Operating System Principle, OS

2025年秋. 操作系统原理

第2章 操作系统硬件基础

课 程 组：邹德清, 李珍, 李志, 苏曙光

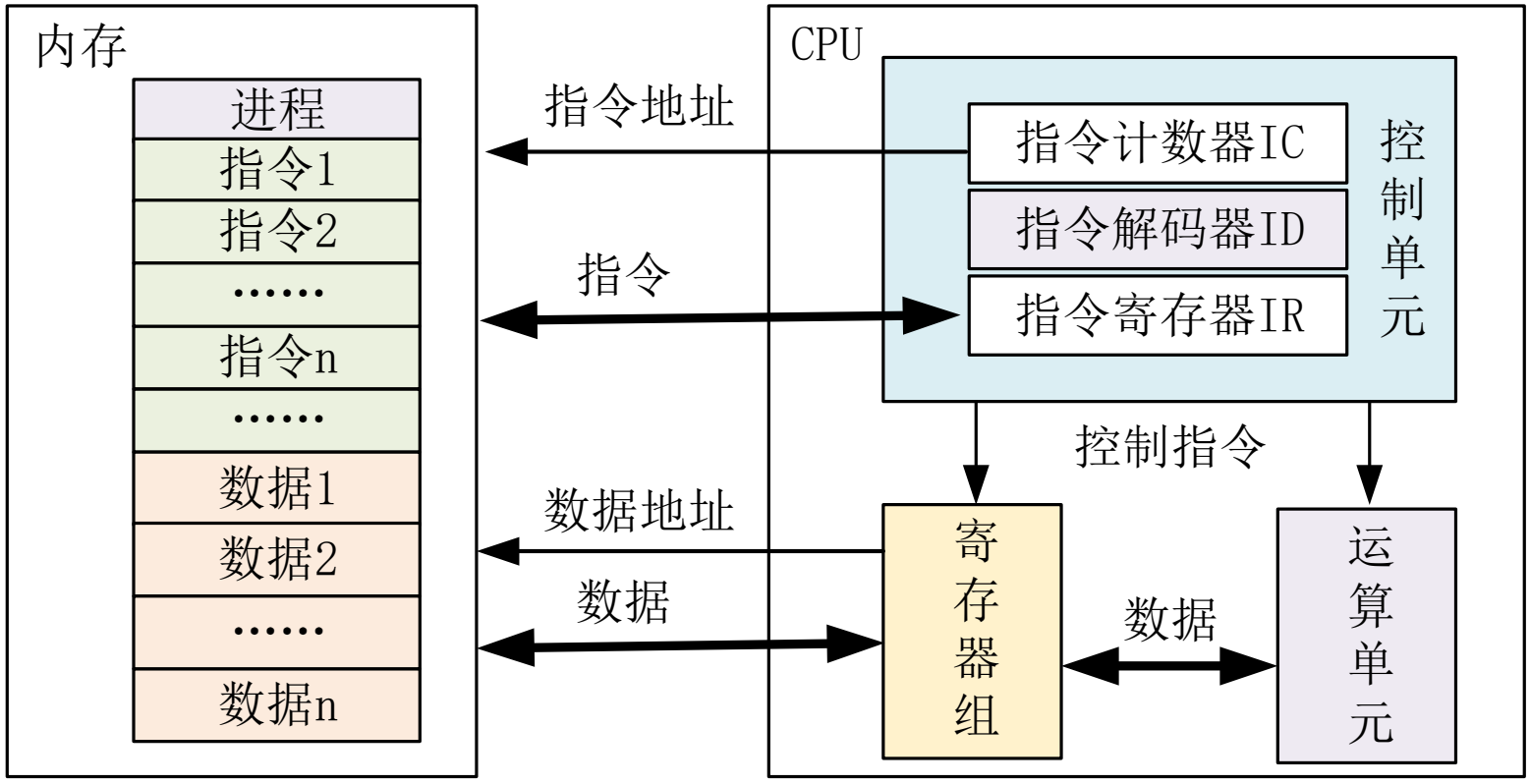
企业教师：华为认证专家(鸿蒙方向)

第2章 操作系统依赖的基础硬件

- 主要内容
  - 1.CPU的态/可信计算
  - 2.中断机制
  - 3.基本输入输出系统/BIOS
  - 4.操作系统启动过程
  - 5.操作系统的生成
- 重点
  - CPU的态
  - 中断响应的过程
  - 操作系统初始引导过程

操作系统依赖的最基本硬件

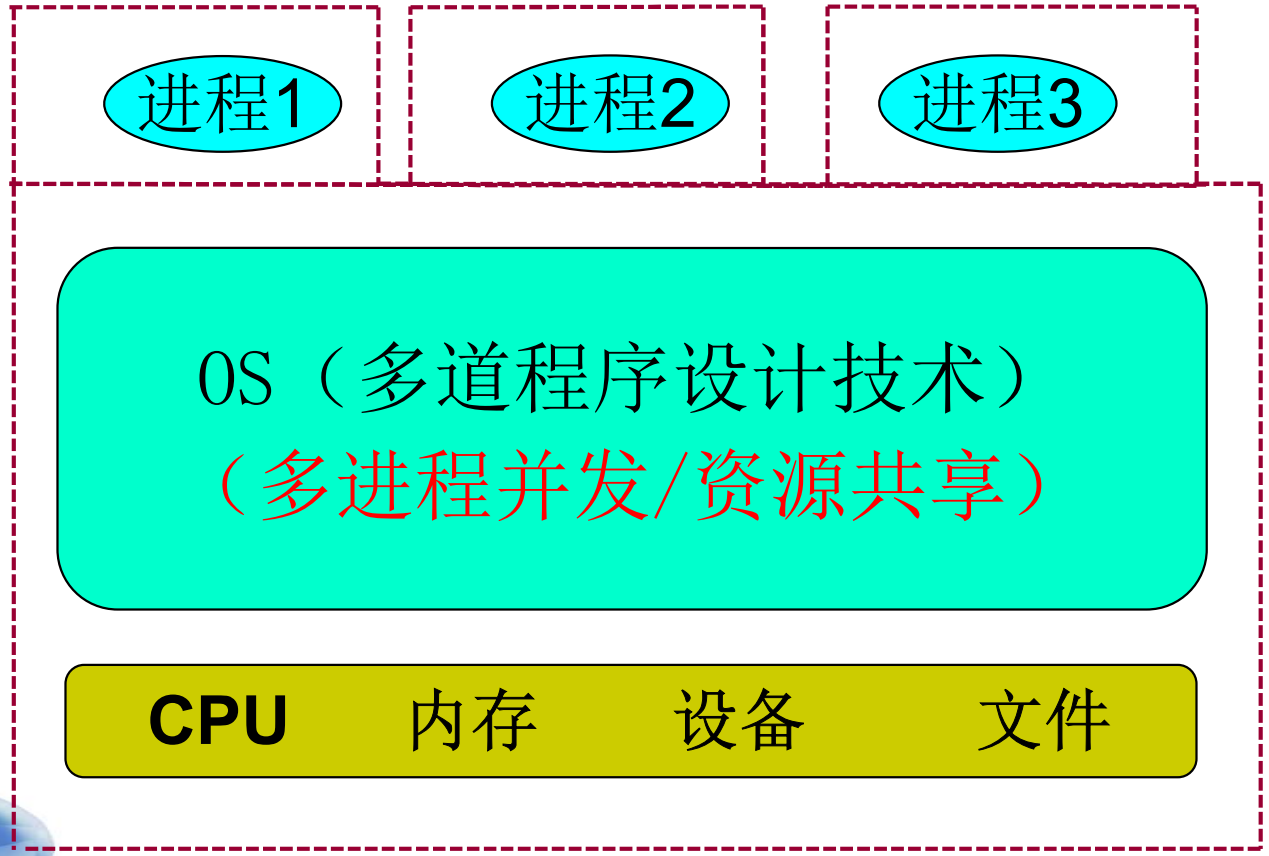
- 操作系统依赖的最基本硬件
  - CPU
  - 内存
  - 中断
  - 时钟



2.1 CPU的态/可信计算

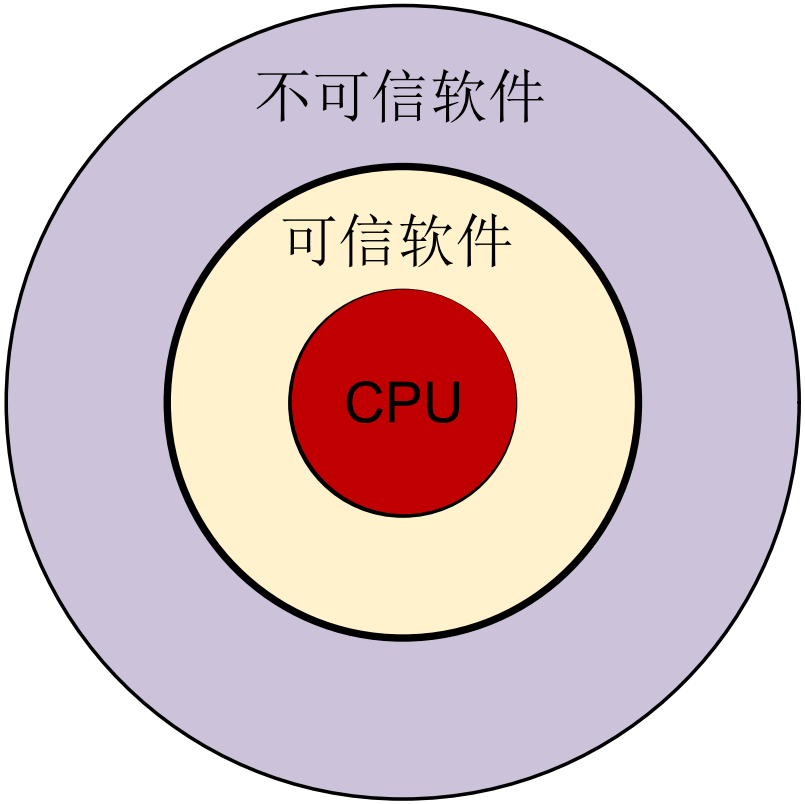
操作系统考虑的安全问题

- 共享/安全
  - 防止进程的信息被非法访问
  - 防止进程随意存取系统资源
  - 防止进程修改系统安全机制



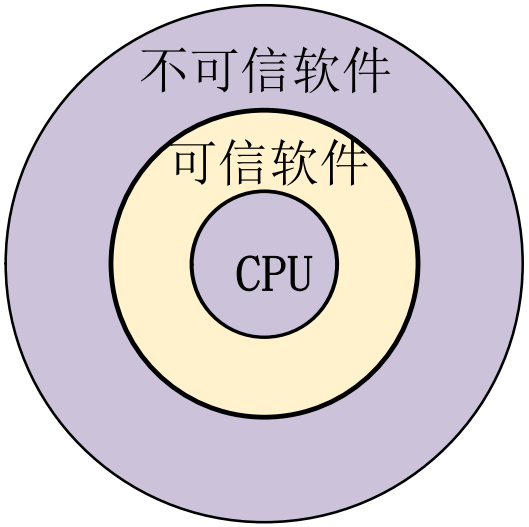
操作系统考虑的安全问题

- 解决策略
  - 1）软件分级/分类（例：可信软件与不可信软件）
    - 可信软件
      - 可以修改安全保护机制
      - 可以存取系统资源
      - 普通指令集 + 特权指令集
    - 不可信软件
      - 功能受限
      - 普通指令集
  - 2）硬件分模式（设置访问屏障）
    - 硬件模式 ↔ 软件级别
    - 模式(态)：描述指令使用和资源访问的权限



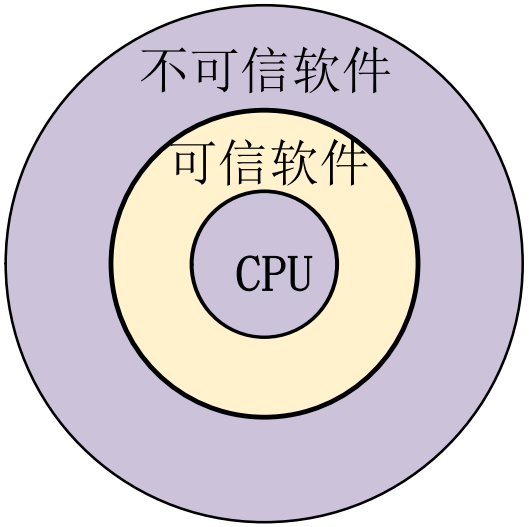
CPU的态（mode）

- 态的分类
  - 核态(Kernel mode)
    - ◆能够访问所有资源和执行所有指令
    - ◆管理程序/OS内核
  - 用户态 (User mode,目态)
    - ◆仅能访问部分资源，其它资源受限。
    - ◆用户程序
  - 管态(Supervisor mode)
    - ◆介于核态和用户态之间



CPU的态（mode）

- 硬件和OS对CPU的观察
  - ■硬件按“态”来区分CPU的状态
  - ■OS按“进程”来区分CPU的状态
    - ◆A,B,C,D：四个进程
    - ◆K/U：Kernel mode/ User mode



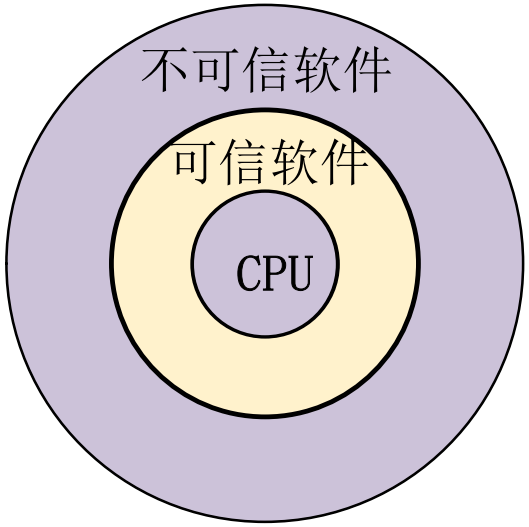
	进程	A	B	C	D
硬件→	核心态	K			K
硬件→	用户态		U	U	

OS ↓ A　OS ↓ B　OS ↓ C　OS ↓ D

CPU的态（mode）

- 硬件对态的支持
  - CPU设置模式位，表明当前的模式/态
    - 例(Intel CPU)：$CS_{0,1}$ ，$DS_{0,1}$ ，$SS_{0,1}$ ，$ES_{0,1}$
      - 指令执行前增加“权限是否满足”的条件判断。
        - 比较CPL|RPL|DPL ;Privilege Level：特权级

```
JMP start          ;JMP CS:start
MOV BL, hello      ;MOV BL, DS:hello
```



