# 1、access 函数：

#include<unistd.h>

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

参数介绍：

    pathname 是文件的路径名+文件名

    mode：指定access的作用，取值如下

**mode 参数：**

F\_OK 值为0，判断文件是否存在

X\_OK 值为1，判断对文件是可执行权限

W\_OK 值为2，判断对文件是否有写权限

R\_OK 值为4，判断对文件是否有读权限

注：后三种可以使用或“|”的方式，一起使用，如W\_OK|R\_OK

返回值：成功0，失败-1

例子：

#include"stdio.h"

#include"unistd.h"

#include "string.h"

#define fileNAME1 "test"

#define fileNAME2 "./liang"

int main(void)

{

char name[BUFSIZ];//文件名字

int flag = 1;//退出标志,0 exit

printf("\t\t\t程序开始\n");

printf("请输入要检查的文件(可包含路径,EOF退出):");

scanf("%s",name);

if( strcmp(name,"EOF") == 0 ){

flag = 0;

}

while(flag){

if(access(name,F\_OK)==0){

printf("文件存在\n");

if(access(name,R\_OK|W\_OK)==0){

printf("文件可读可写\n");

}else

printf("文件不可读或不可写\n");

if(access(name,X\_OK)==0){

printf("文件可执行\n");

}else

printf("文件不可执行\n");

}else

printf("文件不存\n");

printf("\n请输入要检查的文件(可包含路径,EOF退出):");

scanf("%s",name);

if( strcmp(name,"EOF") == 0 ){

flag = 0;

}

}//while

printf("\t\t\t程序结束\n");

return 0;

}

# 2、#ifdef #endif

用法：

#ifdef 语句1

　　//程序2

#endif

可翻译为：如果宏定义了语句1则程序2。

作用：我们可以用它区隔一些与特定头文件、程序库和其他文件版本有关的代码。



# 3、snprintf函数

snprintf(data,128,"{\"etype\":\"doorbell\_press\",\"edata\":\"%x%d\"}",wakeup\_data,(INT\_T)timeutc);

data 是 输出参数

128 是 输出参数的最大值

后面的是相关的数据 类似与printf 一样的使用

# 4、localtime\_r 函数

localtime\_r也是用来获取系统时间，运行于linux平台下

函数原型为struct tm \*localtime\_r(const time\_t \*timep, struct tm \*result);

#include <stdio.h>

#include <time.h>

int main()

{

time\_t time\_seconds = time(0);

struct tm now\_time;

localtime\_r(&time\_seconds, &now\_time);

printf("%d-%d-%d %d:%d:%d\n", now\_time.tm\_year + 1900, now\_time.tm\_mon + 1,

now\_time.tm\_mday, now\_time.tm\_hour, now\_time.tm\_min, now\_time.tm\_sec);

}

# 5、mount、umount、umount2

参考文档：<https://www.yiibai.com/unix_system_calls/mount.html>

6、#include<stdbool.h> c语言包含bool 类型的

# 7、环境变量

1、PATH是可执行文件的环境变量

Export PATH=$PWD/toolchain/arm-buildroot-linux-uclibcgnueabihf-4.9.4/bin:$PATH

2、LIBRARY\_PATH 和 LD\_LIBRARY\_PATH 是Linux下的两个环境变量，二者的含义和作用分别如下：

两者的共同点是库，库是这两个路径和PATH路径的区别；

LIBRARY\_PATH环境变量用于在***程序编译期间***查找动态链接库时指定查找共享库的路径；

例如，指定gcc编译需要用到的动态链接库的目录。设置方法如下（其中，LIBDIR1和LIBDIR2为两个库目录）：

export LIBRARY\_PATH=LIBDIR1:LIBDIR2:$LIBRARY\_PATH

LD\_LIBRARY\_PATH环境变量用于在***程序加载运行期间***查找动态链接库时指定除了系统默认路径之外的其他路径；

注意，LD\_LIBRARY\_PATH中指定的路径会在系统默认路径之前进行查找。

设置方法如下（其中，LIBDIR1和LIBDIR2为两个库目录）：

export LD\_LIBRARY\_PATH=LIBDIR1:LIBDIR2:$LD\_LIBRARY\_PATH

举个例子，我们开发一个程序，经常会需要使用某个或某些动态链接库，为了保证程序的可移植性，可以先将这些编译好的动态链接库放在自己指定的目录下，然后按照上述方式将这些目录加入到LD\_LIBRARY\_PATH环境变量中，这样自己的程序就可以动态链接后加载库文件运行了。

**区别与使用：**

开发时，设置LIBRARY\_PATH，以便gcc能够找到编译时需要的动态链接库。

发布时，设置LD\_LIBRARY\_PATH，以便程序加载运行时能够自动找到需要的动态链接库。