3.4,

三个函数将0, 1, 2指向的文件表项赋为fd指向的文件表项，因此，这三个函数起到文件重定向的作用，使对标准输入输出出错的操作定向到fd打开的文件。

如果fd <= 2，也就是对三个标准的操作定向到其中一个的操作。

如果fd > 2，也就是对三个标准的操作定向到fd打开的文件，此时就有4个文件描述符指向fd打开的文件。

为什么要关闭fd呢？书上给的答案是：“这种情况下就需要关闭描述符3”。但没有说明为什么。

如果不关闭fd会有什么问题吗？个人认为，已经有了三个标准的操作，此时就不需要fd了，而且如果fd再操作的话容易引起混乱，纯属个人见解。

知识点:int dup2(int filedes, int filedes2); 将filedes2对应的打开文件置为filedes对应的打开文件，之后就可以使用filedes2操作filedes打开的文件。

由于要复制一个文件描述符，而且不能使用fcntl()，那么，唯一能使用的就是dup()了：

int dup(int filedes); 复制一个当前可用的最小的文件描述符，使它对应的打开文件是filedes对应的打开文件。

这里采用的主要思想是：dup()返回的是当前可用的最小的文件描述符，那么就可以使用遍历，从dup(filedes)开始遍历，知道返回的文件描述符等于filedes2。

下面介绍主要流程：

当filedes2是无效文件描述符时，输出错误信息，那么什么样的是无效文件描述符呢？就是小于0或者大于进程可以打开的最大文件数时，文件描述符是无效的。

然后就可以分为三种情况：

filedes == filedes2: 直接返回filedes2

filedes > filedes2: 关闭filedes2，再进行dup(filedes)返回值应该就是filedes

filedes < filedes2: 从dup(filedes)开始遍历，直到返回值等于filedes2，然后关闭filedes2，再dup(filedes)返回值应该就是filedes

3.6

#include <stdio.h>

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

int main(void)

{

int fd = -1;

int ret = -1;

int size = -1;

char read\_buf[1024] = {0};

char write\_buf[32] = "hello world";

/\* lseek后，读取文件 \*/

printf("READ\n");

fd = open("/home/ningqijun/apue/chapter3/lseek\_test", O\_RDWR | O\_APPEND);

if (fd == -1)

{

printf("open failed\n");

goto EXIT;

}

//将文件指针设置在文件开头

ret = lseek(fd, 0, SEEK\_SET);

if (ret == -1)

{

printf("lseek function failed.\n");

close(fd);

goto EXIT;

}

printf("After lseek position is:%d\n", ret);

//读取文件

size = read(fd, read\_buf, sizeof(read\_buf));

if (size == -1)

{

printf("read failed\n");

close(fd);

goto EXIT;

}

printf("read number of character == %d\n", size);

printf("read\_buf:%s\n", read\_buf);

close(fd);

/\* lseek后，写入文件 \*/

printf("WRITE\n");

fd = open("/home/ningqijun/apue/chapter3/lseek\_test", O\_RDWR | O\_APPEND);

if (fd == -1)

{

printf("open failed\n");

goto EXIT;

}

//将文件指针设置在文件开头

ret = lseek(fd, 0, SEEK\_SET);

if (ret == -1)

{

printf("lseek failed.\n");

close(fd);

goto EXIT;

}

printf("After lseek position is:%d\n", ret);

//读取文件

size = write(fd, write\_buf, strlen(write\_buf));

if (size == -1)

{

printf("write failed\n");

close(fd);

goto EXIT;

}

printf("write number of character == %d\n", size);

close(fd);

EXIT:

return 0;

}

结论:  
如果使用追加标志O\_APPEND打开一个文件以便读、写; 能用lseek在任一位置开始读,不能用lseek更新文件中任一部分的数据，只能从文件末尾更新数据。