CS/DS/SS/ES

15 … 3	2	1 0
INDEX	TI	CPL/RPL
0 0 0 0 0 0 0 0 0 1 1 0	0	1 1

CPU的态（mode）

- 常规保护错误（General-Protection Fault #GP）

CPU的态（mode）

- 常规保护错误（General-Protection Fault #GP）

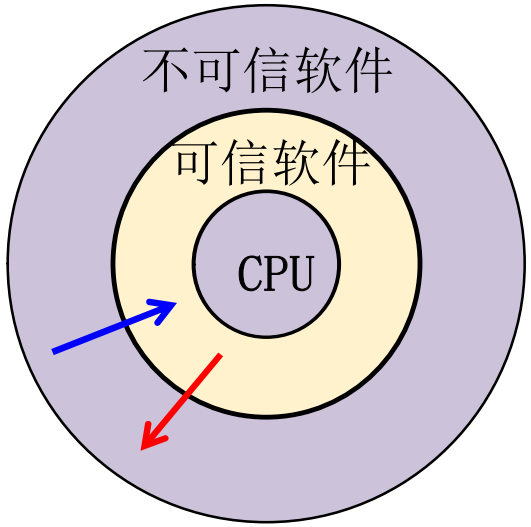
```
1  void trap_init(void)
2  {
3      set_trap_gate(0,&divide_error);//除数是0
4      set_trap_gate(1,&debug);//单步调试
5      set_trap_gate(2,&nmi);
6      set_system_gate(3,&int3);   //int3 ~ 5
7      set_system_gate(4,&overflow);
8      set_system_gate(5,&bounds);
9      set_trap_gate(6,&invalid_op);
10     set_trap_gate(7,&device_not_available);
11     set_trap_gate(8,&double_fault);
12     set_trap_gate(9,&coprocessor_segment_overrun);
13     set_trap_gate(10,&invalid_TSS);
14     set_trap_gate(11,&segment_not_present);
15     set_trap_gate(12,&stack_segment);
16     set_trap_gate(13,&general_protection);
17     set_trap_gate(14,&page_fault);
18     set_trap_gate(15,&reserved);
19     set_trap_gate(16,&coprocessor_error);
20     for (inti=17;i<48;i++)
21         set_trap_gate(i,&reserved);
22     set_trap_gate(45,&irq13);
23     outb_p(inb_p(0x21)&0xfb,0x21);
24     outb(inb_p(0xA1)&0xdf,0xA1);
25     set_trap_gate(39,&parallel_interrupt);
26 }
```

特权指令

```
417     SHL EAX, 4
418     ADD EAX, LABEL_GDT          ; EAX <- gdt 基地址
419     MOV DWORD [GdtPtr + 2], EAX ; [GdtPtr + 2] <- gdt 基地址
420
421     ; 为加载 IDTR 作准备
422     XOR EAX, EAX
423     MOV AX, DS
424     SHL EAX, 4
425     ADD EAX, LABEL_IDT          ; EAX <- idt 基地址
426     MOV DWORD [IdtPtr + 2], EAX ; [IdtPtr + 2] <- idt 基地址
427     ; 保存 IDTR
428     SIDT [_SavedIDTR]
429     ; 保存中断屏蔽寄存器(IMREG)值
430     IN   AL, 21h
431     MOV  [_SavedIMREG], AL
432
433     ; 加载 GDTR
434     LGDT [GdtPtr]
435     ; 关中断
436     CLI
437     ; 加载 IDTR
438     LIDT [IdtPtr]
```

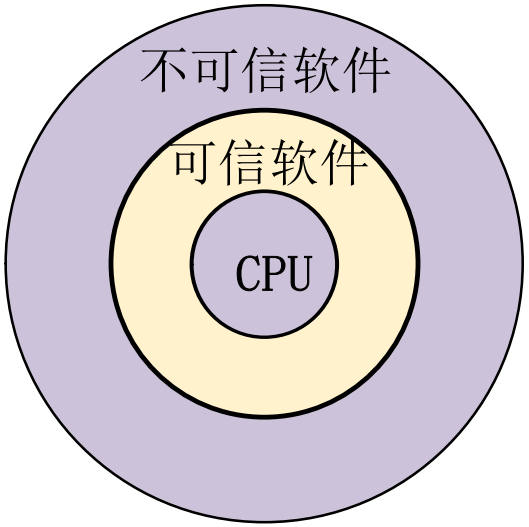
用户态和核态之间的转换

- 用户态向核态转换
  - 用户请求OS提供服务
  - 用户进程产生错误（内部中断）
  - 用户态企图执行特权指令
  - 发生中断
- 核态向用户态转换的情形
  - 一般是中断返回：IRET



特权指令

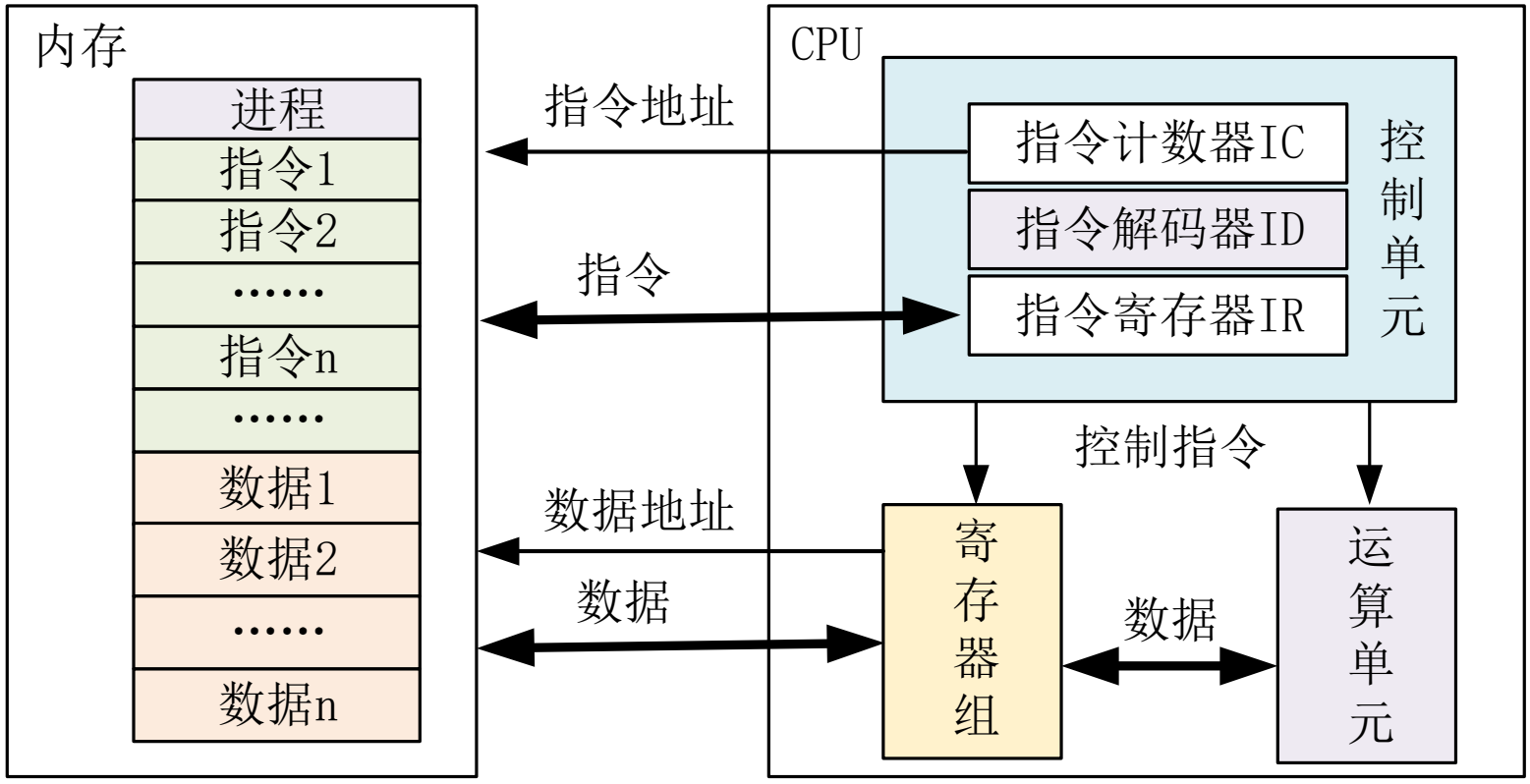
- 类别
  - ■① 涉及外部设备的输入/输出指令
  - ■② 修改特殊寄存器的指令
  - ■③ 改变机器状态的指令
- 例子
  - ■LGDTR/LIDTR/CLTS：装载特殊寄存器
  - ■STI/CTI：允许和禁止中断
  - ■HLT：停止CPU的工作
  - ■IN/OUT：执行I/O操作



2.4 中断机制

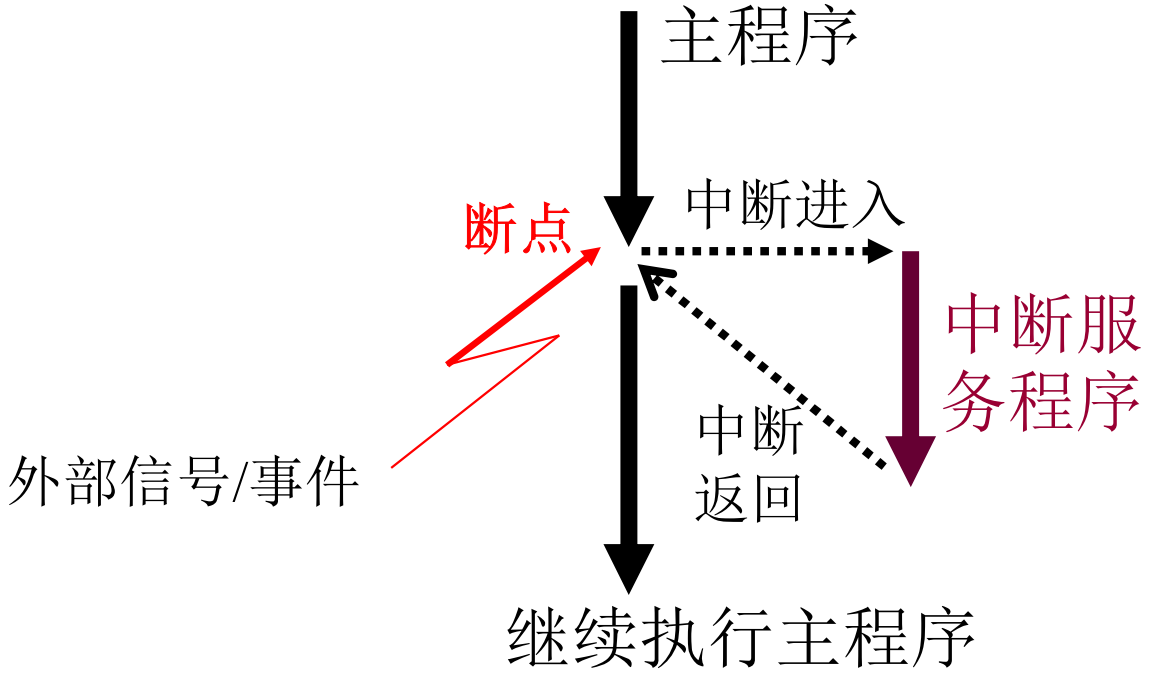
操作系统依赖的最基本硬件

- 操作系统依赖的最基本硬件
  - CPU
  - 内存
  - 中断
  - 时钟



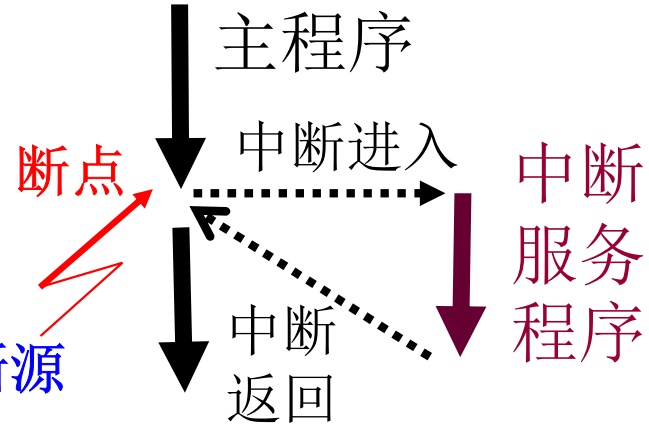
中断机制

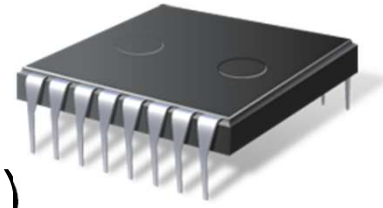
- 中断定义
  - 指CPU对突发的外部事件的反应过程或机制。
  - CPU收到外部信号（中断信号）后，停止当前工作，转去处理该外部事件，处理完毕后回到原来工作的中断处（断点）继续原来的工作。



中断的一些概念

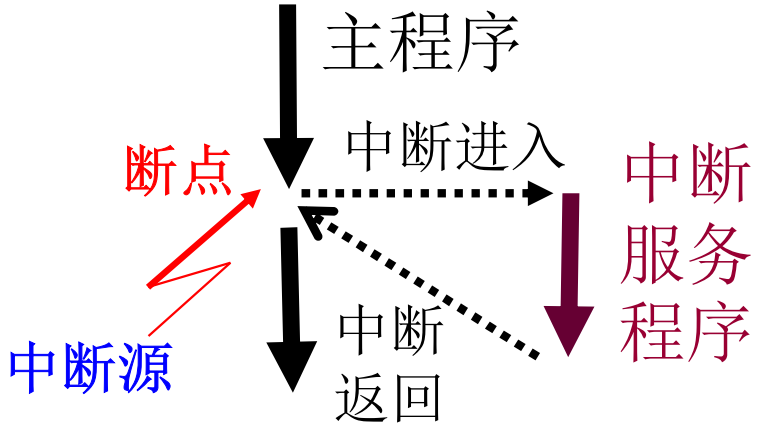
- 中断源
  - 引起系统中断的事件称为中断源
- 中断类型
  - 强迫中断和自愿中断
    - ◆强迫中断：程序没有预期：例：外部中断
    - ◆自愿中断：程序有预期。例：访管指令/INT 21H
  - 外中断（中断）和内中断（俘获）
    - ◆外中断：由CPU外部事件引起。例：外部中断(I/O)
    - ◆内中断：由CPU内部事件引起。例：访管指令、程序中断
  - 外中断：不可屏蔽中断和可屏蔽中断
    - ◆不可屏蔽中断：中断原因很紧要，CPU必须响应
    - ◆可屏蔽中断：中断原因不很紧要，CPU可以不响应





中断的一些概念

- 断点（保护和恢复：硬件处理）
  - 程序中断的地方
    - 将要执行的下一条指令的地址
  - CS:IP（FLAGS、SS、SP）
- 现场 （保护和恢复：程序处理）
  - 程序正确运行所依赖的信息集合。
    - 相关寄存器、断点(FLAGS,SS,SP)
- 现场的处理
  - 现场保护：进入中断服务程序之前：CPU→栈
  - 现场恢复：退出中断服务程序之后：栈→CPU

