# **Linux常用函数**

系统IO函数：

open()

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

// 打开一个已经存在的文件

int open(const char \*pathname, int flags);

int open(const char \*pathname, int flags);

参数：

- pathname：要打开的文件路径

- flags：对文件的操作权限设置还有其他的设置

O\_RDONLY, O\_WRONLY, O\_RDWR 这三个设置是互斥的

返回值：返回一个新的文件描述符，如果调用失败，返回-1

errno：属于Linux系统函数库，库里面的一个全局变量，记录的是最近的错误号。

void perror(const char \*s);作用：打印errno对应的错误描述

s参数：用户描述，比如hello,最终输出的内容是 hello:xxx(实际的错误描述)

// 创建一个新的文件

int open(const char \*pathname, int flags, mode\_t mode);

参数：

- pathname：要创建的文件的路径

- flags：对文件的操作权限和其他的设置

- 必选项：O\_RDONLY, O\_WRONLY, O\_RDWR 这三个之间是互斥的

- 可选项：O\_CREAT 文件不存在，创建新文件

- mode：八进制的数，表示创建出的新的文件的操作权限，比如：0775

最终的权限是：mode & ~umask

0777 -> 111111111

& 0775 -> 111111101

----------------------------

111111101

按位与：0和任何数都为0

umask的作用就是抹去某些权限。

flags参数是一个int类型的数据，占4个字节，32位。

flags 32个位，每一位就是一个标志位。

\*//关闭文件：

int close(int fd);

read()&write()

/\*

#include <unistd.h>

ssize\_t read(int fd, void \*buf, size\_t count);

参数：

- fd：文件描述符，open得到的，通过这个文件描述符操作某个文件

- buf：需要读取数据存放的地方，数组的地址（传出参数）

- count：指定的数组的大小

返回值：

- 成功：

>0: 返回实际的读取到的字节数

=0：文件已经读取完了

- 失败：-1 ，并且设置errno

#include <unistd.h>

ssize\_t write(int fd, const void \*buf, size\_t count);

参数：

- fd：文件描述符，open得到的，通过这个文件描述符操作某个文件

- buf：要往磁盘写入的数据，数据

- count：要写的数据的实际的大小

返回值：

成功：实际写入的字节数

失败：返回-1，并设置errno

\*/

lseek()

#include <sys/types.h>

#include <unistd.h>

off\_t lseek(int fd, off\_t offset, int whence);

参数：

- fd：文件描述符，通过open得到的，通过这个fd操作某个文件

- offset：偏移量

- whence:

SEEK\_SET

设置文件指针的偏移量

SEEK\_CUR

设置偏移量：当前位置 + 第二个参数offset的值

SEEK\_END

设置偏移量：文件大小 + 第二个参数offset的值

返回值：返回文件指针的位置

作用：

1.移动文件指针到文件头

lseek(fd, 0, SEEK\_SET);

2.获取当前文件指针的位置

lseek(fd, 0, SEEK\_CUR);

3.获取文件长度

lseek(fd, 0, SEEK\_END);

4.拓展文件的长度，当前文件10b, 110b, 增加了100个字节

lseek(fd, 100, SEEK\_END)

注意：需要写一次数据

\*/

stat()&lstat()

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

int stat(const char \*pathname, struct stat \*statbuf);

作用：获取一个文件相关的一些信息

参数:

- pathname：操作的文件的路径

- statbuf：结构体变量，传出参数，用于保存获取到的文件的信息

返回值：

成功：返回0

失败：返回-1 设置errno

int lstat(const char \*pathname, struct stat \*statbuf);

参数:

- pathname：操作的文件的路径

- statbuf：结构体变量，传出参数，用于保存获取到的文件的信息

返回值：

成功：返回0

失败：返回-1 设置errno

\*/

struct stat {

dev\_t st\_dev; // 文件的设备编号

ino\_t st\_ino; // 节点

mode\_t st\_mode; // 文件的类型和存取的权限

nlink\_t st\_nlink; // 连到该文件的硬连接数目

uid\_t st\_uid; // 用户ID

gid\_t st\_gid; // 组ID

dev\_t st\_rdev; // 设备文件的设备编号

off\_t st\_size; // 文件字节数(文件大小)

blksize\_t st\_blksize; // 块大小

blkcnt\_t st\_blocks; // 块数

time\_t st\_atime; // 最后一次访问时间

time\_t st\_mtime; // 最后一次修改时间

time\_t st\_ctime; // 最后一次改变时间(指属性)

};

文件属性操作函数：access() chmode() truncate()

#include <unistd.h>

int access(const char \*pathname, int mode);

作用：判断某个文件是否有某个权限，或者判断文件是否存在

参数：

- pathname: 判断的文件路径

- mode:

R\_OK: 判断是否有读权限

W\_OK: 判断是否有写权限

X\_OK: 判断是否有执行权限

F\_OK: 判断文件是否存在

返回值：成功返回0， 失败返回-1

\*/

#include <sys/stat.h>

int chmod(const char \*pathname, mode\_t mode);

修改文件的权限

参数：

- pathname: 需要修改的文件的路径

- mode:需要修改的权限值，八进制的数

返回值：成功返回0，失败返回-1

\*/

/\*

#include <unistd.h>

#include <sys/types.h>

int truncate(const char \*path, off\_t length);

作用：缩减或者扩展文件的尺寸至指定的大小

参数：

- path: 需要修改的文件的路径

- length: 需要最终文件变成的大小

返回值：

成功返回0， 失败返回-1

\*/

目录遍历函数：chdir() mkdir() getcwd()

/\*

#include <unistd.h>

int chdir(const char \*path);

作用：修改进程的工作目录

比如在/home/nowcoder 启动了一个可执行程序a.out, 进程的工作目录 /home/nowcoder

参数：

path : 需要修改的工作目录

/\*

#include <sys/stat.h>

#include <sys/types.h>

int mkdir(const char \*pathname, mode\_t mode);

作用：创建一个目录

参数：

pathname: 创建的目录的路径

mode: 权限，八进制的数

返回值：

成功返回0， 失败返回-1

\*/

#include <unistd.h>

char \*getcwd(char \*buf, size\_t size);

作用：获取当前工作目录

参数：

- buf : 存储的路径，指向的是一个数组（传出参数）

- size: 数组的大小

返回值：

返回的指向的一块内存，这个数据就是第一个参数

\*/

#include <stdio.h>

int rename(const char \*oldpath, const char \*newpath);

\*/

dup() dup2()

/\*

#include <unistd.h>

int dup(int oldfd);

作用：复制一个新的文件描述符

fd=3, int fd1 = dup(fd),

fd指向的是a.txt, fd1也是指向a.txt

从空闲的文件描述符表中找一个最小的，作为新的拷贝的文件描述符

\*/

/\*

#include <unistd.h>

int dup2(int oldfd, int newfd);

作用：重定向文件描述符

oldfd 指向 a.txt, newfd 指向 b.txt

调用函数成功后：newfd 和 b.txt 做close, newfd 指向了 a.txt

oldfd 必须是一个有效的文件描述符

oldfd和newfd值相同，相当于什么都没有做

\*/

#include <unistd.h>

#include <stdio.h>

#include <string.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <fcntl.h>

fnctl()

/\*

#include <unistd.h>

#include <fcntl.h>

int fcntl(int fd, int cmd, ...);

参数：

fd : 表示需要操作的文件描述符

cmd: 表示对文件描述符进行如何操作

- F\_DUPFD : 复制文件描述符,复制的是第一个参数fd，得到一个新的文件描述符（返回值）

int ret = fcntl(fd, F\_DUPFD);

