2. (1) ①进程的上下文切换，②（内部）异常，③（外部）中断；

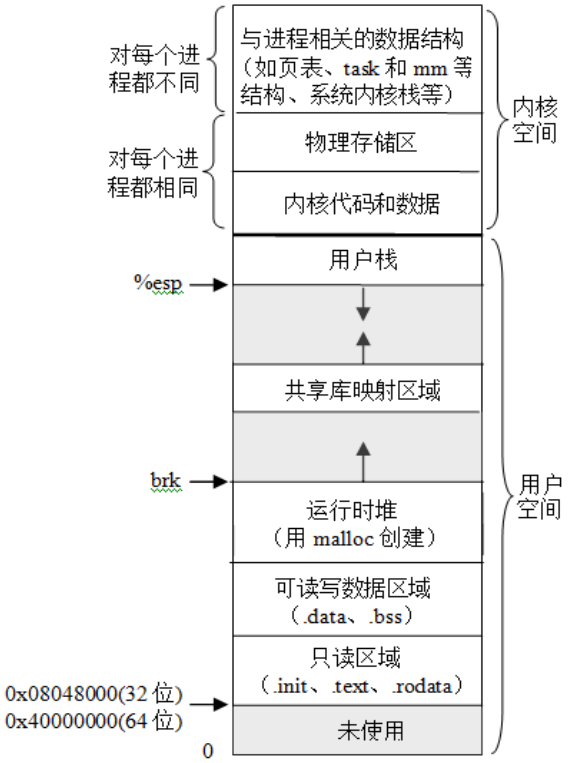
(2) 程序是代码和数据的集合，是一种静态的概念，进程是程序的一次运行过程，即一个具有一定独立功能的程序关于某个数据集合的一次运行活动，具有动态的含义；

(3) 提供的两个假象是把一台计算机的所有资源看成由应用程序所独占，且应用程序是在处理器上执行和在存储空间中存放的唯一的用户程序。好处是简化了程序员的编程以及语言处理系统的处理，即简化了编程、编译、链接、共享和加载等整个过程。

(4) 不同进程的虚拟地址空间是独立的，有些进程的逻辑控制流在时间上有交错，而在空间上不会交错。进程被打断后能从"断点"处继续执行，由进程的上下文切换机制实现。

(5) 操作系统主要完成的工作有：①将当前处理器的寄存器上下文保存到当前进程的系统级上下文的现场信息中；②将新进程系统级上下文中的现场信息作为新的寄存器上下文恢复到处理器的各个寄存器中；③将控制转译到新进程执行；

(6)



(7) 保护断点和程序状态、关中断、识别异常和中断事件并转到相应处理程序执行；

(8) 当 CPU 处于单步跟踪状态时，每条指令都被设置成了陷阱指令，执行每条指令后，都会发生中断类型号为 1 的"调试"异常，从而转去执行"单步跟踪处理程序"；

(9) 在当前栈中保存 EFLAGS、CS 和 EIP 寄存器的内容（断点和程序状态）；

(10) 通用寄存器内容（现场信息），硬件出错码（如果有的话）和异常类型号；

(11) 从编程者角度看来，两者在形式上没有区别；

普通函数使用 CALL 指令实现过程调用，系统调用使用陷阱指令实现；

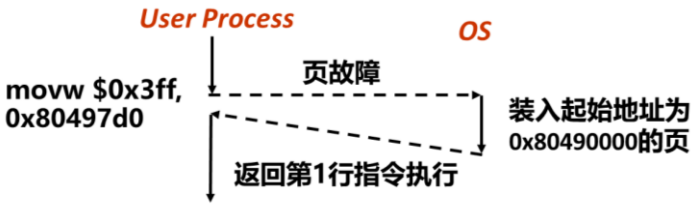
普通函数处理器在用户态下执行，执行的指令和访问的存储空间都受限，系统调用处理器转到内核态下运行，CPU 此时可以执行特权指令并访问内核空间；

