Game-Basic 系统

<VmBeta & VmBasic>

设计规划与技术手册

Linwei 2002

电子科技大学通讯工程学院

20013080 班林伟

二〇〇二年九月

总 体 说 明

（一） 主要思想：

1、 内容紧扣名称，内容围绕“VirtualMachine(虚拟机系统)”与“游戏”(针对 Windows/Unix 不同平台独立性）展开。

2、 本软件不再以 Game 或 Basic 或 VirtualMachine 的名义进行宣传和版权说明等，用“VmBasic”取而代之。这样才能把“VmBasic”的品牌创出来，体现出

3、 软件由一年前完成的虚拟机系统 VmBeta 和游戏编程语言 VmBasic 完美的结合起来，利用虚拟机的与平台无关性，容错性，高效性，安全性。使得 VmBasic 编译出来的软件能基于虚拟机的技术，在不同的平台上面发挥出最大的功效。

4、 实现类似消息机制的灵活扩展接口,使得 VmBasic 可以与你的软件完美的结合起来，成为一套灵活的脚本引擎，使你的软件具有运行脚本的能力。同时对于语言本身，实现了预连接技术，也可以使用户方便的为语言增加功能。

5、 全部核心代码用标准 C++写成，分别在 Windows/Unix 下数款 C++编译器上面调试通过，充分保证了软件的与平台无关性。从 2002 年 9 月开始，对外提供 VmBeta 与 VmBasic 引擎，如有需要者请 mail 一份到 lwwind@yeah.net

（二） 制作时间：

2001 年 11 月-2002 年 1 月制作虚拟机系统：VmBeta

2002 年 4 月-2002 年 8 月制作游戏编程语言：VmBasic

（三） 系统发展概况：

1、 起初，我由于平时编程常常居于不同的平台，而且在所设计的几个软件中都迫切的需要与平台无关的脚本机制的支持来提高软件的灵活性与可维护性。著名的 Autocad/ Animator/ 3D Max 以及金蝶财务软件中都提供脚本功能来让用户更灵活地操作软件，但是他们的脚本机制采用的都是相对独立封闭的技术，要让它们扩充一项脚本功能则需要每个软件都改写大量代码，而且效率安全方面也不健全。

2、 初期，我开始意识到要解决这些问题，使得今后的系统中采用统一的脚本引擎并且统一维护，灵活扩展只有采用 VirtualMachine(虚拟机)技术，于是在去年年底，我花了大量的时间研制成功了一套新的虚拟机系统 VmBeta，它参照了一款小型栈式 CPU，制定了自身一系列 32 位寄存器和指令集合实现了与平台无关性。历经三个月 VmBeta 系统的研制成功对我后来的开发工作起到了制关重要的作用。

3、 中期，在 VmBeta 的设计成果之上，软件的脚本支持问题得到了初步的解决，但是脚本的编写都是基于 VmBeta 的汇编指令上面来完成的，编程效率极其低下。我意识到要使它真正起作用的话，必须为 VmBeta 编写一款高级语言编译器。就象 Java 和 Javac 一般，这是项艰巨的工程，凭着以前我写过几个脚本引擎，研究了大量的编译方面知识终于写出了一款编译器 VmBasic。本来编译器想选择 Pascal 为蓝本，可是我觉得 Basic 更能让人接受，仅仅写成传统的 Basic 很不够，于是我陆续为其

增加了许多灵活的功能，整个虚拟机-脚本引擎完成。

4、 现在，处于我自己的兴趣爱好，我喜欢编写游戏，而且是从 DOS 下直接操作 VGA 显存作图，操作声卡发声一直到现在 Windows 下利用 DirectX 和 OpenGL 编写游戏，我深感游戏设计之困难程度非凡，许多朋友渴望自己制作游戏却被一道道高高的门槛隔与局外，于是我利用 VmBeta 的扩充性将 VmBasic 扩充了一系列多媒体操作指令以及游戏常用指令，使得刚入门的朋友们只需掌握简单的 Basic 语句就可以编写一定规模的游戏。

