Linux系统编程之-read与write使用

概述

Linux哲学“一切皆文件”，我们也知道，文件操作中最基本的就是打开（open）、读（read）、写（write），内核会为每个进程维护一个打开文件的列表，该列表称为文件表。文件表由一些非负整数进行索引，这些非负整数称为文件描述符（file descriptors，简称为fds）。列表的每一项是一个打开文件的信息，包括指向该文件的索引节点（inode），其中inode中保存除了文件名以外的元数据信息。内核空间和用户空间都使用文件描述符作为唯一cookies，即**打开文件会返回文件描述符，而后续的操作都会将文件描述符作为基本的参数。**它包含以下要点：

- 内核会为每个进程维护一个文件表，而该文件表通过非负整数索引，我们把这个非负整数称为文件描述符（fds）
  - fds用整数来表示
  - fds存在最大的上限，默认情况下为1024
  - 负数表示文件操作函数返回错误
  - 因此，文件操作的返回值在Linux中应该是ssize_t (signed size_t)类型

系统调用open()打开文件

open基本使用

```
#include <sys/types.h>  // It is finded in the /usr/include/sys/types.h.
#include <sys/stat.h>   //It is finded in the /usr/include/sys/stat.h.
```

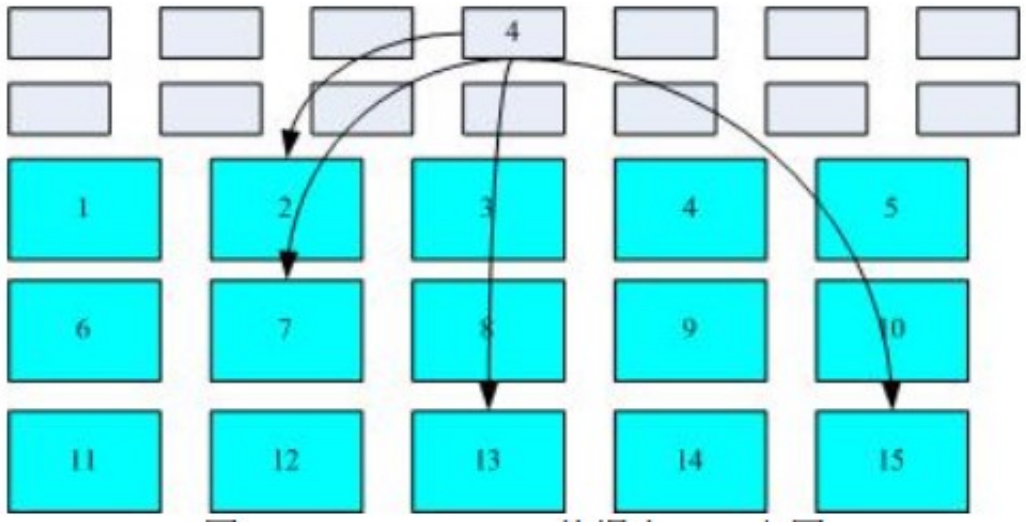
```c
#include <fcntl.h>  //fcntl means file control. It defines the file control operate.
#include <stdio.h>  //It includes perror(const char *str)
#include <unistd.h>   //It includes read(int fd, void* buf, size_t len)
#include <errno.h>
#include <string.h>  //It includes memset(void *str, int value, size_t __n)
#define BUFFER_SIZE 4096

int main (int argc,char *argv[]){
  int fd;   // It is the File Description which is used int the Linux System.
  fd = open("/home/os/Papers/Technical-Documents/OS/linux_system_programming/code/file/testRead.c",O_RDONLY);
  if (fd == -1){
    perror("main");
  }else{
    printf("Success!\n");
  }

  ssize_t ret;   // signed size_t, the negative numbers mean error

  char buf[BUFFER_SIZE];   //It means the buffer for the reading operating.
  memset(&buf,0,BUFFER_SIZE);  //It means the initial value for every value in the buffer

  while ((ret = (read(fd,&buf,BUFFER_SIZE))) != 0){
    if (ret == -1){
       if (errno == EINTR)   //It means interupts
           continue;
       perror("main");
       break;
    }
    for (int i = 0 ; i < BUFFER_SIZE; i++){
      printf("%c",buf[i]);
    }
  }
  if (close(fd)  == -1){
    perror("fd_close:");
```

```
37    }
38    return 0;
39 }
```

需要注意：使用open系统调用打开文件后，会返回对应文件的文件描述符fd，该文件描述符将作为后续read、write、close系统调用的输入参数。open的详细介绍参见：http://man7.org/linux/man-pages/man2/open.2.html 。

inode结构

假设当前文件分配到的inode为4，在这个inode中记录了该文件实际存放的物理块的位置，还包括文件的权限、用户、组等信息（但不包括文件名，文件名仅仅是方便用户管理，系统管理中不用文件名管理），在本例中，它们分别为2,7,13,15。通过“文件名”查找到相应的inode编号，则可以直接读取文件内容。用户在操作Linux时，使用文件名查找文件，实际上系统会查找“文件名与inode索引之间的映射”，在系统内部不使用文件名，仅仅使用inode来标识文件，在inode中也不存在任何与文件名相关的字段。在Linux中采用索引式文件系统(indexed allocation)，即inode（index node）机制。



