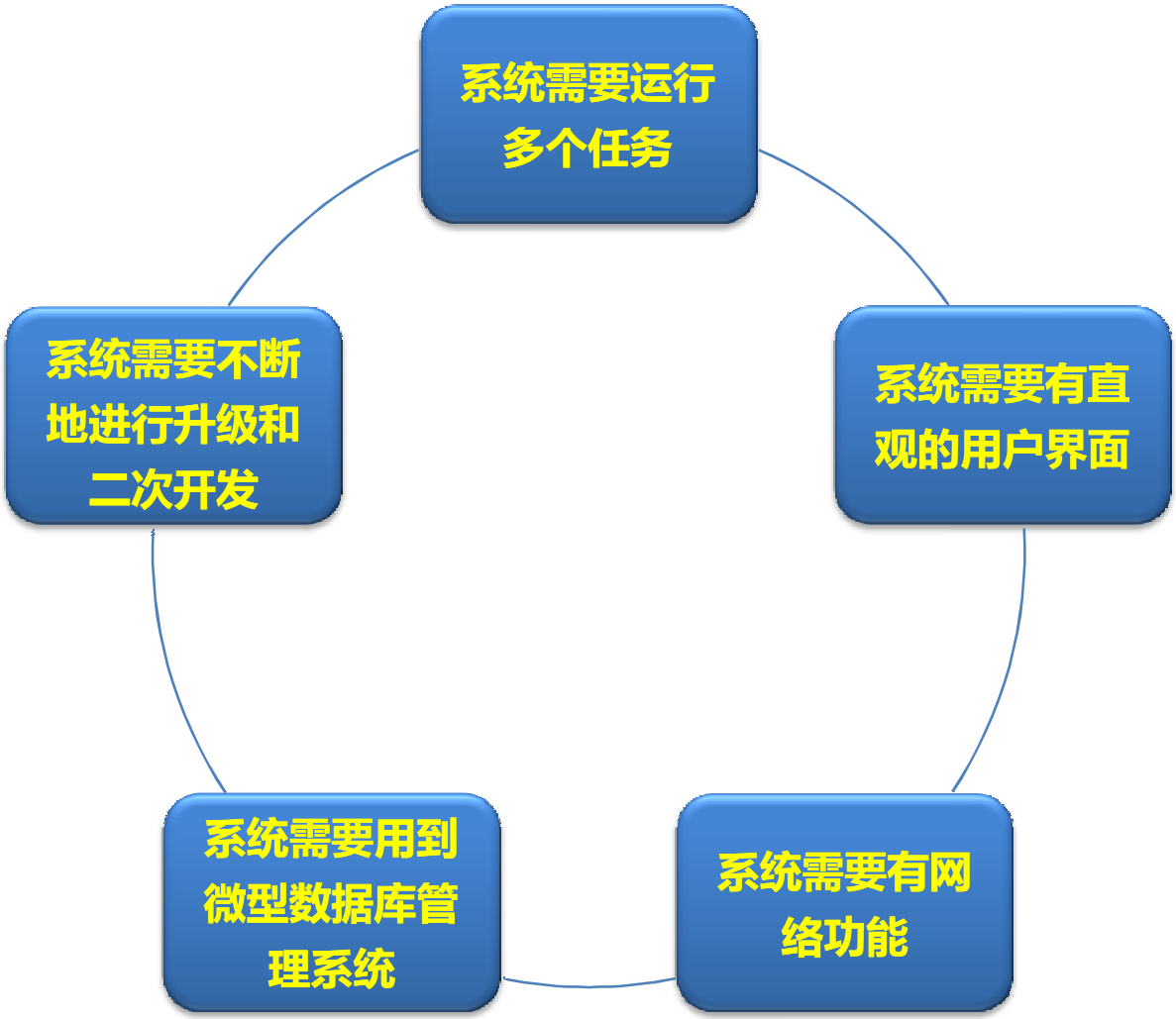
基于Linux的嵌入式系统开发

华清远见 孙天泽
tianzesun@farsight.com.cn
12/06/2008

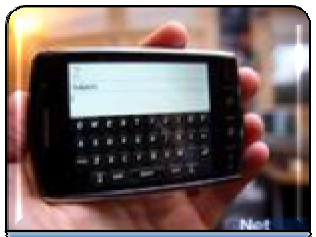
内容提纲

1. 嵌入式移动开发格局
2. 构建嵌入式LINUX开发环境
3. 交叉编译环境
4. 内核开发与调试

何时需要“嵌入式操作系统”



嵌入式移动开发格局



BlackBerry[m]

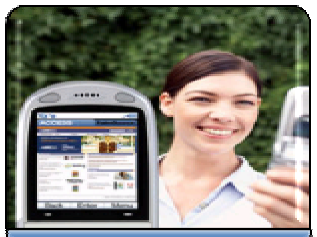
BREW[m]

Symbian[mns]

Windows Mobile[mns]

iPhone[n]

Garnet OS[n]



DoJa[j]

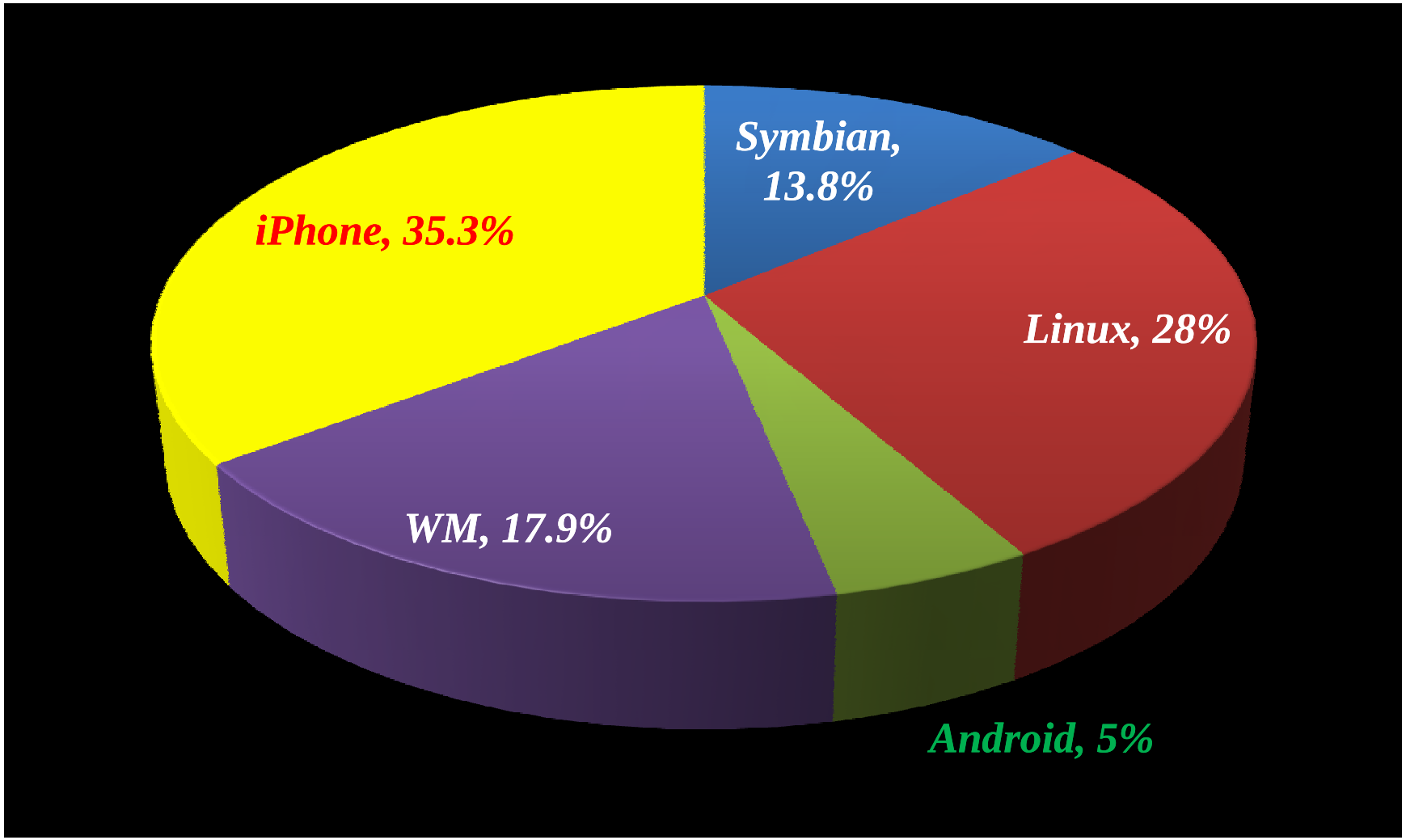
JavaFX[j]

SavaJe[j]

