

Linux系统编程1——系统函数

原创

夏季八起

于 2022-12-03 15:05:10 发布

118

收藏

版权


分类专栏：

Linux系统编程

 文章标签：

linux

服务器

Linux系统编程

专栏收录该内容

0 订阅

9 篇文章

订阅专栏

文章目录

- 前言
- 一、C标准函数与系统函数的区别
 - 1、什么是系统调用
 - 2、文件描述符
 - 3、相关函数
 - 3.1 open函数
 - 3.2 close函数
 - 3.3 read函数
 - 3.4 write函数
 - 3.5 lseek函数
 - 4、perror和errno
- 二、阻塞与非阻塞
- 三、文件和目录
 - 1、文件相关函数
 - 1.1、stat/lstat函数
 - 2、目录相关函数
 - 2.1、opendir函数
 - 2.2、readdir函数
 - 3.3、closedir函数
 - 3.4、dup函数
 - 3.5、dup2函数
 - 3.6、fcntl函数

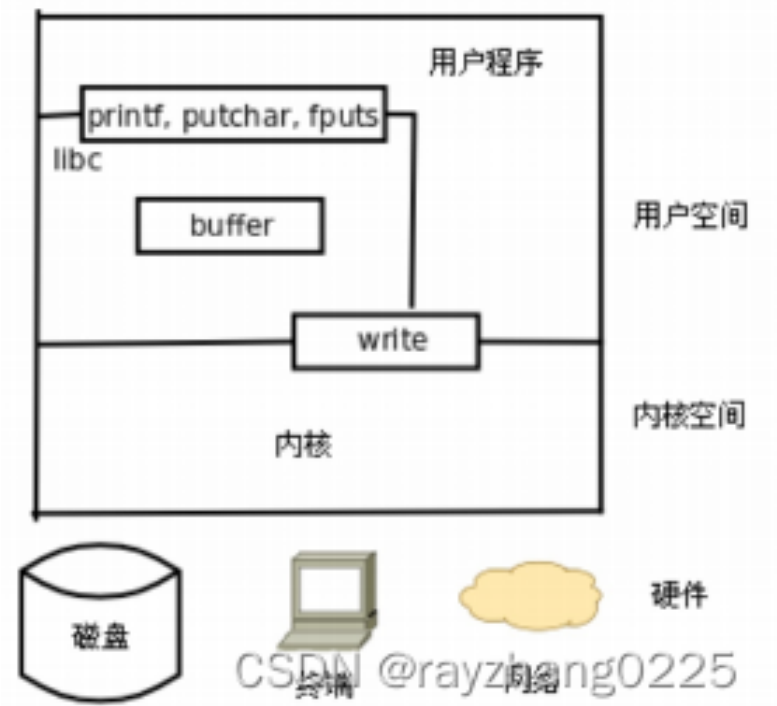
前言

本文用于记录linux系统函数学习

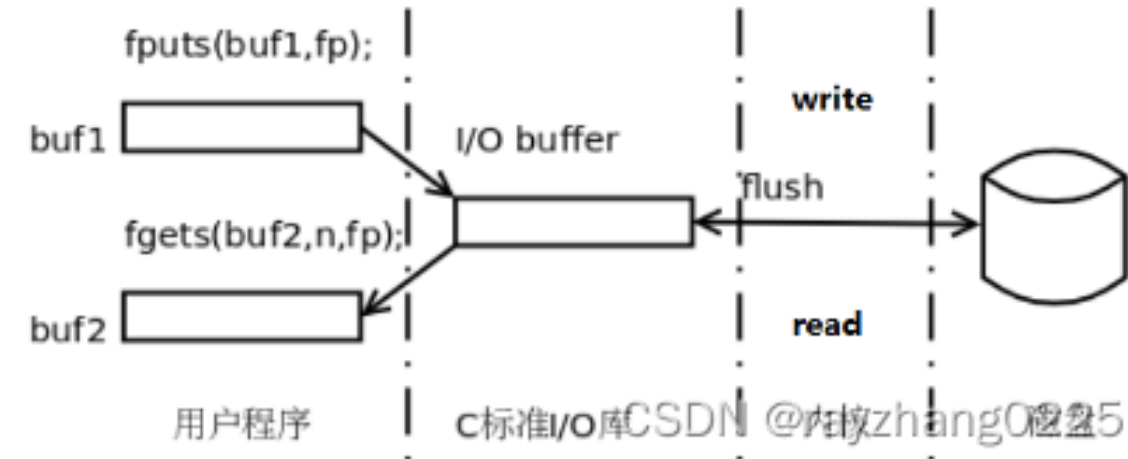
一、C标准函数与系统函数的区别

1、什么是系统调用

系统调用是由操作系统实现并提供给外部应用程序的编程接口，是应用程序同系统之间数据交互的桥梁。
一个hello world如何打印到屏幕上？



每一个FILE文件流（标准C库函数）都有一个缓冲区buffer，默认大小8192Byte。Linux系统的IO函数默认是没有用户级缓冲区的。



2、文件描述符

一个进程启动之后，默认打开三个文件描述符：

```
1 | #define STDIN_FILENO 0
2 | #define STDOUT_FILENO 1
3 | #define STDERR_FILENO 2
```

新打开文件返回文件描述符表中未使用的最小文件描述符，调用open函数就可以打开或创建一个文件，得到一个文件描述符。

3、相关函数

3.1 open函数

函数作用：打开或者新建一个文件

```
1 | 函数原型：
2 | int open(const char *pathname, int flags);
3 | int open(const char *pathname, int flags, mode_t mode);
4 | 函数参数：
5 | pathname参数
```

```
6         是要打开或创建的文件名,和fopen一样,  pathname既可以是相对路径也可以是绝对路径。
7 flags参数