系统调用read()读取文件

函数原型：`ssize_t read(int fd, void *buf, size_t count);`
该函数原型有三个参数：fd表示文件描述符，使用open函数打开文件后，会返回该值；buf用于存放读到的数据；count表示读取的字节数。该函数原型的返回值为ssize_t，当它大于0时，表示读取到buf中的字节数；当它小于0时，表示返回错误；当它为0时，表示读取到文件的末尾即EOF。

```
```

```c
#include <sys/types.h>  // It is finded in the /usr/include/sys/types.h.
#include <sys/stat.h>   //It is finded in the /usr/include/sys/stat.h.
#include <fcntl.h>  //fcntl means file control. It defines the file control operate.
#include <stdio.h>  //It includes perror(const char *str)
#include <unistd.h>   //It includes read(int fd, void* buf, size_t len)
#include <errno.h>
#include <string.h>  //It includes memset(void *str, int value, size_t __n)
#define BUFFER_SIZE 4096*2
// #define BUFFER_SIZE 20   // extreme case. The buffer size is common 4096 * n
int main (int argc,char *argv[]){
  int fd;   // It is the File Description which is used int the Linux System.
  //fd = open("/home/os/Papers/Technical-Documents/OS/linux_system_programming/code/file/testRead.c",O_RDONLY);   //example 1: reading the testRead.c file
  //fd = open("/home/os/Papers/Technical-Documents/OS/linux_system_programming/code/file/input_not_null",O_RDONLY);   //example 2: reading the input file
  fd = open("/home/os/Papers/Technical-Documents/OS/linux_system_programming/code/file/input_null",O_RDONLY);   //example 3: reading the null input file
  if (fd == -1){
    perror("main");
  }else{
    printf("Success!\n");
  }

  ssize_t ret;   // signed size_t, the negative numbers mean error
  char buf[BUFFER_SIZE];   //It means the buffer for the reading operating.
  memset(&buf,0,BUFFER_SIZE);  //It means the initial value for every value in the buffer

  while ((ret = (read(fd,&buf,BUFFER_SIZE))) != 0){
    if (ret == -1){
```

```
27          if (errno == EINTR)   //It means the program encounts
interupts, for software interupts and so on.
28              continue;
29          perror("main");
30          break;
31      }
32      for (size_t i = 0 ; i < BUFFER_SIZE; i++){
33        printf("%c",buf[i]);
34      }
35    }
36    if (close(fd)  == -1){
37      perror("fd_close:");
38    }
39    return 0;
40  }
```

系统调用write()读取文件

函数原型： `ssize_t write(int fd, void *buf, size_t count);`

该函数原型的参数个数与参数含义与read()函数一样，其返回值也表示写入到buf中的字节数，如果返回值小于0，表示返回错误；当返回值为0时，仅仅表示写入0个字节，并无任何其它的含义。

```
1  #include <sys/types.h>
2
3  #include <sys/stat.h>
4
5  #include <unistd.h>
6
7  #include <fcntl.h>
8
9  #include <stdio.h>
10
11 #include <string.h>
12
13 #include <errno.h>
14
15 #define BUFFER_SIZE 4096
16
```

```c
int main (int argc, char *argv[]){



  ssize_t fd_input = open("/home/os/Papers/Technical-Documents/OS/linux_system_programming/code/file/testRead.c",O_RDONLY);

  if (fd_input == -1){

    perror("input:");

  }



  ssize_t fd_output = open("/home/os/Papers/Technical-Documents/OS/linux_system_programming/code/file/output",O_WRONLY | O_APPEND);

  if (fd_output  == -1){

    perror("output:");

  }



  char buffer[BUFFER_SIZE];

  memset(&buffer,0,BUFFER_SIZE);



  ssize_t ret_read;

  ssize_t ret_write;

  while((ret_read = read(fd_input,&buffer,BUFFER_SIZE)) != 0){

    if(ret_read == -1){

```

```
55      if (errno  == EINTR){
56
57         continue;
58
59      }
60
61      perror("read:");
62
63      break;
64
65    }
66
67    ret_write = write(fd_output,&buffer,(ssize_t)ret_read);
68
69    if (ret_write == -1){
70
71      if(errno == EINTR){
72
73        continue;
74
75      }
76
77      perror("write:");
78
79      break;
80
81    }
82
83  }
84
85  int ret = fdatasync(fd_output);
86
87  printf("sync:%d\n",ret);
88
89  if (close(fd_output) == -1){   //The fd_output is closed that is
input file desctiptor. It is the second for the open sort.
90
91    perror("fd_outpt:");
92
93  }
94
95  if (close(fd_input)  == -1){
```

```
96
97      perror("fd_input:");
98
99    }
100   return 0;
101 }
```