**KalinoteOS 系统支持文档**

目录

[1. 系统内置CMD指令 1](#_Toc59265628)

[1.1 查询类 1](#_Toc59265629)

[mem指令 1](#_Toc59265630)

[dir&ls指令 1](#_Toc59265631)

[1.2 功能类 1](#_Toc59265632)

[cls&clear指令 1](#_Toc59265633)

[type [Filename]指令 1](#_Toc59265634)

[exit 指令 1](#_Toc59265635)

[start指令 1](#_Toc59265636)

[run指令 1](#_Toc59265637)

[langmode指令 2](#_Toc59265638)

[shutdown指令 2](#_Toc59265639)

[2. KalinoteOS系统API 2](#_Toc59265640)

[\_api\_putchar: 2](#_Toc59265641)

[\_api\_putstr0: 2](#_Toc59265642)

[cons\_putstr1: 2](#_Toc59265643)

[api\_end: 3](#_Toc59265644)

[api\_openwin: 3](#_Toc59265645)

[\_api\_putstrwin: 3](#_Toc59265646)

[\_api\_boxfilwin: 4](#_Toc59265647)

[\_api\_initmalloc: 4](#_Toc59265648)

[\_api\_malloc: 4](#_Toc59265649)

[\_api\_free: 5](#_Toc59265650)

[\_api\_point: 5](#_Toc59265651)

[\_api\_refreshwin: 5](#_Toc59265652)

[\_api\_linewin: 6](#_Toc59265653)

[\_api\_closewin: 6](#_Toc59265654)

[\_api\_getkey: 6](#_Toc59265655)

[\_api\_alloctimer: 6](#_Toc59265656)

[\_api\_inittimer: 7](#_Toc59265657)

[\_api\_settimer: 7](#_Toc59265658)

[\_api\_freetimer: 7](#_Toc59265659)

[\_api\_beep: 7](#_Toc59265660)

[\_api\_fopen: 8](#_Toc59265661)

[\_api\_fclose: 8](#_Toc59265662)

[\_api\_fseek: 8](#_Toc59265663)

[\_api\_fsize: 9](#_Toc59265664)

[\_api\_fread: 9](#_Toc59265665)

[\_api\_cmdline: 9](#_Toc59265666)

[\_api\_getlang: 10](#_Toc59265667)

[\_api\_cls: 10](#_Toc59265668)

[3. 标准函数API 10](#_Toc59265669)

[string.h 10](#_Toc59265670)

[malloc.h(stdlib.h) 10](#_Toc59265671)

[stdio.h 10](#_Toc59265672)

[ctype.h 11](#_Toc59265673)

[stdlib.h 11](#_Toc59265674)

[4. 后面要做的事 11](#_Toc59265675)

[压缩算法 11](#_Toc59265676)

[API函数标准化(标准函数) 11](#_Toc59265677)

# 系统内置CMD指令

## 查询类

### mem指令

该指令为memory的缩写，作用是查询系统内存使用情况。

### dir&ls指令

该指令用于查询系统中的文件。

## 功能类

### cls&clear指令

cls为clear screen的缩写，该指令的作用是清空命令行窗口。

### type [Filename]指令

type指令用于输出某个文件的内容。

### exit 指令

该指令用于关闭命令窗口。

### start指令

该指令用于在新的命令窗口启动一个应用程序。

### run指令

该指令用于在当前命令窗口执行一个应用程序，并不占用当前命令窗口。

### langmode指令

该指令用于切换系统显示语言模式。

### shutdown指令

该指令用于关闭计算机。

# KalinoteOS系统API

### \_api\_putchar:

寄存器数据：

EDX = 1

AL = 字符

描述：

在命令窗口打印单个字符。

调用方法：

**void api\_putchar(int c);**

### \_api\_putstr0:

寄存器数据：

EDX = 2

EBX = 字符串

描述：

在命令窗口打印字符串，并在结尾为字符编码0时结束。

调用方法：

**void api\_putstr0(char \*s);**

### cons\_putstr1:

寄存器数据：

EDX = 3

EBX = 字符串

描述：

在命令窗口打印字符串，并提前指定字符串长度。

调用方法：

停止使用

### api\_end:

寄存器数据：

EDX = 4

描述：

用于结束运行应用程序。

调用方法：

**void api\_end(void);**

### api\_openwin:

寄存器数据：

EDX = 5

EBX = 窗口缓冲区

ESI = 窗口X(高度)