- F\_GETFL : 获取指定的文件描述符文件状态flag

获取的flag和我们通过open函数传递的flag是一个东西。

- F\_SETFL : 设置文件描述符文件状态flag

必选项：O\_RDONLY, O\_WRONLY, O\_RDWR 不可以被修改

可选性：O\_APPEND, O)NONBLOCK

O\_APPEND 表示追加数据

NONBLOK 设置成非阻塞

阻塞和非阻塞：描述的是函数调用的行为。

\*/

小球从N点出发的速度方向不确定 使用正交分解，

考虑**垂直于电场方向**与**平行于电场方向**的两个方向的速度

垂直于电场方向的速度 电场力对其没有做工 根据小球在M点沿PM方向进入磁场，可知其即为小球进入磁场的速度

在磁场与电场的复合场中，小球能做匀速直线运动 磁力与电场力二力平衡：

得

平行于电场方向的速度 方向与电场力相反 小球在这个方向作减速运动 到达M点时 减速为0 小球只剩下垂直于电场方向的速度

平行于电场方向做匀减速运动 减速为0

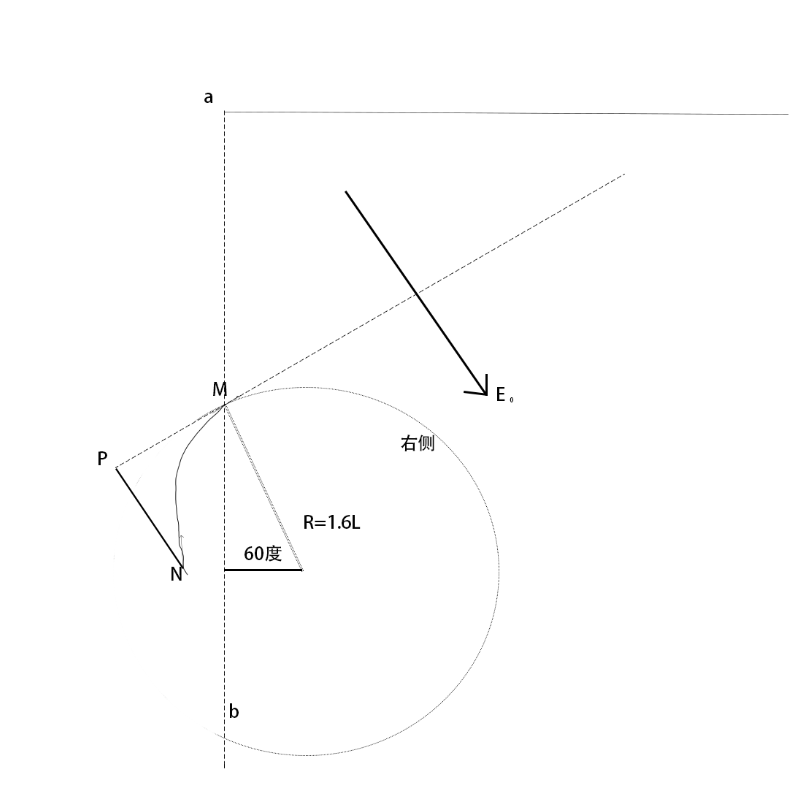
即

整理得： 其中

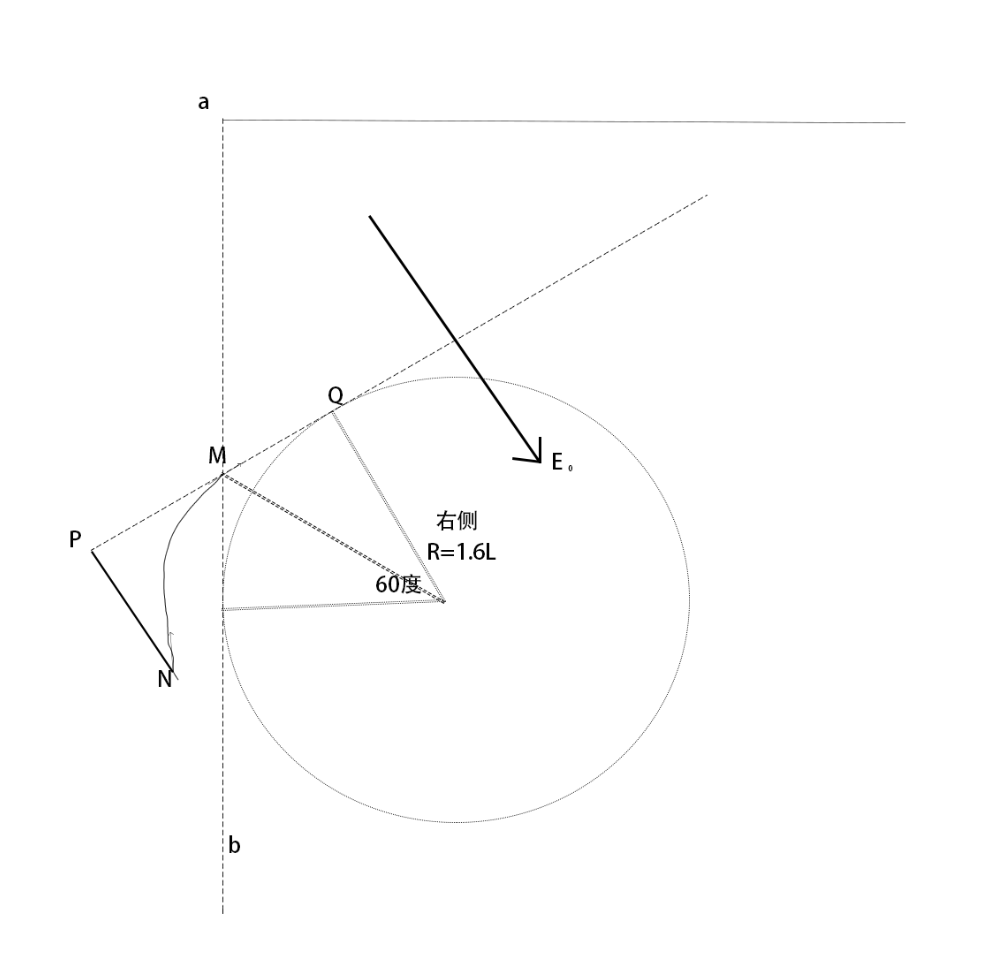
解二次一元方程得到：

撤去电场后：小球在磁场中运动半径：

