Embedded Systems

Ch 4
Introduction to Device Driver
Part A

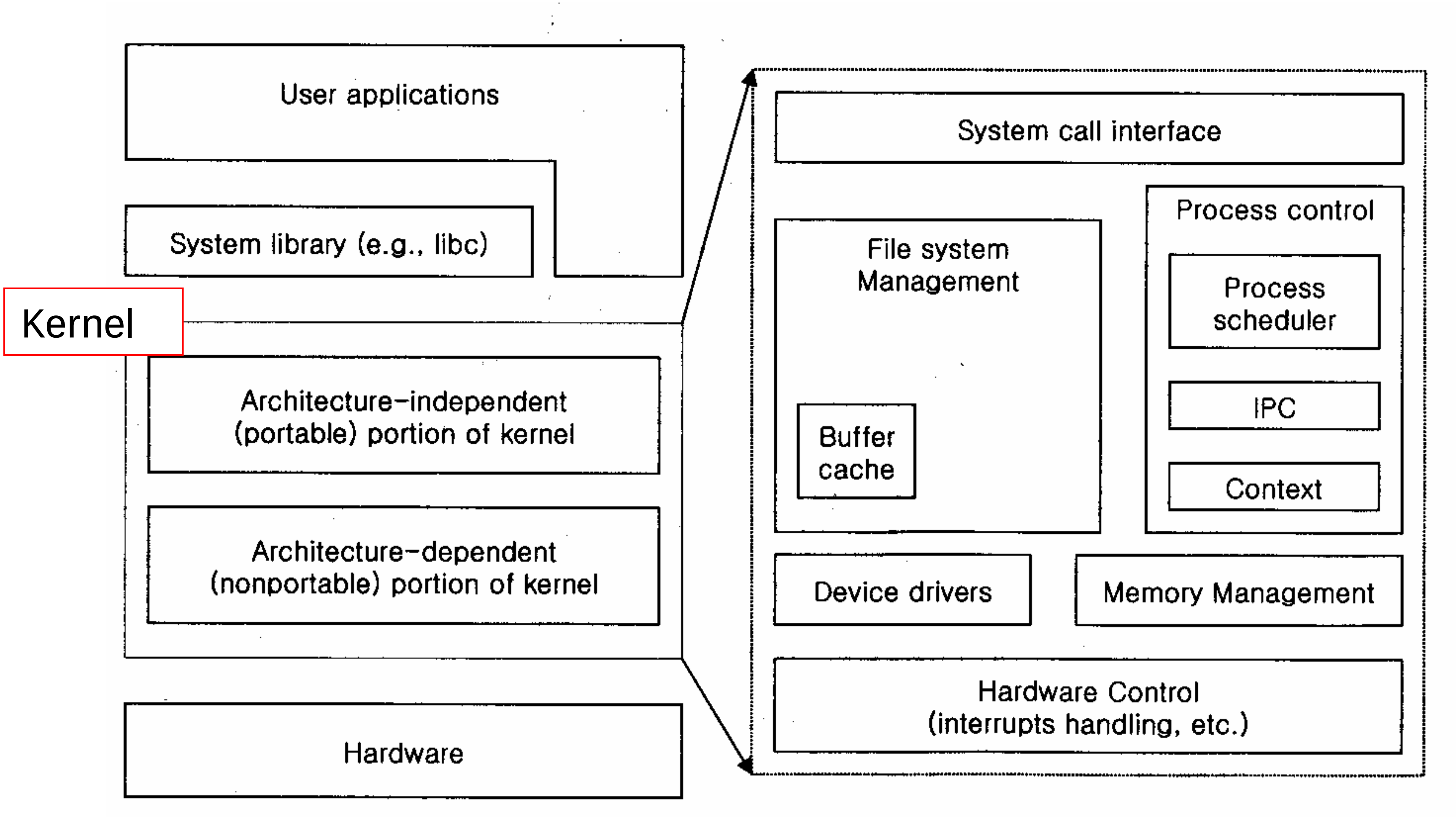
Byung Kook Kim
Dept of EECS
Korea Advanced Institute of Science and Technology