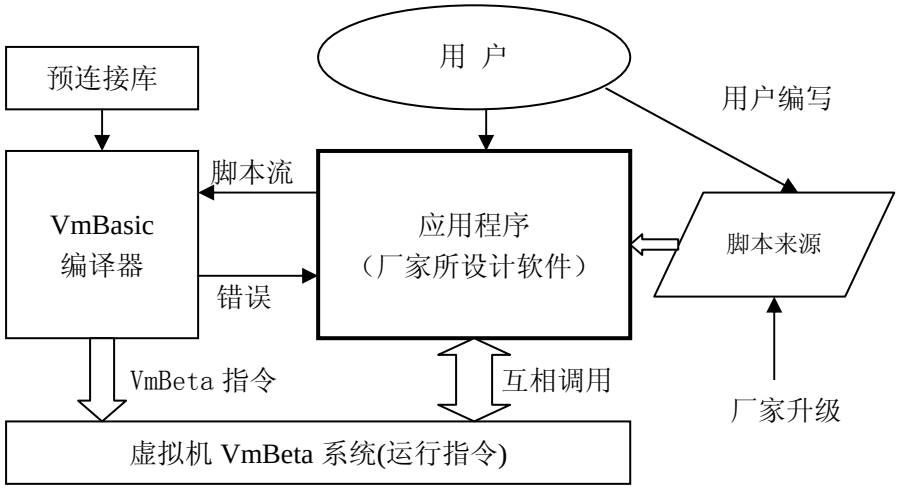
5、 今后，由于现在 VmBeta 是一款高效的虚拟机而 VmBasic 且用途是一个脚本引擎。所以将继续从简单性和可扩充性着手。一来基于 VmBeta 的框架做扩充可以变为某种 CPU 的模拟器备将来研究开发之用，二来现在的由于其自身特点，在许多领域都可以发挥重要用途。

（四） 联系方式：

林伟：电子科技大学通讯工程学院 20013080 班
电话：83200790, 191-8094070
信箱：lwwind@yeah.net skywindt@yeah.net

GameBasic 规 划 说 明

（一） 构架方案：



其基本运行模式如上图所示，按照面向性对象的思想，不同层次之间的通讯采用消息机制最为灵活，比如 Windows 的消息机制。所以 VmBeta 对外部的扩展选择了消息机制，就是那个“互相调用”箭头所表示的含义。在厂家制定了一系列扩展模块以适应自身软件需要的同时，这些功能在 VmBasic 脚本里面并没有体现出来，因此还要为 VmBasic 来编写预连接库，来定义一些关于新功能的函数、过程、常量、变量等等，才算完成。脚本调用预连接库时也就实现了新功能的调用。

（二） 游戏 Basic 的具体规则：

首先游戏 Basic 为 VmBeta 提供了一系列 Windows 多媒体功能，键盘鼠标控制功能，本来打算使用 DirectX 来控制作图，但是考虑到 DirectDraw 于界面程序不好配合，且当时研究 GDI32 作图甚有心得，于是图形全面采用 GDI 来控制。

接着编写了预连接库 StdLib.VBL 为上述功能提供了脚本编写接口，于是整个扩展得以完成，实际上游戏 Basic 也就是 VmBeta 与 VmBasic 的一个具体应用实例。

整个游戏 Basic 的组成文件如下：

文 件 名	功 能 说 明
VmBeta.exe	虚拟机运行文件(For Windows)
VmBIDE.exe	游戏 Basic 开发环境(集成了 VmBasic)
StdLib.VBL	预连接库(可以用文本编辑器查看)

（三） 系统的性能比较：

虽然 VmBasic 编译器并没有做代码优化，而且经常生成一些机械多余的代码影响了速度，但是经过和 DOS 下自带的 QBASIC 进行测试，同样的几个算法程序，这套系统要比 QBASIC 快的多，达到了 3/4 的程度，这也得归功于虚拟机系统的高效性。

VmBeta 规 划 说 明

（一） 虚拟机 Beta 的指令集及 I/O 说明：

1. 虚拟机模拟栈式 CPU 一共有八个寄存器：

   RP:指令寻址寄存　　RF:标志寄存器　　R0,R1,R2,R3:通用寄存器

   RS:栈寄存器　　RB:辅助栈寄存器

   标志寄存器 RF 的第 0 位到第 3 位分别为 FZ(零标志) FB(小于标志) FA(大于)当用 CMP 指令比较两个变量时比较的结果会更改三个标志，在用条件跳转指令 JPC 进行跳转时会根据 JPC 所给的条件查询三个标志进行条件跳转