庞大的Linux阵营

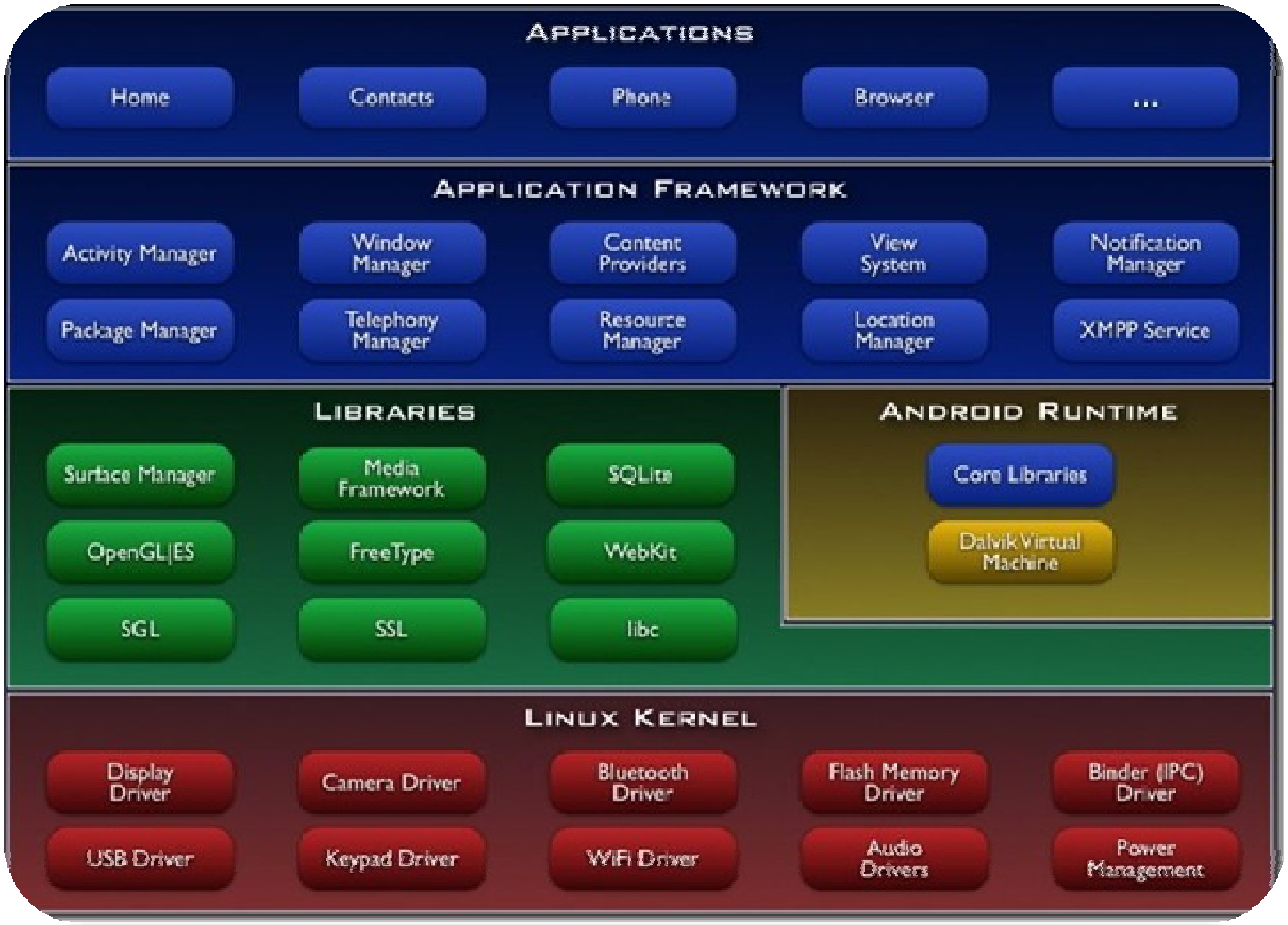

Android™
LiMo Platform™
Qt Extended™
Openmoko™
Ubuntu Mobile™
MotoMagx™
Tux
Mobilinux
RTOS
Access Linux Platform™
Maemo

