# Research of Key Technologies for Embedded Linux Based on ARM

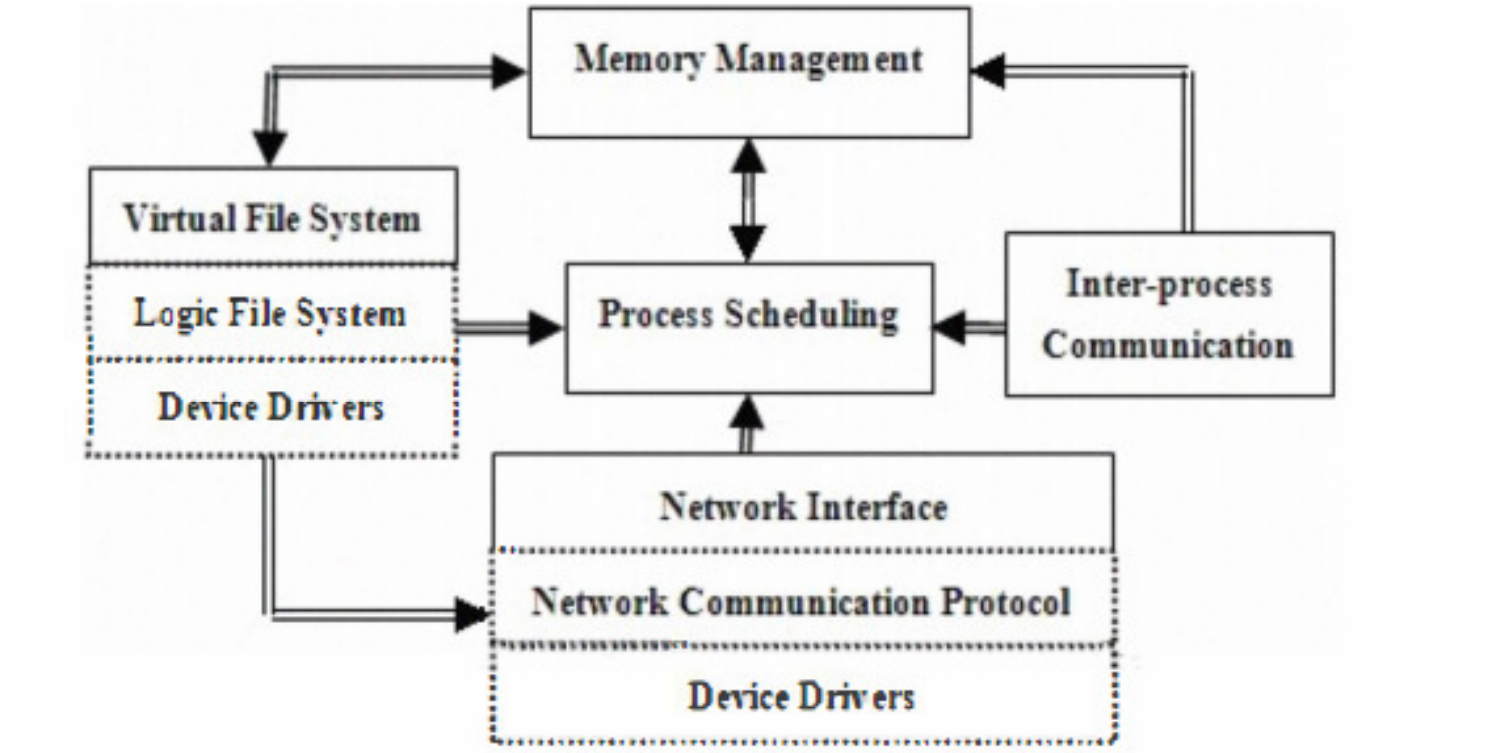
Why is Linux a favorite in embedded area?

What are the research areas in Embedded LINUX?

- Analysis and modification of Linux Kernel
- Linux Porting
- Improvement of Real Time performance
- Development of Device Drivers

- Vinay

# Kernel Structure of LINUX




Each subsystem needs to rely on process scheduler to hang or recover process, so process scheduling is in the center

# PORTING

- Obtaining source codes
- Building cross-compiler environment
- Porting Linux bootloader
- Configure and compile the kernel,
- Porting and loading the embedded file system
- Developing and debugging of application program
- Program downloading

# Porting – Detailed Steps

- Modify Makefile

```
ARCH  ?=arm
CROSS  COMPI LE?= arm-linux-
#Vi ~/.bashrc
export PATH  =Iusr/local/ar-linux -3.4.4/bin:$PATH
```

# Porting – Detailed Steps

- Set Flash Partition

arch/arm/machs3c2410/devs.c

```
#include <linux/mtd/partitions.h>
#include <linux/mtd/nand.h>
#include <asm/arch/nand.h>
```

modify s3c_device_nand structure variables

- Set Initialization : arch/arm/mach-s3c241O/mach-smdk241O.c
- Prohibit Flash ECC check :

s3c241O_nand_init_chip ()

chip->eccmode = NAND _ ECC _NONE;