2. 虚拟机指令有 JMP(无条件跳转)JPC(条件跳转)CALL(调用)RET(返回)LD(数据装载)CAL(数据运算)CMP(数据比较)PUSH(压栈)POP(弹栈)NOOP(空指令)IN(输入指令)OUT(输出指令)EXIT(推出指令)等 13 条指令支持类型有 DWORD(32bit) / WORD(16b) /BYTE(8b) / INT(32b) / FLOAT(32b)。

3. 基本 I/O：

   输入:INPUT

   00.转化 FLOAT 类型的 R3 数据到 INT 类型
   01.转化 INT 类型的 R3 数据到 FLOAT 类型
   02.分配字符串，返回句柄
   03.转化 R3 的字符串到 INT 类型并且返回
   04.将 R3 的 INT 数字转化成 R2 所指字符串
   05.将 R3 的字符串设置成 R2 的字符串
   06.将 R3 的字符串末尾添加 R2 的字符串
   07.求 R3 字符串的长度
   08.释放 R3 的字符串
   09.比较 R3,R2 两个字符串
   0A.将 R3 的 FLOAT 类型转换成 R2 所指字符串
   0B.将 R3 字符串转换成 FLOAT 类型并返回
   0C.返回 R3 字符串的第 R2 个字符
   0D.设置 R3 字符串第 R2 个字符成 R1
   0E.返回版本号前 16 个字节是主版本号，后 16 个字节是附版本号
   0F.返回一个 1KHz 的时钟值，这个时钟从程序开始运行便启动
   10.返回 sin((float)R3)
   11.返回 cos((float)R3)
   12.返回 tan((float)R3)
   13.返回 sqrt((float)R3)
   14.返回 abs((int)R3)
   15.返回 abs((float)R3)

   输出:OUTPUT

   00.输出带'\n'的整数
   01.输出带'\n'的字符串
   02.输出不带'\n'的字符串
   03.输出不带'\n'的整数
   04.输出单个字符

05.输出不带'\n'的浮点
0A.输入整数到 R3
0B.输入字符串到 R3
0C.输入浮点数到 R3

（二） 虚拟机的设计和工作原理：

1. 本工作原理：虚拟机的基本工作原理很简单，从内存中顺序取出虚拟机指令的字节码然后作出相应的操作
2. 基本的框架设定 IVMInstance 虚拟机的定义是在 ivms.h 里面，虚拟机概念被定义为实例 HINSTANCE 叫做 IVMInstance，这个类是纯虚类，没有任何实现，只是提供了基本接口。类 IVMInstance 里面定义了虚拟机的寄存器，内存空间，空间大小，还有就是虚函数的接口:Process,VmIn,VmOut,VmError 这四个成员函数是虚拟机实现之关键，Process 是从虚拟机空间的当前位置解释执行一条虚拟机命令，然后返回，这样以一条语句为单位的基本定义是为了方便你在派生类中加入跟踪调试，断点等其他功能。VmIn 是虚拟机的输入函数，VmOut 是输出函数，VmError 是错误处理函数。
3. 具体的实现类 IVMBeta，IVMInstance 被设计成纯虚函数，起一个框架作用。具体实现的一个虚拟机被命名为 VmBeta 在 ivmsbeta.h 里面得到了定义，通过对 IVMInstance 的继承，重载并且实现了 Process/VmIn/VmOut/VmError 并且加入了大量其他功能。所以说真正的指令解释是在 IVMBeta 得到实现的。具体的解释方法也并没有什么特殊的地方，可以参考 ivmsbeta.cpp 里面的实现。虚拟机的指令里面有 IN/OUT 两条指令控制输出输入，所谓向端口输入，输出解释程序并不做什么而是调用 VmIn/VmOut 来处理发生的输出输入操作，这样通过处理函数的继承，重载实现了虚拟机功能的扩充，因此有点象 PC 系列里面的中断调用功能而已。
4. 虚拟机文件的读取:在 IVMBeta 里面有方法 Create 实现了初始化化操作(及初始化虚拟机内存 VirtualMem)。如果有数据来源则同时拷贝数据到 VirtualMem 中，但是这样之可以装载二进制的数据，因此有了 ivmca.h 和 ivmca.cpp 来实现读取 VMS 文件的方法，VMS 文件中保存的是文本状态的虚拟机汇编指令，类 ivmsca.h 中 IVMBetaAsm 实现读取这些汇编指令并且解释成二进制的代码方便运行，在以后虚拟机的读取并不应该直接读文本状态的汇编指令,而要读取二进制的数据并且实现动态连接，这点由于时间关系，我有去实现。