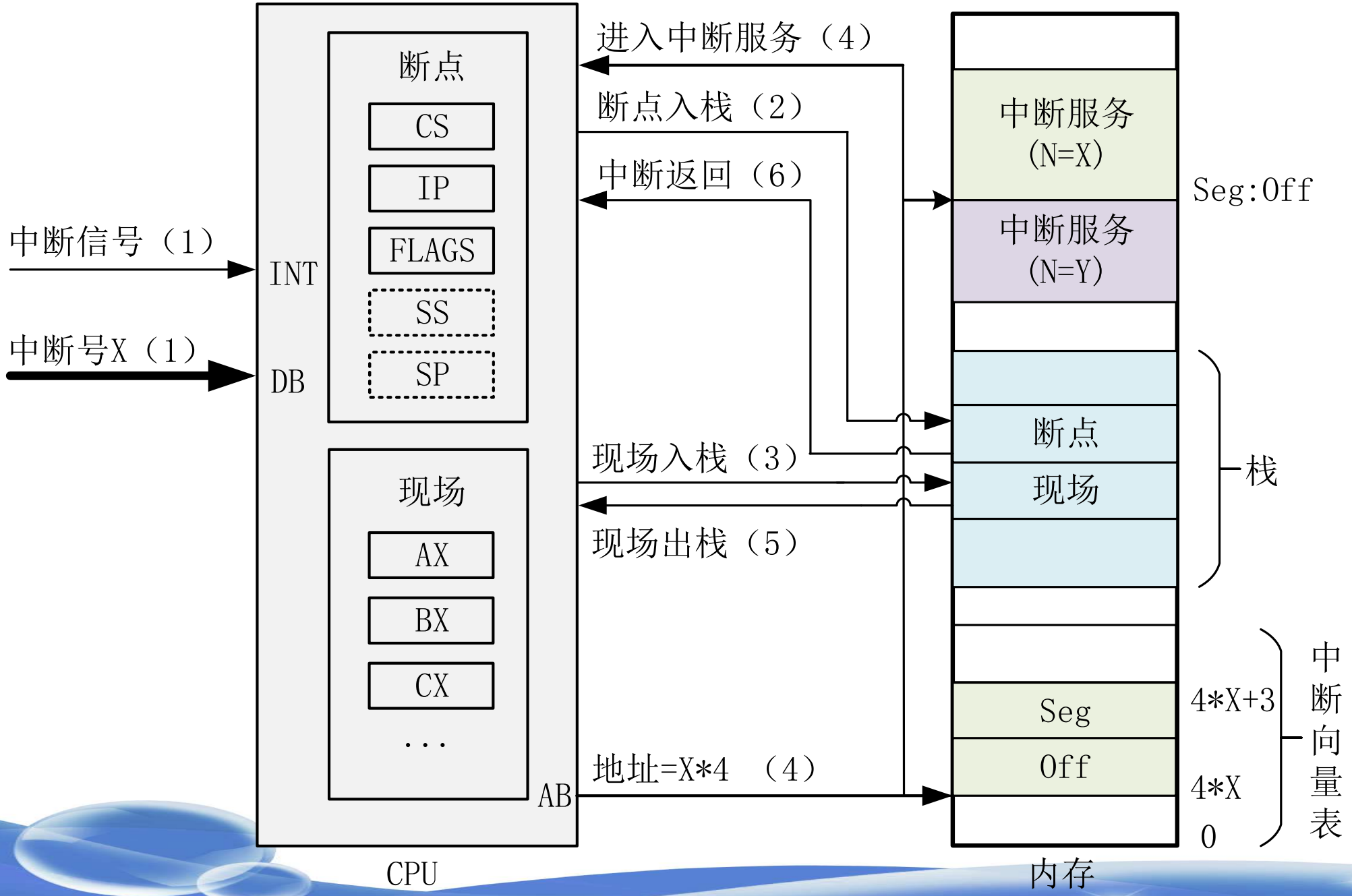



```
BUF   DB   'Hello World! $'
      ……
MOV   DX，  OFFSET  BUF
MOV   AH，  09H
INT   21H
```

中断响应过程

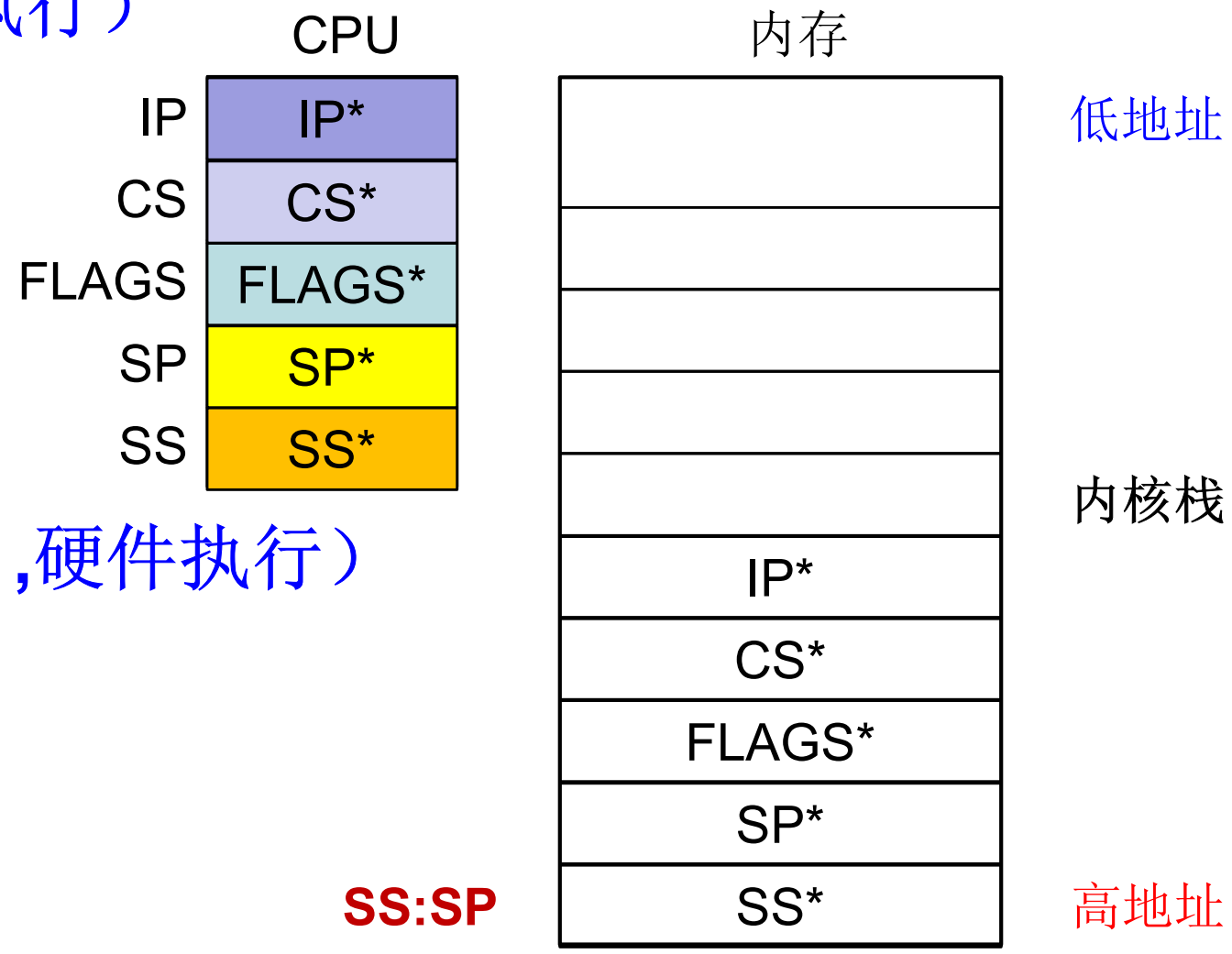
- 中断响应过程
  - （1）识别中断源
  - （2）保护断点
  - （3）保护现场(中断服务程序的前段)
  - （4）中断服务程序(主体)
  - （5）恢复现场(中断服务程序的末段)
  - （6）中断返回(恢复断点)

中断响应过程


断点
CS
IP
FLAGS
SS
SP
现场
AX
BX
CX
...
INT
DB
AB
CPU
中断信号（1）
中断号X（1）
进入中断服务（4）
断点入栈（2）
中断返回（6）
现场入栈（3）
现场出栈（5）
地址=X*4 （4）
中断服务
(N=X)
中断服务
(N=Y)
断点
现场
Seg
Off
Seg:Off
栈
4*X+3
4*X
0
中断向量表
内存

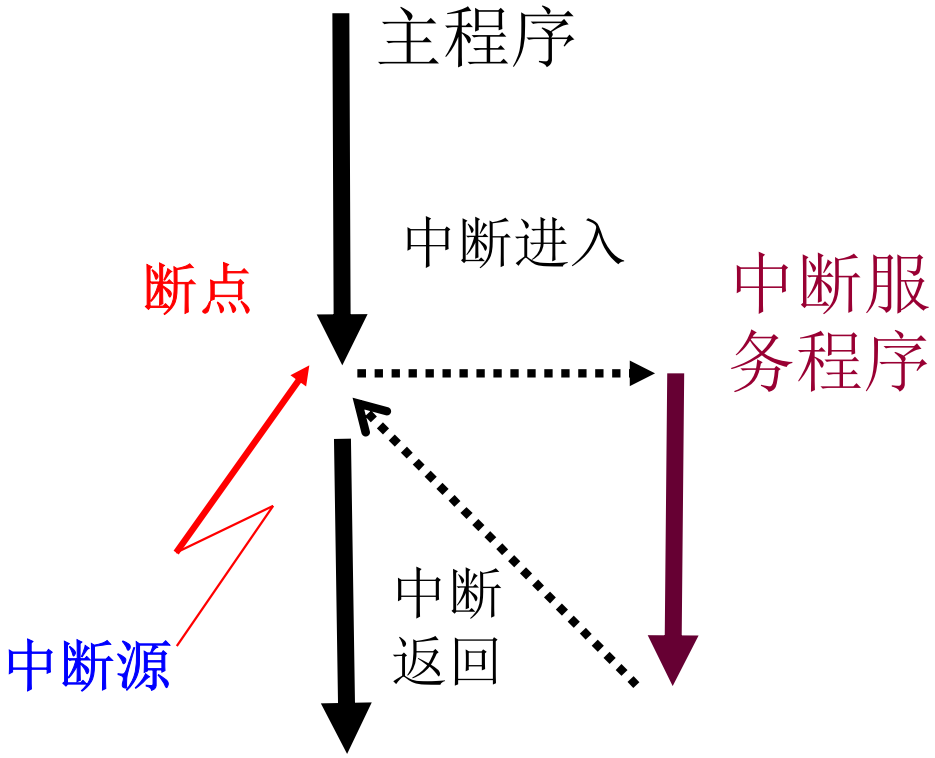
现场的保护和恢复(断点)

- 现场保护（硬件执行）
  - PUSH SS
  - PUSH SP
  - PUSHF
  - PUSH CS
  - PUSH IP
- 现场恢复（**IRET** ,硬件执行）
  - POP IP
  - POP CS
  - POPF
  - POP SP
  - POP SS



中断机制

- 中断响应的实质
  - 交换指令执行地址
  - 交换CPU的态
  - 工作
    - 现场保护和恢复
    - 参数传递（通信）
- 引入中断的目的
  - 实现并发活动
  - 实现实时处理
  - 故障自动处理



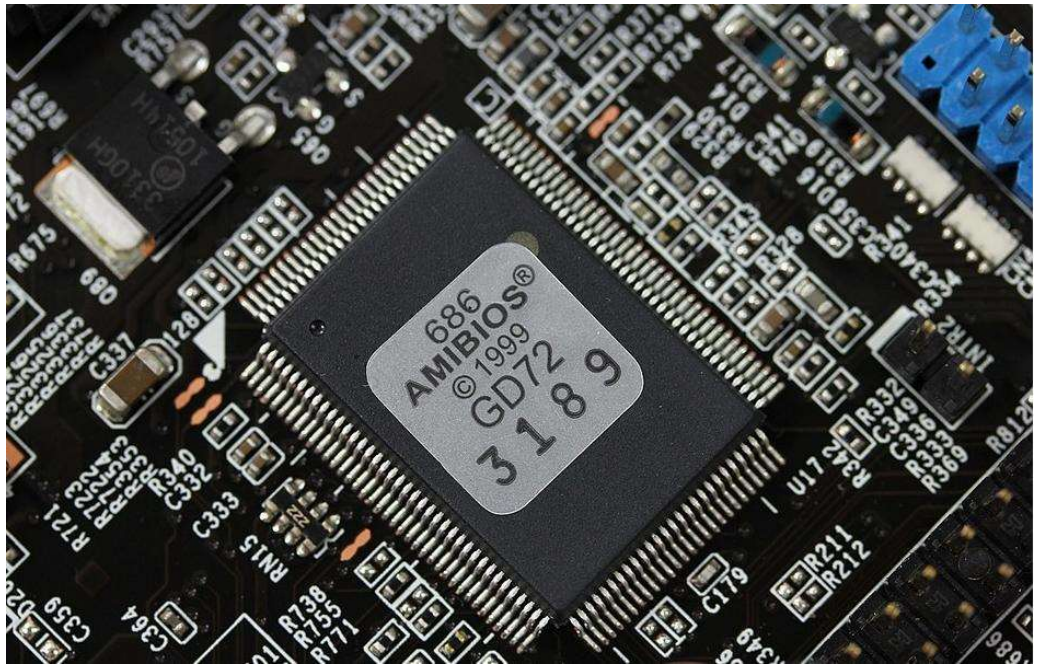
2.5 基本输入输出系统/BIOS

系统BIOS的作用


American
Megatrends
www.ami.com
AMIBIOS(C)2006 American Megatrends, Inc.
ASUS P5B-Deluxe ACPI BIOS Revision 0507
CPU : Intel(R) Core(TM)2 CPU 6400 @ 2.13GHz
Speed : 4.01 GHz Count : 2
Press DEL to run Setup
Press F8 for BBS POPUP
Press ALT+F2 to boot from System Recovery
PC2-4300 Dual Channel Interleaved
Initializing USB Controllers .. Done.
2048MB OK

系统BIOS的作用

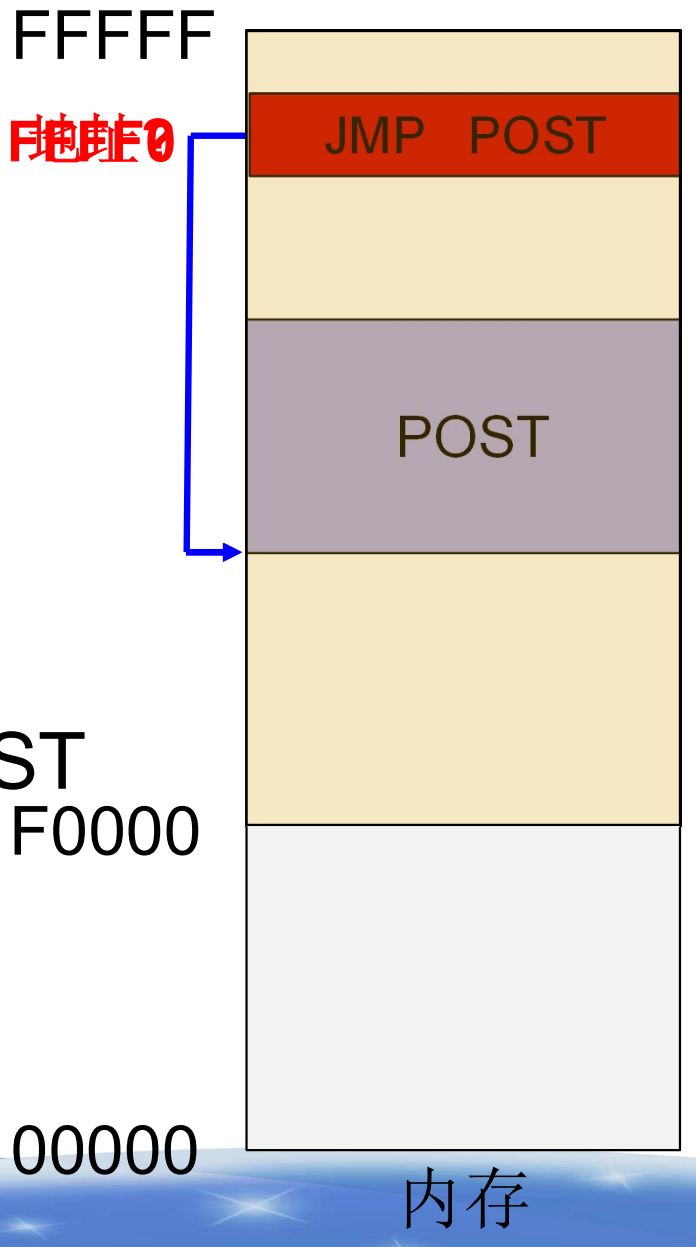
- 系统**BIOS**
  - 固件（firmware）
  - 地址范围：F0000- FFFFF
- 系统**BIOS**作用
  - （1）加电自检(POST)
    - 加电或复位开始执行
  - ……



