# 第三章IO

## 1.文件描述符、文件描述符标志、文件状态标注

**1.文件描述符**

   文件描述符是一个非负整数

**2.文件描述符标志**

   目前只有一个文件描述符标志(close\_on\_exec)，一个文件描述符标志只属于一个进程。

  struct task\_struct--->struct files\_struct---> | struct file \* fd\_array[NR\_OPEN\_DEFAULT]

                | struct fdtable fdtab

  struct fdtable {

      unsigned int max\_fds;

      struct file \*\* fd;      /\* current fd array \*/

      fd\_set \*close\_on\_exec;

      fd\_set \*open\_fds;

      struct rcu\_head rcu;

      struct fdtable \*next;

  };

  fdtable字段  fd\_set \*close\_on\_exec;保存着进程文件描述符标志

  文件描述符标志(close\_on\_exec):仅仅是一个标志,当你fork了一个子进程,然后在子进程中调用了exec函数时就用到了该标志.意义是:执行exec前是否要关闭这个文件描述符.

   0：exec时不关闭已经打开的文件描述符

   1：exec时关闭已经打开的文件描述符

**3.文件状态标志**

   文件状态标志是体现进程打开文件的一些标志，fork时不会复制file结构，而是多个进程文件描述符指向同一个file

  struct file

  {

     unsigned int   f\_flags; //对应系统调用open的int flags参数

     fmode\_t f\_mode; //应系统调用open的mod\_t mode参数

  }

fcntl()函数能设置读和写上述两个标志

**4.exec函数集对文件描述符标志的影响：**

  1>对打开文件的处理与每个描述符的e x e c关闭标志值有关。进程中每个打开描述符都有一个exec关闭标志。

    若此标志设置，则在执行exec时关闭该描述符，否则该描述符仍打开。除非特地用fcntl设置了该标志，

    否则系统的默认操作是在exec后仍保持这种描述符打开。

  2>POSIX.1明确要求在exec时关闭打开目录流（见AUPE 4 . 2 1节中所述的opendir函数）,这通常是由opendir函数实现的，它调用fcntl函数为对应于打开目录流的描述符设置exec关闭标志。

**5.dup函数对文件描述符标志的影响：**

  新描述符的执行时关闭(close\_on\_exec)文件描述符标志总是由dup函数清除

**6.fcntl() 函数既能改变文件描述符标志，也能改变文件标志：**

  int fcntl(int fields, int cmd,.../\* int arg \* / ) ;

  ? 获得/设置文件描述符标记（cmd =FGETFD或FSETFD）

  ? 获得/设置文件状态标志  （cmd =FGETFL或FSETFL）