EDI = 窗口Y(宽度)

EAX = 透明色

ECX = 窗口名称

描述：

用于生成一个基本窗口。

调用方法：

**int api\_openwin(char \*buf, int xsiz, int ysiz, int col\_inv, char \*title);**

### \_api\_putstrwin:

寄存器数据：

EDX = 6

EBX = 窗口句柄

ESI = 显示字符串X坐标

EDI = 显示字符串Y坐标

EAX = 色号

ECX = 字符串长度

EBP = 字符串

描述：

用于在指定窗口上打印一个字符串。

调用方法：

**void api\_putstrwin(int win, int x, int y, int col, int len, char \*str);**

### \_api\_boxfilwin:

寄存器数据：

EDX = 7

EBX = 窗口句柄

EAX =

ECX =

ESI =

EDI =

EBP = 色号

描述：

用于在指定窗口上打印一个字符串。

调用方法：

**void api\_boxfilwin(int win, int x0, int y0, int x1, int y1, int col);**

### \_api\_initmalloc:

寄存器数据：

EDX = 8

EBX = memman地址

EAX = memman所管理的内存空间的起始地址

ECX = memman所管理的内存空间的字节数

描述：

用于初始化内存分配程序。

调用方法：

**void api\_initmalloc(void);**

### \_api\_malloc:

寄存器数据：

EDX = 9

EBX = memman的地址

ECX = 需要请求的字节数

EAX = 分配到的内存空间地址

描述：

用于分配内存空间。

调用方法：

**char \*api\_malloc(int size);**

### \_api\_free:

寄存器数据：

EDX = 10

EBX = memman地址

EAX = 需要释放的内存空间地址

ECX = 需要释放的字节数

描述：

用于释放内存空间。

调用方法：

**void api\_end(void);**

### \_api\_point:

寄存器数据：

EDX = 11

EBX = 窗口句柄

ESI = 显示位置的X坐标

EDI = 显示位置的Y坐标

EAX = 色号

描述：

在指定窗口的某个位置描绘一个像素点。

调用方法：

**void api\_point(int win, int x, int y, int col);**

### \_api\_refreshwin:

寄存器数据：

EDX = 12

EBX = 窗口句柄

EAX =

ECX =

ESI =

EDI =

描述：

用于窗口的区域渲染，在窗口绘制完成后刷新以显示。

使用方法：

**void api\_refreshwin(int win, int x0, int y0, int x1, int y1);**

### \_api\_linewin:

寄存器数据：

EDX = 13

EBX = 窗口句柄

EAX =

ECX =

ESI =

EDI =

EBP = 色号

描述：

用于窗口直线渲染。

使用方法：

**void api\_linewin(int win, int x0, int y0, int x1, int y1, int col);**

### \_api\_closewin:

寄存器数据：

EDX = 14

EBX = 窗口句柄

描述：

用于关闭窗口。

使用方法：

**void api\_closewin(int win);**

### \_api\_getkey:

寄存器数据：

EDX = 15

EAX = 获取模式

EAX = 键盘输入的字符编码

描述：

用于获取键盘数据，关于获取模式：参数为0的话该函数在没有键盘输入时返回 -1，参数为1的话该函数在没有键盘输入时休眠。

使用方法：

**int api\_getkey(int mode);**

### \_api\_alloctimer:

寄存器数据：

EDX = 16

EAX = 定时器句柄(由系统返回)

