1.了解Linux的中断处理流程，解释为什么引入上半部和下半部处理？

因为中断会打乱内核中进程的正常调度和运行，而且往往会屏蔽其他中断，所以必须使中断处理程序尽量短。但现实中，很多中断处理程序都很长。Linux内核为了在缩短中断处理时间和处理大量中断之间寻求平衡，将中断处理程序分成了两部分：上半部和下半部。Linux把中断程序中比较紧急的工作放在上半部分。这些工作会在接收到中断信号后立即执行，而且处理的时候会关中断，因此要追求速度，只把必需的工作放在这里，否则就可能不能及时处理其他中断。而对于那些比较耗时且对时间不那么敏感的工作，就把它们放到下半部分。这样做可以尽可能缩短屏蔽中断的时间，同时兼顾大量中断信号，提高系统运行的效率。

2. 以缺页异常为例，调研ARM和x86的中断异常机制，简述并比较两者的处理流程。

x86处理器在处理缺页异常时，首先确定其对应的中断号，然后通过中断描述符表查找对应的中断描述符，在全局描述符表中查找中断处理程序所在的段基址。然后要进行特权级检查，保证当前特权级的数字不大于所需的特权级。如果发生了特权级的变化，则需要切换堆栈。然后需要保存当前环境的上下文，找到缺页异常处理程序的入口，执行相应的指令。在处理完后，会恢复现场回到引发缺页异常的指令，再次执行。

ARM处理异常时，大致步骤也是查询中断向量表，保存现场，处理缺页异常，然后恢复现场重新执行指令。但是ARM没有特权级的检查。ARM异常向量表只有8项，相对较少。以及，ARM会把原来的程序中寄存器数据保存在相应异常模式的栈中。ARM没有像x86一样给每种中断分配表项，而是在异常模式下读取中断控制器的寄存器，根据寄存器中信息确定发生了哪一种中断。

3. 总结中断、异常和系统调用

中断：为了支持CPU和设备之间的并行操作引入中断。硬件接收中断信号，然后讲处理器控制权交给特定的处理程序。CPU执行完一条指令后，会检查中断码，如果没有中断则继续执行。如有中断，则先保存现场，然后根据中断向量表找到中断处理程序的入口，执行中断处理程序，响应完中断请求后，会恢复现场，返回到下一条指令继续执行。CPU在启动设备之后，便可以独立工作，直到设备工作完再给CPU发送中断信号让CPU决定之后的工作。利用中断，CPU还可以及时处理设备的信号，比起轮询的方式更加节省时间和资源。

异常：一般内部的中断被称为异常，即CPU执行指令时出现的问题。常见的异常有溢出、除零（运算时发生的异常）、缺页异常以及保护性异常等。利用异常还可以执行系统调用。异常处理完后可能返回当前指令，返回下一条指令，也可能终止当前程序，没有返回。

系统调用：系统调用是异常的一种，是操作系统给用户的接口，可以从用户态进入内核态。系统会选择一条特殊的陷入指令，通过该指令可以引发异常，进入内核态，执行中断处理程序。然后系统根据系统调用号和系统调用表找到相应程序的入口，根据参数执行相应的内核函数。不同的系统，其系统调用中断号不同。有些系统调用会经过包装，变成C函数库和API接口。系统调用执行的功能一般都是比较重要的，比如进程控制、文件管理、设备管理等。这些功能如果不用系统调用来实现，可能会影响操作系统的正常工作。

4.系统调用与函数/过程调用的区别是什么？

系统调用会进入到系统内核执行内核函数，需要通过中断异常机制执行，会在用户空间和内核上下文环境间切换，开销较大。而普通的函数调用一般是在用户态下执行的，调用开销较小。

系统调用的参数传递方式很多，可以由陷入指令自带参数、通过寄存器传递或者在内存中开辟专用堆栈传递。而普通的函数调用传递参数主要是用通用寄存器，如果寄存器个数不够则在栈中保存参数。

系统调用要先根据中断向量表找到中断处理程序并执行，然后根据系统调用码找到相应的处理函数。而函数调用可以直接找到函数的位置执行指令。

不同的系统会提供不一样的系统调用，而代码相同的函数调用在各个系统上实现的功能是一样的。

系统调用在内核地址空间执行，运行时间属于系统时间，而函数调用在用户地址空间执行，运行时间属于用户时间。