Overview

- 1. Introduction to Linux Kernel
- 2. Kernel Compile
- 3. Kernel Compile Options
- 4. Module
- 5. Device

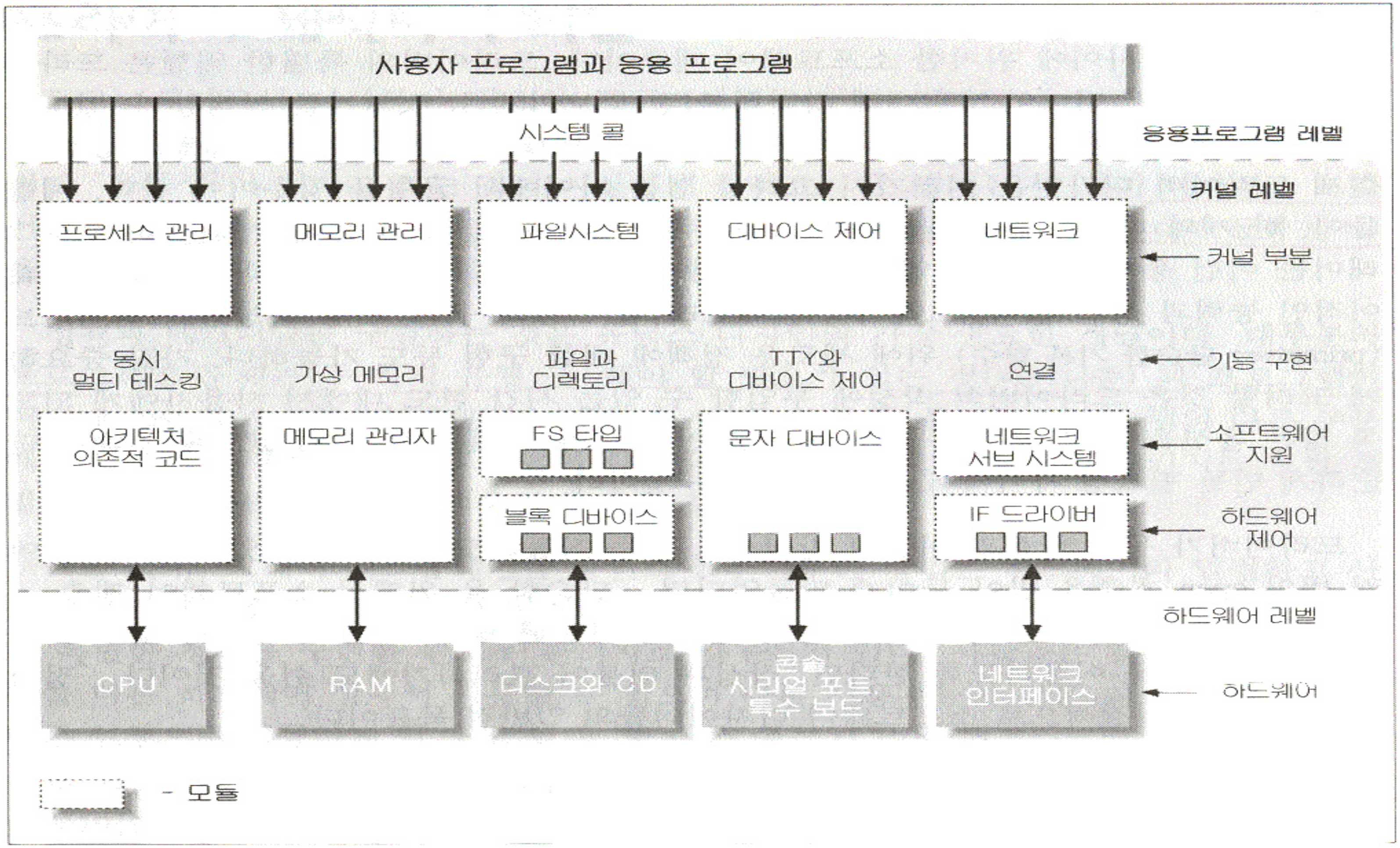
1. Introduction to Linux Kernel

- Linux의 구조


User applications
System library (e.g., libc)
Kernel
Architecture-independent (portable) portion of kernel
