

第 2 章 操作系统的硬件基础

月出皓兮 苏曙光老师的课堂笔记

2022 年 3 月 5 日

支持操作系统最基本的硬件结构为 CPU，内存，中断和时钟。

目录

| | |
|--|----------|
| 1 CPU 的态系统 | 4 |
| 1.1 目的 | 4 |
| 1.2 CPU 态的定义 | 4 |
| 1.3 态的分类 | 4 |
| 1.3.1 核态 (Kernel mode) | 4 |
| 1.3.2 用户态 (User mode, 目态) | 4 |
| 1.3.3 管态 (Supervisor mode) | 4 |
| 1.4 态的转换 | 4 |
| 1.4.1 用户态向核态转换 | 4 |
| 1.4.2 核态向用户态转换 | 5 |
| 1.5 硬件支持 | 5 |
| 1.6 Intel CPU 的态 | 5 |
| 1.6.1 PL(Privilege Level) | 5 |
| 1.6.2 DPL: 描述符特权级 (Descriptor Privilege Level) | 5 |
| 1.6.3 当前特权级 CPL: Current Privilege Level | 6 |
| 1.6.4 请求特权级 RPL: Requested Privilege Level | 6 |
| 2 中断机制 | 6 |
| 2.1 中断的定义 | 6 |
| 2.2 目的 | 6 |

| | | |
|----------|--|-----------|
| 2.3 | 相关概念 | 6 |
| 2.3.1 | 中断源 | 6 |
| 2.3.2 | 中断类型 | 6 |
| 2.3.3 | 断点 | 7 |
| 2.3.4 | 现场 | 7 |
| 2.3.5 | 中断服务程序 | 7 |
| 2.3.6 | 中断向量表 | 7 |
| 2.4 | 中断响应过程 | 7 |
| 2.5 | 中断嵌套 | 8 |
| 2.6 | 中断系统重要的指令 | 8 |
| 2.6.1 | INT n | 8 |
| 2.6.2 | IRET | 8 |
| 2.7 | 中断响应的实质 | 8 |
| 3 | 基本输入输出系统 BIOS | 9 |
| 3.1 | 位置 | 9 |
| 3.2 | 常见类型 | 9 |
| 3.3 | 基本功能 | 9 |
| 3.3.1 | 加电自检及初始化 (POST) | 9 |
| 3.3.2 | CMOS 设置 | 9 |
| 3.3.3 | 系统自举/加载 OS | 10 |
| 3.3.4 | 基本 I/O 服务 (BIOS 中断) | 10 |
| 4 | 操作系统是如何在计算机上启动的 ? | 10 |
| 4.1 | 概述 | 10 |
| 4.1.1 | 初始引导 | 10 |
| 4.1.2 | 核心初始化 | 11 |
| 4.1.3 | 系统初始化 | 11 |
| 4.2 | Linux 系统的启动过程 | 11 |
| 4.3 | 使用 GRUB 手工引导 | 12 |
| 4.4 | 主引导记录 (MBR:Main Boot Record) | 12 |
| 4.4.1 | MBR 的结构 | 12 |
| 4.4.2 | MBR 的作用 | 13 |
| 4.4.3 | MBR 的工作原理 | 13 |

| | |
|---------------------------------|----|
| 4.4.4 安装操作系统对 MBR 的影响 | 13 |
| 4.5 操作系统生成 | 13 |

1 CPU 的态系统

1.1 目的

为 CPU 设定态的根本目的在于为系统建立安全机制，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。

通过态来支持对**可信程序**和**非可信程序**的区分。可信程序能对系统的安全机制进行设置和修改，可以存取系统资源，使用**普通指令集**和**特权指令集**。非可信程序只能使用**普通指令集**。其中特权指令包括涉及外部设备的输入/输出指令，修改特殊寄存器的指令，改变机器状态的指令。

态也为计算机系统的资源访问设置了访问屏障。可信软件和不可信软件之间设置访问屏障。**CPU 设置工作模式（限定可用指令集）**。

1.2 CPU 态的定义

即 CPU 的工作状态，对资源和指令使用权限的描述。

硬件按“**态**”来区分 CPU 的状态，OS 按“**进程**”来区分 CPU 的状态。

1.3 态的分类

1.3.1 核态 (Kernel mode)

能够访问所有资源和执行所有指令，具有最高的特权级别，管理程序/OS 内核一般为核态程序。

1.3.2 用户态 (User mode, 目态)

仅能访问部分资源，其它资源受限。用户程序为用户态程序。

1.3.3 管态 (Supervisor mode)

介于核态和用户态之间

1.4 态的转换

1.4.1 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）。

- 外部设备的中断。
- 用户态企图执行特权指令。

可以总结说，从用户态向核态转变的唯一方法就是中断。

1.4.2 核态向用户态转换

一般是中断返回：IRET.

