回顾：

1.linux内核mmap机制

目的：将硬件外设的物理地址映射到用户空间的虚拟地址上

内核的sys_mmap所做的内容：

1.帮你在MMAP内存映射区找一块空闲的虚拟内存区域,用来映射物理地址

2.一旦找到,内核用struct vm_area_struct数据结构来描述这块空闲的虚拟内存区域,其对象由内核创建

3.内核sys_mmap最终调用底层驱动的mmap接口,并且将描述空闲的虚拟内存区域的对象的首地址传递给驱动的mmap接口

底层驱动的mmap仅作只做一件事：映射

```
struct file_operations {
 int (*mmap)(struct file *file, struct vm_area_struct *vma)
};//vma指针指向的对象由内核创建,用来描述空闲的虚拟内存区域
```

切记：映射时,地址必须是页面大小的整数倍！

2.linux内核platform机制

实现linux内核的分离思想

实现将硬件和软件彻底分开

一旦软件写好,无需在进行改动

将来只需要关注硬件差异即可,将来硬件有所变动,只需改动硬件部分

画图

总结：

1.probe函数是否被调用至关重要！

2.驱动开发者利用platform机制实现设备驱动,只关注两个数据结构：

```
  struct platform_device
      .name 相当重要
      .id
      .resource
              .start
              .end
              .flags
                IORESOURCE_MEM
                IORESOURCE_IRQ
      .num_resources
  struct platform_driver
      .driver = {
              .name 相当重要
      },
      .probe 匹配成功内核调用,形参指向匹配成功的硬件信息
      .remove 删除软件或者硬件节点内核调用
```

3.什么遍历,什么匹配,什么调用,什么传参都是由内核完成！

4.probe函数一般所做的工作：

0.调用代表一个完整的硬件设备驱动产生

1.通过形参获取硬件信息

```
  struct resource *platform_get_resource(
            struct platform_device *pdev,
            int flags;
            int index;
  );
```

函数功能：通过probe函数的形参pdev或者resource描述硬件信息

pdev:指向匹配成功的硬件信息

flags：硬件信息的类型

IORESOURCE_MEM/IORESOURCE_IRQ

index:同类资源的偏移量
返回值：返回获取到的硬件信息的首地址

2.处理获取到的硬件信息
该申请的申请
该注册的注册
该映射的映射
该初始化的初始化
3.注册硬件操作接口(字符设备驱动或者混杂设备驱动)
.open
.release
.read
.write
.unlocked_ioctl
.mmap

remove跟probe对着干！

****
3.I2C总线(IIC总线)
面试题：谈谈对I2C总线理解
3.1.I2C总线的功能
计算机CPU和外设的通信方式很多种：
GPIO,例如LED,按键
总线(地址线数据线),例如内存,NorFlash,DM9000
UART,例如BT,GPS
I2C,例如重力传感器,触摸屏芯片
一线式,例如DS18B20温度传感器
SPI,例如NorFlash
等
I2C总线是CPU和外设通信的一种数据传输方式

3.2.I2C总线的定义
两线式串行总线
解释：
“两线式”：CPU跟外设的数据通信只需2根信号线,分别是SCL时钟控制信号线
和SDA数据线,画简要的连接图
SDA数据线：用来传输数据,CPU和外设都可以控制,但是不能同时
控制,例如CPU向外设写数据,数据线由CPU控制
CPU从外设读取数据,数据线由外设控制
问：由于外设的处理速度远远慢于CPU,CPU和外设如何
保证数据传输的正常呢？
答：关键靠SCL时钟线
注意：I2C总线数据传输从数据的高位开始！

SCL时钟线：同步双方的数据传输,保证数据传输正常，只能由CPU控制
例如：CPU在SCL为高电平时,将数据放到数据线上
设备就在同周期的SCL为低电平时,从数据线上读取数据
SCL为低电平时,数据线上的数据保持稳定不变！

“串行”：CPU和外设的数据通信是一个时钟周期传输1个bit位
“总线”：CPU和外设通信的两根信号线上可以挂接多个外设
画出简要的连接图
注意：SDA和SCL都会连接一个上拉电阻,默认电平都为高电平！