8         有一系列常数值可供选择, 可以同时选择多个常数用按位或运算符连接起来, 所以这些常数的宏定义都以O_开头,表示or。
9         必选项:以下三个常数中必须指定一个, 且仅允许指定一个。
10         O_RDONLY  只读打开
11         O_WRONLY  只写打开
12         O_RDWR   可读可写打开
13         以下可选项可以同时指定0个或多个, 和必选项按位或起来作为flags参数。可选项有很多, 这里只介绍几个常用选项:
14         O_APPEND  表示追加。如果文件已有内容, 这次打开文件所写的数据附加到文件的末尾而不覆盖原来的内容。
15         O_CREAT   若此文件不存在则创建它。使用此选项时需要提供第三个参数mode, 表示该文件的访问权限。
16                 文件最终权限: mode & ~umask
17         O_EXCL    如果同时指定了O_CREAT, 并且文件已存在, 则出错返回。
18         O_TRUNC   如果文件已存在, 将其长度截断为为0字节。
19         O_NONBLOCK 对于设备文件, 以O_NONBLOCK方式打开可以做非阻塞
20                 I/O(NonblockI/O),非阻塞I/O。
21 函数返回值:
22         成功: 返回一个最小且未被占用的文件描述符
23         失败: 返回-1, 并设置errno值。
24
```

3.2 close函数

函数作用：关闭文件

```
1  函数原型: int close(int fd);
2  函数参数: fd文件描述符
3  函数返回值:
4      成功: 返回0
5      失败: 返回-1, 并设置errno值
```

需要说明的是,当一个进程终止时, 内核对该进程所有尚未关闭的文件描述符调用close关闭,所以即使用户程序不调用close, 在终止时内核也会自动关闭它打开的所有文件。但是对于一个长年累月运行的程序(比如网络服务器), 打开的文件描述符一定要记得关闭, 否则随着打开的文件越来越多, 会占用大量文件描述符和系统资源。

3.3 read函数

函数作用：从打开的设备或文件中读取数据

```
1  函数原型: ssize_t read(int fd, void *buf, size_t count);
2  函数参数:
3      fd: 文件描述符
4      buf: 读上来的数据保存在缓冲区buf中
5      count:  buf缓冲区存放的最大字节数
6  函数返回值:
7      >0 : 读取到的字节数
8      =0 : 文件读取完毕
9      -1 :  出错, 并设置errno
```