（三） 对虚拟机进行扩展：

之所以设计了虚拟机，就是为了让它扩展到各个方面，应用层比如某款软件的脚本系统，对它的扩展在于继承类 IVMInstance 或者 IVMBeta 然后重载 VmIn/VmOut 前面提到：IN/OUT 两条指令控制输出输入，所谓向端口输入，输出解释程序并不做什么而是调用 VmIn/VmOut 来处理发生的输出输入操作，这样通过处理函数的继承，重载实现了虚拟机功能的扩充。而 VmIn 里面定义的功能一般是虚拟机的一些最低级别的基础功能，比如浮点数和整数之间的转化，字符串和整数，浮点数的转化 sin/cos/sqrt 等常用数学功能，在上面已经提到过都是一些为语言服务的功能，并不和扩展应用和运行平台有关系，意思就是扩充时可以不必重载。而 VmOut 里面定义的东西则大部分是和平台相关的东西了，比如输出打印字符串，浮点，整数，还有输入等等。那么对于某项针对具体应用的功能，最好也扩展在输出部分。

（四） VmBeta 例子程序,:

一 段 求 解 阶 乘 的 程 序

```
; demo.vms

.stack 8192                ; 设定栈大小

start:                     ; 程序入口
  out 2,string1            ; 输出"This program is use to.."
  out 10,0                 ; 等待输入整数到 r3
  ld dword r0,r3           ; 将 r3 复制到 r0
  cmp int r0,1             ; 比较 r0 和 1
  jpc b error              ; 如果小于则错误(因为阶乘要>=1)
  push r0                  ; 将 r0 压栈
  call fun1                ; 调用 fun1 进行递归求解
  cal dword add rs,4       ; 调用完成后要复位栈指针
  out 2,string2            ; 输出"the result is "
  out 0,r0                 ; 输出储存于 r0 中的结果
  exit                     ; 退出程序
error:                     ;
  out 0,0                  ; 输出 0
  exit
fun1:                      ; 计算阶乘的递归模块
  push rb                  ; 保存辅助寄存器
  ld dword rb,rs           ; 设置辅助寄存器
  ld dword r0,rb           ; 下面三行是读取栈中的参数
  cal dword add r0,12      ;
  ld dword r0,[r0]         ;
  cmp int r0,1             ; 如果参数 n 大于 1 则到 jmp1 处否则 jmp2 处
  jpc a jmp1               ;
  jmp jmp2                 ;
  jmp1:                    ; 将参数 n 减一后压栈，并递归调用自己
  push r0                  ;
  cal int sub r0,1         ;
  push r0                  ;
  call fun1                ;
  cal dword add rs,4       ;
  ld dword r1,r0           ;
  pop r0           ;
  cal int mul r0,r1        ; 将 n 和得出的(n-1)相乘
  jmp2:                    ; 如果 n 小于等于 1 则直接返回 1
  ld dword rs,rb           ; 还原栈地址
  pop rb                   ; 还原辅助寄存器
  ret                      ; 返回
data string1 byte "This program is use to work out n!=?",$a,"please input n=",0 ;
```

```
e6
data string2 byte "the result is ",0 ; 11f
end
```

