JNI 攻略之十－－操作 Java 虚拟机

江苏 无锡 缪小东

本篇和大家讲述一个比较高级的 JNI 的应用！主要讲述在 C 或 C++中启动虚拟机！

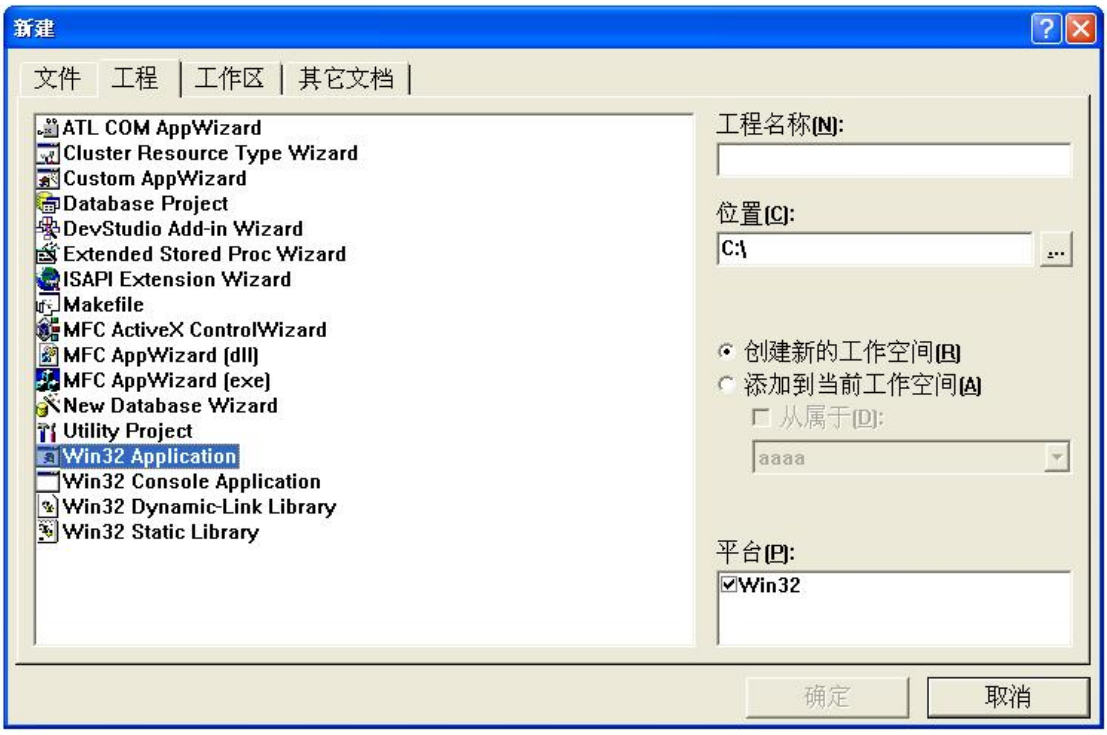
一、启动虚拟机的 C 文件

```
//下面是启动 JDK1.2 后的虚拟机的 c 代码！JDK1.2 后的虚拟机和之前的是不太相同的
//由于现在我们使用的虚拟机一般都是 1.4 甚至更高，所以我们使用 1.2 以上的虚拟机
/* invoke1.2.c */
#include <stdio.h>
#include <jni.h>

int main() {
        int res;                        //启动虚拟机的返回值，关于返回值请查手册
        JavaVM *jvm;                    //虚拟机的指针
        JNIEnv *env;                    //环境指针
        JavaVMInitArgs vm_args;         //虚拟机的启动参数
        JavaVMOption options[3];        //虚拟机的选项
        vm_args.version=JNI_VERSION_1_2;//这个字段必须设置为该值

        /*设置初始化参数*/
        options[0].optionString = "-Djava.compiler=NONE";
        options[1].optionString = "-Djava.class.path=.";
        options[2].optionString = "-verbose:jni";    //用于跟踪运行时的信息
        /*版本号设置不能漏*/
        vm_args.version = JNI_VERSION_1_2;
        vm_args.nOptions = 3;
        vm_args.options = options;
        vm_args.ignoreUnrecognized = JNI_TRUE;
        res = JNI_CreateJavaVM(&jvm, (void**)&env, &vm_args);        //启动虚拟机
        if (res < 0) {                                               //未启动成功
            fprintf(stderr, "Can't create Java VMn");                //输出错误
            //exit(1);
        }
        (*jvm)->DestroyJavaVM(jvm);                                  //释放虚拟机
        fprintf(stdout, "Java VM destory.n");                  //输出“虚拟机释放”信息

}
```

