biololo

新手上路

发表于 2009-10-13 21:27 | 只看该作者 打印 字体大小: 1 #

Runtime Type Information 运行时类型信息(二)

二:获得函数的RTTI代码

以下函数是获得published部分声明的函数名称,不包含参数和返回值,引用单元ObjAuto。AObj声明时包含编译开关{$M+}。(代码修改自D7VCL中一段,虽然TObject中有MethodName和MethodAddress两个函数,但是使用汇编撰写的,翻译成Pascal代码,也差不多就是下面这段的意思):

```
function GetObjMethodNames(AObj: TPersistent): String;
var
VMT: Pointer;
MethodInfo: Pointer;
Count: Integer;
begin
VMT := PPointer(AObj)^;
repeat
  MethodInfo := PPointer(Integer(VMT) + vmtMethodTable)^;
  if MethodInfo <> nil then
  begin
    Count := PWord(MethodInfo)^;
    Inc(Integer(MethodInfo), 2);
    while Count > 0 do
    begin
```

```
      Result:= Result+ PMethodInfoHeader(MethodInfo)^.Name+#13+#10;
      Inc(Integer(MethodInfo), PMethodInfoHeader(MethodInfo)^.Len);
      Dec(Count);
    end;
  end;
  VMT := PPointer(Integer(VMT) + vmtParent)^;
  if VMT = nil then
  begin
    Exit;
  end;
  VMT := PPointer(VMT)^;
until False;
end;
```

以下代码是获取函数及其参数和返回值,引用单元ObjAuto。AObj声明时包含编译开关{$M+}{$METHODINFO ON}:

```
function TForm1.GetObjMethods(AObj: TPersistent): String;
const
ConventionName: array [Low(TCallingConvention)..High(TCallingConvention)] of
String =
  ('Register', 'Cdecl', 'Pascal', 'StdCall', 'SafeCall');
var
StrList: TStringList;
VMT: Pointer;
MethodInfo: PMethodInfoHeader;
Count: Integer;
RoutinPrefix, MethodName, ReturnName, Params: String;
MethodAddr, MethodEnd: Pointer;
ReturnAddr: PReturnInfo;
begin
StrList:= TStringList.Create;
try
  VMT := PPointer(AObj)^;
  repeat
    MethodInfo := PPointer(Integer(VMT) + vmtMethodTable)^;
```

```
    if MethodInfo <> nil then
    begin
      // Scan method table for the method
      Count := PWord(MethodInfo)^;
      Inc(Integer(MethodInfo), 2);
      while Count > 0 do
      begin
        RoutinPrefix:= '';
        ReturnName:= ';';
        Params:= '';
        //now methodinfo points to head of method
        MethodName:= MethodInfo^.Name;
        MethodEnd := Pointer(Integer(MethodInfo) + MethodInfo^.Len);
        ReturnAddr:= Pointer(Integer(MethodInfo)+SizeOf(TMethodInfoHeader)-
          SizeOf(ShortString)+1+Length(MethodName));
        MethodAddr:=Pointer(Integer(ReturnAddr)+SizeOf(TReturnInfo));
        // RTTI involves methodinfo
        if Integer(MethodAddr) < Integer(MethodEnd) then
        begin
          if ReturnAddr^.ReturnType= nil then RoutinPrefix:='Procedure'
          else
          begin
            RoutinPrefix:= 'Function';
            ReturnName:= ': '+ReturnAddr^.ReturnType^^.Name+ ';';
          end;
          //add routin's convention type
          ReturnName:= ReturnName+ '
'+ConventionName[ReturnAddr^.CallingConvention];
          //the first parameter is self pointer and be hidden
          Inc(Integer(MethodAddr), SizeOf(TParamInfo) -SizeOf(ShortString) + 1 +
            Length(PParamInfo(MethodAddr)^.Name));
          while Integer(MethodAddr) < Integer(MethodEnd) do
          begin
            Params:= Params+ PParamInfo(MethodAddr)^.Name+': '+
              PParamInfo(MethodAddr)^.ParamType^^.Name;
```

```
          Inc(Integer(MethodAddr), SizeOf(TParamInfo) -SizeOf(ShortString) + 1 +
            Length(PParamInfo(MethodAddr)^.Name));
        end;
      end;
      //output information
      StrList.Append(Format(RoutinPrefix+' %s(%s)%s;', [MethodName,Params,
ReturnName]));
      //jump to the next method
      Inc(Integer(MethodInfo),PMethodInfoHeader(MethodInfo)^.Len);
      Dec(Count);
    end;
  end;
  // Find the parent VMT
  VMT := PPointer(Integer(VMT) + vmtParent)^;
  if VMT = nil then
  begin
    Break;
  end;
  VMT := PPointer(VMT)^;
 until False;
 Result:= StrList.Text;
finally
 StrList.Free;
end;
end;
```


bhylolo@gmail.com

replysubmit