VmBasic 概 述

（一） 设计思路：

我开始书写 VmBasic 代码的时候碰到了许多的困难，但是编写一款编译器一直是我多年的梦想，于是我坚持了下来。由于我自己的原创情绪，所以没有使用任何自展开技术完成代码，诸如 FLEX 以及 YACC 等等，所以从词法分析到语法分析的大量代码都是手工完成编写的。随着我长时间的学习与不停修改调试，VmBasic 现在已经是一款很强大的脚本编译器了。

主要的原理仍然是采用传统的递归下降技术，将语句分为许多模块，模块中又相互包含或者包含自身，编译时碰到某模块就调用某模块的分析程序，而大部分情况是模块中含有模块。

比如 K=X[GetIndex(Y+1)-1]一段程序：赋值模块里有数组模块，数组模块里有表达式分析模块，表达式分析模块中有函数调用模块，函数调用模块中又有表达式分析模块。所以分析程序进行大量的互相调用与自身调用，直到程序结束，也就是递归下降。同样也是 VmBasic 编译引擎依照的思想之一。

（二） VmBasic 语法说明：

支持整型 X,字符串型 X$和浮点型 X!等三种运算，实现了数组，支持供函数和过程，行号并没有得到支持，BASIC 语法版本很多大概语法和 DOS 下的 QBASIC 那个版本差不多，实现的语句有 IF, GOTO, GOSUB, RETURN, SUB, FUNCTION, PRINT, INPUT, THEN, ELSE, DIM, WHILE, WEND。注意，变量 X 代表整型而 Qbasic 里面代表浮点变量，另外不支持双精度，所有浮点都是单精度的 X! 这个版本没有做 FOR...NEXT 语句和 SELECT 语句，可以先用 WHILE 和 IF 还有 GOTO 代替。

VmBasic 的扩展：字符串可以无限长，是虚拟机做特殊支持的。支持灵活的类型转换，直接用等于号就可以了，比如：

```
B$="100"+5
A=B$
B$=A+5
```

结果是 A=1005,B$="1010"这样比原来的 BASIC 更具有灵活的字符串处理能力

（三） 预连接标准库：

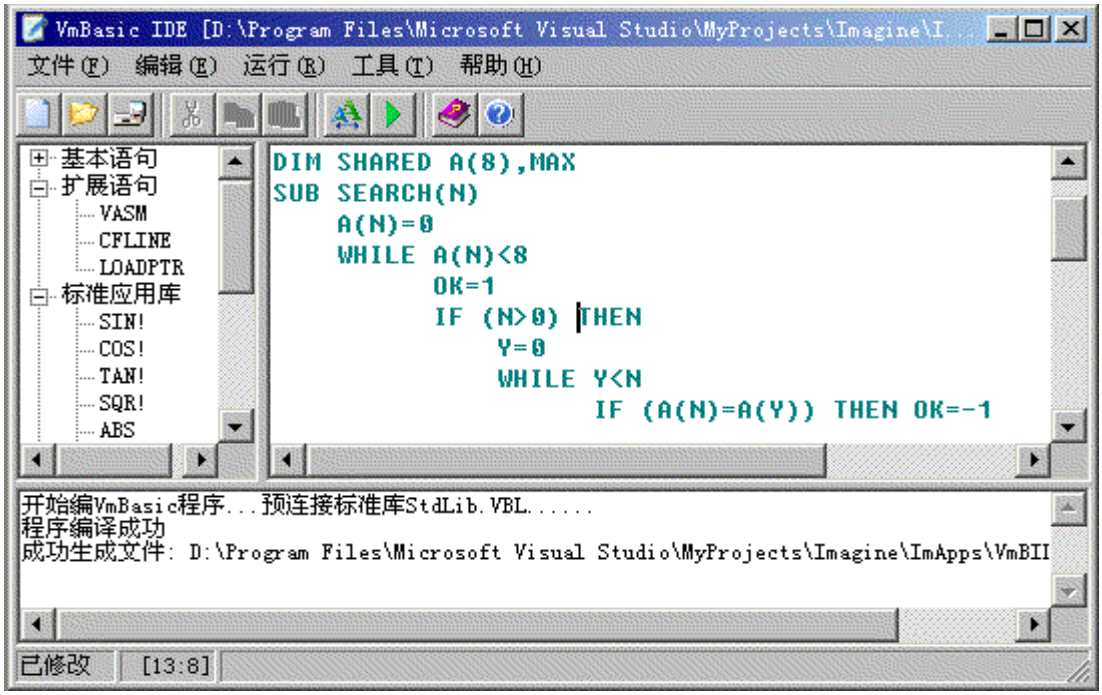
在编译开始默认连接的库，通过它的定义和虚拟机的重载实现语言的升级。默认的预连接库是 StdLib.VBL 里面大部分是一些函数，过程的定义，通过内嵌汇编指令 VASM("")来实现直接控制虚拟机输出输入，目前在 StdLib.VBL 里面定义的函数或过程有：SIN!, COS!, TAN!, SQR!, ABS, ABS!, RANDOM, CHR$等常用命令以外提供了一些多媒体命令(游戏 Basic 主要扩充指令集)：