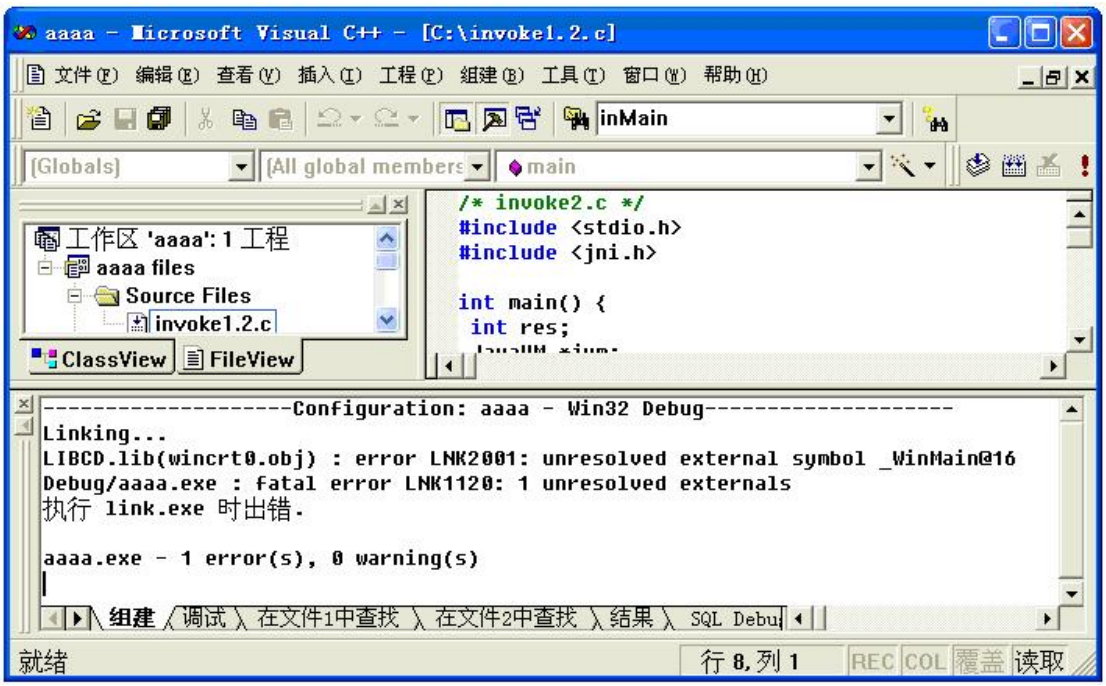
从上面的代码可以看出－－虚拟机的启动有三步：1.甚至启动参数；2.启动虚拟机；3.释放虚拟机相关的资源。