假设小球一进入磁场就撤去磁场 运动路径如下所示：



要想使得小球刚好在磁场内运动出不来 应该在Q点撤去电场 使得运动轨迹圆与ab刚好相切：



根据几何关系可得：MQ=

滑块P的运动过程分为三个阶段：

第一阶段：被物体Q下落拉着加速上升直到物体Q撞击地面：

因为物体Q本身也在做加速运动不能确定绳子拉力就是物体Q重力

将物体Q与P看作一个整体进行受力分析

物体Q的重力　滑块P受到其重力分量与摩擦力

第一阶段的运动距离为h 最终速度为：

第二阶段：滑块P继续向上减速运动 此时只受到重力分量与摩擦力

运动距离：

第三阶段：滑块P在重力分量作用下开始向下滑动 滑动距离为h+x2= 然后回到斜劈底部 绳子突然绷直 然后Q与P速度变得一致 类似于完全非弹性碰撞 使用动量守恒进行计算

根据动量守恒:

最后 滑块P撞停 物体Q做竖直上抛运动

可将斜劈 滑块与物体Q视为一个整体分析 三者之间的力属于内力不考虑

作为一个整体 只受到重力 支持力T与地面的摩擦力f作用

（牛三考虑整体受力 然后部分的加速度作为整体的受力结果）

只考虑竖直方向(取向下为正方向)：在竖直方向上 Q在向下加速 P具有竖直向上的加速分量

只考虑水平方向：只有滑块P有水平方向的加速度分量：

其方向水平向左 与水平分量方向一致

（不保证各个步骤计算正确 最好自己照着这个思路推导一边）