Architecture-dependent (nonportable) portion of kernel
Hardware
System call interface
File system Management
Buffer cache
Process control
Process scheduler
IPC
Context
Device drivers
Memory Management
Hardware Control (interrupts handling, etc.)

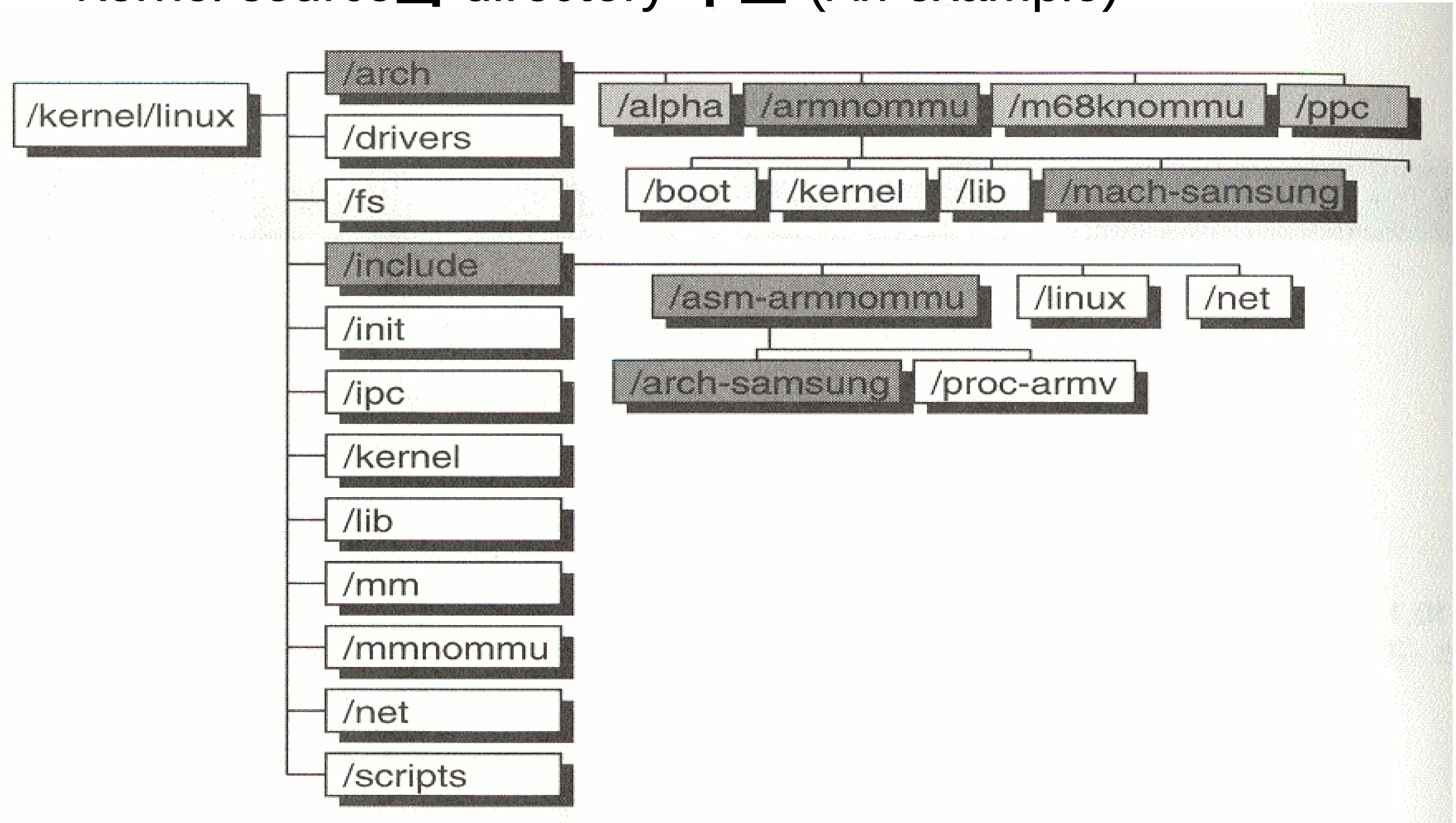
Introduction to Linux Kernel (II)