游 戏 Basic 主 要 语 句	简 要 说 明
FTimer()	取 1ms 为单位的时钟
KeyDown()	判断一个按键是否被按下
CreateCanvas(W,H)	创建一个宽 W 高 H 的画布并且返回一个标识
FreeCanvas(Can)	释放一个画布
LoadCanvas(FILE$)	从 TGA 文件中读取一个画布并且返回标试
Pixel(Can,X,Y,C)	在画布 Can 上面画点
ReadPixel(Can,X,Y)	读取点
FillCanvas(Can,X,Y,W,H,C)	填充画布
BlitCanvas(CanDest,CanSrc,DX,DY,W,H,X,Y,ColorKey)	图块拷贝 ColorKey 小于零则做不透明拷贝
ShowCanvas(Can)	将画布显示到虚拟机窗口上
CloseGraph()	刷新虚拟机窗口，除去画下的痕迹

（四） 其他说明：关于其他信息请查看前面的虚拟机说明，以及“软件总体介绍.HTM”

开 发 环 境 使 用 手 册

首先请运行开发环境 VmBIDE.EXE，其具体界面如下图所示：



上边的图是完整的开发环境左边是语句帮助，中间是代码编写区，下面是编译的错误和过程记录，系统热键说明： 1.F8 编译成 VMS 文件 2.F9 编译并运行程序 3.F1 对 VmBasic 的帮助 4.Shift+F1 帮助 IDE 另外点击运行图表左边的图表可以查看编译出来的虚拟机汇编代码。点击工具目录，可以做一系列设置：虚拟机程序设置，预连接库设置，开发环境设置等，都是简单的东西

上面的图片是运行一个空战游戏代码的截图，文件是 Examples\Fire.BAS

关 于 其 他

半年前在论坛上面看见过一些高手们关于编译的争论，忽然有所感悟，那时刚好写了虚拟机，于是就决定为它写款语言，本来考虑写类 C 或者类 Pascal 的，但是想着 Basic 用起来简单，而且分析起来似乎也简单，后来我才发现虽然没有 C 的编译难写但由于 Basic 经历了长时间的发展，语法变化很大，总的来说没有同意的规范，模块表示也不明确，就连 IF 语句都有好多种版本，所以一个支持函数/过程的 Basic 编译器我觉得比 Pascal 难写的多。目录 DOS 下有 DOS 环境的编译器和虚拟机,可以用来编译运行非扩展的 vmbasic 程序:alex1-4.bas，可以在 IDE 的工具->设置里面设定虚拟机的运行程序。

为什么虚拟机？

在我编程的经验中，很多地方用到了脚本，而且我也写了几个脚本引擎，我迫切的感觉需要一个虚拟机来提供强大的铅入式编程，于是上半年写了一个虚拟机。而且既然要它实现与平台无关的铅入式，我的代码都经过几个 C++编译器的测试：G++/VC++/BCB 等，希望它能不偏不倚。虚拟机指令集是我对某个汇编指令集做了精简设计出来的，模拟一个栈式 CPU 的常用功能，指令里面有 IN/OUT 两条指令控制输出输入，所谓向端口输入，输出解释程序并不做什么而是调用外面的程序来处理发生的输出输入操作，这样通过处理函数的继承，重载消息机制实现了虚拟机功能的扩充，因此有点象 PC 系列里面的中断调用功能而已。