3.4 write函数

函数作用：向打开的设备或文件中写数据

```
1  函数原型: ssize_t write(int fd, const void *buf, size_t count);
2  函数参数:
3      fd: 文件描述符
4      buf: 缓冲区, 要写入文件或设备的数据
5      count: buf中的数据长度
6  函数返回值:
7      成功:  返回写入的字节数
8      错误:  返回-1并设置error
```

3.5 lseek函数

所有打开的文件都有一个当前文件偏移量(current file offset),以下简称为cfo. cfo通常是一个非负整数, 用于表明文件开始处到文件当前位置的字节数. 读写操作通常开始于 cfo, 并且使 cfo 增大, 增量为读写的字节数. 文件被打开时, cfo 会被初始化为 0, 除非使用了 O_APPEND. 使用 lseek 函数可以改变文件的 cfo.

```
1  #include <sys/types.h>
2  #include <unistd.h>
3  off_t lseek(int fd, off_t offset, int whence);
4  函数描述: 移动文件指针
5  函数原型: off_t lseek(int fd, off_t offset, int whence);
6  函数参数:
7      fd: 文件描述符
8      参数 offset 的含义取决于参数 whence:
9      如果 whence 是 SEEK_SET, 文件偏移量将设置为 offset。
10     如果 whence 是 SEEK_CUR, 文件偏移量将被设置为 cfo 加上 offset, offset 可以为正也可以为负。
11     如果 whence 是 SEEK_END, 文件偏移量将被设置为文件长度加上 offset, offset 可以为正也可以为负。
12  函数返回值: 若lseek成功执行, 则返回新的偏移量。
13
14  lseek函数常用操作
15  文件指针移动到头部
16  lseek(fd, 0, SEEK_SET);
17
18  获取文件指针当前位置
19  int len = lseek(fd, 0, SEEK_CUR);
20
21  获取文件长度
22  int len = lseek(fd, 0, SEEK_END);
23
24  lseek实现文件拓展
25  off_t currpos;
26  // 从文件尾部开始向后拓展1000个字节
27  currpos = lseek(fd, 1000, SEEK_END);
28  // 额外执行一次写操作, 否则文件无法完成拓展
29  write(fd, "a", 1);  // 数据随便写
```


4、perror和errno

errno是一个全局变量, 当系统调用后若出错会将errno进行设置, perror可以将errno对应的描述信息打印出来.
如:perror("open"); 如果报错的话打印: open:(空格)错误信息

二、阻塞与非阻塞

阻塞和非阻塞时文件本身的属性，不是read函数的属性。

- 普通文件：hello.c 默认是非阻塞的
- 终端设备：如/dev/tty 默认是阻塞的
- 管道和套接字：默认是阻塞的

三、文件和目录

1、文件相关函数

1.1、stat/lstat函数

函数作用：获取文件属性

```
1  函数原型: int stat(const char *pathname, struct stat *buf);
2          int lstat(const char *pathname, struct stat *buf);
3  函数返回值:
4      成功返回0
5      失败返回-1
6
7  struct stat {
8      dev_t      st_dev;          // 文件的设备编号
9      ino_t      st_ino;          // 节点
10     mode_t      st_mode;         // 文件的类型和存取的权限
11     nlink_t     st_nlink;        // 连到该文件的硬连接数目，刚建立的文件值为1
12     uid_t       st_uid;          // 用户ID
13     gid_t       st_gid;          // 组ID
14     dev_t       st_rdev;         // (设备类型) 若此文件为设备文件，则为其设备编号
15     off_t       st_size;         // 文件字节数(文件大小)
16     blksize_t   st_blksize;      // 块大小(文件系统的I/O 缓冲区大小)
17     blkcnt_t    st_blocks;       // 块数
18     time_t      st_atime;        // 最后一次访问时间
19     time_t      st_mtime;        // 最后一次修改时间
20     time_t      st_ctime;        // 最后一次改变时间( 指属性)
21 };
22 - st_mode -- 16位整数
23     0-2 bit -- 其他人权限
24     S_IROTH     00004   读权限
25     S_IWOTH     00002   写权限
26     S_IXOTH     00001   执行权限
27     S_IRWX0     00007   掩码，过滤 st_mode中除其他人权限以外的信息
28     3-5 bit -- 所属组权限
29     S_IRGRP     00040   读权限
30     S_IWGRP     00020   写权限
31     S_IXGRP     00010   执行权限
32     S_IRWXG     00070   掩码，过滤 st_mode中除所属组权限以外的信息
33     6-8 bit -- 文件所有者权限
34     S_IRUSR     00400   读权限
35     S_IWUSR     00200   写权限
36     S_IXUSR     00100   执行权限
37     S_IRWXU     00700   掩码，过滤 st_mode中除文件所有者权限以外的信息
38     If (st_mode & S_IRUSR) -----为真表明可读
39         If (st_mode & S_IWUSR) -----为真表明可写
40         If (st_mode & S_IXUSR) -----为真表明可执行
41     o 12-15 bit -- 文件类型
42     S_IFSOCK    0140000   套接字
43     S_IFLNK     0120000   符号链接（软链接）
44     S_IFREG     0100000   普通文件
45     S_IFBLK     0060000   块设备
46     S_IFDIR     0040000   目录
47     S_IFCHR     0020000   字符设备
48     S_IFIFO     0010000   管道
49     S_IFMT     0170000   掩码,过滤 st_mode中除文件类型以外的信息
50
51 if ((st_mode & S_IFMT)==S_IFREG) ---- 为真普通文件
52 if(S_ISREG(st_mode)) -----为真表示普通文件
53 if(S_ISDIR(st.st_mode)) -----为真表示目录文件
54
```