加电自检POST

- 加电自检(Power On Slef-Test)
  - 初始化基本硬件
    - CPU
    - 内存
    - 显卡
    - …
  - 若自检错误则通过喇叭鸣叫提示
  - 按下PowerOn或Reset开始执行POST
  - JMP指令在（　　）单元

```
JMP   POST      ; 加电自检
```

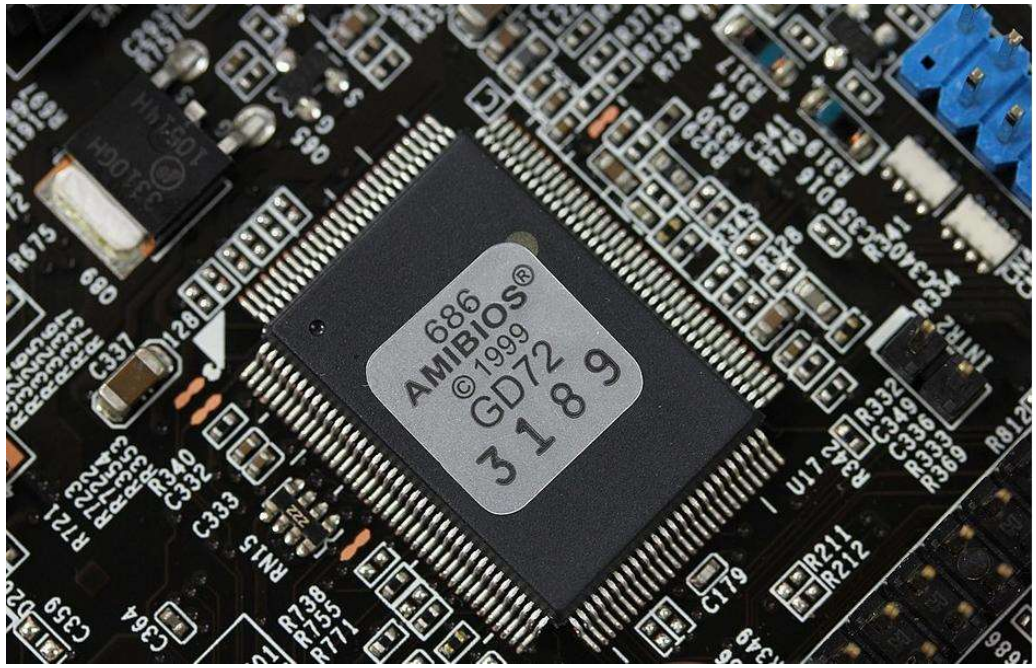


系统BIOS的作用


CMOS SETUP UTILITY
AWARD SOFTWARE, INC.
STANDARD CMOS SETUP
BIOS FEATURES SETUP
CHIPSET FEATURES SETUP
POWER MANAGEMENT SETUP
PNP/PCI CONFIGURATION
LOAD BIOS DEFAULTS
LOAD OPTIMUM SETTINGS
INTEGRATED PERIPHERALS
SUPERVISOR PASSWORD
USER PASSWORD
IDE HDD AUTO DETECTION
SAVE & EXIT SETUP
EXIT WITHOUT SAVING
Esc : Quit
F10 : Save & Exit Setup
↑ ↓ → ← : Select Item
(Shift)F2 : Change Color
Time, Date, Hard Disk, Type...

系统BIOS的作用

- 系统**BIOS**
  - 固件（firmware）
  - 地址：F0000- FFFFF
- 系统**BIOS**作用
  - （1）加电自检(POST)
    - 加电或复位开始执行
  - （2）CMOS设置
    - 交互式设置系统参数
  - ……

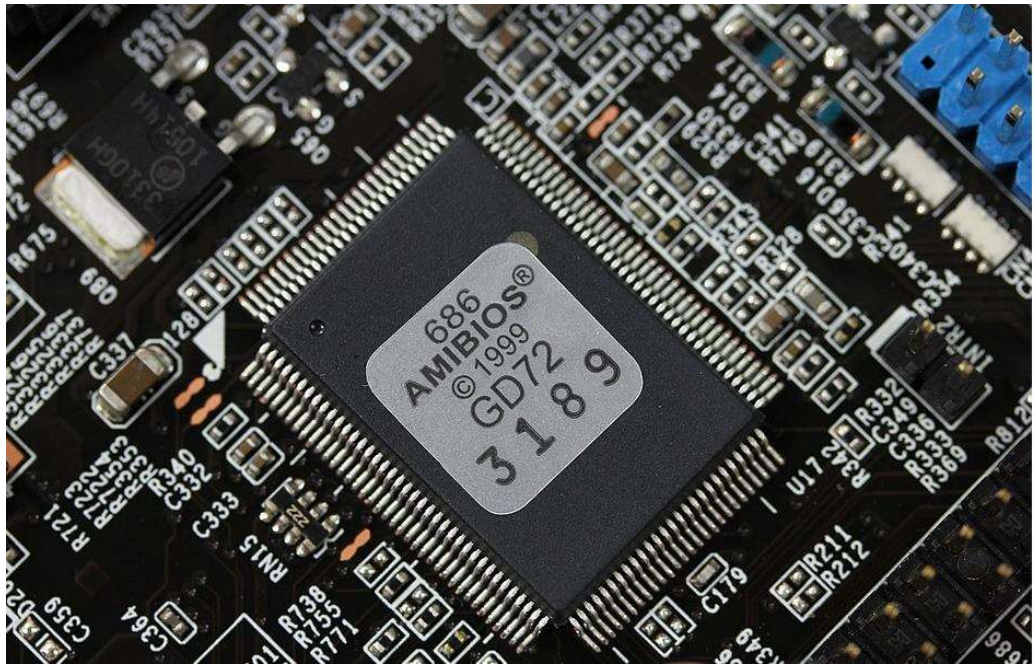


系统BIOS的作用

```
1   org 0x7c00
2   ; 清屏
3   mov ax, 0600h
4   mov bx, 0700h
1     mov ax, cs
2     mov ds, ax
3     mov ah, 0
4     int 16h
5     cmp al, 1ch ;al:ASCII码...
6     je HandleEnter
7     jmp $
8
9   HandleEnter:
10    pop ax
11    mov al, 0
12    pop ax
13    ret
18  int 10h
19  ; 原地停留
20  jmp $
21  BootMessage: db "Start Boot ..."
```

系统BIOS的作用

- 系统**BIOS**
  - 固件（firmware）
  - 地址：F0000- FFFFF
- 系统**BIOS**作用
  - （1）加电自检(POST)
    - 加电或复位开始执行
  - （2）CMOS设置
    - 交互式设置系统参数
  - （3）基本I/O服务
    - BIOS中断
  - ……




```
3       mov ah, 0
4       int 16h
5       cmp al, 1ch ;al:ASCII码
```

基本I/O服务（BIOS中断）

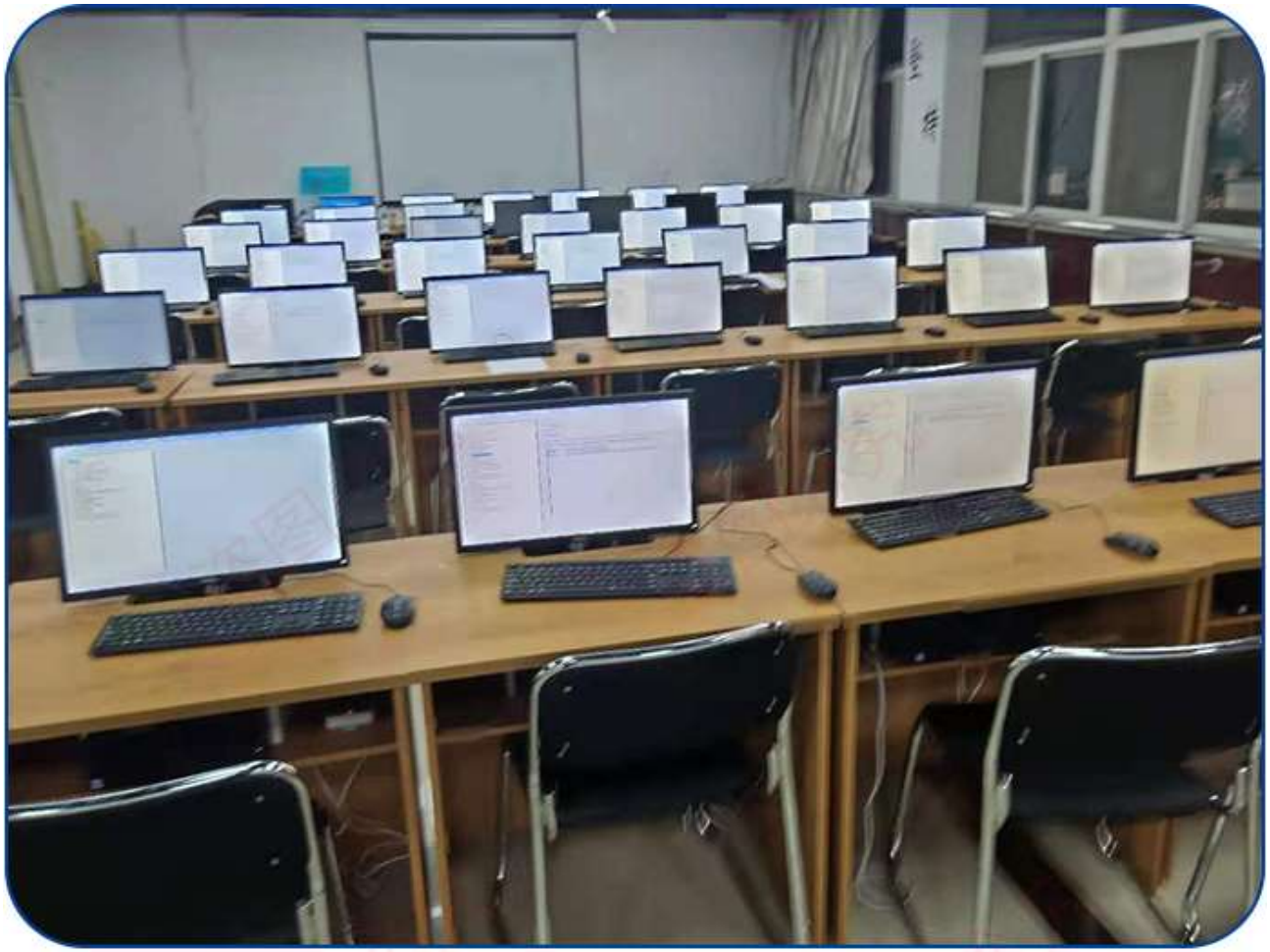
- **BIOS使用的中断类型号为10H ~ 1FH** 例：INT 13H

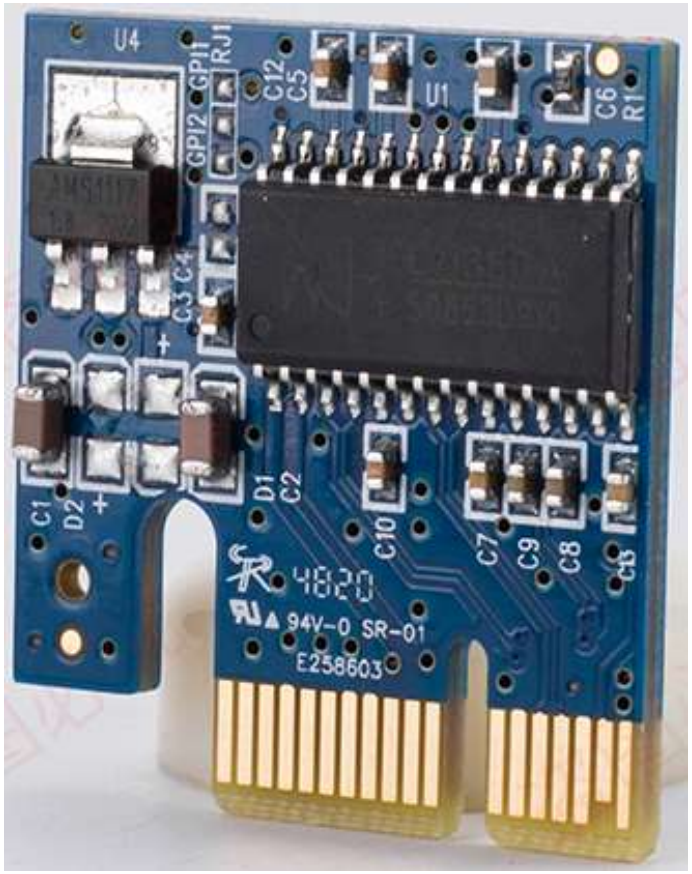
中断号	功能	中断号	功能
10H	显示器 I/O 调用	18H	ROM BASIC 入口
11H	获取设备配置调用	19H	自举程序入口
12H	获取存储器大小调用	1AH	时间日期调用
13H	软盘 I/O 调用	1BH	Ctrl-Break 控制
14H	异步通信口调用	1CH	定时处理
15H	磁带 I/O 调用	1DH	显示器参数表
16H	键盘 I/O 调用	1EH	软盘参数表
17H	打印机 I/O 调用	1FH	字符点阵结构参数表

例：INT 13H 软盘I/O调用的子功能（例：02H读扇区）

子功能号	子功能	子功能号	子功能
00H	磁盘系统复位	0AH	读长扇区
01H	读取磁盘系统状态	0BH	写长扇区
02H	读扇区	0CH	查寻
03H	写扇区	0DH	硬盘系统复位
04H	检验扇区	0EH	读扇区缓冲区
05H	格式化磁道	0FH	写扇区缓冲区
06H	格式化坏磁道	10H	读取驱动器状态
07H	格式化驱动器	11H	校准驱动器
08H	读取驱动器参数	12H	控制器 RAM 诊断
09H	初始化硬盘参数	13H	控制器驱动诊断