# Porting – Detailed Steps

- Configuring Kernel

Add devfs configuration

Configure kernel options

- Kernel Compile

make clean

#make zImage

- Kernel download

vivi> loadflash kernel u

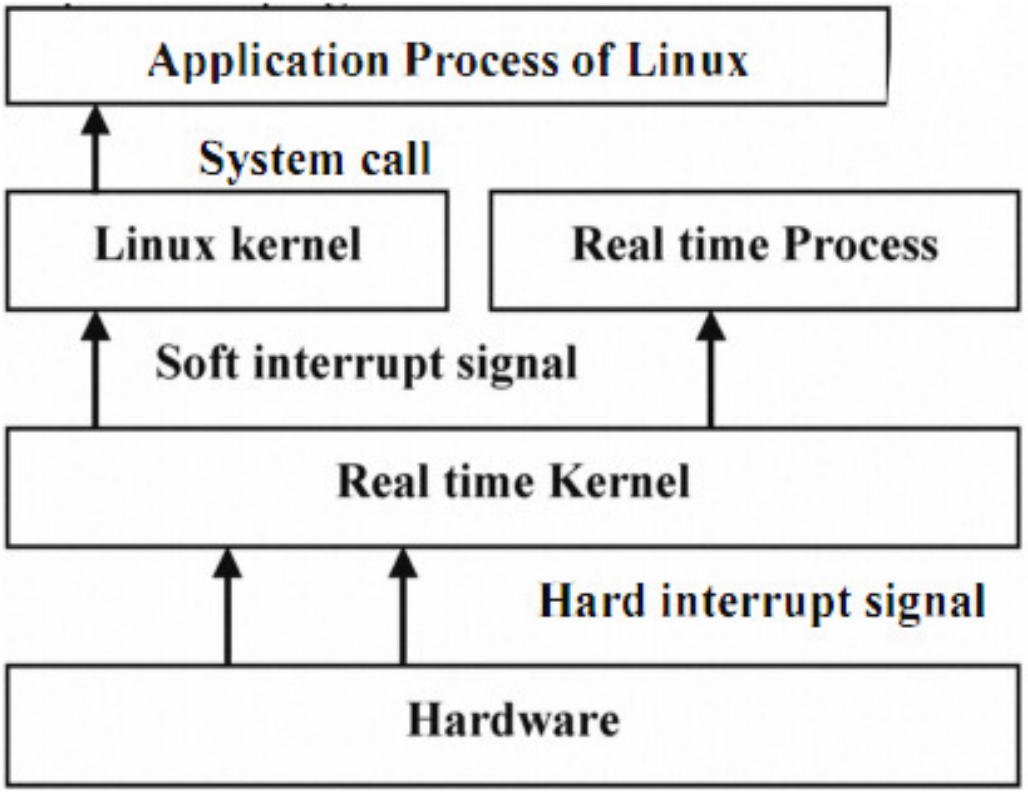
# Real Time problems of Linux

- Non-preemption of kernel
- Process synchronization and mutual exclusion
- Priority Inversion
- Clock granularity is rough
- Interrupt processing

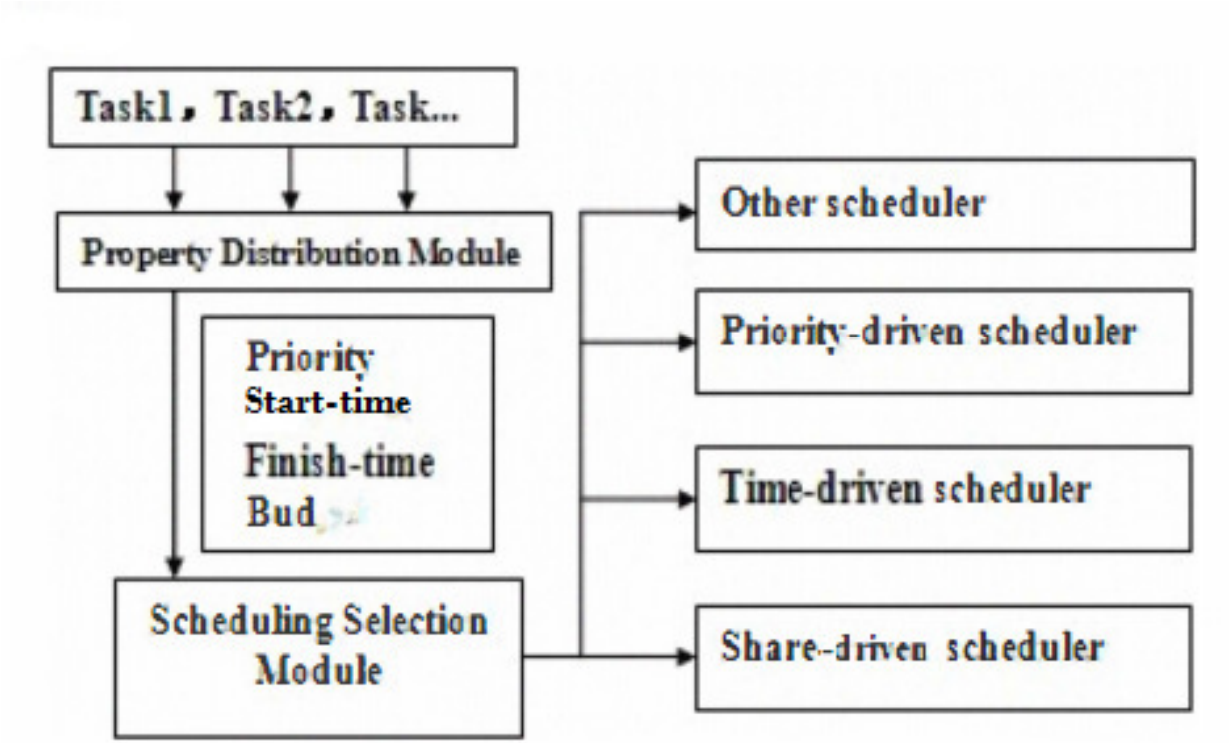
Linux is designed as a time-sharing system,
which must be modified to meet the real-time performance