stat函数和lstat函数的区别

- 对于普通文件，这两个函数没有区别，是一样的
- 对于链接文件，调用lstat函数获取的是 **链接文件本身** 的属性信息；而stat函数获取的是 **链接文件指向的文件的** 属性信息

2、目录相关函数

2.1、opendir函数

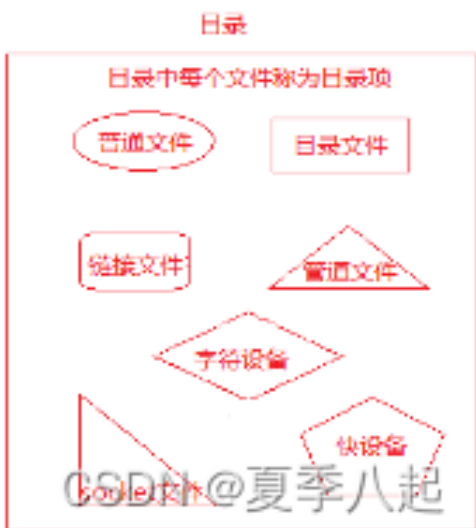
函数作用：打开一个目录

```
1  函数原型: DIR *opendir(const char *name);
2  函数返回值: 指向目录的指针
3  函数参数: 要遍历的目录（相对路径或者绝对路径）
```

2.2、readdir函数

函数作用：读取目录内容——目录项

```
1  函数原型: struct *readdir(DIR *dirp);
2  函数返回值: 读取的目录项指针
3  函数参数: opendir函数的返回值
4      struct dirent
5      {
6          ino_t d_ino;           // 此目录进入点的inode
7          off_t d_off;           // 目录文件开头至此目录进入点的位移
8          signed short int d_reclen;  // d_name 的长度, 不包含NULL 字符
9          unsigned char d_type;     // d_name 所指的文件类型
10         char d_name[256];        // 文件名
11     };
12
13     d_type的取值:
14         DT_BLK  - 块设备
15         DT_CHR  - 字符设备
16         DT_DIR  - 目录
17         DT_LNK  - 软连接
18         DT_FIFO - 管道
19         DT_REG  - 普通文件
20         DT_SOCK - 套接字
21         DT_UNKNOWN - 未知
```



3.3、closedir函数

函数作用：关闭目录

```
1  函数原型: int closedir(DIR *dirp);
2  函数返回值:
3      成功返回0
4      失败返回-1
5  函数参数:
6      opendir函数的返回值
```

读取目录内容的一般步骤

```
1  1 DIR *pDir = opendir("dir");    // 打开目录
2  2 while((p=readdir(pDir))!=NULL){} // 循环读取文件
3  3 closedir(pDir);    // 关闭目录
```