1.5 硬件支持

CPU 设置**模式位**，表明当前的状态。

指令执行前增加“**权限状态是否满足**”的条件判断。

Intel CPU 的相关机制

1. 保护模式 (CR0 的 PE 位,PG 位)
2. CPL (当前特权级)
3. 地址映射机制
4. 权限核验 (DPL+RPL+CPL)

1.6 Intel CPU 的态

1.6.1 PL(Privilege Level)

Ring 0 – Ring 3 (Ring 0 最核心, Ring 3 最外层)

Unix/Linux/Windows OS: 仅用到了 Ring 0 和 Ring 3

1.6.2 DPL: 描述符特权级 (Descriptor Privilege Level)

段描述符 Descriptor, 8 个字节, 其中包括段基址, 段界限, 段属性 (段类型, 访问该段所需最小特权级, 是否在内存)

1.6.3 当前特权级 CPL: Current Privilege Level

1.6.4 请求特权级 RPL: Requested Privilege Level

2 中断机制

2.1 中断的定义

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。

2.2 目的

实现并发活动

实现实时处理

故障自动处理

2.3 相关概念

2.3.1 中断源

引起系统中断的事件称为中断源

2.3.2 中断类型

强迫中断和自愿中断

- 强迫中断：程序没有预期。例：I/O、外部中断
- 自愿中断：程序有预期。例：执行访管指令

外中断（中断）和内中断（俘获）

- 外中断：由 CPU 外部事件引起。例：**I/O**，外部事件。
- 内中断：由 CPU 内部事件引起。例：访管中断、程序中断

外中断又可分为不可屏蔽中断和可屏蔽中断。

- 不可屏蔽中断：中断的原因很紧要，CPU 必须响应
- 可屏蔽中断：中断原因不很紧要，CPU 可以不响应

2.3.3 断点

程序中中断的地方，将要执行的下一指令的地址
包含 CS:IP (FLAGS、SS、SP)

2.3.4 现场

程序正确运行所依赖的信息集合。

包含 **PSW (程序状态字)**、相关寄存器、断点

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

2.3.5 中断服务程序

处理中断源中断事件的程序称为中断服务程序。中断服务程序是事先已准备好的一个特殊函数，该函数的调用由系统自动完成。

2.3.6 中断向量表

中断服务程序的入口地址用段基址 (Segment) 和段偏移 (Offset) 两个参数记录，称之为中断向量。

2.4 中断响应过程

1. 识别中断源
2. 保护断点
3. 保护现场
4. 进入中断服务程序
5. 恢复现场
6. 中断返回

其中现场入栈和中断服务在内容可能会有重叠。

编写中断服务程序的关键：保护现场，实现特定功能，还原现场，IRET。

在保护模式下发生中断的时候可能涉及特权级的变化。中断发生时，如果代码在相同特权级间跳转时，则堆栈不变（则只需入栈 FLAGS, CS, IP）；若在不同特权级间跳转，则会用到两个不同的堆栈（需要压入 SS, SP, FLAGS, CS, IP）。压栈的过程地址由低到高（SP 不断变小）。

