

}_attribute_((packed)) BITMAPFILEHEADER;

//_attribute_((packed))的作用是告诉编译器取消结构在编译过程中的优化对齐

1赞

赞赏

更多好文

登录

注册

/ 写:

```
//40byte信息头
typedef struct
charciSize[4];//BITMAPFILEHEADER所占的字节数
int ciWidth;//宽度
int ciHeight;//高度
charciPlanes[2];//目标设备的位平面数, 值为1
int ciBitCount;//每个像素的位数
charciCompress[4];//压缩说明
charciSizeImage[4];//用字节表示的图像大小, 该数据必须是4的倍数
charciXPelsPerMeter[4];//目标设备的水平像素数/米
charciYPelsPerMeter[4];//目标设备的垂直像素数/米
charciClrUsed[4]; //位图使用调色板的颜色数
charciClrImportant[4]; //指定重要的颜色数, 当该域的值等于颜色数时(或者等于0时), 表示所有
颜色都一样重要
}_attribute_((packed)) BITMAPINFOHEADER;
static void FbUpdate(int fbfd, struct fb_var_screeninfo *vi) //将要渲染的图形缓冲区的内容绘制
到设备显示屏来
vi->yoffset=0; //from 0 to yres will be display.
ioctl(fbfd, FBIOPUT_VSCREENINFO, vi);
static int CursorBitmapFormatConvert(char *dst, char *src, int screenXres, int screenYres, int
bytes_per_pixel_screen,
intbmpWidth, intbmpHeight, intbytes_per_pixel_bitmap)
int i ,j ;
char*psrc = src;
char*pdst = dst;
```

```
char*p = psrc;
int oldi =0;
int left_right = (screenXres-bmpWidth)/2;
int top_bottom = (screenYres-bmpHeight)/2;
printf("left_right: %d, top_bottom: %d\n", left_right, top_bottom);
printf("screenXres: %d, screenYres: %d\n", screenXres, screenYres);
printf("bmpWidth: %d, bmpHeight: %d\n", bmpWidth, bmpHeight);
if(left_right <0&& top_bottom <0){
printf("error: BMP is too big, screen X: %d, Y: %d, bmp width: %d, height: %d\n", screenXres,
screenYres, bmpWidth, bmpHeight);
return-1;
/* 由于bmp存储是从后面往前面, 所以需要倒序进行转换 */
pdst += (screenXres * bmpHeight * bytes_per_pixel_screen);
pdst += (screenXres * top_bottom * bytes_per_pixel_screen); //move to middle bmpHeight
for(i=0; i<bmpHeight; i++){
p = psrc + (i+1) * bmpWidth * bytes_per_pixel_bitmap;
if(i == 0)
pdst -= (left_right * bytes_per_pixel_screen); //move to middle bmpWidth
else
pdst -= (left_right * 2 * bytes_per_pixel_screen); //move to middle bmpWidth
for(j=0; j<bmpWidth; j++){
pdst -= bytes_per_pixel_screen;
p -= bytes_per_pixel_bitmap;
int kk = 0;
int k;
for(k=0; k<bytes_per_pixel_screen; k++){
if(kk >= bytes_per_pixel_bitmap)
pdst[k] =255; // 3: alpha color
```

```
else
pdst[k] =p[kk]; // 0: blue, 1: green, 2: red
kk++;
pdst -= (left_right * bytes_per_pixel_screen); // move to 0 position
return0;
int ShowBmp(char *path, int fbfd, struct fb_var_screeninfo *vinfo, char *fbp)
int i;
FILE *fp;
int rc;
int line_x, line_y;
longint location =0, BytesPerLine =0;
char*bmp_buf =NULL;
char* buf = NULL;
int flen =0;
int ret =-1;
int total_length =0;
BITMAPFILEHEADER FileHead;
BITMAPINFOHEADER InfoHead;
int width, height;
printf("into ShowBmp function\n");
if(path ==NULL) {
printf("path Error,return\n");
return-1;
```

```
printf("path = %s\n", path);
fp = fopen( path, "rb" );
if(fp == NULL){
printf("load cursor file open failed\n");
return-1;
/* 求解文件长度 */
fseek(fp,0,SEEK_SET);
fseek(fp,0,SEEK_END);
flen = ftell(fp);
printf("flen is %d\n",flen);
bmp_buf = (char^*)calloc(1,flen - 54);
if(bmp_buf == NULL){
printf("load > malloc bmp out of memory!\n");
return-1;
/* 再移位到文件头部 */
fseek(fp,0,SEEK_SET);
rc = fread(&FileHead, sizeof(BITMAPFILEHEADER),1, fp);
if ( rc !=1) {
printf("read header error!\n");
fclose(fp);
return( -2 );
//检测是否是bmp图像
if (memcmp(FileHead.cfType, "BM", 2) !=0) {
printf("it's not a BMP file\n");
```

```
fclose(fp);
return( -3 );
rc = fread( (char *)&InfoHead, sizeof(BITMAPINFOHEADER),1, fp );
if ( rc !=1) {
printf("read infoheader error!\n");
fclose(fp);
return( -4 );
width = InfoHead.ciWidth;
height = InfoHead.ciHeight;
printf("FileHead.cfSize =%d byte\n",FileHead.cfSize);
printf("flen = %d\n", flen);
printf("width = %d, height = %d\n", width, height);
//跳转的数据区
fseek(fp, FileHead.cfoffBits, SEEK_SET);
printf(" FileHead.cfoffBits = %d\n", FileHead.cfoffBits);
printf(" InfoHead.ciBitCount = %d\n", InfoHead.ciBitCount);
int bytes_per_pixel_bitmap =InfoHead.ciBitCount/8;
total_length = width * height * bytes_per_pixel_bitmap;
printf("total_length = %d\n", total_length);
//每行字节数
buf = bmp_buf;
while ((ret =fread(buf,1,total_length,fp)) >=0) {
if (ret ==0) {
usleep(100);
continue;
```

```
printf("ret = %d\n", ret);
buf = ((char*) buf) + ret;
total_length = total_length - ret;
if(total_length ==0)
break;
CursorBitmapFormatConvert(fbp, bmp_buf, (*vinfo).xres, (*vinfo).yres, (*vinfo).bits_per_pixel/8,
width, height, bytes_per_pixel_bitmap);
free(bmp_buf);
fclose(fp);
printf("show logo return 0\n");
return0;
int ShowPicture(int fbfd, char *path)
struct fb_var_screeninfo vinfo;
struct fb_fix_screeninfo finfo;
longint screensize =0;
struct fb_bitfield red;
struct fb_bitfield green;
struct fb_bitfield blue;
//打开显示设备
printf("fbfd = %d\n", fbfd);
if (fbfd ==-1) {
//printf("Error opening frame buffer errno=%d (%s)\n",errno, strerror(errno));
return-1;
if (ioctl(fbfd, FBIOGET_FSCREENINFO, &finfo)) {
//printf("Error: reading fixed information.\n");
```

```
return-1;
if (ioctl(fbfd, FBIOGET_VSCREENINFO, &vinfo)) {
//printf("Error: reading variable information.\n");
return-1;
//printf("R:%x;G:%d;B:%d\n", (int)vinfo.red, vinfo.green, vinfo.blue);
printf("%dx%d, %dbpp\n", vinfo.xres, vinfo.yres, vinfo.bits_per_pixel );
//计算屏幕的总大小(字节)
screensize = vinfo.xres * vinfo.yres * vinfo.bits_per_pixel / 8;
printf("screensize=%ld byte\n",screensize);
//对象映射
char*fbp = (char*)mmap(0, screensize, PROT_READ | PROT_WRITE, MAP_SHARED, fbfd, 0);
if (fbp == (char*)-1) {
printf("Error: failed to map framebuffer device to memory.\n");
return-1;
printf("sizeof file header=%ld\n", sizeof(BITMAPFILEHEADER));
//显示图像
ShowBmp(path, fbfd, &vinfo, fbp);
FbUpdate(fbfd, &vinfo);
///在屏幕上显示多久
sleep(10);
//删除对象映射
munmap(fbp, screensize);
return0;
int main()
```

```
{
int fbfd =0;
fbfd = open("/dev/fb0", O_RDWR);
if (!fbfd) {
printf("Error: cannot open framebuffer device.\n");
return-1;
ShowPicture(fbfd, "/tmp/license_ckx.bmp");
close(fbfd);
return0;
 1人点赞>
Linux
```