移动平台操作系统指数对比



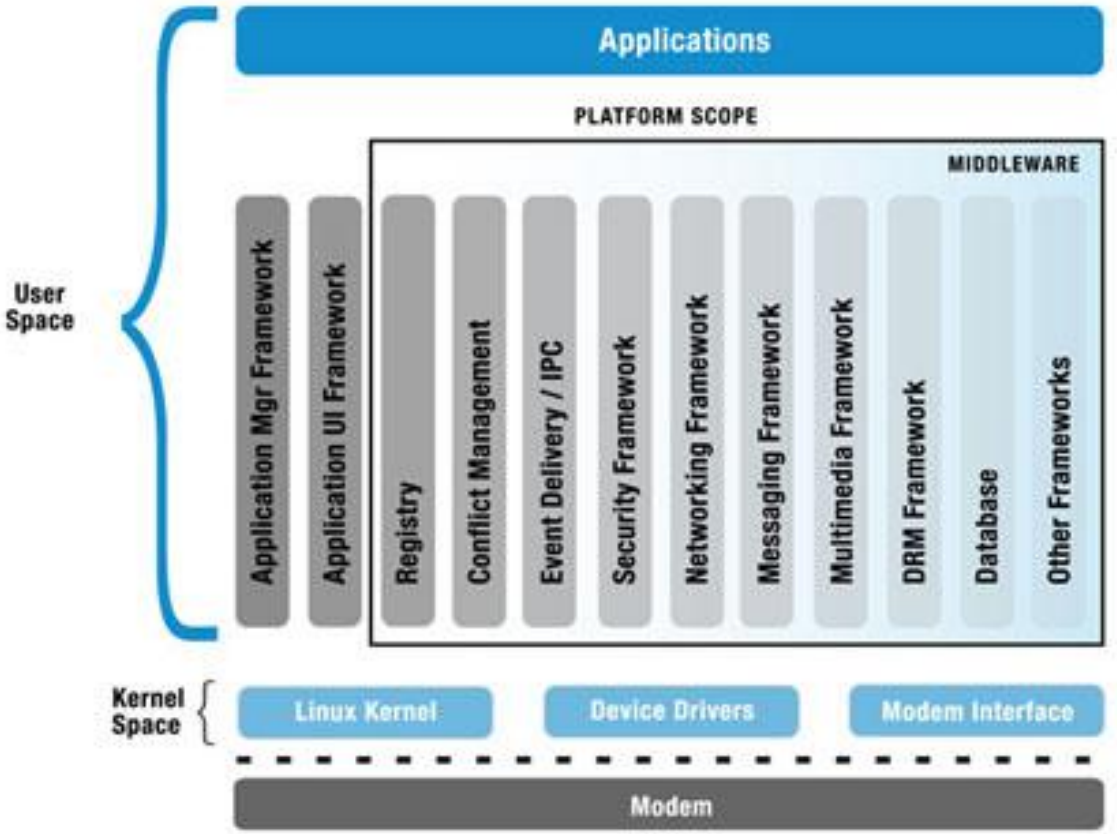
《CSDN中国IT技术指数报告2008年7月第一期》

Android平台架构

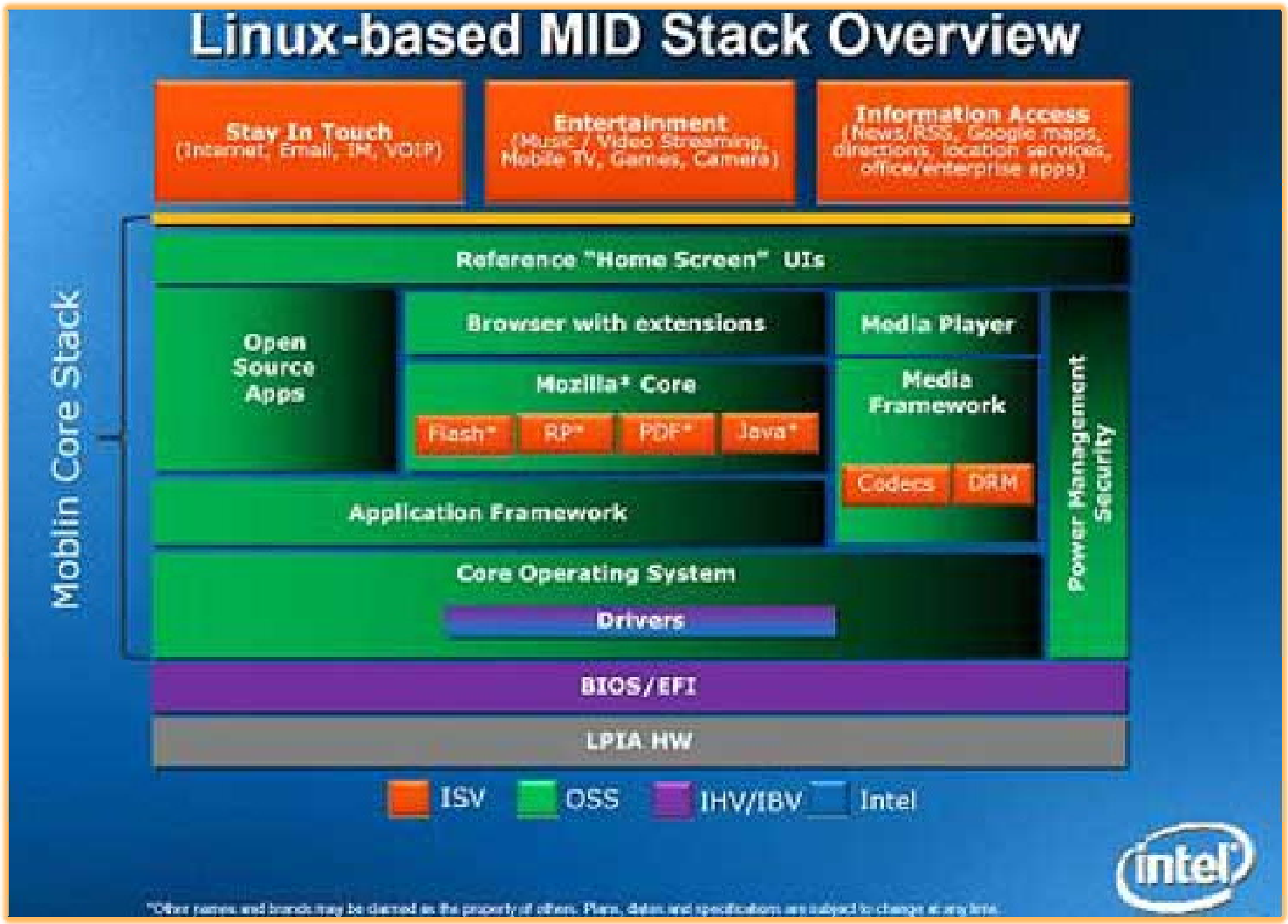


LiMo平台架构

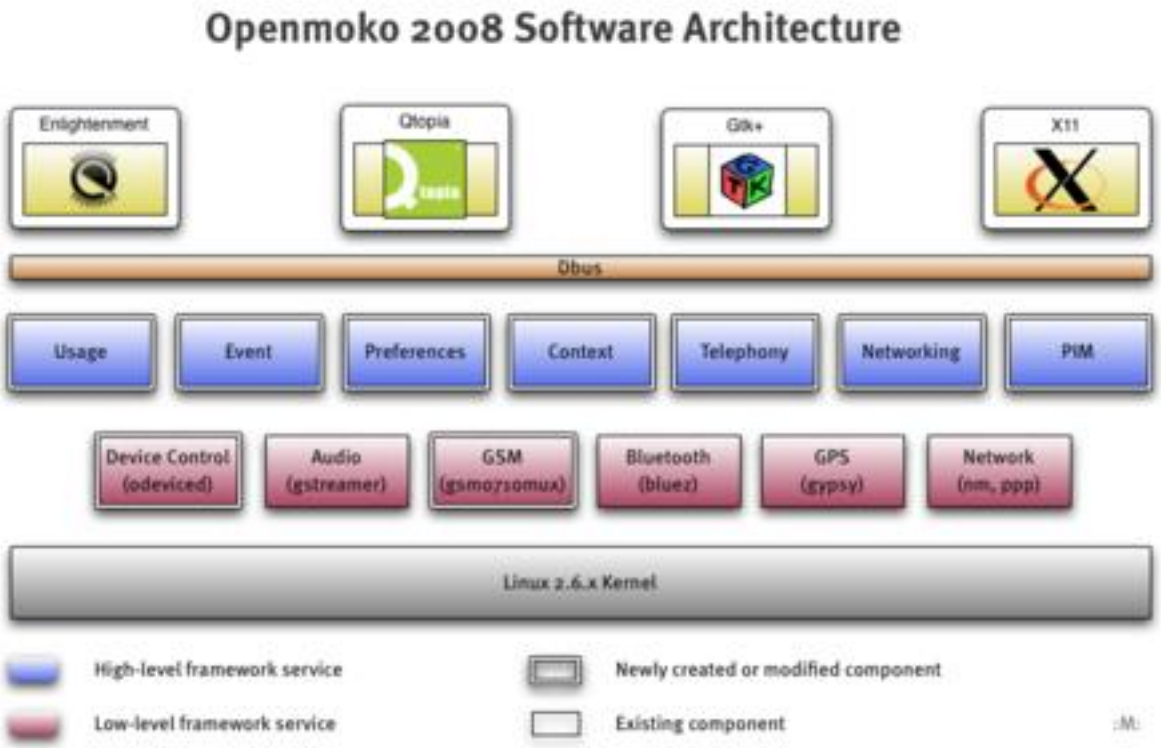
- 摩托罗拉
- 三星
- LG
- 松下
- NEC
- 华为
- …



Moblin软件架构视图



Openmoko软件架构



典型的Linux发行版

Red Hat: http://www.redhat.com

Mandrake: http://www.linux-mandrake.com/en/

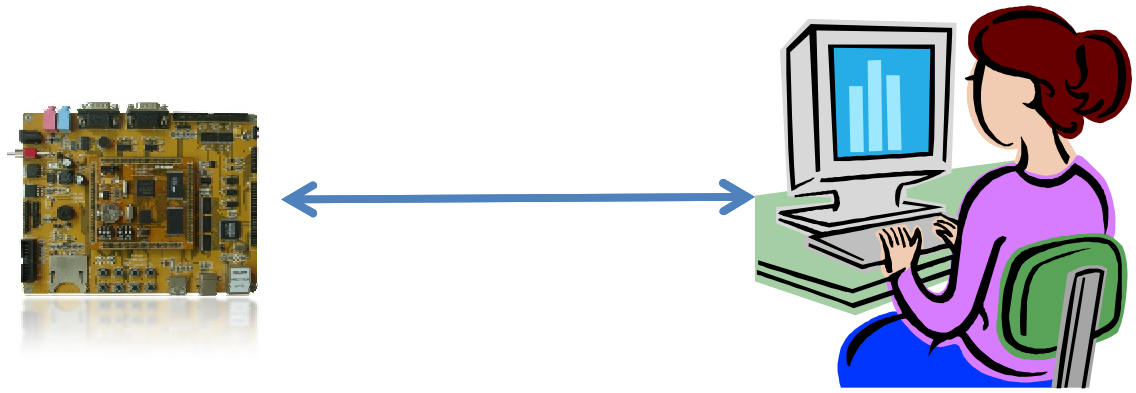
Slackware: http://www.slackware.com/

OpenSuSE: http://www.opensuse.org/

Ubuntu: http://www.ubuntu.org.cn/

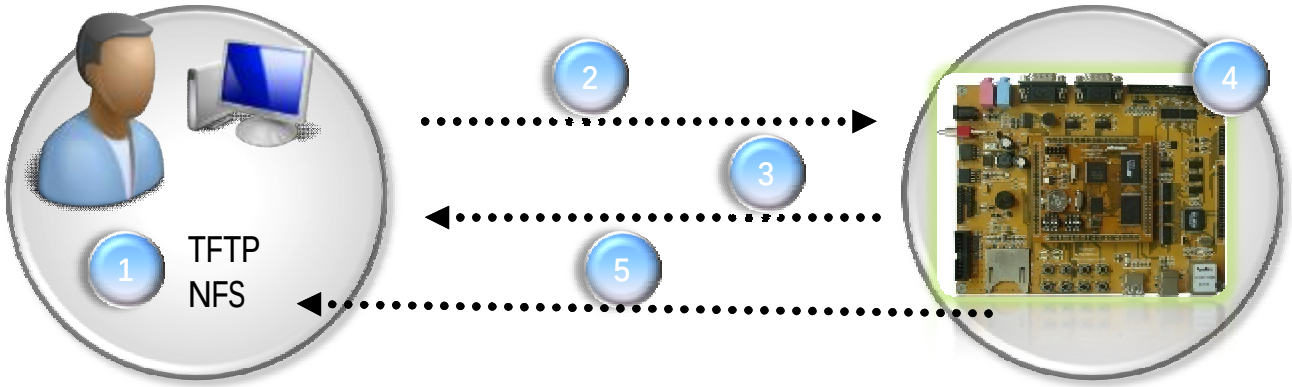
Debian: http://www.debian.org/

嵌入式Linux开发



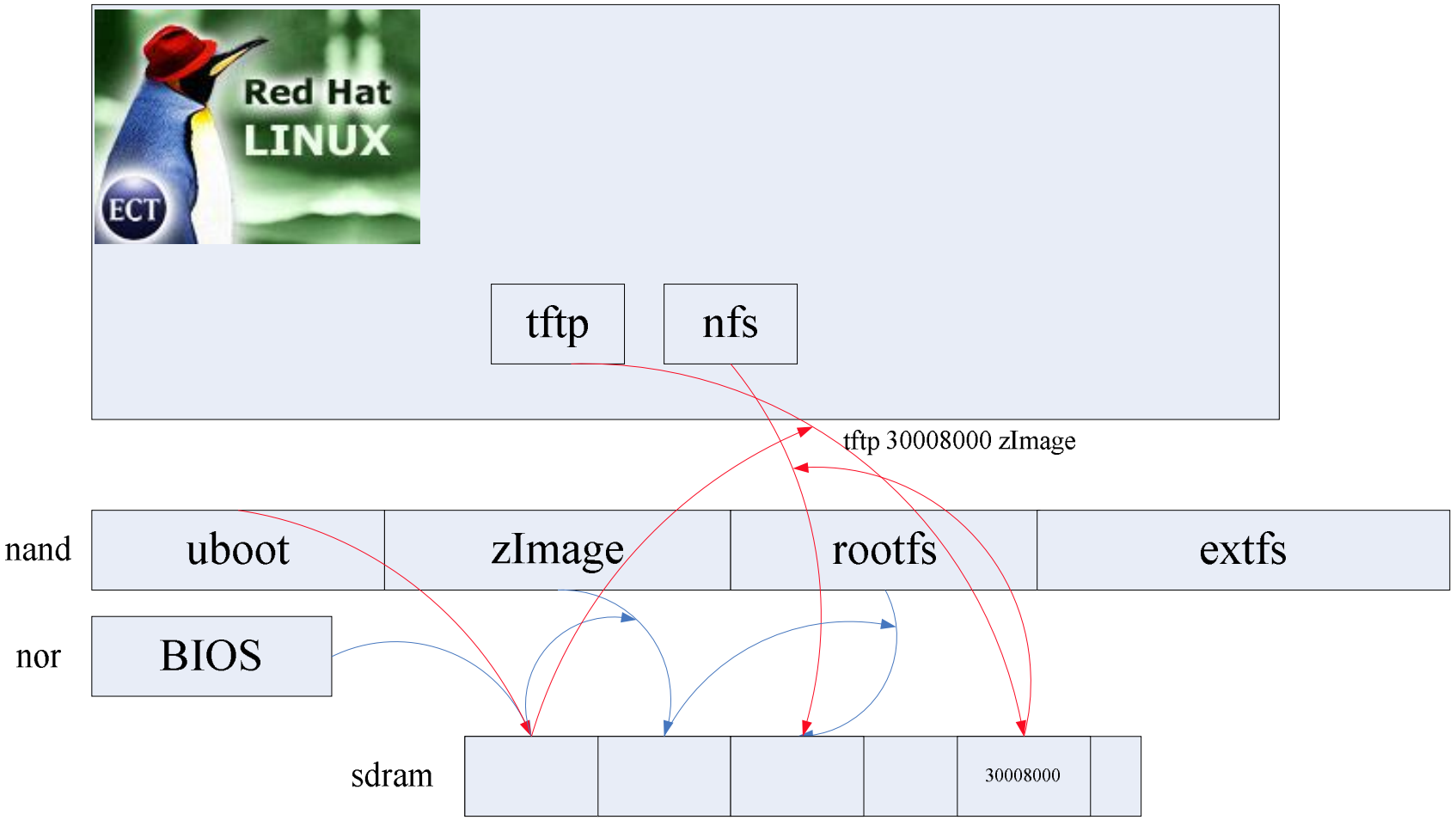
目标机	软件	宿主机
X	开发工具	√
X	源代码	√
√	调试器	√
√	操作系统	√
可选	系统服务	NFS TFTP
√	可执行程序	X