- Construction of Linux kernel



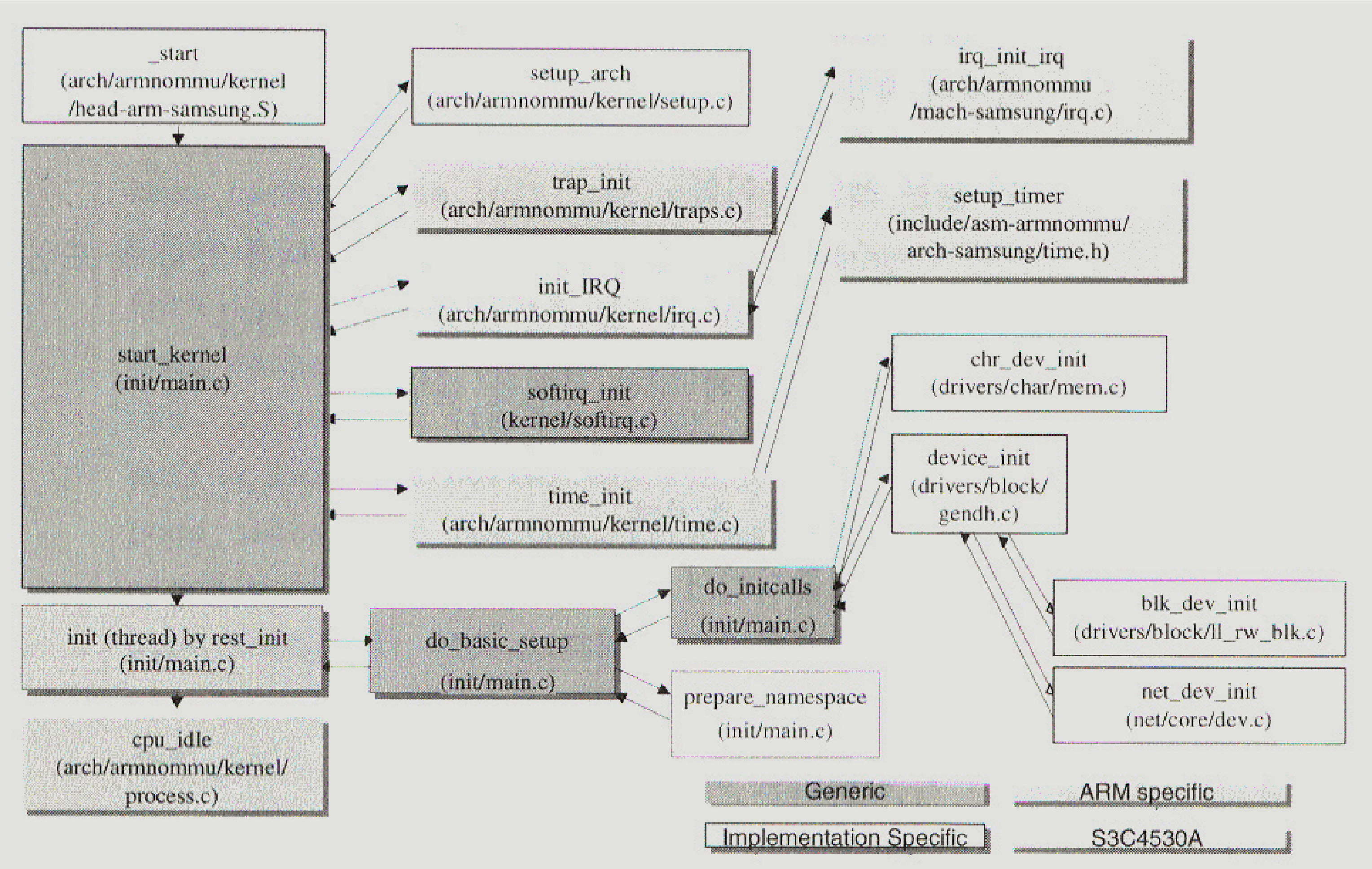
Introduction to Linux Kernel (III)