# Improvement of Real-time Performance for Linux Kernel

1) Dual Kernel Method



## 2) Scheduling strategy of real-time task



# DEVICE DRIVER DEVELOPMENT TECHNOLOGY

Classification of Linux device drivers

- Character devices
- Block Devices
- Network Devices

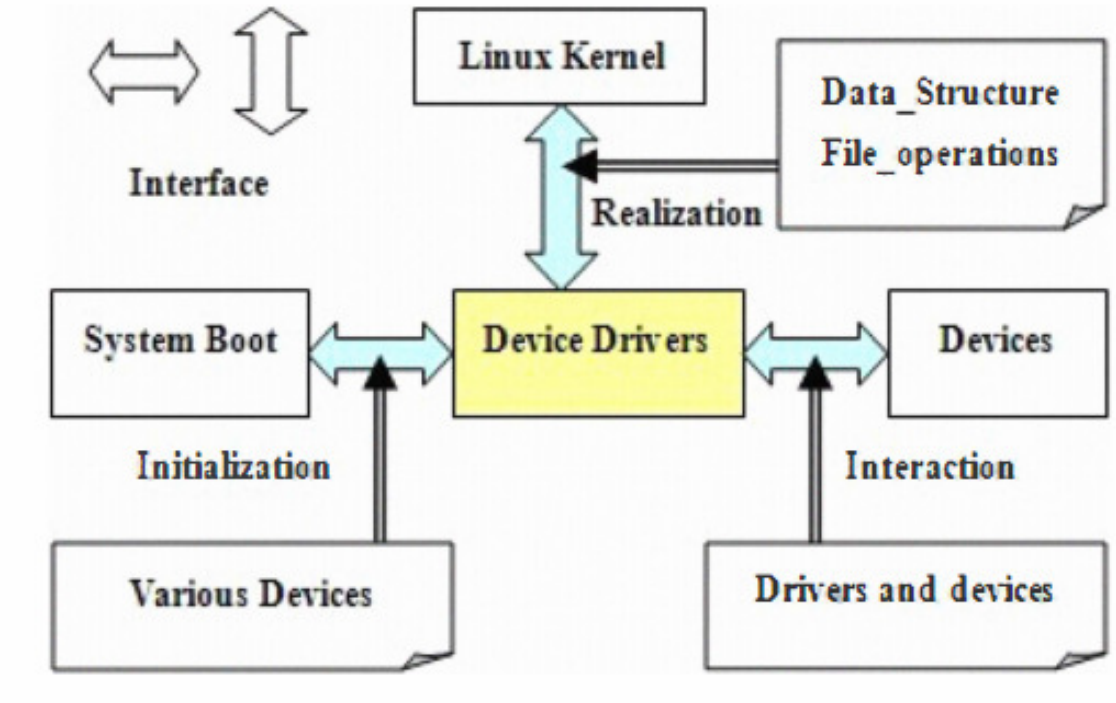
Commonality of Linux device drivers

- Read/write
- Interrupt
- Clock

# Elements of Embedded Linux Device Drivers

- Module Initialization function: Init-module ( )
- Module Unload function: Cleanup-module ( )
- Device Driver Interface: file-operations

## Framework of Device Drivers



# Development flow -embedded Linux device driver

- Define major device number and minor device number, and we can also obtain them dynamically.
- Implement driver initialization and remove functions, if the driver uses module style, it will implement the module initialization and remove functions.
- Design the file operations to be achieved and define file-operations structure.
- Implement - operations calls, (read, write, etc.)
- Implement interrupt service and register to the kernel with "equest_irq".
- Compile drivers into kernel and loaded with the command "insmod".
- Generate the device node files.

# Thank You!!!

-Vinay