嵌入式Linux开发流程

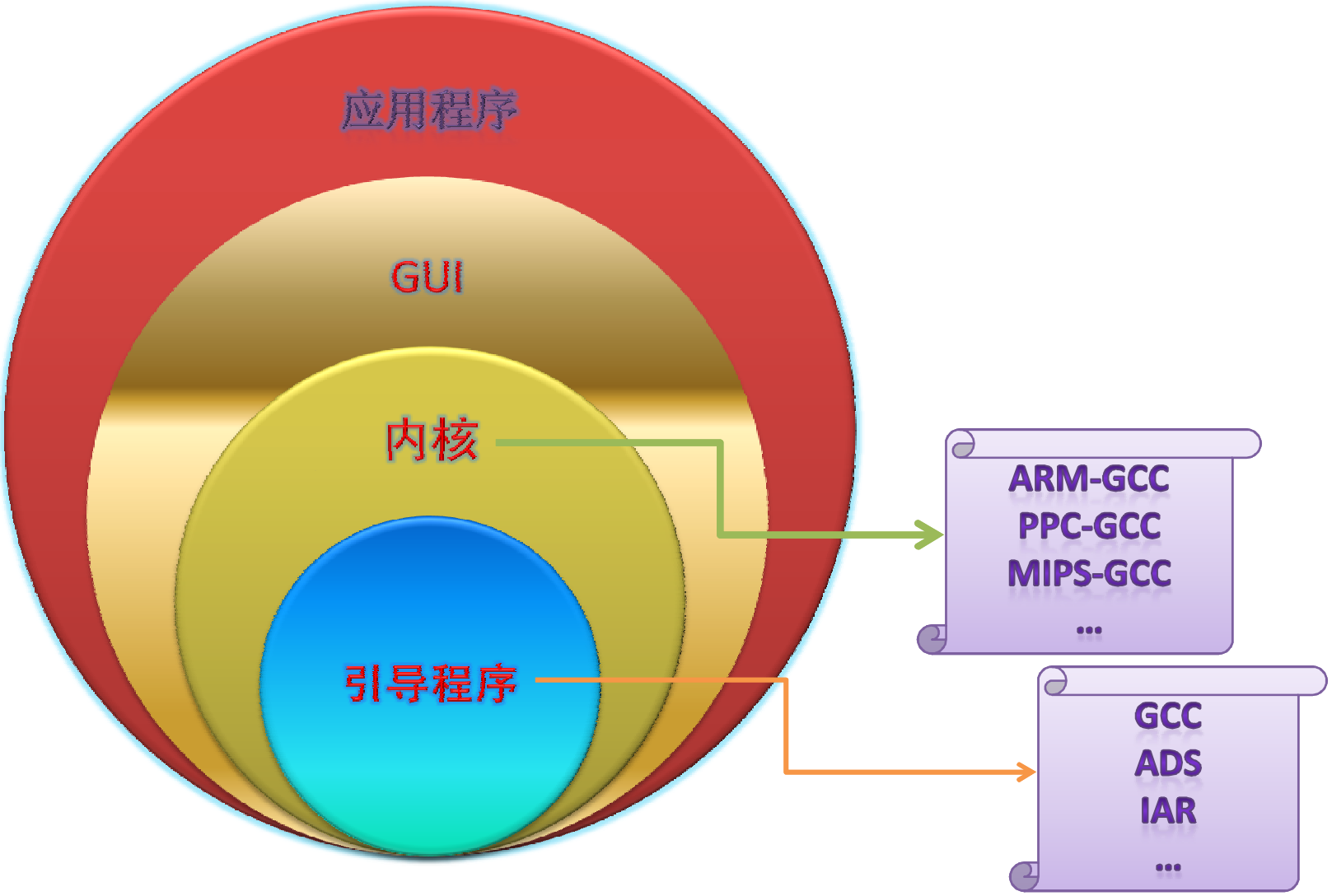


1. 设置开发主机的NFS服务、TFTP服务
2. 通过并口、串口、USB下载Bootloader
3. 通过FTP将Linux内核下载到开发板的内存中
4. 启动开发板上的Linux系统
5. 挂载开发主机的NFS文件系统

系统开发图解

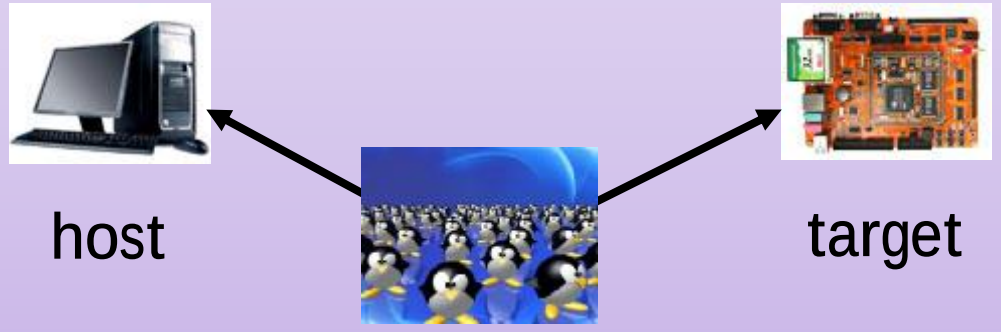

Red Hat
LINUX
ECT
tftp
nfs
tftp 30008000 zImage
nand
uboot
zImage
rootfs
extfs
nor
BIOS
sdram
30008000

