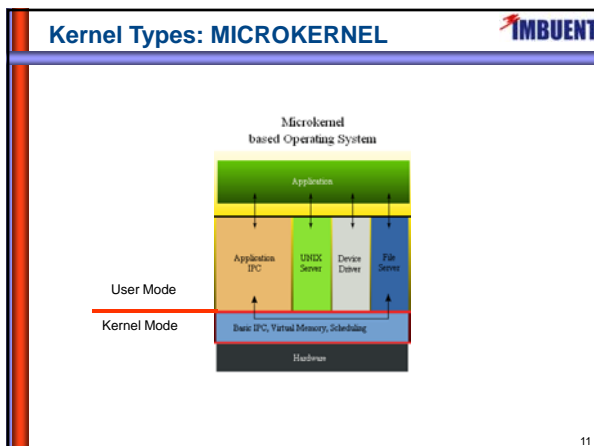
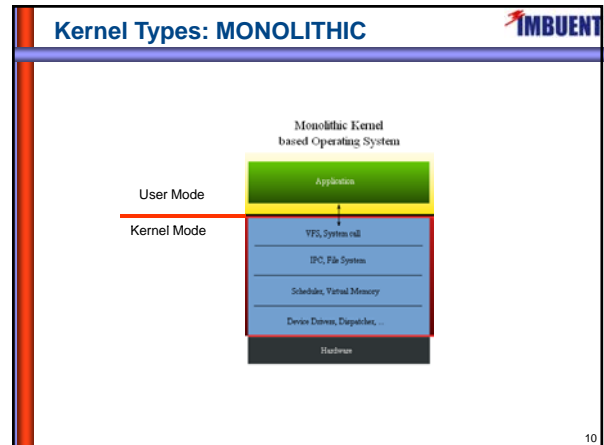


Kernel Types

Two type of kernel mainly:

- >> Monolithic Kernel
- >> Microkernel

9



Kernel Types: tradeoff

Monolithic Kernel

- Kernel is implemented as a single process. i.e. only in one address space
- Design is simple
- Inter process communication is fast.
- Interrupt handling is easy.
- Used in general system, where safety is not main concern.

Microkernel

- Only kernel basic services (scheduling, IPC, VFS) are kept in kernel space,
- Design is complicated
- extra communication is required as everything is a process.
- Interrupt handling might not easy some time.
- Used in safety critical system.

12

EMBEDDED LINUX SETUP

Linux shell at serial terminal

Development System (Host System)

Embedded Board (Target Board)

13

Minimum Requirement for Target Board

- 32 bit processor with MMU (Memory Management Unit) (ARM, PowerPC, ColdFire etc)
- 32MB RAM
- 8MB Flash
- Serial Port
- Ethernet Port
- 20 Pin JTAG Port

14

Example ARM Board SAM9-L9260

15

Example ARM Board SAM9-L9260 Back Side

16

But without board >>>

It's Possible...

Answer is Processor Emulator

QEMU – Open source processor emulator

QEMU says...

When used as a machine emulator, QEMU can run OSes and programs made for one machine (e.g. an ARM board) on a different machine (e.g. your own PC). By using **dynamic translation**, it achieves very good performances.

<http://bellard.org/qemu/>

17

QEMU – Processor Emulator tool

QEMU User Mode (Linux host only)

Can Emulate target CPUs in order to launch Linux Processes

Supports ...

ARM	PowerPC	MIPS	SPARC	X86 etc
-----	---------	------	-------	---------

QEMU System Emulation Mode

Can run complete processor system and peripherals

Supports...

ARM	PowerPC	MIPS	SPARC	X86 etc
-----	---------	------	-------	---------

<http://bellard.org/qemu/>

18

QEMU – Example

QEMU User Mode (Linux host only)

Can Emulate target CPUs in order to launch Linux Processes

Supports ...

ARM PowerPC MIPS SPARC etc

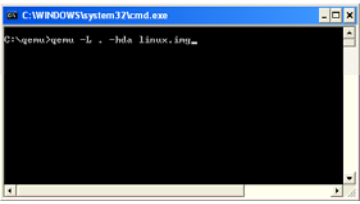
<http://bellard.org/qemu/>

19

Running QEMU

On Command prompt type...

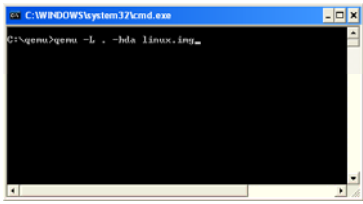
```
qemu -L . -hda linux.img
```



20

Running QEMU

Linux boots...