问：CPU如何找到总线上要访问的某个外设？

问：如果CPU找到这个外设,CPU如何通过两根信号线和外设通信呢？
问：SDA和SCL如何搭配使用呢？
答：答案在I2C总线传输协议中

3.3.I2C总线协议相关概念
CPU=主设备=master
外设=slave
MSB:高位
LSB:低位

START信号：又称起始信号,每当CPU要访问总线上某个外设,首先CPU向总线
发送一个START信号,此信号只能由CPU发起
SCL为高电平,SDA由高向低跳变产生START信号
类似“同学们,上课了”
STOP信号：又称结束信号,每当CPU要结束对某个外设的访问,CPU
只需向总线发送一个STOP信号即可,此信号同样只能由CPU发起
SCL为高电平,SDA由低向高跳变产生STOP信号
类似“同学们,下课”
画图START和STOP时序图
读写位：如果CPU要读设备,读写位=1;如果CPU要写设备,读写位=0
设备地址：同一个I2C总线上的外设都有一个唯一的设备地址(类似身份证号)
表示外设在总线上的唯一性
如果将来CPU要访问总线上某个外设,CPU只需向总线
发送这个外设对应的设备地址即可
类似“某某同学,请回答问题”
切记：设备地址不包含读写位

读设备地址=设备地址<<1|1
写设备地址=设备地址<<1|0

问：外设的设备地址如何确定？
答：
以LM77温度传感器为例P8：
确定LM77设备地址=10010A1A0(A1A0都接地)=》
1001000(高位补0)=01001000=0x48
=>
LM77的：
读设备地址=0x48<<1 | 1
写设备地址=0x48<<1 | 0

以AT24C02存储器为例P11:
确定AT24C02设备地址=1010A2A1A0(A2A1A0都接地)=》
1010000(高位补0)=01010000=0x50
读设备地址=0x50<<1 | 1
写设备地址=0x50<<1 | 0

以ADP8860背光灯控制芯片为例：
0101010x=>
设备地址=0101010(去x,高位补0)=00101010=0x2A
读设备地址=0x2A<<1 | 1
写设备地址=0x2A<<1 | 0

ACK信号:又称应答信号,表示CPU和外设的通信状态
低电平有效
类似“老师，我在”

总结：
1.CPU要想访问总线上某个外设,CPU只需向总线发送这个外设的
  设备地址即可
  I2C总线数据传输一周期一bit,一次一字节

问：CPU一旦通过设备地址找到某个外设,如何通过两根信号线
    和外设进行数据通信呢？
答：答案都在外设的芯片手册中,重点关注其中的操作时序图
以CPU读取LM77温度传感器2字节数据为例P12：
1.CPU向总线发送START信号
2.CPU向总线发送外设的设备地址包括读写位
3.如果外设存在于总线上,外设在第九个时钟周期会给CPU发送
  一个ACK信号,低电平有效
4.设备向CPU发送两字节数据的高字节
5.CPU读取数据以后,CPU同样在第九个时钟周期给外设一个有效的
  ACK信号
6.设备继续向CPU发送两字节数据的低字节
7.CPU读取数据以后,没有在第九个时钟周期给外设一个有效的
  ACK信号
8.数据读取完毕,CPU向总线发送STOP信号结束此次的数据读取操作
边说边画图(框框圈圈)

以AT24C02存储器的读写为例P11，字节写时序图：
AT24C02存储器特性：
      存储容量256字节
      内部地址编址：0x00~0xff
用户需求：将数据'A'写入到AT24C02的片内的0x10地址存储空间中
          硬件操作流程：
1.CPU向总线发送START信号
2.CPU向总线发送设备地址包括读写位
3.如果设备存在于总线上,设备在第九个时钟周期给CPU发送一个
  有效的ACK信号
4.CPU向设备发送要访问的片内地址0x10
5.设备接收到了CPU要访问的片内地址,设备同样在第九个时钟
  周期给CPU一个有效的ACK信号
6.CPU最后向设备发送要写入的数据'A'
7.设备将数据写入到片内0x10地址以后,设备同样在第九个时钟
  周期给CPU一个有效的ACK信号
8.CPU向总线发送STOP结束数据的操作
边说边画框框圈圈图

再以CPU读取AT24C02任意片内地址存储空间的数据为例P2随机读试图：
具体参见时序图

最后以CPU让HMC6352指南针传感器进入休眠模式为例：
1.CPU发送START信号
2.CPU发送设备地址<<1|0
3.设备如果存在于总线上,设备在第九个时钟周期给CPU发送一个
  有效的ACK信号
4.CPU向设备发送命令'S'=0x53,让设备进入休眠模式
5.设备接收到命令以后，设备在第九个时钟周期给CPU发送一个
  有效的ACK信号
6.最后CPU向总线发送STOP信号结束访问
边说边画圈圈框框图

切记：任何外设的访问必须严格按照时序图进行！！！

问：SDA和SCL如何搭配使用呢？
答：以CPU向设备写入数据为例：
CPU应该在SCL为低电平的时候将数据放到数据线上
那么设备就应该在SCL同周期的高电平从数据线上读取数据
"低放高取"
一定要画出相应的时序图,以CPU向HMC6352发送‘S’为例P4：
可以仅画START时序,写0,1时序即可

会画波形，将来也会分析示波器的波形图！！！！