关于脚本引擎：

记得 3DS/MAX 里面实现了一个类似 BASIC 的脚本，Animator 里面实现了一个类 C 的脚本语言，Autodesk 公司的软件对于脚本支持的很出色，因此我也只是希望能对程序起一个很好的扩充功能。我觉得虚拟机的效率不算太低，就是编译程序没有做代码优化，生成许多烦琐的中间代码。通过八皇后程序的速度测试，编译好的 alex3.bas 运行速度大概是 QBASIC 的三四倍左右，先这样吧，等有了时间与闲情再做优化吧~不过如果有了时间，我会首先考虑实现让它可以动态连接。

独立平台性测试：

VmBeta 与 VmBasic 引擎代码经过数个不同平台编译器的顺利测试，而且工作稳定，性能良好

Borland C++ 3.1	DOS
GNU C++ for DOS	DOS
DJGPP for DOS	DOS
GNU C++ for Unix	UNIX
GNU C++ for Linux	Linux
Microsoft Visual C++ 6.0	Windows
Borland C++ Builder 5.0	Windows

特别申明：

编译器 VmBasic 和虚拟机 VmBeta 的所有源程序,文档以及可执行程序,示例文件由 skywind(林伟)skywindt@yeah.net 独立设计完成，你可以任意复制拷贝和在你的刊物、CD、网站上发布，可以任意更改本压缩档里面的程序，也可以用在你的程序中，但请先高知我。如果你对于它有什么好的意见请写信给我，或者到 http://softnb.51.net 或 http://www.joynb.net 留言，

你的建议是对我的最大鼓励，我将继续完善它。如果你对整套系统更为具体的技术感兴趣，请一样与我联系。

各个文件说明：

1. VmBeta.EXE 虚拟机主程序
2. VmBIDE.EXE 编译器和 IDE 程序
3. StdLib.VBL 编译时的标准应用库程序
4. VmBIDE.INI 编译器环境设置文件
5. DOS/ Console 模式的虚拟机和编译器
6. Examples/ 语言的测试文件：
   - alex1.bas 非波拉数列测试程序
   - alex2.bas 矩阵测试程序
   - alex3.bas 八皇后问题的递归解法
   - alex4.bas 八皇后的函数版
   - Fire.bas 多媒体功能测试-一个飞机游戏

以及说明文档若干等等。。。。

编写这套系统是我若干年来的一个梦想，如今实现了。。。。。。

参考文献：

1. 《80x86 汇编语言程序设计教程》 清华大学出版社 杨季文等
2. 《微型计算机原理》电子科技大学出版社 李广军等
3. 《算法与数据结构》电子工业出版社 王晓东等
4. 《Essential C++》Stanley B. Lippman
5. 《编译原理》清华大学出版社
6. 《Windows98/2000 编程实例详解》电子工业出版社 周万宁等
7. 《Win32 程序员指南》微软公司推出，清华大学翻译出版
8. 《高性能 Windows 图形程序设计》机械工业出版社 李国岫
9. 《Graphics Programming Black Book》Michael Abrash
10. 《自学游戏编程 21 日通》北京学苑出版社
11. 《电脑游戏编程技巧大全》北京学苑出版社
12. 《计算机图形学》电子工业出版社 蔡士杰等译
13. 《如何用 Visual C++设计图形,游戏与动画》北京希望出版社
14. 《Linux 程序设计权威指南》机械工业出版社 2001.4
15. 《Linux 参考大全》北京希望电子出版社
16. 《C++ Builder 5 实用教程》中国铁道出版社
17. 《Visual C++ 6 从入门到精通》9787980023175
18. 《再谈 GDI 模式作图》林伟 http://www.joynb.net/maker/regdi.htm
19. 《Implementing A Scripting Engine》

http://www.flipcode.com/tutorials/tut_scr01.shtml
20. 《RPG 脚本之道》林伟 http://www.joynb.net/maker/script.htm

电子科技大学 通讯工程学院 20013080 班
林伟 (Skywind) lwwind@yeah.net skywindt@yeah.net
WEB: http://www.joynb.net/
MSN: skywind3000@hotmail.com
联系方式：028-83200790 宿舍，191-8094070 传呼