嵌入式Linux开发的任务

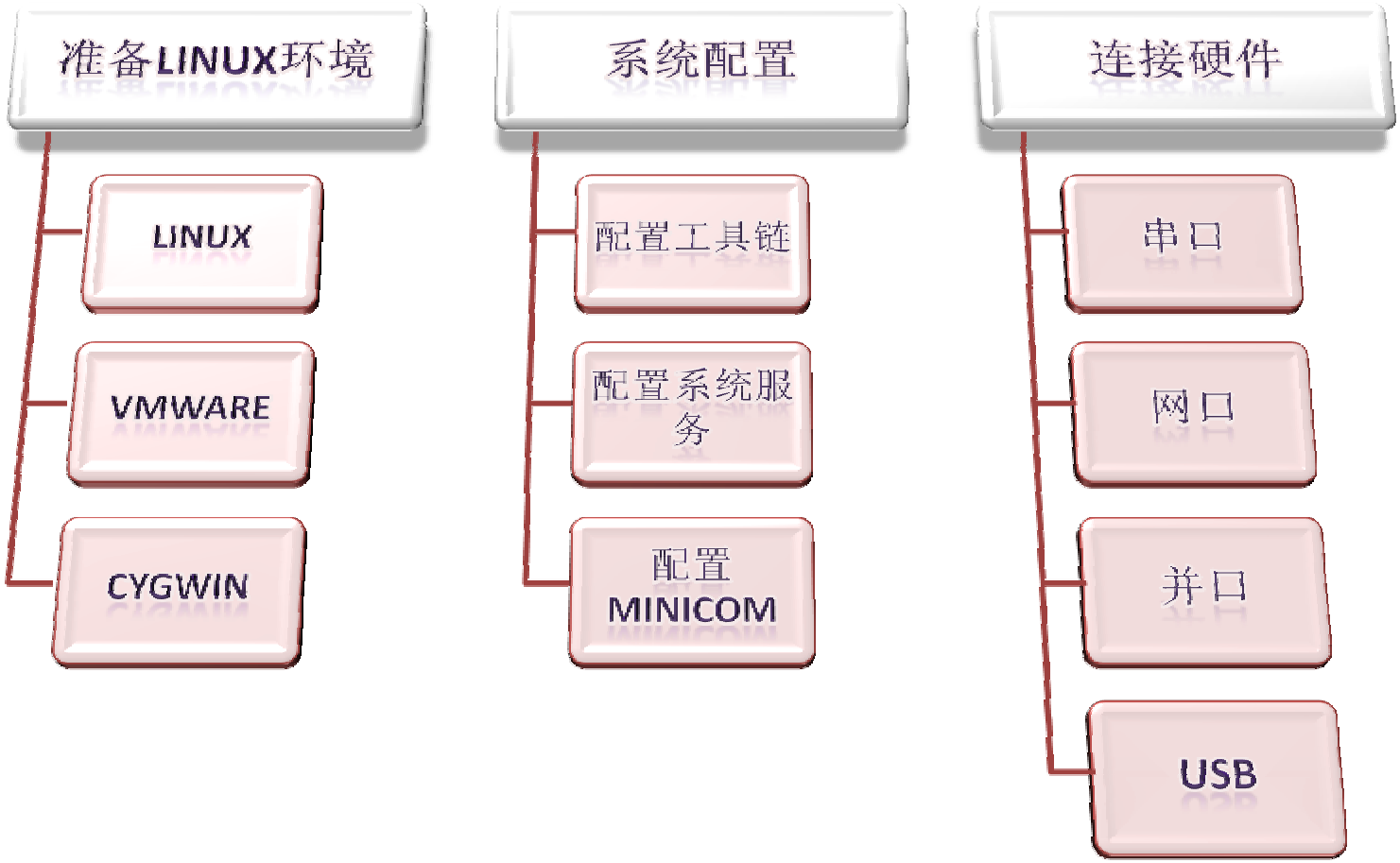


交叉编译的概念

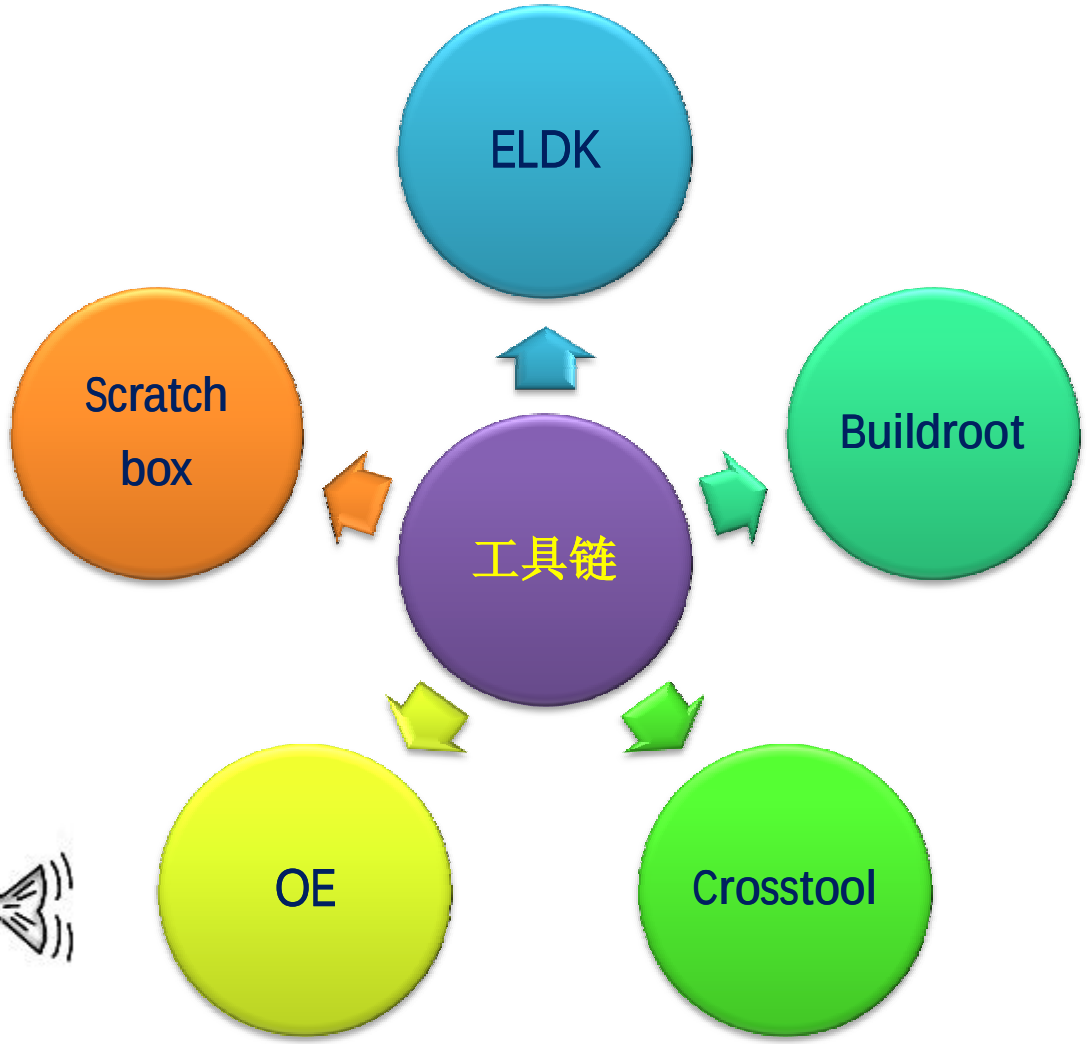
- 交叉编译：在一个平台生成可以在另一个平台运行的代码。
- 平台：体系结构、操作系统。同一个体系结构可以运行不同的操作系统；同一个操作系统也可以在不同的体系结构上运行。

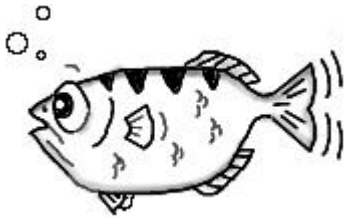


构建嵌入式Linux开发环境



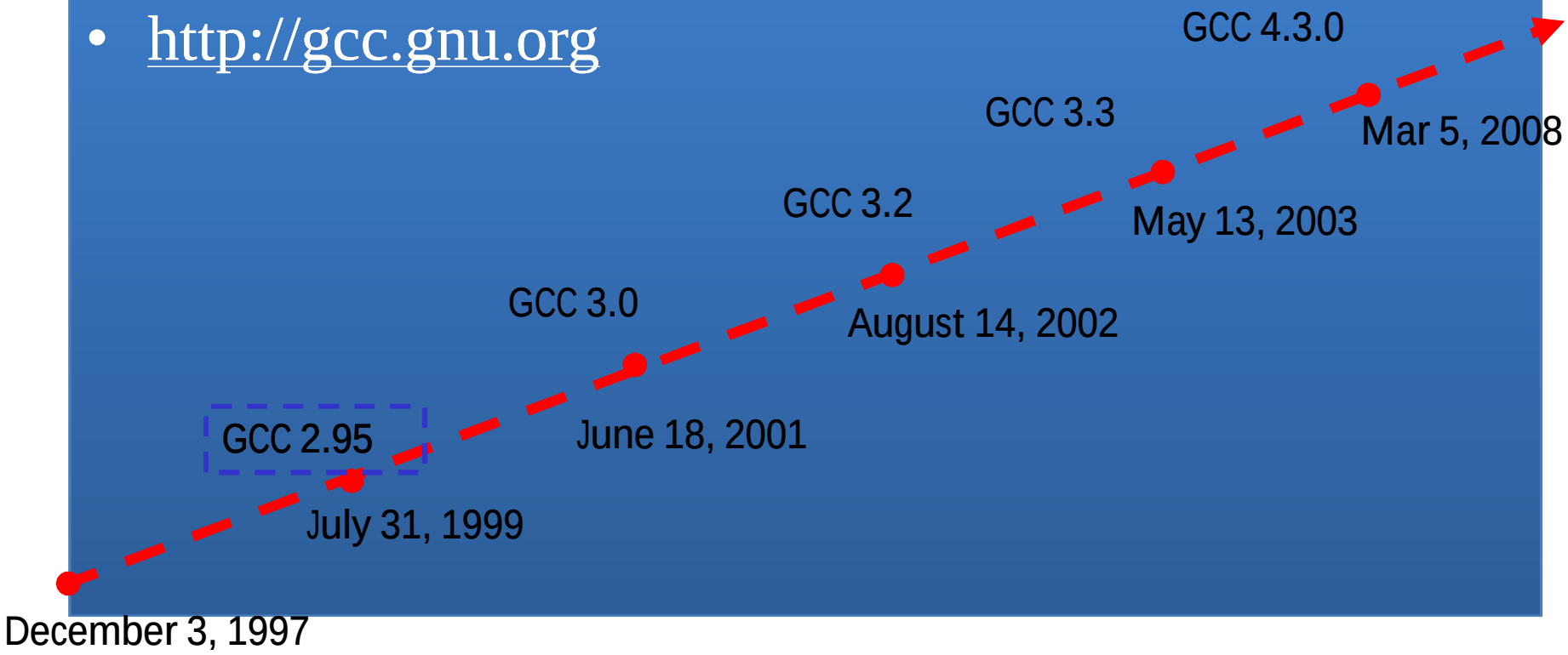
交叉编译环境





GCC编译器的版本

- GNU Compiler Collection
- C, C++, Objective-C, Fortran, Java, Ada
- http://gcc.gnu.org



