**[MONO参数信息](http://www.cnblogs.com/likaincu/articles/404997.html)**

放源代码协会通过Linux软件开发公司Ximian开始了一个旨在将微软.Net平台移植到Linux上来的项目。这个叫做Mono工程的项目包括创建一个开发平台允许人们为基于Linux和基于Windows系统的.Net开发应用程序。

据Ximian公司的首席技术官员Miguel de Icaza说Mono项目的起因是GNOME社团特别是Ximain公司对于一个改进的开发工具的需要。

但是Mono项目的影响却是更加深远的，因为这些开发工具是基于CLI（通用语言架构）和微软提交给欧洲计算机制造协会（ECMA）的C＃实现标准的。如果Mono能够创建完成一套基于这些标准Linux开发工具集的话，开发者将能够编写同时在Windows和Linux上运行的.Net程序，这些程序甚至还可能在其它非Windows的操作系统上运行，比方Unix。

除了实际的开发过程，Mono项目的工具必须在开放源代码社团的内部完成而不能违反微软的知识产权专利－－这实际上意味着开发者们必须基于标准重新构造这些工具而不能查看微软的任何源代码。

Mono项目包括三个核心的部分：一个C＃语言的编译器，一个CLI和一个类库。 CLI是包含了类加载器的虚拟机，实时的编译器，和一个运行时环境的垃圾收集器。CLI这个组件将允许用C＃编写的应用程序能够在象Linux这样的非Windows操作系统上运行。这与Java世界中Java虚拟机能够让一个应用程序在不同的操作系统上运行是类似的。

Mono项目的类库能够让应用程序完成核心的诸如XML处理，文件输入输出，和连接网络之类的任务。它将会和微软的CLI兼容（与微软在.Net 框架中发布的东西紧密的配套），还有一点就是将由开放源代码团体编写的额外的类库也将和CLI兼容

Mono命令  
正确安装Mono之后，就可以使用mono的C#编译器mcs编译cs文件了。Mono一般有以下的工具：  
mono：mono 的JIT编译器，运行mono程序  
mcs：c#编译器  
gmcs：符合C#2.0规范的编译器，正在实现中  
mbas：basic编译器  
MonoDevelop：Mono IDE开发工具  
monodoc：帮助文件查看工具  
mint：Mono解释器（Interpreter），以解释的形式运行mono程序  
ilasm：Mono汇编程序  
monodis：Mono反汇编器  
al：程序集链接器  
cert2spc：发行者证书测试工具  
certmgr：证书管理器工具管理证书、证书信任列表和证书吊销列表  
chktrust：证书验证工具检查用 Authenticode 证书签名的文件的有效性  
cilc：Mono CIL2C的binding产生器，可以为配件产生c的接口  
disco：Web 服务发现工具  
gacutil：全局程序集缓存工具可以查看和操作全局程序集缓存  
genxs：Mono XML序列化产生器  
makecert：证书创建工具  
monograph：显示配件的调用关系和类型层次结构  
monop：Mono 类大纲查看器  
monoresgen：Mono资源产生器  
resgen：IL资源产生工具  
secutil：从程序集提取 X.509 证书的强名称信息或公钥  
setreg：设置注册表工具使您得以更改公钥密码的注册表设置  
signcode：文件签名工具使用  
sn：强名称工具有助于使用强名称对程序集进行签名  
soapsuds：编译使用Remote与 XML Web services 通信的客户端应用程序  
sqlsharp：sql命令行交互式工具  
wsdl：Web 服务描述语言工具  
xsd：XML 架构定义工具