21

Setting up of Host system

- GCC:** Tool chain for native development
- make:** Make utility for Makefile
- file:** Used to check the file type of a file
- flex:** Generate scanner for input C file
- bison:** General purpose parser
- patch:** utility to apply patches in source codes
- libc6-dev:** C library
- perl:** Perl programming language
- gzip:** zip utility for linux
- wget:** Utility to update package
- libcurses4-dev:** todo
- telnetd:** telnet program

22

Setting up of Host system

```

/root/cross
binutils-2.8.tar.gz
kernelxxx.tar.gz
gcc-4.3.2.tar.gz
glibc-2.7.tar.gz
glibc-linuxthreadx-2.3.tar.gz
gdb.6.8.tar.gz

```

```

/arm-linux-binutils
./configure --target=arm-linux --prefix = usr v
make
make install

```

23

Setting up of Host system:

Building binutils tools for arm-linux

```

Tar file: binutils-2.8.tar.gz
Current directory : /root/cross


configure toolchain for arm-linux
# ./configure --target=arm-linux --prefix = /usr v
target is arm for linux, prefix where to install toolchain

#make
# compiling
# make install
# installing

Tools will be installed in
/usr/bin/

```

24

Setting up of Host system: 

Building binutils tools for arm-linux

Tar file: **binutils-2.8.tar.gz**
 Current directory : /root/cross


configure toolchain for arm-linux
`# ./configure --target=arm-linux --prefix = /usr v`
 target is arm for linux, prefix where to install toolchain

```
#make
  compiling
# make install
  installing
```

Tools will be installed in
 /usr/bin/

```
/usr/bin/powerpc-linux-addr2line
/usr/bin/powerpc-linux-ar
/usr/bin/powerpc-linux-as
/usr/bin/powerpc-linux-c++filt
/usr/bin/powerpc-linux-gasp
/usr/bin/powerpc-linux-ld
/usr/bin/powerpc-linux-nm
/usr/bin/powerpc-linux-objcopy
/usr/bin/powerpc-linux-objdump
/usr/bin/powerpc-linux-ranlib
/usr/bin/powerpc-linux-readelf
/usr/bin/powerpc-linux-size
/usr/bin/powerpc-linux-strings
/usr/bin/powerpc-linux-strip
```

25

Setting up of Host system: 

Building gcc cross compiler tools for arm-linux

Tar file: **gcc-4.3.2.tar.gz**
 Current directory : /root/cross


configure toolchain for arm-linux
`# ./configure --target=arm-linux \`
`--prefix = /usr \`
`--with-headers =/usr/src/arm-linux/include \`
`--enable-language=c \`
`--disable-threads`

[target is arm for linux, prefix where to install toolchain
 --with-headers kernel source header files will be copied into gcc installation
 directory
 --disable-threads = threading support is disable]

```
#make
  compiling
# make install
  installing
```

Tools will be installed in
 /usr/bin/

26

Setting up of Host system: 

Building glibc tools for arm-linux


Tar file: **glibc-2.7.tar.gz**
 Tar file: **glibc-linuxthreadx-2.3.tar.gz**
 Current directory : /root/cross

configure toolchain for arm-linux
`# ./configure --target=arm-linux --prefix = /usr v`
 target is arm for linux, prefix where to install toolchain

```
#make
  compiling
# make install
  installing
```

Tools will be installed in
 /usr/bin/

27

Setting up of Host system: 

Building gdb tools for arm-linux


Tar file: **gdb-6.8.tar.gz**
 Current directory : /root/cross

configure toolchain for arm-linux
`# ./configure --target=arm-linux --prefix = /usr v`
 target is arm for linux, prefix where to install toolchain

```
#make
  compiling
# make install
  installing
```

Tools will be installed in
 /usr/bin/

28

Setting up of Host system: 

Building gdb tools for arm-linux



Tar file: **gdb-6.8.tar.gz**
 Current directory : /root/cross

configure toolchain for arm-linux
`# ./configure --target=arm-linux --prefix = /usr v`
 target is arm for linux, prefix where to install toolchain

```
#make
  compiling
# make install
  installing
```

Tools will be installed in
 /usr/bin/

29

<http://www.embeddedcraft.org> <http://www.imbuent.com>

30