- Kernel source의 directory 구조 (An example)



Introduction to Linux Kernel (IV)

- Booting procedure



_start (arch/armnommu/kernel /head-arm-samsung.S)

start_kernel (init/main.c)

setup_arch (arch/armnommu/kernel/setup.c)

trap_init (arch/armnommu/kernel/traps.c)

init_IRQ (arch/armnommu/kernel/irq.c)

softirq_init (kernel/softirq.c)

time_init (arch/armnommu/kernel/time.c)

irq_init_irq (arch/armnommu /mach-samsung/irq.c)

setup_timer (include/asm-armnommu/ arch-samsung/time.h)

chr_dev_init (drivers/char/mem.c)

device_init (drivers/block/ gendh.c)

blk_dev_init (drivers/block/ll_rw_blk.c)

net_dev_init (net/core/dev.c)

init (thread) by rest_init (init/main.c)

do_basic_setup (init/main.c)

do_initcalls (init/main.c)

prepare_namespace (init/main.c)

cpu_idle (arch/armnommu/kernel/ process.c)

Generic

ARM specific

Implementation Specific

S3C4530A

Introduction to Linux Kernel (V)

- Unix system
  - Multiple **tasks** executed by several concurrent **processes**
  - Each process requires system resources – CPU, memory, disk, network connection, etc.

- Kernel
  - *Set of executable programs to handle multiple processes*

Introduction to Linux Kernel (VI)

- Functions of kernel
  - *Process management*
    - Creates and deletes processes, and performs input/output connections (signal, pipe, IPC primitives) and inter-process communications.
    - Schedules processes on single (or multiple) CPUs
  - *Memory management*
    - Provide virtual memory spaces to all processes (function calls and malloc/free routines)
  - *File system*
    - Constructs structures file systems on unstructured hardware
    - Supports multiple file systems

Introduction to Linux Kernel (VII)

- Functions of kernel (Cont’d)
  - *Device control*
    - Each device driver for specific device performs required actions for that device.
    - Supports keyboard, hard disk, tape drive, etc.
    - Can be separated from the kernel as a module.
  - *Networking*
    - Network operations are not limited to one process: OS should handle the network operation.
    - Collects/distributes outgoing/incoming packets.
    - Sleeps and wake-ups the process corresponding to the current packet.
    - Network address handling and routing.

2. Kernel Compile

- Kernel Source Usage
  - 1. 새로운 kernel을 제작하기 위하여
  - 2. C 프로그램 작성을 위해서 ARM에 관련된 header file을 얻음
  - 3. 부트로더인 이지부트를 작성하기 위해서.

- ARM Linux Kernel
  - I386 Linux에서 ported
  - Official site:
    - http://www.arm.linux.org.uk

- Getting Linux kernel for EZ-X5
  - 첫째, 리눅스 커널을 구한다.
  - 둘째, 암 패치를 수행한다.
  - 셋째, XScale용 패치를 수행한다.
  - 넷째, EZ 보드를 위한 패치를 수행한다.