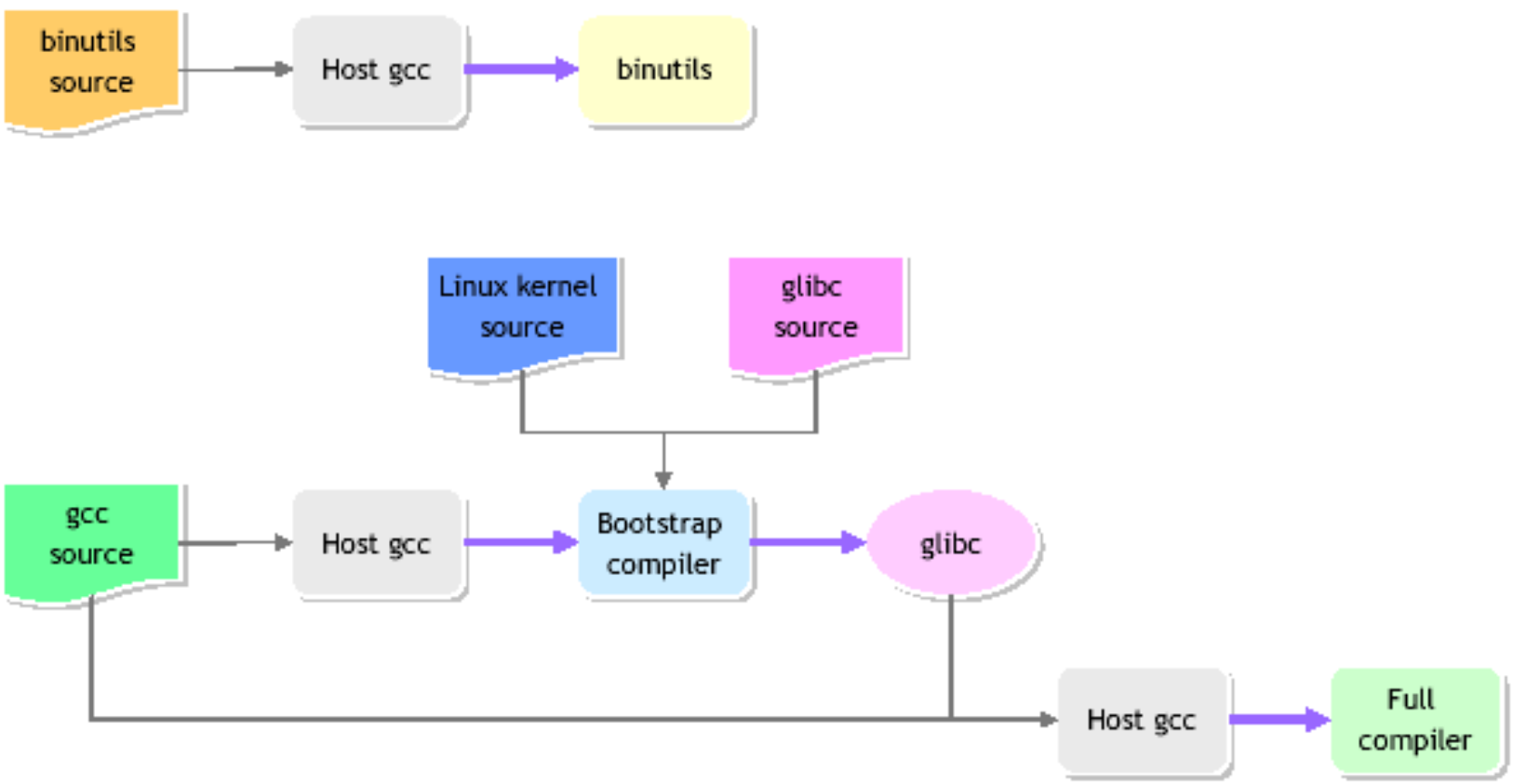
GCC 4.3的新特性

- GCC requires the GMP and MPFR libraries for building all the various front-end languages it supports. See the prerequisites page for version requirements.
- ColdFire targets now treat long double as having the same format as double. In earlier versions of GCC, they used the 68881 long double format instead.
- The m68k-uclinux target now uses the same calling conventions as m68k-linux-gnu. You can select the original calling conventions by configuring for m68k-uclinuxoldabi instead. Note that m68k-uclinuxoldabi also retains the original 80-bit long double on ColdFire targets.
- The -fforce-mem option has been removed because it has had no effect in the last few GCC releases.
- The i386 -msvr3-shlib option has been removed since it is no longer used.
- Fastcall for i386 has been changed not to pass aggregate arguments in registers, following Microsoft compilers.
- Support for the AOF assembler has been removed from the ARM back end; this affects only the targets arm-semi-aof and armel-semi-aof, which are no longer recognized. We removed these targets without a deprecation period because we discovered that they have been unusable since GCC 4.0.0.
- Support for the TMS320C3x/C4x processor (targets c4x-* and tic4x-*) has been removed. This support had been deprecated since GCC 4.0.0.
- Support for a number of older systems and recently unmaintained or untested target ports of GCC has been declared obsolete in GCC 4.3. Unless there is activity to revive them, the next release of GCC will have their sources permanently removed.

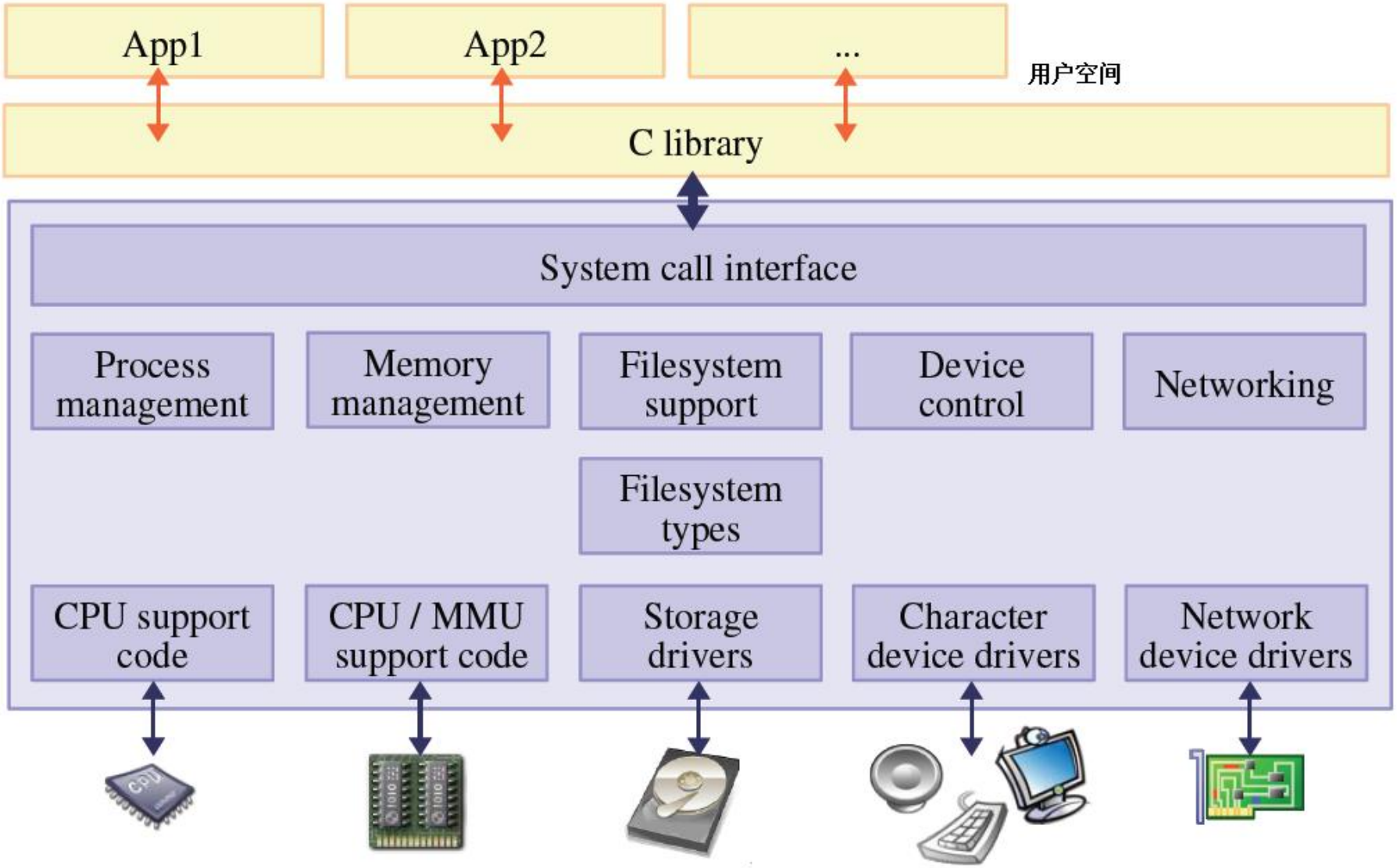
编译器针对ARM平台的新特性

- Compiler and Library support for ***Thumb-2*** and the ***ARMv7*** architecture has been added.