PCI卡的BIOS程序





例：INT 13H 软盘I/O调用的子功能（例：02H读扇区）

- 读第**21**号扇区到内存**10000H**处
  - ■注：假定每个磁道有18个扇区

```
1  MOV   AX,  1000H
2  MOV   ES,  AX
3  MOV   BX,  0     ;内存目标地址
4  MOV   AL,  1     ;扇区数
5  MOV   CH,  0     ;柱面号（C）
6  MOV   DH,  1     ;磁头号（H）
7  MOV   CL,  3     ;扇区号（S）
8  MOV   DL,  0     ;驱动器号
9  MOV   AH,  02H   ;子功能号
10 INT   13H        ;BIOS中断号
```

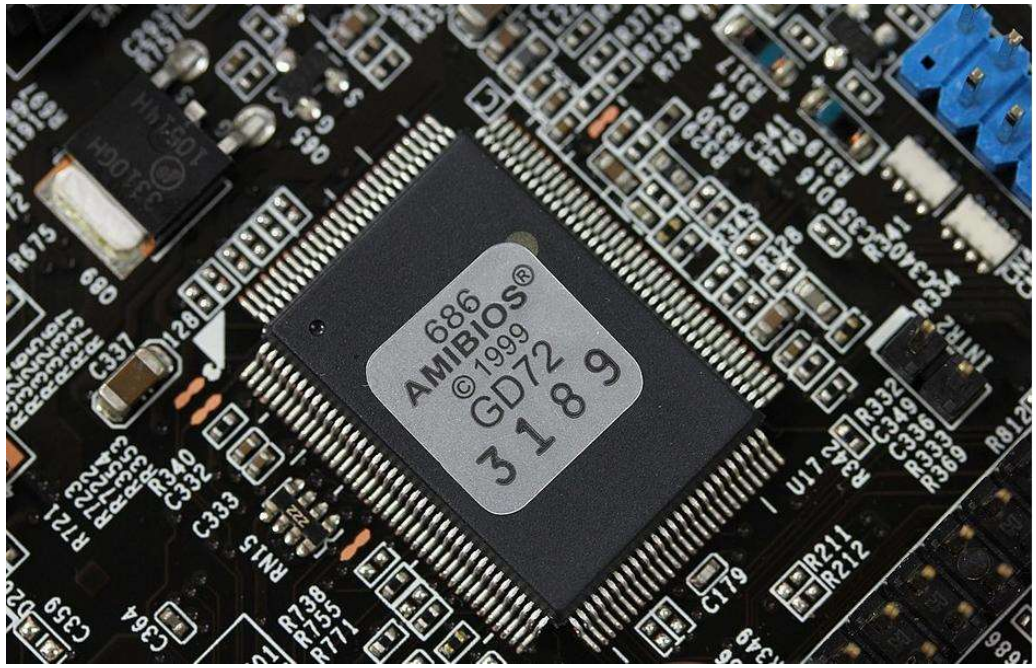
例：INT 13H 软盘I/O调用的子功能（例：02H读扇区）

- 入口参数（**AH＝02H**）
  - AL＝扇区数量
  - CH＝柱面号（C，0起）
  - DH＝磁头（H，0起）
  - CL＝扇区号（S，1起）

  CHS寻址
  - DL＝驱动器:软盘(00~7F),硬盘(80~FF)
  - ES:BX＝缓冲区的地址
- 出口参数
  - CF＝0：成功：AH＝00H，AL＝传输的扇区数
  - CF＝1：失败：AH＝状态代码

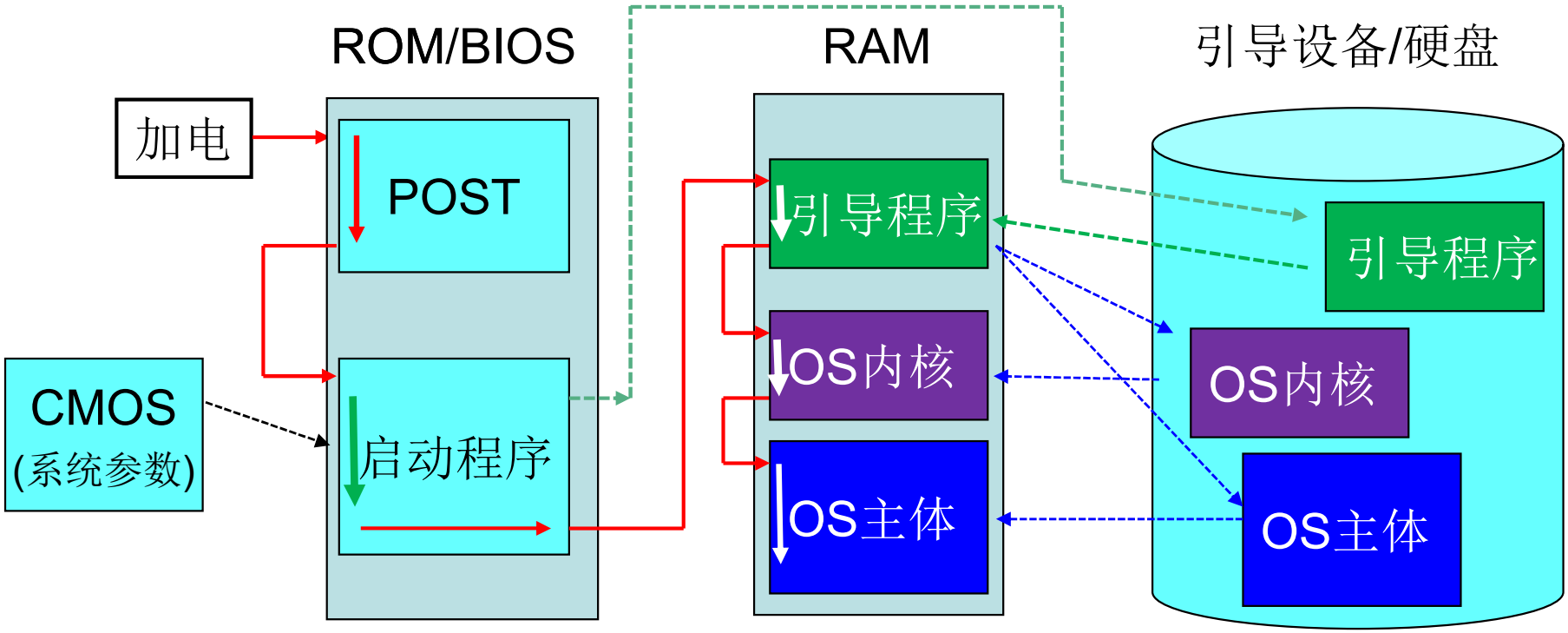
系统BIOS的作用

- 系统**BIOS**
  - 固件（firmware）
  - 地址：F0000- FFFFF
- 系统**BIOS**作用
  - （1）加电自检(POST)
    - 加电或复位开始执行
  - （2）CMOS设置
    - 交互式设置系统参数
  - （3）基本I/O服务
    - BIOS中断
  - （4）系统自举/加载OS



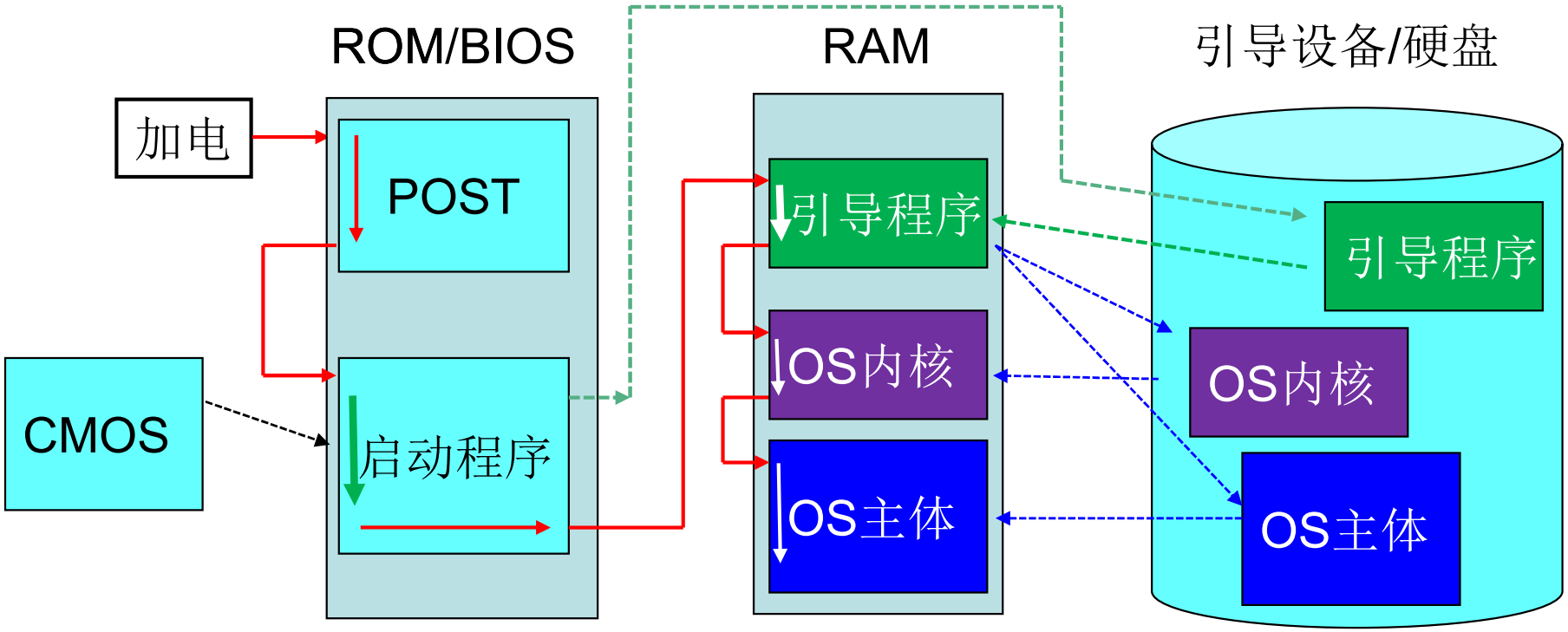
系统自举/加载OS

- 过程


ROM/BIOS
RAM
引导设备/硬盘
加电
POST
CMOS
(系统参数)
启动程序
引导程序
OS内核
OS主体
引导程序
OS内核
OS主体

系统自举/加载OS的两种方式

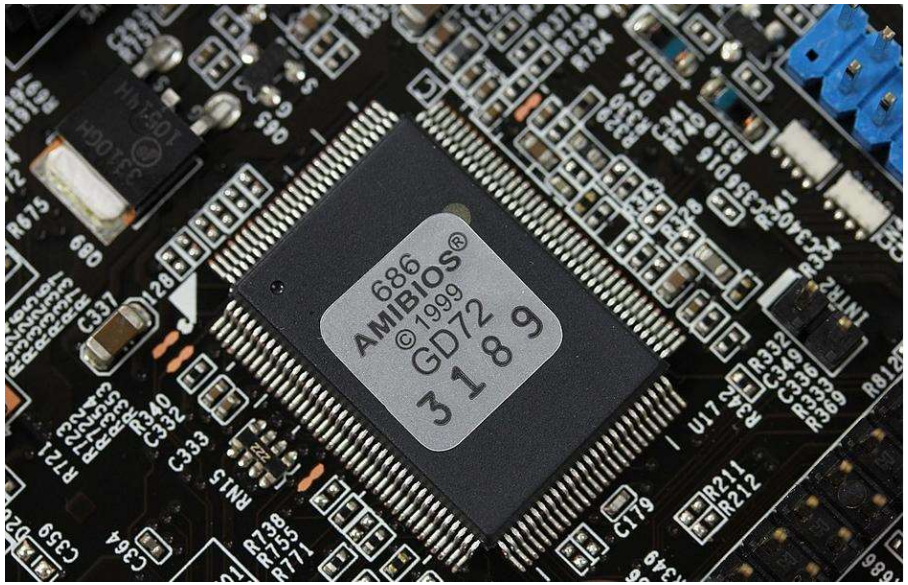
- 现场引导方式
  - OS文件存储在本地存储设备



- 下载引导方式
  - OS文件存储在网络存储设备中

总结：系统BIOS（BIOS）

- 系统**BIOS**作用
  - （1）加电自检(POST)
    - ◆加电或复位开始执行
  - （2）CMOS设置
    - ◆交互式设置系统参数
  - （3）基本I/O服务
    - ◆BIOS中断
  - （4）系统自举/加载OS