mcs的语法

mcs [参数] 源文件

参数可以使用-option或者/option的方式。

       参数：

         --about 显示c#编译器的信息

         -checked[+][-]指定溢出数据类型边界的整数算法是否将在运行时导致异常。

         -codepage:ID指定编译中的所有源代码文件所使用的代码页

         -clscheck[+][-] 禁止编译器依附关系检查

         -define:S1[;S2] 定义一个或多个符号，可以简写为/d

         -debug[+][-] 产生调试信息

         -delaysign[+][-] 仅仅将公钥插入到配件中，并不进行签名

         -doc:File 产生XML文档，利用ndoc工具可以产生类似MSDN的文档

         -g 产生调试信息

         -keycontainer:NAME 指定用来强命名(strongname)配件的密钥对

         -keyfile:FILE 指定用来强命名的密钥文件

         -lib:PATH1,PATH2 增加配件链接的路径

         -main:class 指定入口点(entry point)，也就是缺省可执行的那个类

         -noconfig[+][-] 不使用默认的参考编译，和微软的csc的/noconfig对应

         -nostdlib[+][-] 不导入标准库

         -nowarn:W1[,W2] 显示一个或者多个警告信息

         -out:FNAME 指定输出文件名

         -pkg:P1[;P2] 引用包P1…P2

         --expect-error X 期望抛出X异常

         -resurce:SPEC搜索指定目录中要编译的源文件

         -reference:ASS 编译时引用ASS配件

         -target:KIND  编译输出类型，可以是exe、winexe、liberary和module

         -unsafe[+][-] 编译使用 unsafe 关键字的代码

         -warnaserror[+][-] 将警告作为错误对待

         -warn:LEVEL 警告等级，最高是4，缺省是2

         -help2 获得更多帮助信息

         -linkresource:FILE[,ID] 链接一个资源文件

         -resource:FILE[,ID] 嵌入一个资源文件

         -win32res:FILE 链接一个win32资源文件

         -win32icon:FILE 指定图标

         @FILE指定响应文件,文件列出了编译器选项或要编译的源代码文件的文件

mono的语法

       mono [参数] 配件

       参数如下：

         --compile METHOD 只编译配件中指定的方法

         --ncompile N 编译此方法的次数，缺省是1

         --regression 运行纠正测试

         --print vtable 输出所有用到的类的vtable

         --trace[=EXPR] 允许跟踪

         --compile-all 编译配件中所有的方法

         --breakonex 在异常上插入一个断点

         --break METHOD 在方法的入口上插入一个断点

         --debug 允许调试

         --stats 输出JIT操作的统计信息

         --statfile FILE 设置统计文件

         --aot 将配件编译成本地程序

         --profile[=profiler]  指定profiler

         --graph[=TYPE] METHOD 画一图形（cfg、dtree、code、ssa、optcode）

         --config FILE 指定Mono的配置文件

         --verbose显示更详细消息

         --help 帮助信息

         --version 显示版本信息

         --optimize=OPT 性能优化

gacutil的用法

       gacutil [命令] [参数]

       命令如下：

-i [-check\_refs] [-package NAME] [-root ROOTDIR] [-gacdir GACDIR]

        安装配件到全局程序缓存中

        配件所在路径,配件钥包含配件清单(manifast)

                  例如: -i myDll.dll

         -il [-check\_refs] [-package NAME] [-root ROOTDIR] [-gacdir GACDIR]

        安装多个配件到全局程序缓存中

        一个文件，包含要增加的配件列表

        例如：-il assembly\_list.txt

                assembly\_list.txt 的内容：

                assembly1.dll

                assembly2.dll

         -u [-package NAME] [-root ROOTDIR] [-gacdir GACDIR]

        从全局程序缓存中卸载指定配件

        配件名称或者配件的部分名称，如果是部分名称，所有符合这个名称的配件都会被卸载

        例如 -u myDll,Version=1.2.1.0

-ul [-package NAME] [-root ROOTDIR] [-gacdir GACDIR]

       从全局程序缓存中卸载配件

        包含配件列表的文件名

        例如: -ul assembly\_list.txt

                assembly\_list.txt的内容

            assembly1,Version=1.0.0.0,Culture=en,PublicKeyToken=0123456789abcdef

                assembly2,Version=2.0.0.0,Culture=en,PublicKeyToken=0123456789abcdef

-us [-package NAME] [-root ROOTDIR] [-gacdir GACDIR]

        从全局程序缓存中卸载配件

        配件所在路径

        Example: -us myDll.dll

-l [assembly\_name] [-root ROOTDIR] [-gacdir GACDIR]

        列出全局程序缓存中所有的配件

        如果指定 参数，将只显示符合参数的配件

-?

        显示帮助信息

         参数如下：

         -package

        用来在 prefix/lib/mono 目录下增加一个名为NAME的目录,然后增加一个从 NAME/assembly\_name 到全局程序缓存中的配件的一个链接

-gacdir

        指定全局程序缓存的目录，如果不是标准的全局程序缓存目录，将使用MONO\_GAC\_PATH环境变量来引用配件。

-root

       用来集成automake工具

-check\_refs

        检查将要导入的配件不被任何未强命名的配件引用

-f Mono gacut不考虑兼容性，gacutil总强行安装新配件