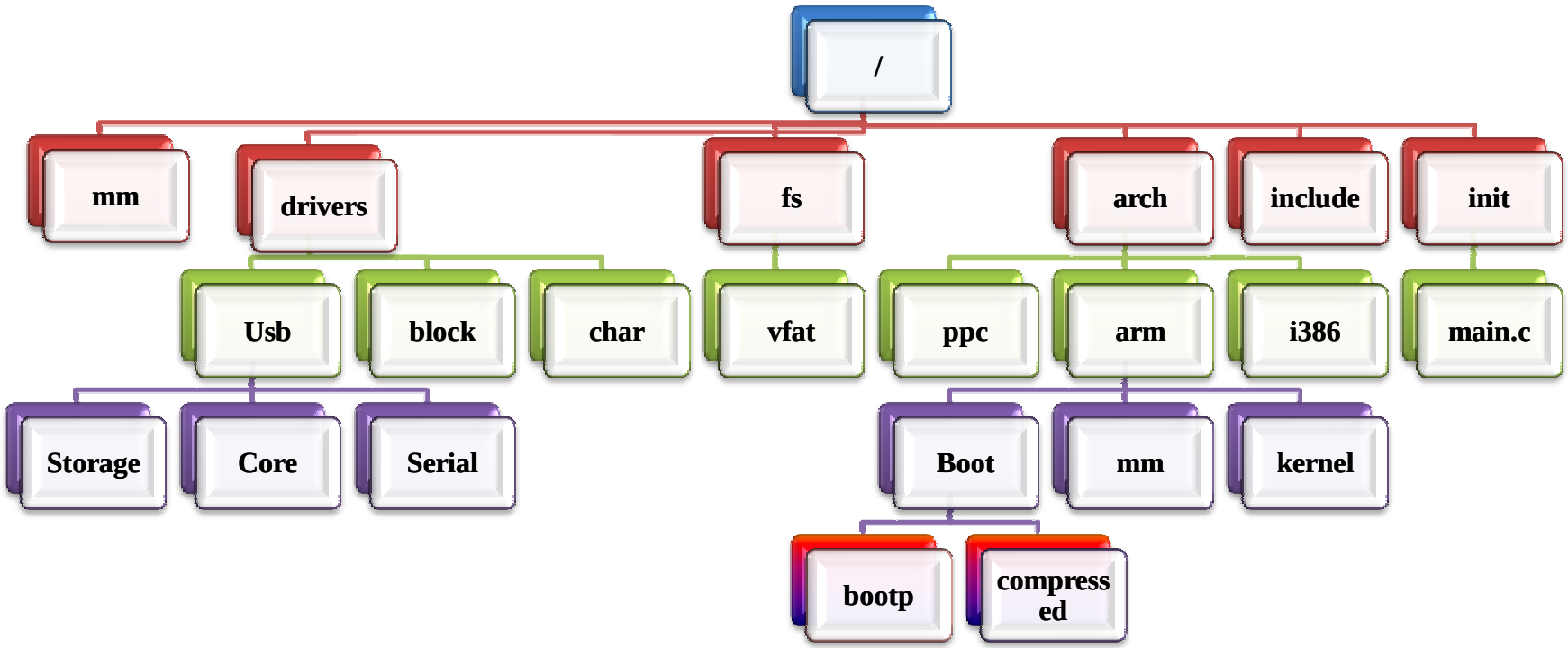
交叉编译的主要步骤

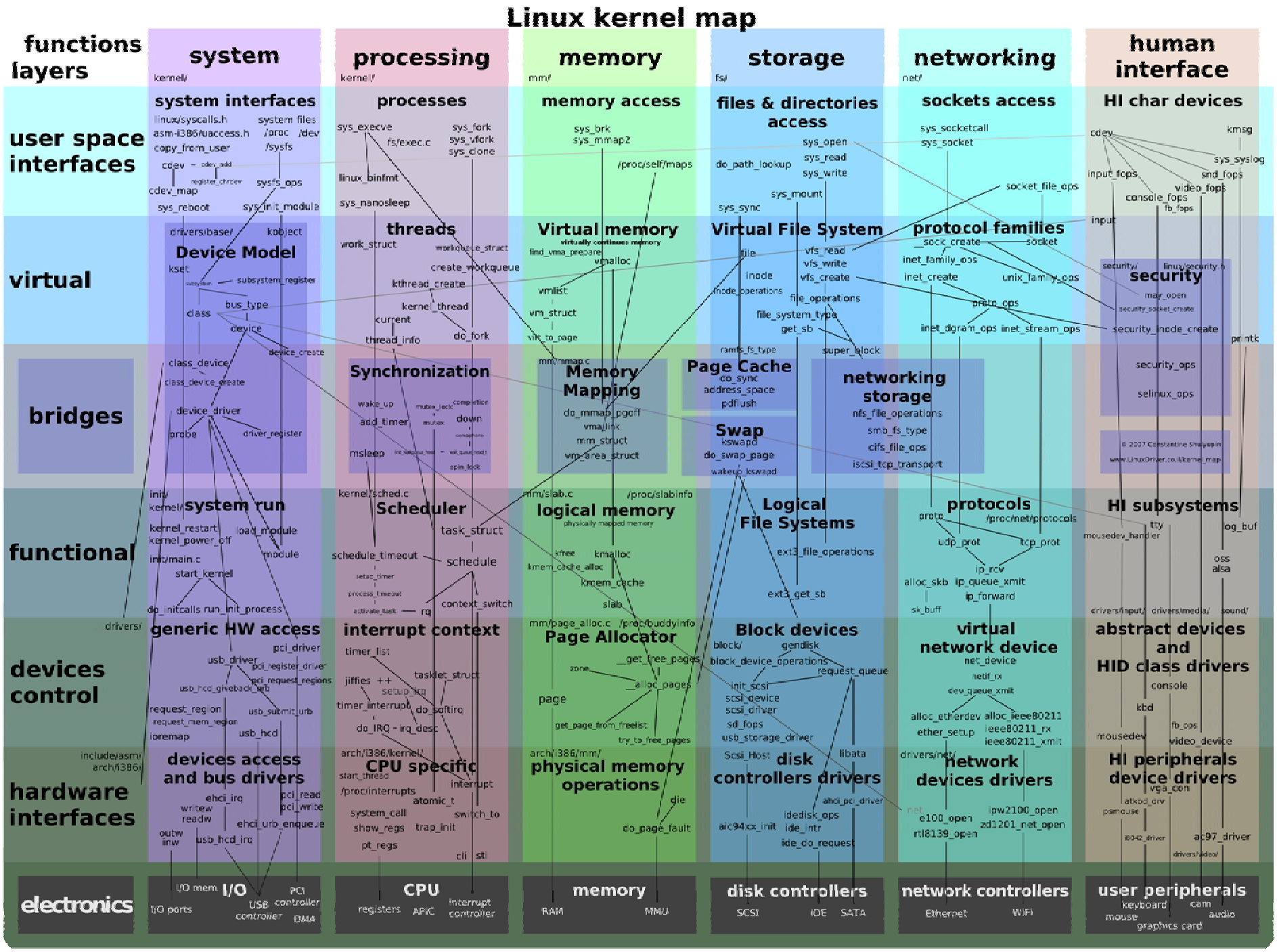

binutils source
Host gcc
binutils
Linux kernel source
glibc source
gcc source
Host gcc
Bootstrap compiler
glibc
Host gcc
Full compiler

Linux内核框架


App1
App2
...
用户空间
C library
System call interface
Process management
Memory management
Filesystem support
Device control
Networking
Filesystem types
CPU support code
CPU / MMU support code
Storage drivers
Character device drivers
Network device drivers

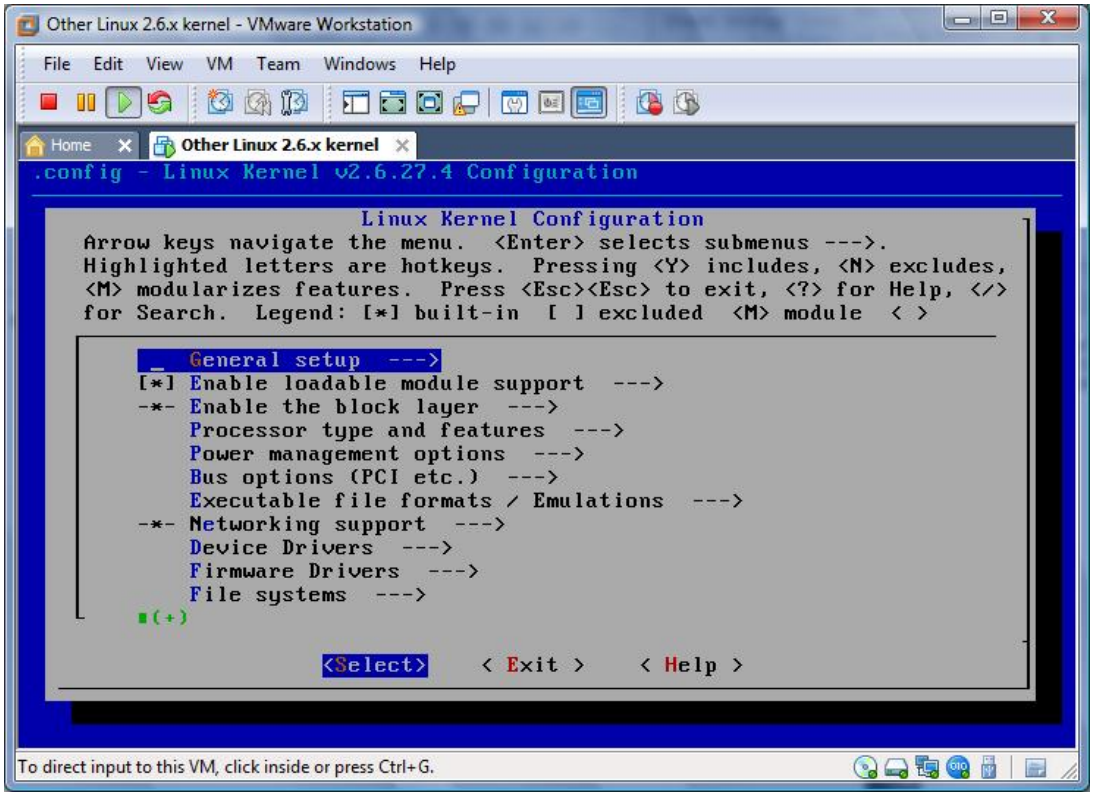
Linux内核代码结构




Linux kernel map
functions
layers
system
kernel/
processing
kernel/
memory
mm/
storage
fs/
networking
net/
human interface
user space interfaces
system interfaces
processes
memory access
files & directories access
sockets access
HI char devices
virtual
Device Model
threads
Virtual memory
Virtual File System
protocol families
security
bridges
Synchronization
Memory Mapping
Page Cache
networking storage
Swap
functional
system run
Scheduler
logical memory
Logical File Systems
protocols
HI subsystems
devices control
generic HW access
interrupt context
Page Allocator
Block devices
virtual network device
abstract devices and HID class drivers
hardware interfaces
devices access and bus drivers
CPU specific
physical memory operations
disk controllers drivers
network devices drivers
HI peripherals device drivers
electronics
I/O
CPU
memory
disk controllers
network controllers
user peripherals