3.4、dup函数

函数作用：复制文件描述符

```
1  函数原型: int dup(int oldfd);
2  函数参数: oldfd — 要复制的文件描述符
3  函数返回值:
4      成功: 返回最小且没被占用的文件描述符
5      失败: 返回-1, 设置errno值
```

3.5、dup2函数

函数作用：复制文件描述符

```
1  函数原型: int dup2(int oldfd, int new fd);
2  函数参数:
3      old—原来的文件描述符
4      newfd—复制成新的文件描述符
5  函数返回值:
6      成功: 将oldfd复制给newfd, 两个文件描述符指向同一个文件
7      失败: 返回-1, 设置error值
8
9  假设newfd已经指向了一个文件, 首先close原来打开的文件, 然后newfd指向oldfd指向的文件
10 若newfd没有被占用, newfd指向oldfd指向的文件
```

3.6、fcntl函数

函数作用：改变已经打开的文件属性

```
1  函数原型: int fcntl(int fd, int cmd, ... /* arg */ );
2      若cmd为F_DUPFD, 复制文件描述符, 与dup相同
3      若cmd为F_GETFL, 获取文件描述符的flag属性值
4      若cmd为 F_SETFL, 设置文件描述符的flag属性
5  函数返回值: 返回值取决于cmd
6      成功
7          若cmd为F_DUPFD, 返回一个新的文件描述符
8          若cmd为F_GETFL, 返回文件描述符的flags值
9          若cmd为 F_SETFL, 返回0
10     失败
11         返回-1, 并设置errno值。
12
13  fcntl函数常用的操作:
14     1 复制一个新的文件描述符:
15         int newfd = fcntl(fd, F_DUPFD, 0);
16     2 获取文件的属性标志
17         int flag = fcntl(fd, F_GETFL, 0)
18     3 设置文件状态标志
19         flag = flag | O_APPEND;
20         fcntl(fd, F_SETFL, flag)
```


21	4 常用的属性标志
22	0_APPEND - - - - 设置文件打开为末尾添加
23	0_NONBLOCK - - - - 设置打开的文件描述符为非阻塞

🔗 文章知识点与官方知识档案匹配，可进一步学习相关知识

CS入门技能树 > Linux入门 > 初识Linux 31995 人正在系统学习中

Linux Socket编程入门——浅显易懂 01-06
文章目录1. 概述2. Socket3. 网络字节序4. sockaddr 数据结构5. 网络套接字API函数 5.1 socket () ... Socket本身有“插座”的意思，在...

Linux下编程——主函数传参 Dr_Cassie的博客 1569
主函数的原型为： int main(int argc,char *argv[],char *envp); argc：传递的参数列表中参数的个数 argv：传递的参数列表 envp：传递的环...

【Linux】主函数和文件操作函数_linux主函数与子函数怎么写_"爱"编程... 5-19
(4) 程序中调用函数 fflush(stdout); //主动刷新 二、文件操作函数 (1)库函数:fopen、fread、fwrite、fclose、fseek FILE *fopen(char *file, char...

七、Linux文件 - main函数参数讲解、代码实现cp指令 6-8
1、main函数参数 intmain(intargc,char*argv[]) { return0; } C语言规定了main函数的参数只能由2个,一个是argc,一个是argv,并且argc只能是...

Linux下简单Shell实现（二）基本功能---主函数及init()函数 Skillness 1440
主函数流程介绍完了Shell的基本情况，就开始描述代码了。首先如下代码段所示为主函数的内容，一共只有几句话，但却是本Shell的一个...

Linux 常用系统函数 qq_40944811的博客 103
linux系统函数

linux文件编程(3)—— main函数传参、myCp(配置成环境变量)、修改配 ... 5-25
掌握向main函数传参 intmain(intargc,char**argv)** 参数 argc: 参数个数 **argv:二级指针,数组的指针(argv[0]、argv[1]、argv[2]...),即这个指...

Linux C编程学习--main()函数简析_驱动main函数_zqixiao_09的博客... 5-4
而对我们嵌入式linux C 呢?这里是有操作系统的,是会调用main()函数的,所以这里一般会有返回值,所以 int main(int argc,char *argv[])是最标...

