**一、vc编译jpeglib库**

**1、下载源代码**

**下载地址：http://www.ijg.org/。注意：一定要下载win32 版本**

**2、编译源代码.**

**A、解压源代码，修改源代码中jconfig.vc为jconfig.h；**

**B、添加环境变量PATH，C:/Program Files/Microsoft Visual Studio/VC98/Bin ；**

**C、修改makefile.vc，将 Pull in standard variable definitions下面的一行换为：!include <C:/Program Files/Microsoft Visual Studio/VC98/Include/win32.mak> ；**

**D、进入命令提示环境下，输入：vcvars32 回车，这是一个用来设置VC路径环境的批处理；**

**E、编译生成库文件 命令：nmake /f makefile.vc nodebug=1；**

**这样就OK了**

**3、jpeglib库VC下使用**

**对于库的使用只需要有相应的.lib文件和头文件就可以了。Vc中要添加libjpeg.lib库的连接**

**将这几个文件拷贝到你的项目中：libjpeg.lib，jconfig.h，jmorecfg.h，jpeglib.h，在你需要进行压缩的文件中加入**

**extern "C" {**

**#include "jpeglib.h"**

**#include "jmorecfg.h"**

**#include "jconfig.h"**

**}**

**参考：**

[**http://blog.csdn.net/xingyu19871124/archive/2009/06/30/4310800.aspx**](http://blog.csdn.net/xingyu19871124/archive/2009/06/30/4310800.aspx)

**小知识：bmp文件的前54个字节是头，后面才是像素值。**

**二、使用jpeg库压缩**

**在源代码的文件夹下面有一个example.c的文件和libjpeg.txt，很有参考价值。**

**1、基本思路**

**首先调用截屏程序，将屏幕的位图信息存放在一个buffer里面，然后调用jpg压缩函数，在当前的目录下生成一个ok.jpg的文件。**

**2、出现的问题：**

**A、运行是总是报错：**

**我参考源代码的例子，也用JSAMPLE \* image\_buffer;来指向位图的像素的首地址，编译可以通过但是运行时就会报错，后来我用BYTE \*image\_buffer;来定义就可以正常运行了。**

**B、生成的jpg图像失真：**

**由于window的位图的像素格式是：BGRA，4个字节，jpeglib库使用的是RGB，3个字节的格式，所以需要将源像素去掉其透明字节，同时改变RGB的顺序。代码如下：**

**//RGB顺序调整**

**for (int i=0, int j=0; j < 1440\*900\*4; i+=3, j+=4)**

**{**

**\*(image\_buffer+i)=\*(image\_buffer+j+2);**

**\*(image\_buffer+i+1)=\*(image\_buffer+j+1);**

**\*(image\_buffer+i+2)=\*(image\_buffer+j);**

**}**

**C、生成的jpg文件图像是倒的：**

**原因没有找到，后来修改了压缩函数的代码，生成了正确的jpg文件**

**while (cinfo.next\_scanline < cinfo.image\_height) {**

**//这里我做过修改，由于jpg文件的图像是倒的，所以改了一下读的顺序**

**//这是原代码：row\_pointer[0] = & bits[cinfo.next\_scanline \* row\_stride];**

**row\_pointer[0] = & bits[(cinfo.image\_height - cinfo.next\_scanline - 1) \* row\_stride];**

**(void) jpeg\_write\_scanlines(&cinfo, row\_pointer, 1);**

**}**

**3、测试结果：**

**我编写了测试代码，连续截屏并生成jpg文件100次，大约花费7秒左右，也就是说1秒可以截屏13次左右。同时生成的jpg文件有100多K的样子。**

**三、代码：**

**#include "stdafx.h"  
#include <atlbase.h>  
#include <afxwin.h>  
#include <WINDOWSX.H>  
   
#define JPEG\_QUALITY 50     //它的大小决定jpg的质量好坏  
   
extern "C" {  
    #include "jpeglib.h"  
    #include "jmorecfg.h"  
    #include "jconfig.h"  
}  
   
int savejpeg(char \*filename, unsigned char \*bits, int width, int height, int depth);  
void CapScreen(char filename[]);  
   
BYTE \*image\_buffer; //指向位图buffer的全局指针，window下像素格式: BGRA(4个字节)  
   
int main(int argc, char\* argv[])  
{  
   
    image\_buffer = (BYTE \*)malloc(1440 \* 900 \* 4);  
   
    for(int i = 0; i < 100; i++){  
        CapScreen("ok.bmp");     
        //RGB顺序调整  
        for (int i=0, int j=0; j < 1440\*900\*4; i+=3, j+=4)  
        {  
            \*(image\_buffer+i)=\*(image\_buffer+j+2);  
            \*(image\_buffer+i+1)=\*(image\_buffer+j+1);  
            \*(image\_buffer+i+2)=\*(image\_buffer+j);  
        }  
        savejpeg("ok.jpg", image\_buffer, 1440, 900, 3);  
    }  
   