配置内核的方法

- make menuconfig
- make xconfig
- make config



Linux调试工具

用户空间(User-space)

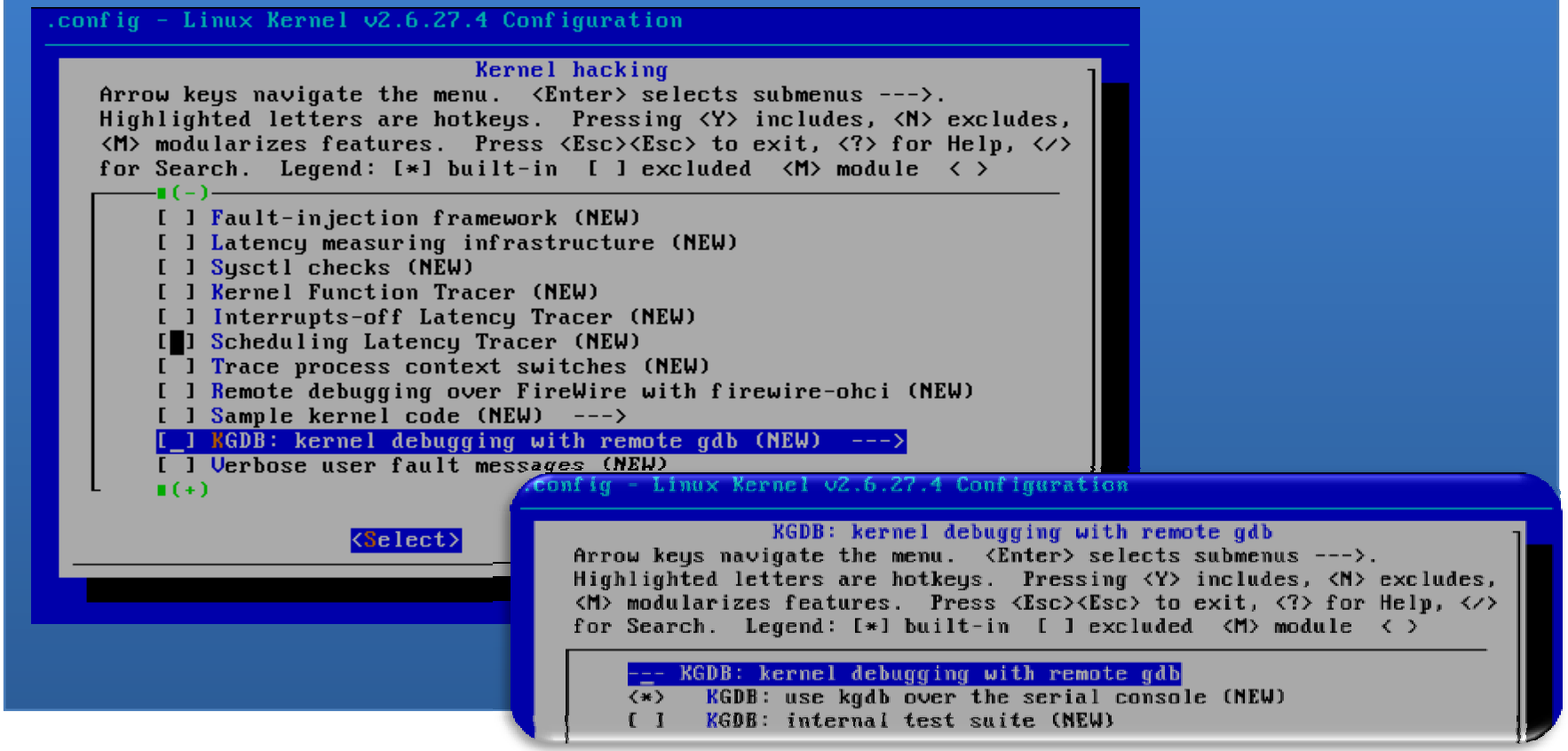
- Memory Debug: MEMWATCH / YAMD/ Valgrind/ Electric Fence
- strace
- GNU debugger (gdb)
- Magic key sequence
- Proc

内核空间(Kernel-space)

- Kernel source level debugger (kgdb)
- Built-in kernel debugger (kdb)
- Oops

内核对KGDB的支持

- 通过kgdb可以在内核代码中设置断点，单步调试和观察变量。



KGDB调试原理

两台运行Linux的PC，一个是Target，另一个是Host。

Target的内核需要有kgdb补丁，Host端运行gdb

两台PC通过串口相连，Target端内核停止到某个断点上，等待gdb连接。

KGDB环境设置


```
stty ispeed 115200 ospeed 115200 -F /dev/ttyS0
echo hello > /dev/ttyS0
```

```
stty ispeed 115200 ospeed 115200 -F /dev/ttyS1
cat /dev/ttyS1
```

oops

产生oops的原因：

- 内存访问越界
- 非法指令
- 使用了NULL指针
- 使用了不正确的指针值

oops的内容：

- CPU寄存器内容
- 页描述符表的位置
- 其他信息

Decoding oopsen: fault

```
Unable to handle kernel NULL pointer dereference at virtual
address  00000014
*pde = 00000000
Oops: 0000
CPU:     0
EIP:     0010:[<c017d558>]
EFLAGS: 00210213
eax: 00000000   ebx: c6155c6c   ecx: 00000038   edx: 00000000
esi: c672f000   edi: c672f07c   ebp: 00000004   esp: c6155b0c
ds: 0018   es: 0018   ss: 0018
Process tar (pid: 2293, stackpage=c6155000)
Stack: c672f000 c672f07c 00000000 00000038 00000060 00000000
c6d7d2a0 c6c79018
       00000001 c6155c6c 00000000 c6d7d2a0 c017eb4f c6155c6c
00000000 00000098
       c017fc44 c672f000 00000084 00001020 00001000 c7129028
00000038 00000069
Call Trace: [<c017eb4f>] [<c017fc44>] [<c0180115>] [<c018a1c8>]
[<c017bb3a>] [<c018738f>] [<c0177a13>]
       [<d0871044>] [<c0178274>] [<c0142e36>] [<c013c75f>]
[<c013c7f8>] [<c0108f77>] [<c010002b>]

Code: 8b 40 14 ff d0 89 c2 8b 06 83 c4 10 01 c2 89 16 8b 83 8c 01
```

(low address implies accessing a structure member)

Resolving addresses

(from oops output)

```
EIP:     0010:[<c017d558>]
```

(from System.map)

```
....
c017cdf0 T reiserfs_dir_fsync
c017ce80 t reiserfs_readdir
c017d2f0 t create_virtual_node
c017d780 t check_left
c017d8d0 t check_right
....
```

EIP = function base address + instruction offset

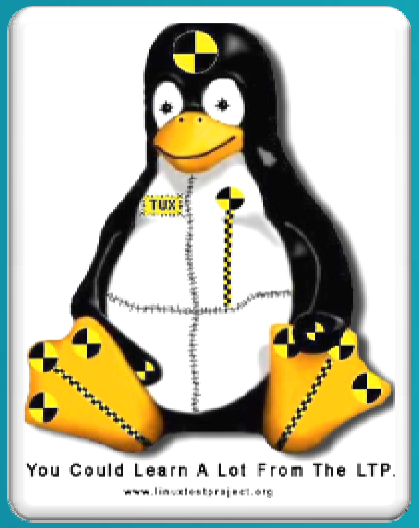
Decoding with ksymoops

```
...
Process tar (pid: 2293, stackpage=c6155000)
...

>>EIP; c017d558 <create_virtual_node+298/490>   <=====
Trace; c017eb4f <ip_check_balance+34f/ae0>
Trace; c017fc44 <reiserfs_kfree+14/50>
Trace; c0180115 <fix_nodes+115/450>
Trace; c018a1c8 <reiserfs_insert_item+88/110>
Trace; c017bb3a <reiserfs_new_inode+3da/500>
Trace; c018738f <pathrelse+1f/30>
Trace; c0177a13 <reiserfs_lookup+73/d0>
Trace; d0871044 <END_OF_CODE+9a77/???
Trace; c0178274 <reiserfs_mkdir+d4/1d0>
Trace; c0142e36 <d_alloc+16/160>
Trace; c013c75f <vfs_mkdir+7f/b0>
Trace; c013c7f8 <sys_mkdir+68/b0>
Trace; c0108f77 <system_call+33/38>
Trace; c010002b <startup_32+2b/139>

Code;  c017d558 <create_virtual_node+298/490>
00000000 <_EIP>:
Code;  c017d558 <create_virtual_node+298/490>   <=====
   0:   8b 40 14                  mov    0x14(%eax),%eax   <=====
Code;  c017d55b <create_virtual_node+29b/490>
   3:   ff d0                     call   *%eax

(cut)
```



SD2C
软件开发2.0技术大会
CSDN Software Development 2.0 Conference
2008

Csdn Programmer 程序员

Thank you