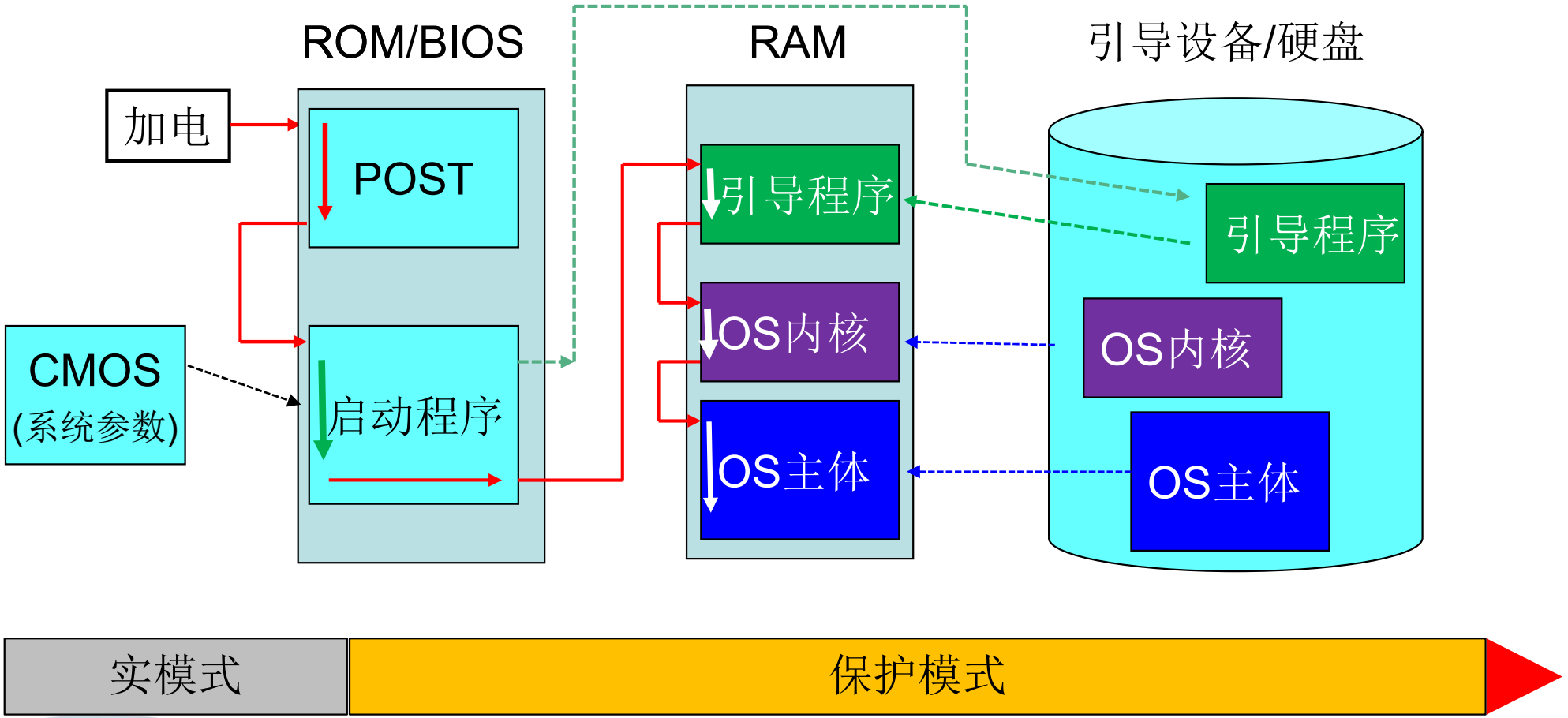


- 系统**BIOS**
  - 固件（firmware）：F0000- FFFFF
  - 三大厂家：AMI、AWARD和PHONIX

2.8 操作系统的启动

操作系统的启动

- 启动的全过程

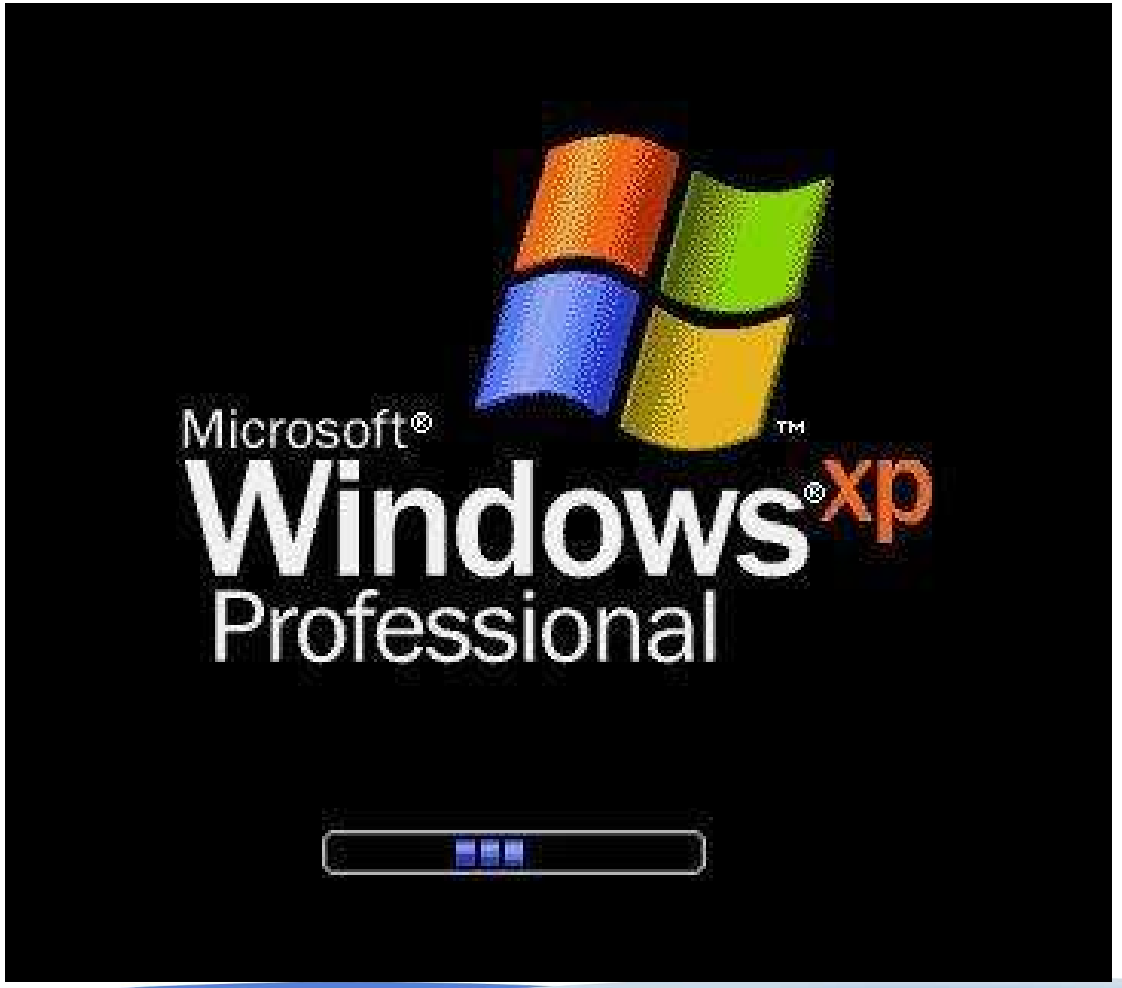


实模式和保护模式

- 实模式（实地址模式，**REAL MODE**）
  - 按照8086方法访问0~FFFFFFh地址空间（1MB）
  - 寻址方式：物理地址(20位)=段地址(16位):偏移地址(16位)
  - CPU单任务运行
- 实模式的**1M**空间
  - 前面640K【00000 -- 9FFFF】：基本内存
  - 中间128K【A0000 -- BFFFF】：显卡显存
  - 末尾256K【C0000 -- FFFFF】：**BIOS**
- 保护模式（内存保护模式，**Protect Mode** ）
  - 寻址方式：段(16位)和偏移量(32位)，寻址4GB
    - ◆段的含义有变化
  - CPU支持多任务

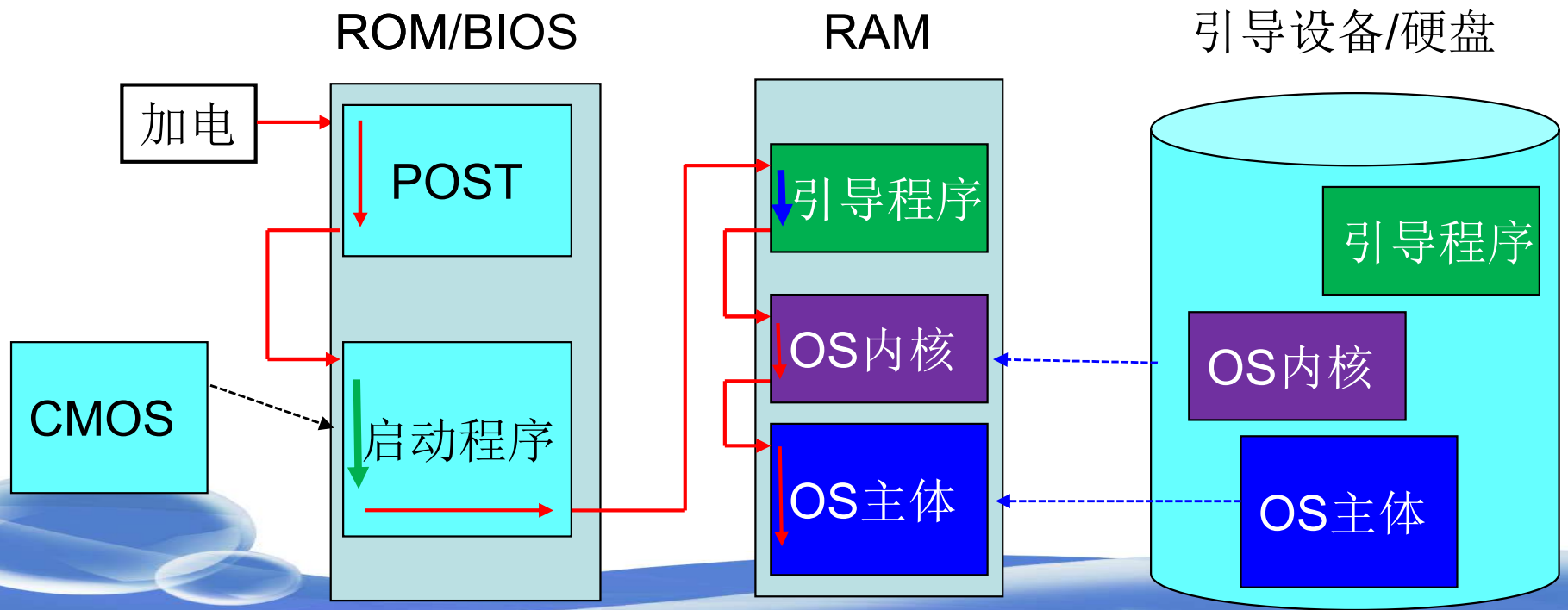
操作系统的启动

- 启动过程
  - 从加电到用户工作环境准备好的过程
    - **(1)初始引导**
    - (2)核心初始化
    - (3)系统初始化



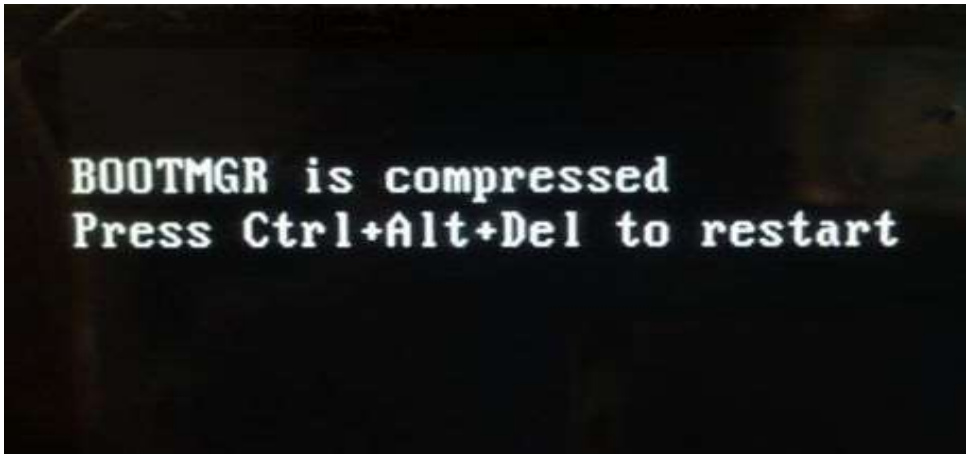
1)初始引导

- 目的
  - ■把OS内核装入内存并使之开始工作接管计算机系统
  - ■引导程序：MBR（Master BOOT Record）
    - ◆GRUB
    - ◆bootmgr|NTLDR|LILO



1)初始引导

- 引导程序缺失



2)核心初始化

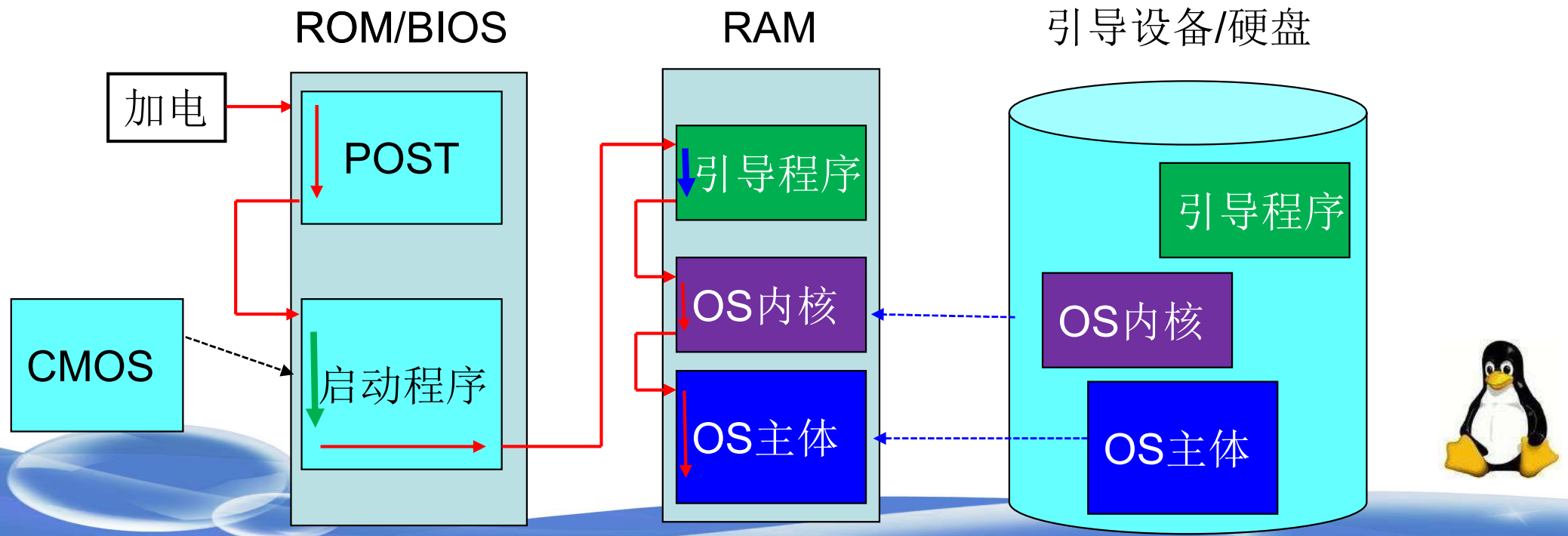
- 核心初始化
  - 目的：OS内核初始化系统的核心数据
  - 典型工作
    - 各种寄存器的初始化
    - 存储系统和页表初始化
    - 核心进程构建
    - ……

3)系统初始化

- 系统初始化
  - 为用户使用系统作准备，使系统处于待命状态。
  - 主要工作
    - 初始化文件系统
    - 初始化网络系统
    - 初始化控制台
    - 初始化图形界面
    - ……

Linux的启动过程

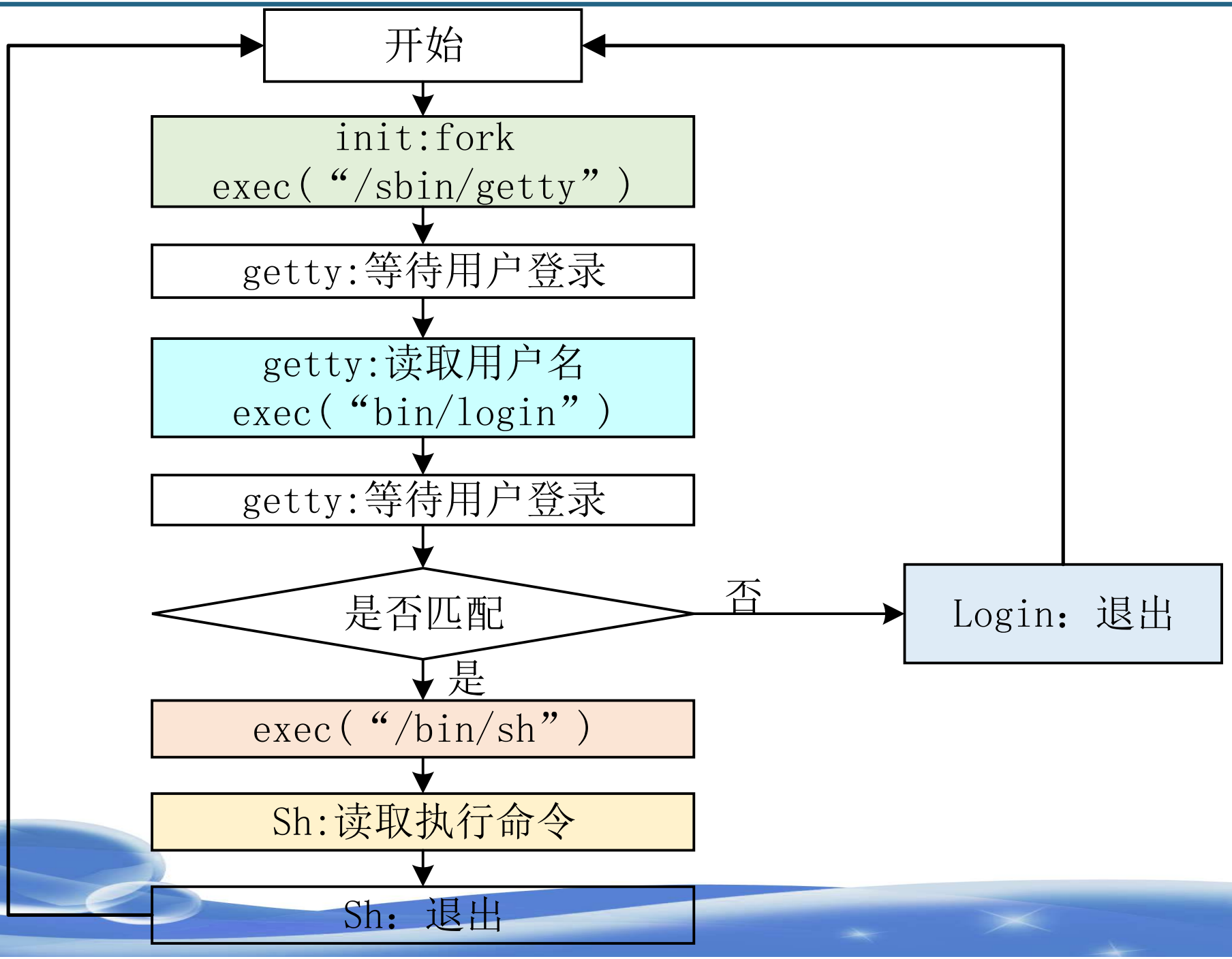
- **LINUX的启动过程**
  - POST → 启动程序→ MBR → KERNEL映像 →
  - KERNEL映像边自解压并边执行→
  - 内核初始化→ 内核启动 →
  - init进程 →