描述：

从系统申请一个定时器。

使用方法：

**int api\_alloctimer(void);**

### \_api\_inittimer:

寄存器数据：

EDX = 17

EBX = 定时器句柄

EAX = 数据

描述：

设定一个数据，在定时器超时后会返回这个数据。

使用方法：

**void api\_inittimer(int timer, int data);**

### \_api\_settimer:

寄存器数据：

EDX = 18

EBX = 定时器句柄

EAX = 时间(ms，毫秒)

描述：

设定一个时间，产生延时。

使用方法：

**void api\_settimer(int timer, int time);**

### \_api\_freetimer:

寄存器数据：

EDX = 19

EBX = 定时器句柄

描述：

释放定时器。

使用方法：

**void api\_freetimer(int timer);**

### \_api\_beep:

寄存器数据：

EDX = 20

EAX = 声音频率(0为停止)

描述：

控制蜂鸣器发声。

使用方法：

**void api\_beep(int tone);**

### \_api\_fopen:

寄存器数据：

EDX = 21

EBX = 文件名

EAX = 文件句柄(为0时表示打开失败，由操作系统返回)

描述：

用于打开一个文件。

使用方法：

**int api\_fopen(char \*fname);**

### \_api\_fclose:

寄存器数据：

EDX = 22

EAX = 文件句柄

描述：

用于关闭一个文件。

使用方法：

**void api\_fclose(int fhandle);**

### \_api\_fseek:

寄存器数据：

EDX = 23

EAX = 文件句柄

ECX = 定位模式

EBX = 定位偏移量

描述：

用于寻找文件时的文件定位，其中定位模式有以下几种：

ECX = 0 定位的起点为文件开头

ECX = 1 定位的起点为当前访问的位置

ECX = 2 定位的起点为文件末尾

使用方法：

**void api\_fseek(int fhandle, int offset, int mode);**

### \_api\_fsize:

寄存器数据：

EDX = 24

EAX = 文件句柄

ECX = 文件大小获取模式

描述：

用于获取文件大小，其中获取模式有以下几种：

ECX = 0 普通文件大小

ECX = 1 当前读取位置从文件开头起算的偏移量

ECX = 2 当前读取位置从文件末尾起算的偏移量

使用方法：

**int api\_fsize(int fhandle, int mode);**

### \_api\_fread:

寄存器数据：

EDX = 25

EAX = 文件句柄

EBX = 缓冲区地址

ECX = 最大读取字节数

EAX = 本次读取到的字节数(由系统返回)

描述：

用于读取文件。

使用方法：

**int api\_fread(char \*buf, int maxsize, int fhandle);**

### \_api\_cmdline:

寄存器数据：

EDX = 26

EBX = 存放命令行内容的地址

ECX = 最多可以放多少字节

EAX = 实际存放了多少字节(操作系统返回)

描述：

用于获取用户在命令行输入的参数

使用方法：

**int api\_cmdline(char \*buf, int maxsize);**

### \_api\_getlang:

寄存器数据：

EDX = 27

EAX = langmode(由系统返回)

描述：

查询当前系统语言模式。

使用方法：

**int api\_getlang(void);**

### \_api\_cls:

寄存器数据：

EDX = 29

描述：

清空cmd内容（这个api暂时不能使用）。

使用方法：

void api\_cls(void);

# 标准函数API

下面是KalinoteOS支持的C语言标准API，具体使用方法可以查询C语言支持手册。

### string.h

int strchr(const char \*str, char c);

### malloc.h(stdlib.h)

void \*malloc(int size);

void free(void \*p);

### stdio.h

int putchar(int c);

int printf(char \*format, ...);

int scanf(const char \*format, ...);

int getchar();

int puts(const char \*str);

char \*gets(char \*str);

### ctype.h

int isspace(char c);

int isdigit(char c);

### stdlib.h

void exit(int status);

# 后面要做的事

### 压缩算法

### API函数标准化(标准函数)