<https://blog.csdn.net/hhhlizhao/article/details/71552588>

1、write()

函数定义：ssize\_t write (int fd, const void \* buf, size\_t count);

函数说明：write()会把参数buf所指的内存写入count个字节到参数放到所指的文件内。

返回值：如果顺利write()会返回实际写入的字节数。当有错误发生时则返回-1，错误代码存入errno中。

附加说明：

（1）write()函数返回值一般无0，只有当如下情况发生时才会返回0：write(fp, p1+len, (strlen(p1)-len)中第三参数为0，此时write()什么也不做，只返回0。man手册给出的write()返回值的说明如下：

（2）write()函数从buf写数据到fd中时，若buf中数据无法一次性读完，那么第二次读buf中数据时，其读位置指针（也就是第二个参数buf）不会自动移动，需要程序员编程控制，

而不是简单的将buf首地址填入第二参数即可。如可按如下格式实现读位置移动：write(fp, p1+len, (strlen(p1)-len)。 这样write第二次循环时变会从p1+len处写数据到fp, 之后的也

由此类推，直至(strlen(p1)-len变为0。

（3）在write一次可以写的最大数据范围内（貌似是BUFSIZ ,8192），第三参数count大小最好为buf中数据的大小，以免出现错误。(经过笔者再次试验，write一次能够写入的并不只有8192这么多，笔者尝试一次写入81920000，结果也是可以，看来其一次最大写入数据并不是8192，但内核中确实有BUFSIZ这个参数，具体指什么还有待研究)

以下通过一个例子具体说明write函数用法

**[objc]** [view plain](https://blog.csdn.net/hhhlizhao/article/details/71552588) [copy](https://blog.csdn.net/hhhlizhao/article/details/71552588)

1. #include <string.h>
2. #include <stdio.h>
3. #include <fcntl.h>
4. **int** main()
5. {
6. **charchar** \*p1 = "This is a c test code";
7. **volatile** **int** len = 0;
9. **int** fp = open("/home/test.txt", O\_RDWR|O\_CREAT);
10. **for**(;;)
11. {
12. **int** n;
14. **if**((n=write(fp, p1+len, (strlen(p1)-len)))== 0)   //if((n=write(fp, p1+len, 3)) == 0)
15. {                                                 //strlen(p1) = 21
16. printf("n = %d \n", n);
17. **break**;
18. }
19. len+=n;
20. }
21. **return** 0;
22. }

此程序中的字符串"This is a c test code"有21个字符，经笔者亲自试验，若write时每次写3个字节，虽然可以将p1中数据写到fp中，但文件test.txt中会带有很多乱码。唯一正确的做法还是将第三参数设为(strlen(p1) - len，这样当write到p1末尾时(strlen(p1) - len将会变为0，此时符合附加说明（1）中所说情况，write返回0， write结束。 

2、read()

函数定义：ssize\_t read(int fd, void \* buf, size\_t count);

函数说明：read()会把参数fd所指的文件传送count 个字节到buf 指针所指的内存中。

返回值：返回值为实际读取到的字节数, 如果返回0, 表示已到达文件尾或是无可读取的数据。若参数count 为0, 则read()不会有作用并返回0。

注意：read时fd中的数据如果小于要读取的数据，就会引起阻塞。