Kernel Compile (II)

- Step 1. Get required sources
  - Getting Linux Kernel Sources
    - ftp://ftp.kernel.org/pub/linux/kernel/v2.4/linux-2.4.19.tar.gz
      - ARM용으로 patch 가능한 Linux kernel
    - ftp://ftp.kr.kernel.org/pub/linux/kernel/v2.4/linux-2.4.19.tar.gz
      - Mirror site
    - EZ-X5 CD
  - Getting ARM patch
    - ftp://ftp.arm.uk.linux.org/pub/linux/arm/kernel/v2.4/patch-2.4.19-rmk7.gz
      - rmk: Russel King
  - Getting Xscale PXA255 patch
    - ftp://ftp.arm.uk.linux.org/pub/linux/arm/people/nico/diff-2.4.19-rmk7-pxa1.gz
  - Getting EZ-X5 patch
    - Diff-2.4.19-rmk7-pxa1-ez-x5.gz in EZ-X5 CD in sw/kernel

Kernel Compile (III)

- Step 2. Uncompress kernel and patches
  - Use directory /project/ez-x5/test/kernel
    - Copy all sources into /project/ex-z5/test/kernel
  - Uncompress kernel source
    - # tar zvf linux-2.4.19.tar.gz
  - Perform ARM patch
    - # cd linux-2.4.19
    - # gzip –cd ../patch-2.4.19-rmk7.gz | patch –p1
  - Perform Xscale patch
    - # gzip –cd ../diff-2.4.19-rmk7-pxa1.gz | match –p1
  - Perform EZ-x5 board patch
    - # gzip –cd ../diff-2.4.19-rmk7-pxa1-ez-x5.gz | patch –p1
  - Version control and link
    - # cd ..
    - # mv linux-2.4.19 linux-2.4.19-rmk7-pxa1-ez-x5
    - # ln –s linux-2.4.19-rmk7-pxa1-ez-x5 linux

Kernel Compile (IV)

- OR
  - # tar xvzf linux-2.4.19-x5-v05.tar.gz
  - # mv linux-2.4.19 linux-2.4.19-x5-v05
  - # ln –s linux-2.4.19-x5-v05 linux
- Step 3. Set compile environment
  - Set default configuration for EZ-X5
    - # make ez-x5_config
  - Set configuration parameters
    - # make old_config
      - Change some compile parameters using defaults found in .config
  - Set main configuration parameters
    - # make menuconfig
      - Details? Later.

Kernel Compile (V)

- Step 4. Compile kernel
  - Make file dependency in makefile
    - # make dep
  - Clear old object files in the directories
    - # make clean
  - Compile kernel to produce zImage
    - # make zImage
      - Have a cup of coffee….
      - zImage is produced in ./arch/arm/boot/zImage
  - Check the produced kernel
    - # cd arch/arm/boot
    - # ls –la
    - zImage ……………………
      - # Check if correctly compiled. Size?

Kernel Compile (VI)

- Step 5. Generate modules
  - Modules
    - Designated as ‘M’ in kernel compile option
    - Will be included in /lib/modules in EZ-X5
  - Compile modules
    - # make modules
  - Install modules in RamDisk
    - # INSTALL_MOD_PATH=”your_ramdisk_directory”/make modules_install

3. Kernel Compile Options

- 1. Code maturity level options
  - [*] Prompt for development and/or incomplete code/drivers
    - [*] built-in, [ ] excluded, <M> module, < > module-capable

- 2. Loadable module support
  - [*] Enable loadable module support

- 3. System type
  - (PXA250/210-based) ARM system type
    - (X) PXA250/210-based
  - 3.1 Intel PSA250/210 Implementations
    - [*] FALINUX EZ-X5

Kernel Compile Options (II)