    delete [] image\_buffer;  
    return 0;  
}  
   
   
/\*===================================================================================  
function:       jpeg压缩  
input:          1:生成的文件名,2:bmp的指针,3:位图宽度,4:位图高度,5:颜色深度  
return:         int  
description:    bmp的像素格式为(RGB)  
===================================================================================\*/  
int savejpeg(char \*filename, unsigned char \*bits, int width, int height, int depth)  
{  
        struct jpeg\_compress\_struct cinfo;  
        struct jpeg\_error\_mgr jerr;  
        FILE \* outfile;                 /\* target file \*/  
        JSAMPROW row\_pointer[1];        /\* pointer to JSAMPLE row[s] \*/  
        int     row\_stride;             /\* physical row width in image buffer \*/  
   
        cinfo.err = jpeg\_std\_error(&jerr);  
        jpeg\_create\_compress(&cinfo);  
   
        if ((outfile = fopen(filename, "wb")) == NULL) {  
                fprintf(stderr, "can't open %s/n", filename);  
                return -1;  
        }  
        jpeg\_stdio\_dest(&cinfo, outfile);  
   
        cinfo.image\_width = width;      /\* image width and height, in pixels \*/  
        cinfo.image\_height = height;  
        cinfo.input\_components = 3;         /\* # of color components per pixel \*/  
        cinfo.in\_color\_space = JCS\_RGB;         /\* colorspace of input image \*/  
   
        jpeg\_set\_defaults(&cinfo);  
        jpeg\_set\_quality(&cinfo, JPEG\_QUALITY, TRUE /\* limit to baseline-JPEG values \*/);  
   
        jpeg\_start\_compress(&cinfo, TRUE);  
   
        row\_stride = width \* depth; /\* JSAMPLEs per row in image\_buffer \*/  
   
        while (cinfo.next\_scanline < cinfo.image\_height) {  
            //这里我做过修改，由于jpg文件的图像是倒的，所以改了一下读的顺序  
            //这是原代码：row\_pointer[0] = & bits[cinfo.next\_scanline \* row\_stride];  
            row\_pointer[0] = & bits[(cinfo.image\_height - cinfo.next\_scanline - 1) \* row\_stride];  
            (void) jpeg\_write\_scanlines(&cinfo, row\_pointer, 1);  
        }  
   
        jpeg\_finish\_compress(&cinfo);  
        fclose(outfile);  
   
        jpeg\_destroy\_compress(&cinfo);  
        return 0;  
}  
   
   
void CapScreen(char filename[])  
{  
    CDC \*pDC;  
    pDC = CDC::FromHandle(GetDC(GetDesktopWindow()));  
    if(pDC == NULL) return;  
    int BitPerPixel = pDC->GetDeviceCaps(BITSPIXEL);  
    int Width = pDC->GetDeviceCaps(HORZRES);  
    int Height = pDC->GetDeviceCaps(VERTRES);  
   
    CDC memDC;  
    if(memDC.CreateCompatibleDC(pDC) == 0) return;  
     
    CBitmap memBitmap, \*oldmemBitmap;  
    if(memBitmap.CreateCompatibleBitmap(pDC, Width, Height) == NULL) return;  
   
    oldmemBitmap = memDC.SelectObject(&memBitmap);  
    if(oldmemBitmap == NULL) return;  
    if(memDC.BitBlt(0, 0, Width, Height, pDC, 0, 0, SRCCOPY) == 0) return;  
   
    BITMAP bmp;  
    memBitmap.GetBitmap(&bmp);  
     
    //fp = fopen(filename, "w+b");  
   
    BITMAPINFOHEADER bih = {0};  
    bih.biBitCount = bmp.bmBitsPixel;  
    bih.biCompression = BI\_RGB;  
    bih.biHeight = bmp.bmHeight;  
    bih.biPlanes = 1;  
    bih.biSize = sizeof(BITMAPINFOHEADER);  
    bih.biSizeImage = bmp.bmWidthBytes \* bmp.bmHeight;  
    bih.biWidth = bmp.bmWidth;  
     
    BITMAPFILEHEADER bfh = {0};  
    bfh.bfOffBits = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER);  
    bfh.bfSize = bfh.bfOffBits + bmp.bmWidthBytes \* bmp.bmHeight;  
    bfh.bfType = (WORD)0x4d42;  
     
    //fwrite(&bfh, 1, sizeof(BITMAPFILEHEADER), fp);  
    //fwrite(&bih, 1, sizeof(BITMAPINFOHEADER), fp);  
     
    image\_buffer = new BYTE[bmp.bmWidthBytes \* bmp.bmHeight];  
   
    GetDIBits(memDC.m\_hDC,  
        (HBITMAP) memBitmap.m\_hObject,  
     0,  
     Height,  
     image\_buffer,  
     (LPBITMAPINFO) &bih,  
     DIB\_RGB\_COLORS);  
    memDC.SelectObject(oldmemBitmap);  
    //fwrite(p, 1, 1280 \* 800 \* 4, fp);  
    //fclose(fp);  
}**