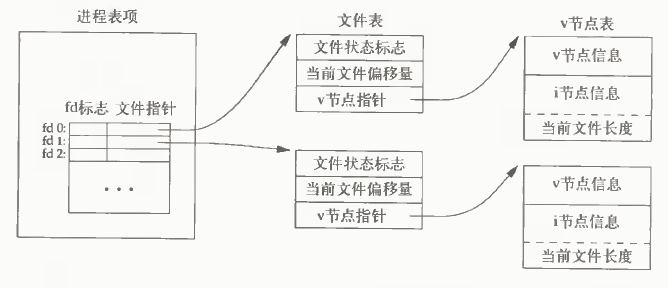
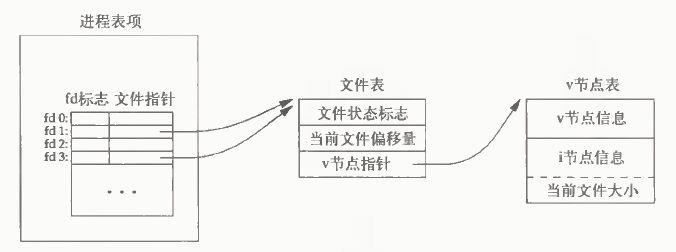
在unix高级编程中有介绍dup和dup2，但是没有实例说明， 笔者自己结合实例进行了测试了解。

在linux下，通过open打开以文件后，会返回一个文件描述符，文件描述符会指向一个文件表，文件表中的节点指针会指向节点表。看下图：

[](http://cdn.01happy.com/wp-content/uploads/2012/11/%E6%89%93%E5%BC%80%E6%96%87%E4%BB%B6%E7%9A%84%E5%86%85%E6%A0%B8%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84.png)

打开文件的内核数据结构

dup和dup2两个函数都可以用来复制打开的文件描述符，复制成功后和复制源共享同一个文件表。看下图：

[](http://cdn.01happy.com/wp-content/uploads/2012/11/%E6%89%A7%E8%A1%8Cdup%E5%90%8E%E7%9A%84%E5%86%85%E6%A0%B8%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84.png)

执行dup后的内核数据结构

****dup函数****

dup(现存的文件描述符)

dup返回的新文件描述符一定是当前可以用描述符中的最小值。下面先打开一个文件来看下文件描述符，为保证测试成功，创建一个测试文件log.txt。

#include <fcntl.h>

#include <stdio.h>

int main(int argc, char \*argv[])

{

int fd;

fd = open("./log.txt", O\_RDWR);

printf("%d\n", fd);

return 0;

}

上面的代码用读写打开了log.txt这个文件，编译上面的代码然后执行，执行成功的话，应当是输出3，因为0，1，2分别被标准输入，标准输出，标准错误输出占用了。使用dup复制这个文件描述符，并尝试移动fd偏移量：

#include <unistd.h>

#include <fcntl.h>

#include <stdio.h>

int main(int argc, char \*argv[])

{

int fd, copyfd;

fd = open("./log.txt", O\_RDWR);

//复制fd

copyfd = dup(fd);

//输出copyfd，应当为4

printf("%d\n", copyfd);

//打印出fd和copyfd的偏移量，都为0

printf("%d\n", (int)lseek(fd, 0, SEEK\_CUR));

printf("%d\n", (int)lseek(copyfd, 0, SEEK\_CUR));

//将fd的偏移量+3

lseek(fd, 3, SEEK\_SET);

//打印出fd和copyfd的偏移量，都为3

printf("%d\n", (int)lseek(fd, 0, SEEK\_CUR));

printf("%d\n", (int)lseek(copyfd, 0, SEEK\_CUR));

return 0;

}

编译执行上例代码可以发现当移动fd的偏移量时，copyfd的偏移量也发生了变化。往文件里写入内容试试，先把log.txt内容清空。

#include <unistd.h>

#include <fcntl.h>

#include <stdio.h>

int main(int argc, char \*argv[])

{

int fd, copyfd;

fd = open("./log.txt", O\_RDWR);

//复制fd

copyfd = dup(fd);

char buf1[] = "hello ";

char buf2[] = "world!";

//往fd文件写入内容

if (write(fd, buf1, 6) != 6) {

printf("write error!");

}

//打印出fd和copyfd的偏移量，经过上面的写操作，都变成6了

printf("%d\n", (int)lseek(fd, 0, SEEK\_CUR));

printf("%d\n", (int)lseek(copyfd, 0, SEEK\_CUR));

//往copyfd写入内容

if (write(copyfd, buf2, 6) != 6) {

printf("write error!");

}

return 0;

}

编译执行程序，log.txt的就有hello world!字符串了。

****dup2函数****

dup2(现存的文件描述符，可用的文件描述符)

dup2和dup函数一样，只是返回的文件描述符可以通过第二个参数”可用的文件描述符“指定。如果“可用的文件描述符“是打开状态，则会被关闭；如果”现存的文件描述符“和”可用的文件描述符“一样，则不会关闭，笔者认为这两个参数值一样的话，代码是没有任何意义的。

#include <unistd.h>

#include <fcntl.h>

#include <stdio.h>

int main(int argc, char \*argv[])

{

int fd, copyfd;

fd = open("./log.txt", O\_RDWR);

//指定文件描述符号为1000

copyfd = dup2(fd, 1000);

//打印fd和copyfd，应当输出3 1000

printf("%d %d\n", fd, copyfd);

return 0;

}

上面程序就是指定返回的文件描述为1000，再来看下指定的文件描述符是打开的情况，修改上例代码，将文件描述符指定为1：

copyfd = dup2(fd, 1);

编译执行程序将看不到任何输出，因为1是终端标准输出的标识符，经过这样一复制后，标准输出就被关闭了，使用printf自然看不到输出信息了。