Linuxc-socket网络通信 er_ha_er的博客 76
网络通信： 底层遵循TCP/IP协议，在系统中以socket接口方式呈现 基于TCP协议的网络通信模型 服务端 客户端 创建socket对象 创建sock...

Linux 系统函数 这是一个c++热爱者的博客哟 515
可以使用 man 2 函数名 查看 系统函数也可以在 文件内 使用 2+ shift +k 进入指定函数 （前提是光标停在指定函数上） open打开文件 open...

Linux C代码实现主函数参数选项解析_李迟的博客 5-13
Linux C代码实现主函数参数选项解析 软件开发中难免会对命令行输入参数做解析、判断。本文给出2个此方面的示例。 1、手动解析版本 ...

linux编程规范--main函数_机器人107的博客 6-4
linux命令使用时,可以加参数。 linux系统中应用程序编程规范:main函数可以支持外部传参! #include <stdio.h> int main(int argc, char* argv[...

总结Linux下常用函数 用于记录学习的过程，不做商业用途。 1156
常用函数 建议学习

Linux网络编程函数 蓬莱道人的博客 220
1、socket函数： 2、bind函数： 3、网络字节序和主机字节序 4、listen函数： 5、accept函数： 6、connect函数： 7、读写socket的函数 8...

Linux Ubuntu操作系统下运行C语言Makefile文件编写_编写一个由头文件g... 6-1
这里是应题目要求. 你也可以将 'c' 文件中的函数直接写在 *.h' 中,在Linux ubuntu相同的目录下,可以直接引用并编译。 gcc myapp.c 1 关于...

Linux下网络编程常用函数 weixin_42655901的博客 87
Linux下网络编程常用函数

python全栈工程师基础——操作系统内核与系统调用 01-09
1) 掌握操作系统的定义：操作系统是一个用来协调、管理和控制计算机硬件和软件资源的系统程序，它位于硬件和应用程序之间。2) 掌...

linux 网络编程源代码 07-06
2.2.1 UNIX/Linux 文件系统简介19 2.2.2 流和标准 I/O...

LINUX操作系统（电子教案，参考答案） 07-11
主要包括Linux的基本概念和操作，Linux的树型结构，Linux的文本编辑，Linux的安装和启动，用户管理，Shell编程技术，进程管理，C编...

LINUX实验程序—————初学者 07-12
学习LINUX的课程小程序，有助于理解linux的很多概念与相关的系统编程的函数理解！借助这些实例，更好理解在linux下进行系统编程的...

清华大学Linux操作系统原理与应用 10-07
D.1 Linux编程常识 233 D.1.1 相关标准（ANSI C、POSIX、SVID、XPG） 233 D.1.2 函数库和系统调用 234 D.1.3 在线文档（man、info...

Linux编程实现DS18B20对树莓派温度的探测 01-06
DS18B20是常用的数字温度传感器，其输出的是数字信号，具有体积小，硬件开销低，抗干扰能力强，精度高的特点，本次实验项目是依...

Linux之函数 chd_sun的博客 114
1.linux shell可以用户定义函数，然后再shell脚本中可以随便调用 2.可以带 function fun()定义,也可以直接fun()定义,不带任何参数 3.参数返...

linux中一些常见函数的含义 linux内核 1001
一.1.container_of：根据一个结构体变量中的一个域成员变量的指针来获取指向整个结构体变量的指针 例如： 有一个结构体变量，其定义...

Linux系统编程——函数 Strive_LiJiaLe的博客 41
笔记

Linux学习之cp函数编写 qq_46417372的博客 542
cp命令是我们常用的文件复制命令，他可以将一个文件的内容复制到另一个文件。 其调用格式为：cp 文件1.后缀 文件2.后缀。 我们都知道...

linux系统调用函数 最新发布 06-01
回答1： Linux系统调用函数是指在Linux操作系统中，用于与操作系统内核进行交互的一组函数。这些函数包括文件操作、进程管理、...

“相关推荐”对你有帮助么？

😞 非常没帮助 😐 没帮助 😐 一般 😊 有帮助 😄 非常有帮助