- 4. General setup
  - (0) Compressed ROM boot loader base address
  - (0) Compressed ROM boot loader BSS address
  - [*] Support CPU clock change (EXPERIMENTAL)
  - [*] Networking support
  - [*] System V IPC
  - [*] ysctl support
  - <*> N FPE math emulation
  - <*> Kernel support for a.out binaries
  - <*> Kernel support for ELF binaries
  - [*] Kernel-mode alignment trap handler

- 5. Parallel port support
  - < > Parallel port support

Kernel Compile Options (III)

- 6. Memory Technology Devices
  - <*> Memory Technology Devices (MTD) support
  - <*> Direct char device access to MTD drives
  - <*> Caching block device access to MTD devices
  - 6.1 RAM/ROM/Flash chip drivers
    - <*> Detect flash chips by Common Flash Interface (CFI) probe
    - <*> Support for Intel/Sharp flash chips
    - <*> Support for AMD/Fujitsu flash chips
  - 6.2 Mapping drivers for chip access
    - < > CFI Flash device in physical memory map
  - 6.3 Self-contained MTD device drivers
    - < > Uncached system RAM
  - 6.4 NAND Flash Device Drivers
    - <*> NAND Device support
    - [*] Verify NAND page writes
    - <*> NAND Flash device on EZ-X5 board

Kernel Compile Options (IV)

- 7. Plug and Play configuration
  - < > Plug and Play support
- 8. Block devices
  - <*> Loopback device support
  - <*> Network block device support
  - <*> RAM disk support
  - (8192) Default RAM disk size
  - [*] Initial RAM disk (initrd) support
- 9. Multi-device support (RAID and LVM)
  - [ ] Multiple devices driver support (RAID and LVM)

Kernel Compile Options (V)

- 10. Networking options
  - <*> Packet socket
  - <*> Unix domain sockets
  - <*> TCP/IP networking

  - 10.1 QoS and/or fair queueing
    - < > QoS and/or fair queueing
  - 10.2 Network testing
    - < > Packet generator (USE WITH CAUTION)

- 11. Network device support
  - [*] Network device support
  - Ethernet (10 or 100 Mbit)
    - [*] EZ-X5 CS8900A support

Kernel Compile Options (VI)

- 12. Armature Radio support
  - [ ] Armature Radio support
- 13. IrDA (infrared) support
  - [ ] IrDA support
- 14. ATA/ATAPE/MFM/RLL support
  - < > ATA/ATAPI/MFM/RLL support
- 15. SCSI support
  - < > SCSI support
- 16. I2O device support
  - < > I2O support
- 17. ISDN subsystem
  - < > ISDN support
- 18. Input core support
  - < > Input core support

Kernel Compile Options (VII)

- 19. Character devices
  - [*] Virtual terminal
  - [*] Support for console on virtual terminal
  - [*] Standard/generic (8250/16550 and compatible UARTS) serial support
  - [*] Support for console on serial port
  - 19.1 Serial drivers
    - < > 8250/16550 amd compatible serial support (EXPERIMENTAL)
  - 19.2 I2C support
    - < > I2C support
  - 19.3 L3 serial bus support
    - < > L3 support
  - 19.4 Mice
    - < > Mouse support
  - 19.5 Joysticks
    - --- Input core support is needed for gameport
  - 19.6 Watchdog cards
    - [ ] Watchdog timer support

Kernel Compile Options (VIII)

- 20. Multimedia devices
  - < > Video for Linux

- 21. File systems
  - <*> Yaffs file-system on NAND
  - [*] /dev/pts file system for Unix98 PTYs
  - [*] Second extended fs format

  - 21.1. Network file systems
    - <*> NFS file system support
    - [*] Provide NFSv3 client support
  - 21.2 Partition types
    - [*] Advanced partition selection

Kernel Compile Options (IX)

- 22. Console drivers
  - [ ] VGA test support
  - 22.1 Frame-buffer support
    - [*] Support for frame buffer devices (EXPERIMENTAL)
    - <*> XA LCD support
    - <*> 16 bpp packed pixels support
    - <*> VGA characters/attributes support
    - [*] Select compiled-in fonts
    - [*] VGA 8x8 font
    - [*] VGA 8x16 font

Kernel Compile Options (X)

- 23. Sound
  - < > Sound support

- 24. Multimedia Capabilities Port drivers
  - [ ] Multimedia drivers

- 25. Bluetooth support
  - < > Bluetooth subsystem support

- 26. Kernel hacking
  - [*] Verbose user fault messages.

