# Core dump

首先解释什么是core dump，当一个进程要异常终止时，可以选择把进程的用户空间内存数据全部保存到磁盘上，文件名通常是core，这叫做core dump-核心转储。进程异常终止时因为有bug，比如非法访问内存导致段错误。事后可以用调试器检查core文件以查清错误原因，这叫做事后调试。

一个进程允许产生多大的core文件，取决于进程的resource limit。默认是不允许产生core文件的。在开发调试阶段可以用ulimit命令改变这个限制，允许产生core文件。

基于以上，总结了几点：

1. 核心转储只能用于开发调试阶段，如果在程序允许期间或者是项目上线后，仍然可以核心转储，会带来许多问题。由于core dump是将用户数据全部写入硬盘，可能一次错误产生的数据就会将硬盘写满。其次，上线了的项目如果能够把所有数据通过core dump写入硬盘，显然是不安全的，尤其是在包含密码等敏感信息时。所以core dump只能在开发调试阶段使用。
2. 核心转储默认是不被开启的。同样基于以上原因，核心转储默认不开启。
3. 造成core dump的原因
4. 内存访问越界
5. 由于使用错误下标，导致数组访问越界
6. 搜索字符串时，依靠字符串结束符来判断字符串是否结束，但是字符串没有正常使用结束符
7. 使用strcpy，strcat，strintf，strcmp，strcasecmp等字符串操作函数，将目标字符串读/写爆
8. 多线程程序使用了线程不安全的函数
9. 多线程读写的数据未加锁保护，对于会被多线程同时访问的全局数据，应该注意加锁保护，否则很容易造成core dump
10. 非法指针
11. 使用空指针
12. 随意使用指针转换。一个指向一段内存的指针，除非确定这段内存原先就分配为某种结构或类型，或者这种结构或类型的数组，否则不要将它转换为这种结构或类型的指针，而应该将这段内存拷贝到一个这种结构或类型种，再访问。这是因为如果这段内存的开始地址不是按照这种结构或类型对齐的，那么访问它时就很容易因为bus error而core dump。
13. 堆栈溢出，不要使用大的局部变量（因为局部变量都分配在栈上），这样容易造成堆栈溢出，破坏系统的栈和堆结构，导致出现莫名其妙的错误
14. core dump如何定位

通常情况下，core文件会包含了程序运行时的内存，寄存器状态，堆栈指针，内存管理信息还有各种函数调用堆栈信息等，我们可以理解为是程序工作当前状态存储生成第一个文件，许多的程序出错时都会产生一个core文件，通过工具分析这个文件，我们可以定位到程序异常退出时对应的堆栈调用等信息，找出问题所在并进行及时解决。