内核完成引导后，加载init程序

- **init进程是系统所有进程的祖先**
  - 进程号=1
  - init进程执行/etc/inittab脚本进行系统初始化
  - 执行/etc/rc.d/rc.sysinit
  - 执行/etc/rc.d/rcX.d/[KS]
  - 执行/etc/ec.d/rc.local
  - 执行 /bin/login 程序
    - 启动 /etc/profile 文件
    - 启动用户目录下bash_profile或bash_login或~/.profile
  - 设置键盘、字体、装载模块、设置网络等。

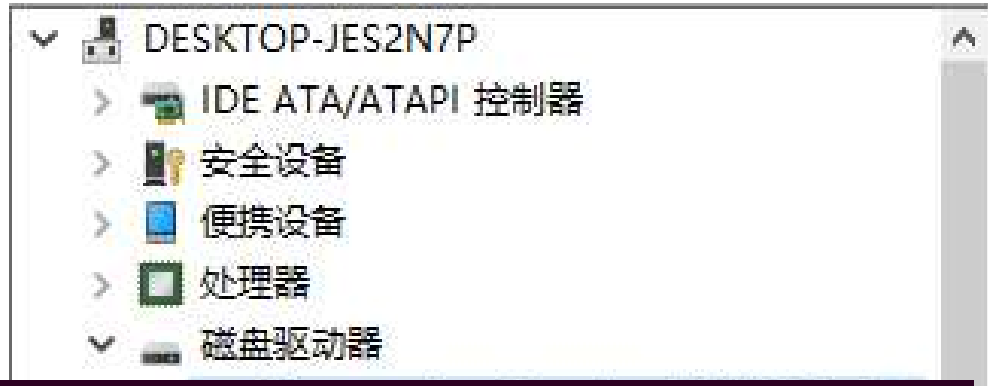
Linux的登录/shell



2.10 操作系统的生成

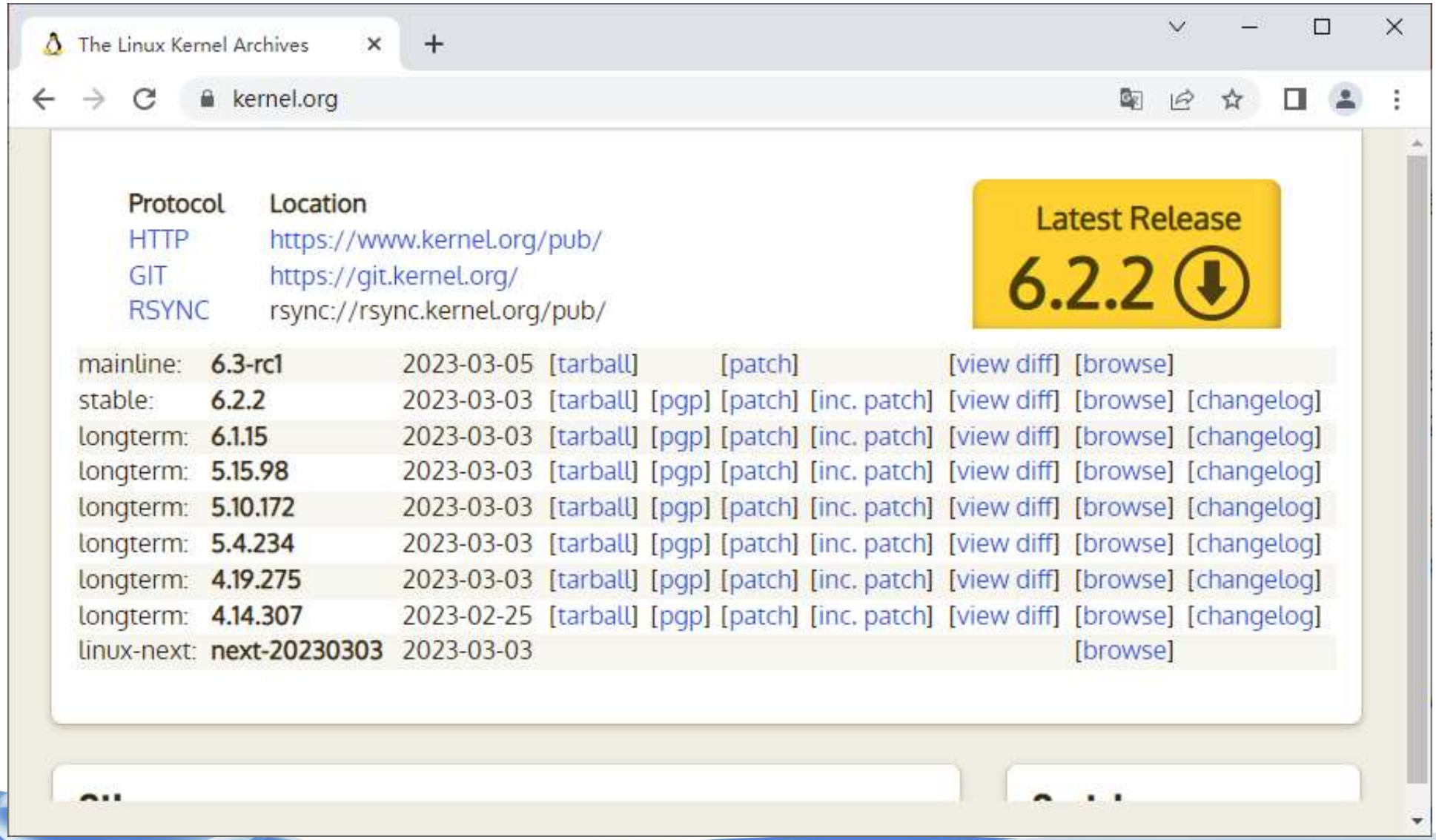
操作系统生成

- 操作系统生成
  - 根据硬件环境和用户需要，配置和构建操作系统。

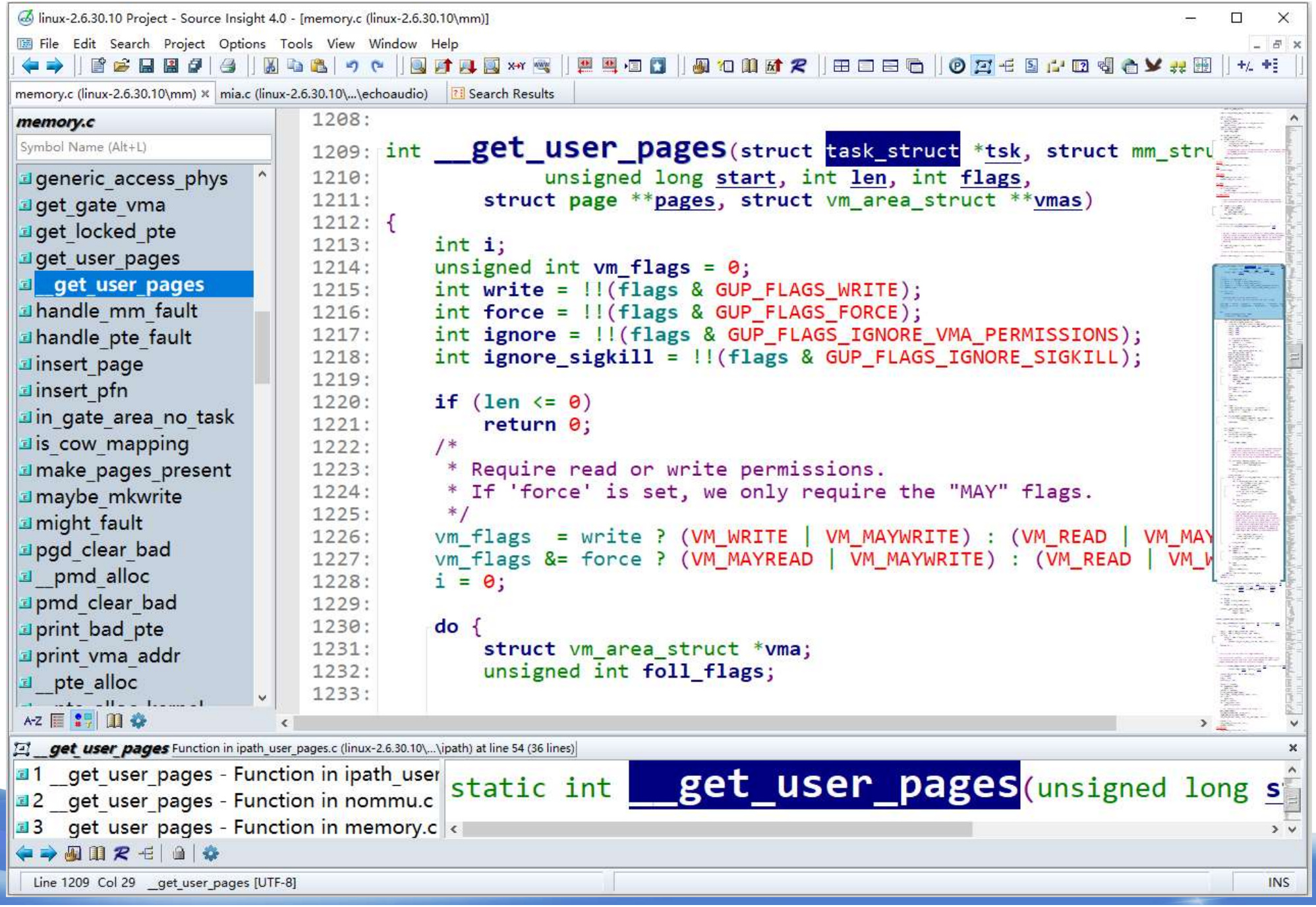


最新版本Linux内核 6.2.2

- **https://www.kernel.org/**



较早版本Linux内核 2.6.30



Linux操作系统的生成

- 步骤
  - 1、获取内核源码
  - 2、配置内核
  - 3、重新编译新内核
  - 4、编译和安装模块
  - 5、配置启动选项

Linux操作系统的生成

- 1、获取内核源码（差距不太远的版本）
  - http://www.kernel.org/

 # cd /usr/src

 # tar zxvf linux-2.6.38-12.tar.gz

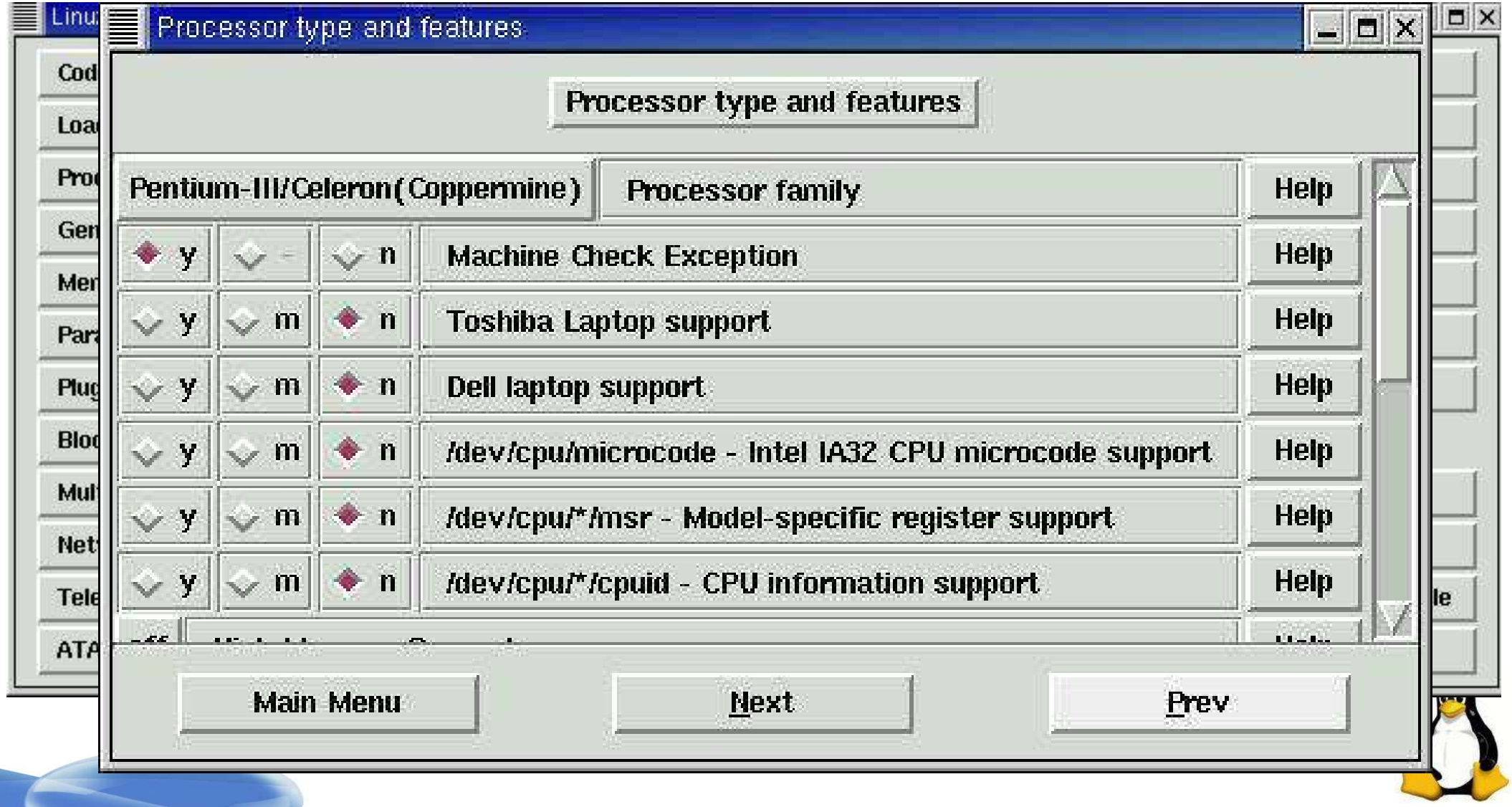
- 2、配置内核 （模块和参数）

#cd /usr/src/linux-2.6

#make xconfig（xWindow图形窗口模式）

make xconfig

Linux内核配置对话框



Linux操作系统的生成

- 1、获取内核源码（差距不太远的版本）
  - http://www.kernel.org/

 # cd /usr/src

 # tar zxvf linux-2.6.38-12.tar.gz

- 2、配置内核 （模块和参数）

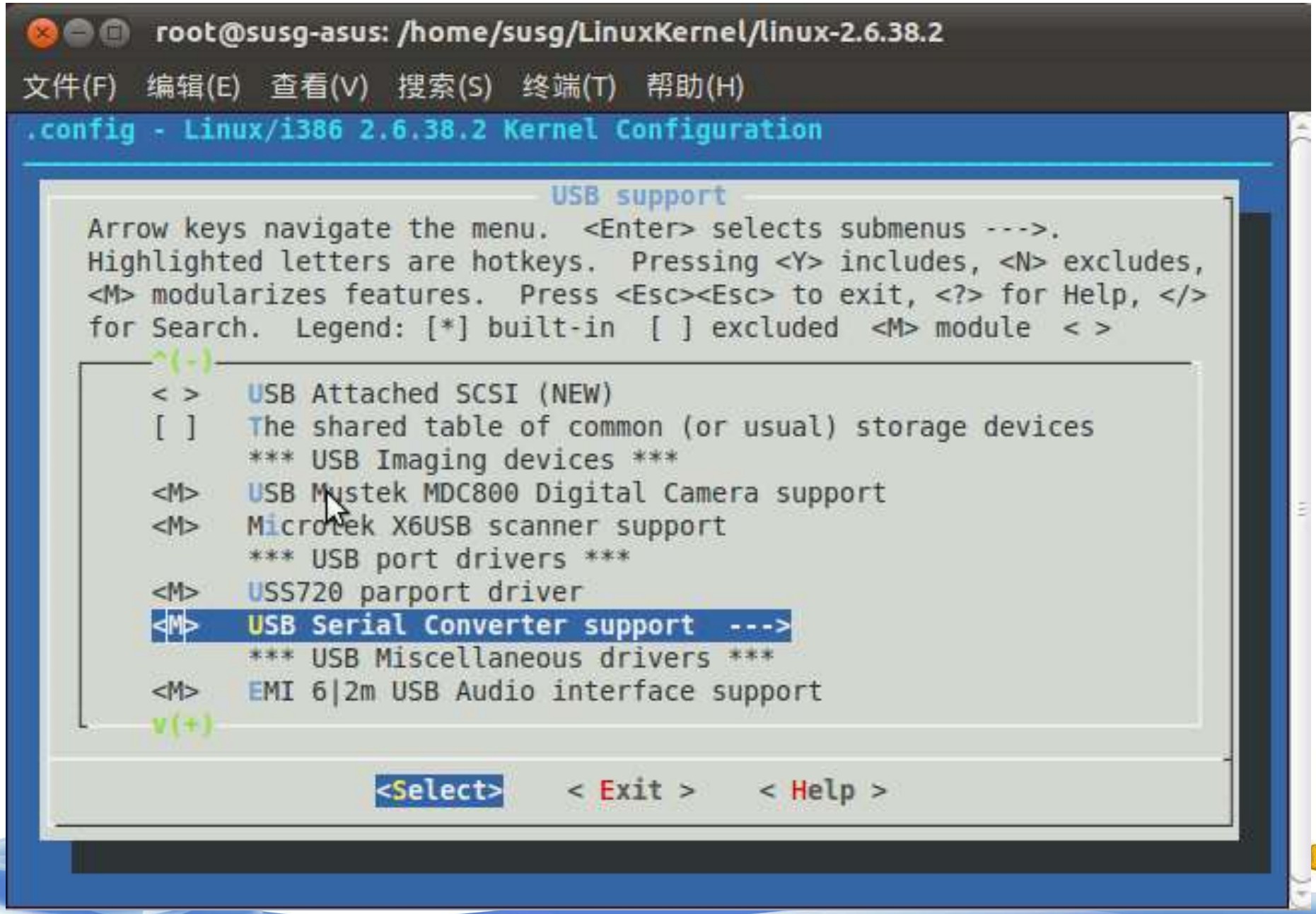
#cd /usr/src/linux-2.6

#make xconfig（xWindow图形窗口模式）

或：

#make menuconfig（文本选择界面，字符终端）

make menuconfig



模块和参数的配置：依据硬件配置和用户需要

- ① Loadable module support 设置是否支持模块机制
  - Enable loadable module support (y)
  - Set version info on all module symbols (n)
  - Kernel module loader (y)
- ② **Processor type and features** 设置**CPU**的类型
  - Processor family 选择CPU类型
  - High Memory Support (n)
  - Math emulation (n)
  - MTTR support： (n)
  - Symmetric multi-processing support (n)

模块和参数的配置：依据硬件配置和用户需要

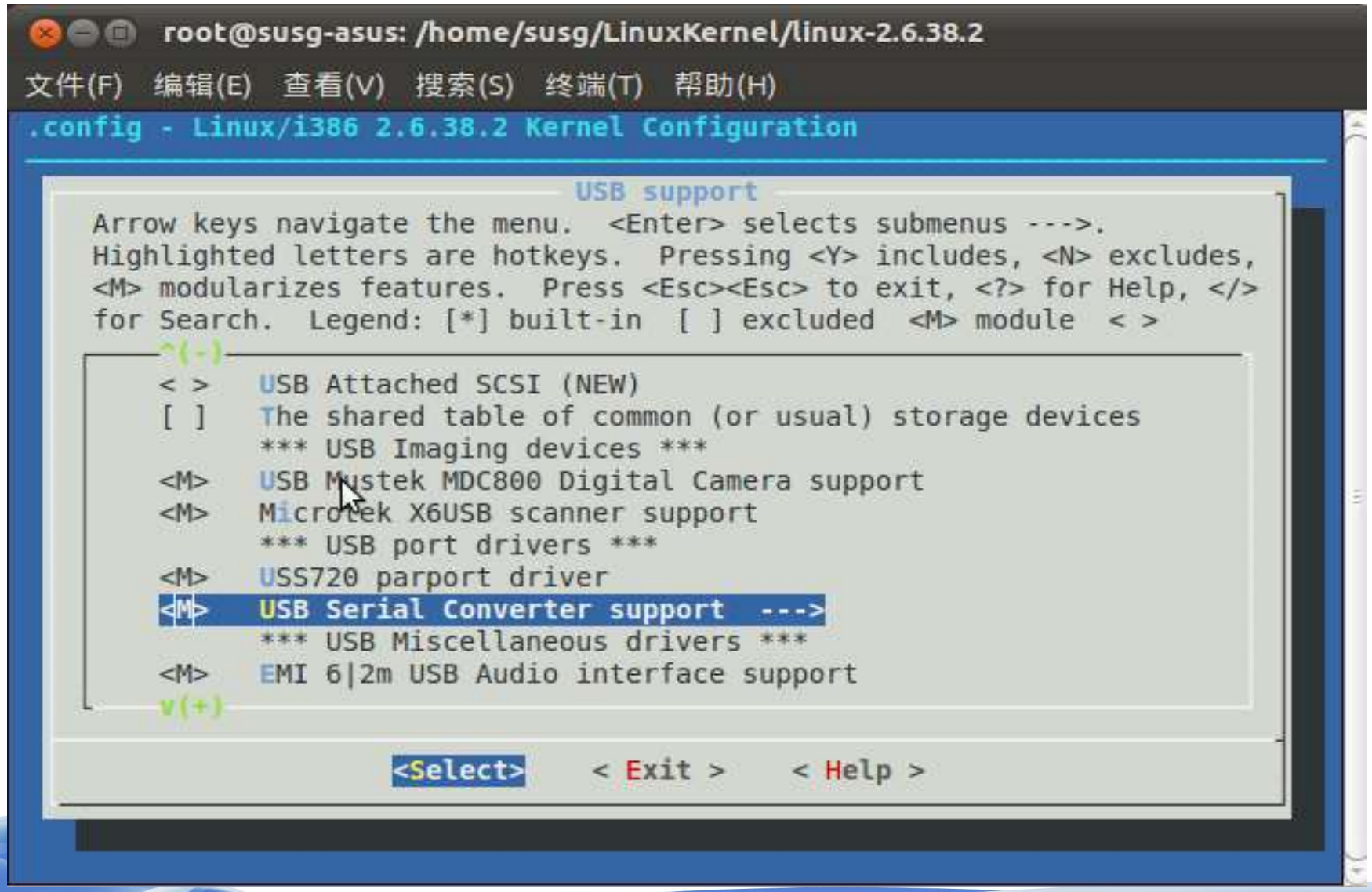
- ③ **General setup**　对一些普通属性进行设置。
  - Networking support：（y）