更多精彩内容, 就在简书APP

"小礼物走一走,来简书关注我"

赞赏支持 还没有人赞赏, 支持一下



天楚锐齿物联网、云计算、大数据、通信、IT、嵌入式
<a href="http://ww... 总资产3 共写了5.8W字 获得105个赞 共69个粉丝

关注

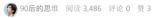
写下你的评论...

全部评论 0 只看作者 按时间倒序 按时间正序

推荐阅读更多精彩内容》

iOS ffmpeg 理解

教程一:视频截图(Tutorial 01: Making Screencaps) 首先我们需要了解视频文件的一些基...



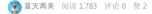
ffmpeg 命令大全

超高速音视频编码器用法: ffmpeg [options] [[infile options] -i infile...



2018-06-14

在C语言中,五种基本数据类型存储空间长度的排列顺序是: A)char B)char=int<=float C)ch...



视音频数据处理入门:RGB、YUV像素数据处理

前一阵子在梳理以前文章的时候,发现自己虽然总结了各种视音频应用程序,却还缺少一个适合无视音频背景 人员学习的"最基础…





Day 345:最重要的东西, 往往是看不到的

最重要的东西,往往是看不到的。如果我们可以洞察事物或者现象背后这些看不到的东西,那么一定可以有所为。很多初次见面...





中国金融数据月度报表-2021年08月

阅读 99

中国金融数据月度报表-2021年09月

阅读 33

中国金融数据月度报表-2021年07月

阅读 74

写下你的评论...



...