- *Save current configuration!!!*

4. Module

- **Module**
  - Def.
    - Kernel에 추가할 수 있는 각각의 코드 부분
  - Features
    - Provide expandability on the Linux kernel
    - 실행 도중에 커널 코드를 확장할 수 있다
      - 시스템 동작 중에도 커널에 기능을 추가할 수 있다
    - 여러 type (class)의 많은 모듈을 지원
      - Device driver 들도 포함.
  - Construction
    - 각 module은 (완전한 실행 file로 link되지 않은) object code로 구성되어 있다.

Module (II)

- Module Commands:
  - Insmod: 각 module을 실행중인 kernel에 동적으로 link 시킨다.
    - # insmod module_name.o
  - Rmmod: 실행중인 module을 unlink 시킨다.
    - # rmmod module_name
  - Lsmod: 실행중인 module들을 열거한다.
    - # lsmod
    - Module1.o module2.o .....

5. Device

- **Device**
  - 각각 다른 task에 집중하는 세가지 type으로 구별할 수 있다.
    - 문자 device
    - Block device
    - Network interface
  - Linux는 각 device type을 module 형태로 적재할 수 있으므로, 최신의 kernel version을 사용하며, 개발에 따라 사용자가 새로운 hardware를 실험할 수 있도록 하였다.
  - 각 module은 하나의 device driver를 대상으로 만들어 진다.

Device (II)

- **1. Character device**
  - File처럼 접근 가능하다.
  - 각 character device는 open, read, write 등의 system call 로 구현된다.
  - 예: console, parallel port.
  - Stream abstraction 으로 표현 가능.
  - File system node: /dev/tty1, /dev/lp1, etc
  - 문자 device: data를 순차적으로 접근할 수 밖에 없다.
    - 일반 file: data의 위치 pointer를 앞뒤로 이동 가능하다.

Device (III)

- **2. Block Device**
  - Disk처럼 file system을 가질 수 있는 device
  - Can be accessed by block units (1 KB typical)
  - Linux에서는 block device를 character device처럼 access 가능하다.
    - 한번에 어떤 크기의 byte 단위로도 전송할 수 있다.
  - 내부적 데이터 처리방법, kernel/driver software interface에서 차이.
  - File system node를 통해서 접근.

Device (III)

- **3. Network Interface**
  - 모든 network transaction은 하나의 interface를 통해서 처리된다.
  - Hardware interface와 software interface가 존재한다
    - 예: Loopback interface
  - 한 개의 network interface는 data packet를 주고받는 책임을 지게 되는데, packet은 kernel의 network subsystem에 의해서 주어지며, 이것이 각각 어떤 task와 관련이 있는지 알지 못한다.
    - Telnet와 ftp는 stream에 기반한 연결을 사용 (connection oriented TCP/IP)
    - UDP와 같은 device를 사용
    - Device는 data packet만 다루며, 각각의 stream에 대해서는 살피지 않는다.
  - Stream에 기반한 device가 아니므로 file system node에 간단히 mapping 불가능: eth0 등의 이름 지정.
  - Read, write 대신 kernel call 함수는 packet 전송과 관련된다.

References

- Introduction to Linux Kernel
  - 최병욱 외, “임베디드 리눅스 – 실습 및 활용”, 홍릉과학출판사, 2003.
- Kernel Compile & options
  - EZ-X5 User’s Manual, Falinux.com, 2003. Ch. 8 & 9.
- Module & device
  - Alessandro Rubini, “Linux Device Drivers”, O’Reilly, 1998.