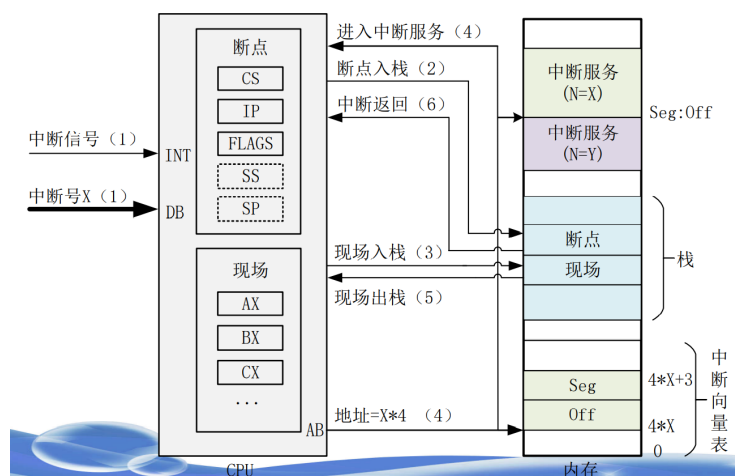


图 1: 中断响应过程

2.5 中断嵌套

而堆栈机制也是中断嵌套处理的关键所在，允许我们从一个低优先级的中断处理中跳出，去处理一个高优先级的中断。

2.6 中断系统重要的指令

2.6.1 INT n

压栈 (SS, SP, FLAGS, CS, IP)，按照中断向量表跳转。

2.6.2 IRET

可通过 IRET 指令进入特定的代码段与任务，其常用于由高优先级（内核态）到低内核态的跳转。

2.7 中断响应的实质

交换指令执行地址。

交换 CPU 的态。

实现了现场保护和恢复与参数传递（通信）。

3 基本输入输出系统 BIOS

Basic I/O System (BIOS, Firmware, 固件, 以硬件形式存在的软件)。

3.1 位置

实地址模式, 内存中的 C0000-FFFFF 部分。其中:

C0000 – C7FFF: 显示卡 BIOS

C8000 – CBFFF: IDE 控制器 BIOS

F0000 – FFFFF: **系统 BIOS**。大小为 64K, 是每一个计算机必须的。

3.2 常见类型

Award BIOS, AMI BIOS, Phoenix BIOS。

3.3 基本功能

3.3.1 加电自检及初始化 (POST)

- 按下 PowerOn 键

CS:IP 复位 **FFFF:0000**, 开始执行 **FFFF0** 单元的指令 **JUMP POST**

- Power On Self-Test (POST, 加电自检)

初始化基本硬件, 如 CPU, 内存, 显卡...

自检正常不提示, 错误则通过喇叭提示。

POST 之后, 根据用户指定顺序从硬盘或光驱等媒介启动 OS。OS 启动后, 由 OS 接管计算机。

3.3.2 CMOS 设置

Complementary Metal-Oxide Semiconductor 互补金属氧化物半导体芯片。芯片保存着系统的重要参数, 要随时保持供电。

CMOS RAM 只是一块存储芯片, 只有数据保存功能, 因而需要使用专门的 CMOS 设置程序去设置其中的各项参数。在计算机加电引导过程中, 可以通过特殊热键启动 CMOS 设置程序, 参数设置好后保存在 CMOS RAM 中。

3.3.3 系统自举/加载 OS

开机时将 OS 载入内存并运行为用户**建立用户环境**。

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。

3.3.4 基本 I/O 服务 (BIOS 中断)

BIOS 使用的中断类型号为 10H——1FH。每一组服务又根据具体功能或使用方式细分为不同的子功能，并用子功能编号加以标识。在汇编程序中, 通过 **AH 寄存器**指定子功能编号以调用不同的子功能。**INT 13H 磁盘直接服务**, AH=02H 读扇区

其中需要了解磁盘的 CHS(Cylinder/Head/Sector) 定位方案。磁头数 (Heads) 表示硬盘总共有几个磁头，也就是有几面盘片，最大为 256 (用 8 个二进制位存储)；柱面数 (Cylinders) 表示硬盘每一面盘片上有几条磁道，最大为 1024(用 10 个二进制位存储)；扇区数 (Sectors per track) 表示每一条磁道上有几个扇区，最大为 63 (用 6 个二进制位存储)。

4 操作系统是如何在计算机上启动的？

4.1 概述

操作系统的启动过程是指从加电开始到操作系统的控制台或桌面准备好的过程。整个启动过程分为 3 个阶段：**初始引导**、**核心初始化**和**系统初始化**。

4.1.1 初始引导

其目的是把 **OS 内核**装入内存并使之开始工作接管计算机系统。

过程如下：

1. 加电，主板发出 RESET 信号，CPU 恢复初始状态
2. 电源稳定后，撤去 RESET 信号，CPU 执行 FFFF0H 处指令（这个地址实际上位于 RAM-BIOS）
3. 进行加电自检 (POST)
4. 跳入 **BIOS 的启动程序 (2 进制程序)**，读取 0 面 0 道第 1 扇区内容，即主启动记录 (Master Boot Record, MBR)。其中包含特定操作系统

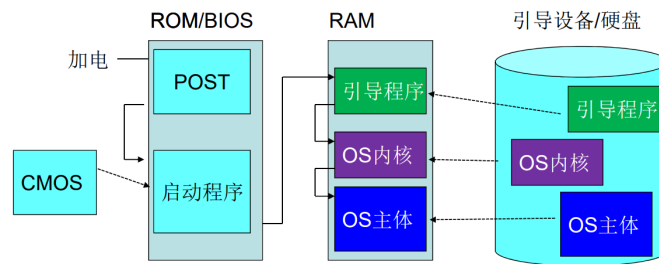


图 2: OS 的初始引导

的引导程序。(BIOS 从 MBR 读入引导程序，装入内存的特定位置。)