  - PCI support（y）
  - PCI access mode　PCI卡存取模式：BIOS、Direct和Any
  - Support for hot-pluggable devices　(n)
  - PCMCIA/CardBus support　(n)
- ④ **Parallel port support**　并口支持　（y）
- ⑤ **Plug and Play configuration**：即插即用配置（y）
- ⑥ **Block devices** 块设备支持的选项
  - Normal PC floppy disk support　软盘支持（y）
  - Network block device support　网络块设备支持（y）

模块和参数的配置：依据硬件配置和用户需要

- ⑦ **Networking options** 配置**TCP/IP networking**选项
- ⑧ **Network device support** 网络设备支持的选项
  - Ethernet（10 or 100Mbit） (y)
  - RealTeck RTL-8139 PCI Fast Ethernet Adapter support (y)
- ⑨ **Mice** 鼠标设置选项：串口、**PS/2**等类型鼠标
- ⑩ **File systems** 文件系统类型。
  - DOS FAT fs 选项：FAT16， FAT32
  - NTFS file system support
  - /proc file system support： （y）
- ⑪ **Sound** 声卡驱动，选项：声卡型号
- ⑫ **USB support** **USB**接口的支持，根据需要选择。

模块和参数的配置：依据硬件配置和用户需要

- **Kconfig**文件



Linux操作系统的生成

- 3、重新编译新的内核
  - 内核配置的结果：makefile

  # make dep　　生成依赖dependency信息

  # make clean　　清除旧的编译结果

  # make bzImage　. /arch/i386/boot/bzImage
- 4、编译和安装模块

  # make modules

  # make modules_install

  模块被编译且安装到 /usr/lib/<内核版本号> 目录下。

Linux操作系统的生成

- **5、配置启动选项（并启动新内核）**
  - cp bzImage /boot/bzImage
  - GRUB (与发行版本有关)
    - 配置/boot/grub/grub.conf

```
title newLinux build by Zhang San Feb.28, 2023
root (hd0,1)
kernel /boot/bzImage ro root=/dev/hda2
```

  - LILO(早期版本)
    - 配置/etc/lilo.conf

```
image=/boot/bzImage
label=newLinux build by Zhang San Feb.28, 2012
```

    - 命令 #lilo　　使配置生效

Linux操作系统的生成

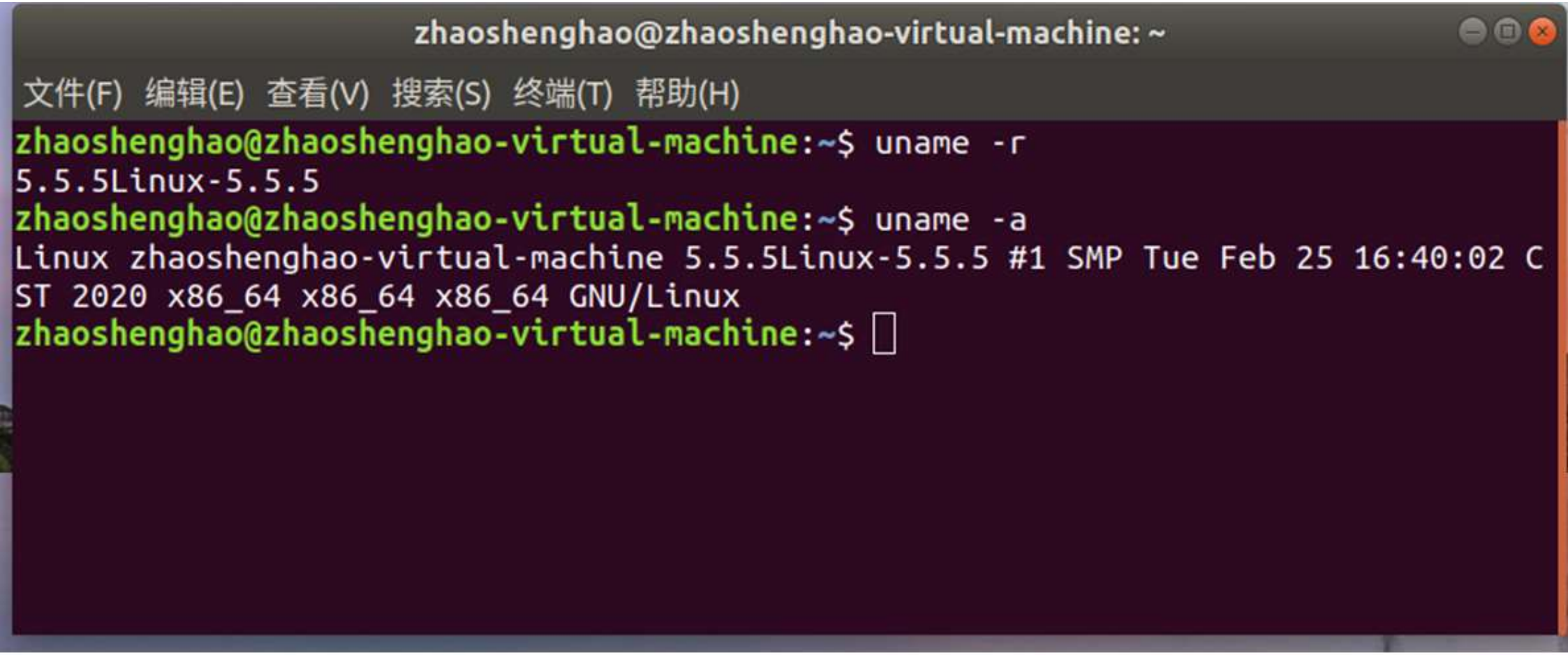
- 重启计算机
- 重启新内核
  - 选择 ubuntu高级选项

Linux操作系统的生成

- 重启计算机
- 重启新内核
  - 在 ubuntu高级选项 中出现下面的界面

Linux操作系统的生成

- 重启计算机
- 重启新内核



参考网址

- blog.csdn.net/qq_39819990/article/details/106605430
- **Ubuntu20.04编译内核**