普通函数调用所用的参数用栈来传递，系统调用所用的参数通过寄存器传递；

(12) 中断向量表用于实地址模式下，记录每个异常或中断的编号，和与其对应的异常处理程序或中断服务程序入口地址，中断描述符表用于保护模式下，获取异常处理或中断服务程序入口地址，每一个表项是一个中断门描述符、陷阱门描述符或任务门描述符。

4. (1) Linux 初始化时段基址均设为 0，即虚拟（逻辑）地址就是线性地址，第一行指令对应的虚拟地址为 0x80482c0，又页的大小为 4KB，第一行指令的地址不是它的整数倍，因此该指令不位于某个页的起始处，在执行到第一行之前的其他指令时，上述 7 条指令被全部装入主存，对它们的执行不会在取指令时发生缺页异常；

(2) 执行第 1 行：可恢复的缺页异常。对 b[1000]（地址为 0x80497d0）的访问是对该地址所在页面（起始地址为 0x8049000）的第一次访问，对应页面还未装入主存。此时 CPU 将控制转移到操作系统内核，调用页故障对应的处理程序，检查此处不存在地址越界或访问越权的情况，则将起始地址为 0x8049000 的页面从磁盘装入主存，故障处理完毕，控制交还给当前程序并从第一行指令重新开始执行，此时数据可以正常访问；



执行第 2 行：可恢复的缺页异常，同(1)；

执行第 6 行：数据可以正常访问，经过(2)中的页故障相关处理，地址 0x804a324 所在的页面已经装载到主存中，此时访问数据不会发生缺页异常；

执行第 7 行：可能发生不可恢复的页故障。a[10000]（实际并不存在）的地址与数组首址的偏移量很大，可能已经超出可读写数据区的范围，在转到操作系统内核执行页故障处理程序时，发生地址越界或访问越权，发送“段错误”信号给用户进程，用户进程则调用信号处理额程序，在屏幕上显示段错误信息，并终止用户进程；

(3) 不可恢复的“除零”故障。此处 k 是未初始化的变量，位于.bss 节中，初始值为 0，且在程序中没有人为赋值，则执行第 5 行指令中 div 时会出现“除零”故障，不可恢复。

5. (1) 执行该段代码时，由于代码为用户程序代码，则系统处于用户态，执行完第 5 行指令后的下一个时钟周期，软中断指令进行了系统调用，系统进入内核态；

(2) 是。通过系统门描述符来激活异常处理程序，对应的中断类型号是 128。则对应的门描述符 P=1，DPL=3，TYPE=1111B，由于系统调用执行的是内核代码，则对应门描述符中的段选择符取出的 GDT 中的段描述符为内核代码段对应的段描述符，其基地址为 0，限界为 FFFFFH，G=1，S=1，TYPE=1010B，DPL=0，D=1，P=1；

(3) ①根据中断类型号 0x80，从 IDTR 指向的 IDT 中取出第 128 个表项，该表项为系统门描述符，P=1，DPL=3，TYPE=1111B，段选择符为 0x60，指向内核代码段描述符；

②根据 IDT 中的段选择符，从 GDTR 指向的 GDT 中取出相应的段描述符，得到对应异常处理程序或中断服程序所在段的基址、DPL 等信息。比较当前特权级 CPL 与段描述符中的 DPL，若 CPL 小于 DPL 则出现 13 号异常（在 Linux 内核中不会发生这种情况）。

检查是否发生特权级变化，此处 CPL=3，DPL=0，从用户态切换到内核态。

用内核栈保存信息，即读 TR 寄存器访问正在运行进程的 TSS 段，将 TSS 段中保存的内

核栈的段选择符和栈指针分别装入 SS 和 ESP，在内核栈中保存原来的用户栈 SS 和 ESP；

③将第 5 行后一条指令的起始地址写入 CS 和 EIP，保证内核处理完之后能够回到下条指令继续执行。在当前内核栈中保存 EFLAGS、CS 和 EIP 寄存器的内容；

④将 IDT 中的段选择符（0x60）装入 CS，将 IDT 中的偏移地址装入 EIP，它指向内核代码段中的系统调用处理程序的第一条指令，则从下一个时钟周期开始转为执行系统调用处理程序，在内核执行完毕后通过 iret 返回，回到第 5 行指令的下一条指令继续执行。