5. 运行**引导程序 (2 进制程序)**，根据相关参数，读取硬盘指定位置的**OS 内核 (以文件形式存在)**到内存，并初始化相关参数。

6. OS 内核获得 CPU 控制权，逐步加载 OS 剩余部分，直到最后完全控制计算机。

4.1.2 核心初始化

核心初始化其目的是 OS 内核初始化系统的核心数据。

典型工作有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……

4.1.3 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**。

主要工作有初始化文件系统，初始化网络系统，初始化控制台，初始化图形界面等。

4.2 Linux 系统的启动过程

- POST
- MBR
- KERNEL 映像
- KERNEL 映像自解压并执行
- 内核初始化
- 内核启动

| | LINUX | GRUB |
|-----------|-------|-------|
| 第一块硬盘 | hda | hd0 |
| 第一块硬盘分区 1 | hda1 | hd0,0 |
| 第一块硬盘分区 2 | hda2 | hd0,1 |
| 第一块硬盘分区 3 | hda3 | hd0,2 |
| 第二块硬盘 | hdb | hd1 |
| 第三块硬盘 | hdc | hd2 |
| 第四块硬盘 | hdd | hd3 |

图 3: Linux 和 GRUB 的硬盘分区

4.3 使用 GRUB 手工引导

GRUB(Grand Unified Boot Loader) 是一款强大的多重开机引导器, 是一个独立于操作系统之外的引导管理程序。

4.4 主引导记录 (MBR:Main Boot Record)

MBR 为和 OS 启动相关的数据和代码, 其存放在主启动扇区 (Main Boot Sector) 大小为 512 BYTES, 若其结束 AA55H 则表明这个设备可以用于启动。

4.4.1 MBR 的结构

MBR 的结构包括:

1. 引导程序 (Boot Loader) 446B
2. 该磁盘的分区表 (Partition Table) 64B
3. 主引导签名, 55AA 表示 MBR 结束。

其中分区表的作用记录硬盘分区信息。每个分区的第一个扇区叫作分区引导记录 (Partition Boot Record, PBR), 也叫次引导记录, 其结构与 MBR 类似。

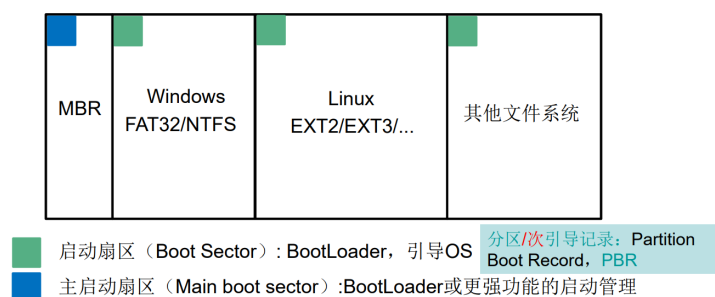


图 4: 被分为多个分区的硬盘中 MBR 和 PBR 的例子

4.4.2 MBR 的作用

提供 BootLoader 或启动管理

1. 直接指向引导代码加载 OS
2. 提供启动选项菜单
3. 跳转到 PBR 以加载特定 OS BootLoader

4.4.3 MBR 的工作原理

MBR 读取**分区表 (Partition Table)**, 查找并确认**唯一活动分区 (Active Partition)**, MBR 读取活动分区 PBR, 并加载到内存。

PBR 控制后续引导过程, 读 OS 内核。

4.4.4 安装操作系统对 MBR 的影响

安装操作系统时, 除了要把操作系统映像复制到计算机外存储空间中, 还要向 MBR 和 PBR 中写入启动相关的数据和代码。

通常对 MBR 有重写和追加两种方式, 这两种方式的区别会在安装多操作系统时体现出来。

4.5 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。

操作系统生成的前提: 操作系统由可拆装模块构成, 有交互式配置工具, 有映像构建 (build) 工具。

Linux 操作系统的生成

- 1、获取 Linux 内核的源代码（最好是当前版本）
- 2、选择和启动内核配置程序
- 3、根据需要配置内核模块和参数
- 4、重新编译新的内核
- 5、编译和安装模块
- 6、启动新内核