二、可能错误及其解决办法（3 个最常见的错误）

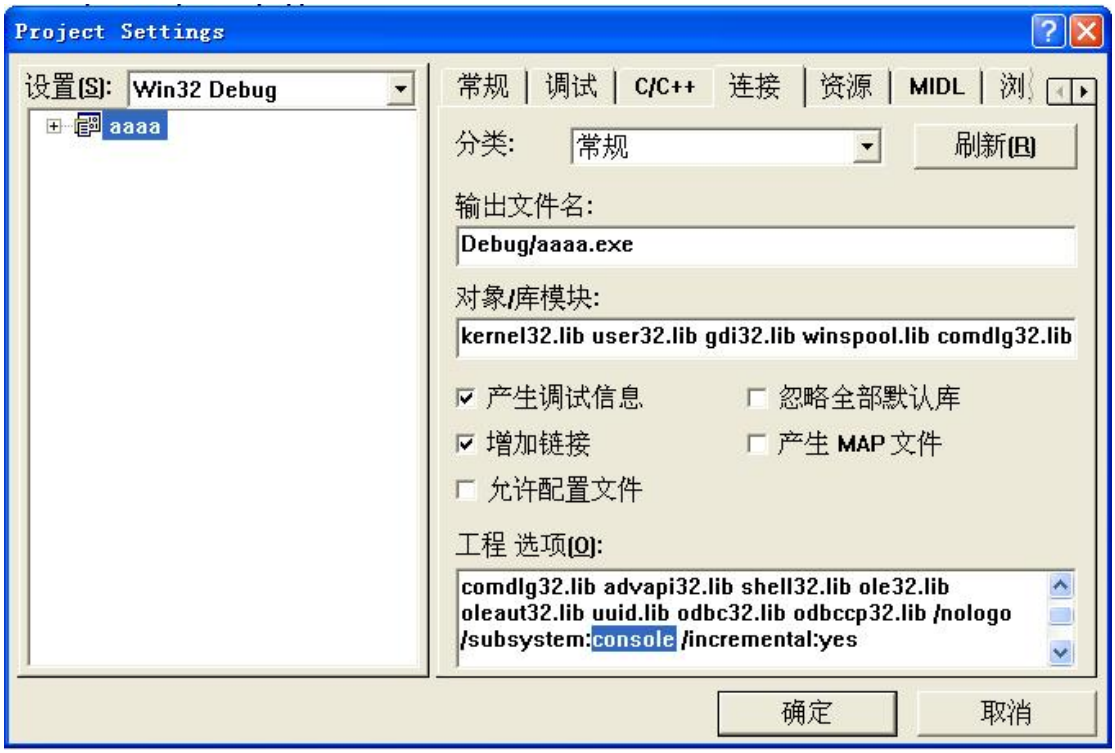
首先，与前面所有 JNI 本地方法不同的是，我们在此使用 VC 工具的“新建”，选择“工程”选项卡，继续选择“Win32 Application”（不是前面的“Win32 Dynamic-Link Library”，这主要是因为我们现在不是生成一个供 Java 使用的 DLL 文件，而是一个可以独立运行的 EXE 文件）。下面一路点击。



其次，在编译通过后，连接时可能会出现下面的错误。（看到了吧！——在 VC 工具的最下面）下面的图给出了修改办法——依次点击“工程”、“设置”、“连接”，到最下面的“工程 选项”中将“subsystem:windows”改为“subsystem:console”就可以解决这个问题了！（为什么！你不要问我哦！我会 Java，可不是 VC、C 高手哦！）



下图就是更改的图示！



最后，当你编译、连接了你的“.c”文件产生一个“.exe”文件时，你去点击，就应该有输出！此时可能跳出错误提示“不能连接 jvm.dll”之类的对话框，请将“Java\jdk1.6.0\jre\bin\client “目录下的 jvm.dll 放到你的 path（系统路径）下。（不会此时你学 java 都不会路径设置吧！请阅读最最基本的书吧！）

三、输出结果

由于启动 JVM 的输出结果很多，我们省略掉中间部分！

[Dynamic-linking native method java.lang.Float.intBitsToFloat ... JNI]
[Dynamic-linking native method java.lang.Double.longBitsToDouble ... JNI]
[Dynamic-linking native method java.lang.Float.floatToIntBits ... JNI]
[Dynamic-linking native method java.lang.Double.doubleToLongBits ... JNI]
……
[Registering JNI native method java.lang.Compiler.command]
[Registering JNI native method java.lang.Compiler.enable]
……
[Dynamic-linking native method java.io.WinNTFileSystem.list ... JNI]
[Dynamic-linking native method java.io.WinNTFileSystem.canonicalizeWithPrefix0 ... JNI]
Java VM destory.n

看到了吧！包括大量的动态连接本地方法和注册本地方法！最后一行红色的是我们在程序中输出的哦！（不会连这个结果都不知道如何得到吧！－－到你程序的目录路径下使用>aaa 111.txt 所有信息就到你的 111.txt 文本文件中了哦！）

更多精彩请关注：

http://blog.163.com/miaoxiaodong78/