open/read/write和fopen/fread/fwrite的区别

open：系统调用，返回的是文件描述符，即文件句柄，是文件在文件描述副表里的索引。
fopen：C语言库函数，返回的是一个指向文件结构的指针。fopen是ANSI C标准中的C语言库函数，在不同的操作系统中应该调用不同的内核API，UNIX环境下，fopen是对open的封装。
文件描述符是UNIX/Linux下的一个概念，Linux环境下，一切设备皆是文件，一切设备皆是以文件的形式进行操作，如网络套接字、硬件设备等。有关文件描述符和文件指针的区别可以参见博文：《文件描述符和文件指针的区别》。
设备文件不可以当成流式文件来处理，因此，只能使用open，而fopen只是用来操纵正规文件的，并且设置有缓冲，跟open还是有区别的。

open和fopen的区别在于：

1.缓冲文件系统
缓冲文件系统是借助于文件结构体指针FILE *来对文件进行管理，通过文件指针对文件进行访问，即可以读写字符、字符串、格式化数据，也可以读写二进制数据。
缓 冲文件系统特点：在内存中开辟一个“缓冲区”，为程序里每一个文件使用，当执行读文件操作时，从磁盘文件将数据先读入内存“缓冲区”，装满后再从内存“缓 冲区”依次读入接收的变量。执行写文件操作时，也是先将数据写入内存“缓冲区”，待内存“缓冲区”装满后再写入文件。由此可以看出，内存“缓冲区”的大 小，影响着实际操作外在的次数，内存“缓冲区”越大，则操作外存的次数就越少，执行速度就越快，效率就越高。一般来说，文件“缓冲区”的大小跟机器是相关 的。
缓冲文件系统的IO函数主要包括：fopen, fclose, fread, fwrite, fgetc, fgets, fputc, fputs, freopen, fseek, ftell, rewind等。

2.非缓冲文件系统
非缓冲文件系统依赖于操作系统，通过操作系统的功能对文件进行读写，是系统级的输入输出，它不设文件结构体指针，只能读写二进制文件（对于UNIX系统内核而言，文本文件和二进制代码文件并无区别），但效率高、速度快，由于ANSI标准不再包括非缓冲文件系统，因此，在读取正规的文件时，建议大家最好不要选择它。
非缓冲文件系统的IO函数主要包括：open, close, read, write, getc, getchar, putc, putchar等。

举个例子来说明open系列函数与fopen系列函数的效率问题：如果文件的大小是8k。
你如果用read/write，且只分配了2K的缓存，则要将此文件读出需要做4次系统调用来实际从磁盘上读出。如果你用fread/fwrite，则系统自动分配缓存，则读出此文件只要一次系统调用从磁盘上读出。也就是用read/write要读4次磁盘，而用fread/fwrite则只要读1次磁盘。效率比read/write要高4倍。如果程序对内存有限制，则用read/write比较好。都用fread 和fwrite，它自动分配缓存，速度会很快，比自己来做要简单。如果要处理一些特殊的文件，用read 和write，如套接口，管道之类的设备文件。
系统调用write的效率取决于你buffer的大小和你要写入的总数量，如果buffer太小，你进入内核空间的次数大增，效率就低下。而fwrite会替你做缓存，减少了实际出现的系统调用，所以效率比较高。
如果只调用一次（这种可能性比较小），这俩差不多，严格来说write要快一点点，因为实际上fwrite封装了write，最后还是用write做真正的写入文件系统工作，但是这其中的差别无所谓。

open和fopen最主要的区别在于fopen在用户态下就有了缓存，在进行read和write时，减少了用户态和内核态的切换，而open则每次都需要进行内核态和用户态的切换，其表现为：如果顺序访问文件，fopen系统的函数要比直接调用open系统函数快，如果随机访问文件，open系列函数要比fopen系列函数快。

因此，open系列函数与fopen系列的区别可以简单概括为：

open系列函数	fopen系列函数
一般用于打开设备文件（少数情况）	一般用于打开普通文件（大多数情况）
利用文件描述符操纵文件	利用文件指针操作文件
open返回一个文件描述符	fopen返回一个文件指针
POSIX系统调用	ANSI C库函数
低层次IO	高层次IO，对open的扩展和封装
只能在POSIX操作系统上移植	可移植到任何操作系统
非缓冲IO	缓冲IO
只能读取二进制或普通文本	可以读取一个结构
可以指定要创建文件的访问权限	不能指定要创建文件的访问权限

fread返回的是一个FILE结构指针
而read返回的是一个int的文件号

前者fopen/fread的实现是靠调用底层的open/read来实现的.

fopen/fread
是C标准的库函数，操作的对象是： file stream

open/read
是和操作系统有关的系统调用。操作的对象是：“file descriptor”

f是ANSI的C标准库。后面的是UNIX下的系统调用。

带f的带有缓冲，是后面的衍生，
直接和硬件打交道，必须是后面的！

UNIX环境下的C 对二进制流文件的读写有两套班子：1) fopen,fread,fwrite ; 2) open, read, write
这里简单的介绍一下他们的区别。
1. fopen 系列是标准的C库函数；open系列是 POSIX 定义的，是UNIX系统里的system call。
也就是说，fopen系列更具有可移植性；而open系列只能用在 POSIX 的操作系统上。
2. 使用fopen 系列函数时要定义一个指代文件的对象，被称为“文件句柄”（ file handler ），是一个结构体；而open系列使用的是一个被称为“文件描述符” （ file descriptor ）的int型整数。
3. fopen 系列是级别较高的I/O，读写时使用缓冲；而open系列相对低层，更接近操作系统，读写时没有缓冲。由于能更多地与操作系统打交道，open系列可以访问更改一些fopen系列无法访问的信息，如查看文件的读写权限。这些额外的功能通常因系统而异。
4. 使用fopen系列函数需要"#include <sdtio.h>"；使用open系列函数需要"#include <fcntl.h>" ，链接时要之用libc（ -lc ）
小结：
总的来说，为了使程序获得更好的可移植性，未到非得使用一些fopen系列无法实现的功能的情况下，fopen系列是首选。

read/write和fread/fwrite区别

1.fread是带缓冲的,read不带缓冲.
eg:
如果文件的大小是8k。你如果用read/write，且只分配了2k的缓存，则要将此文件读出需要做4次系统调用来实际从磁盘上读出。
如果你用fread/fwrite，则系统自动分配缓存，则读出此文件只要一次系统调用从磁盘上读出。
也就是用read/write要读4次磁盘，而用fread/fwrite则只要读1次磁盘。效率比read/write要高4倍。

如果程序对内存有限制，则用read/write比较好
2.fopen是标准c里定义的,open是POSIX中定义的.
3.fread可以读一个结构. read在linux/unix中读二进制与普通文件没有区别.
4.fopen不能指定要创建文件的权限,open可以指定权限.
5.fopen返回指针,open返回文件描述符(整数).
6.linux/unix中任何设备都是文件,都可以用open,read.

都用fread 和fwrite,它自动分配缓存,速度会很快,比自己来做要简单。**如果要处理一些特殊的描述符,用read 和write,如套接口,管道之类的**
系统调用write的效率取决于你buf的大小和你要写入的总数量，如果buf太小，你进入内核空间的次数大增，效率就低下。而fwrite会替你做缓存，减少了实际出现的系统调用，所以效率比较高。
如果只调用一次(可能吗?)，这俩差不多，严格来说write要快一点点(因为实际上fwrite最后还是用了write做真正的写入文件系统工作)，但是这其中的差别无所谓。