




## THE DA DELPHI CODE:

Build an Exe more "other language friendly" and less IDE or platform dependant.



# so what's about ?




{ We need a symbol for the Pascal entry point (main unit's body). An external symbol `main` fixed up by the linker would be fine. Alas, external declarations can't do that; they must be resolved either in the same file with a \$L directive, or in a shared object. Hack: use a bogus, distinctive symbol to mark the fixup, find and patch it in the linker. }


// extract from unit sysinit, does it sound useful ?

- **You have to raise, ignore or fix a problem !**
- **Delphi doesn't have a pre-processor like in C++: therefore it makes use of compiler directives.**

# Versioning Problem or smart linker? **EKON 13**

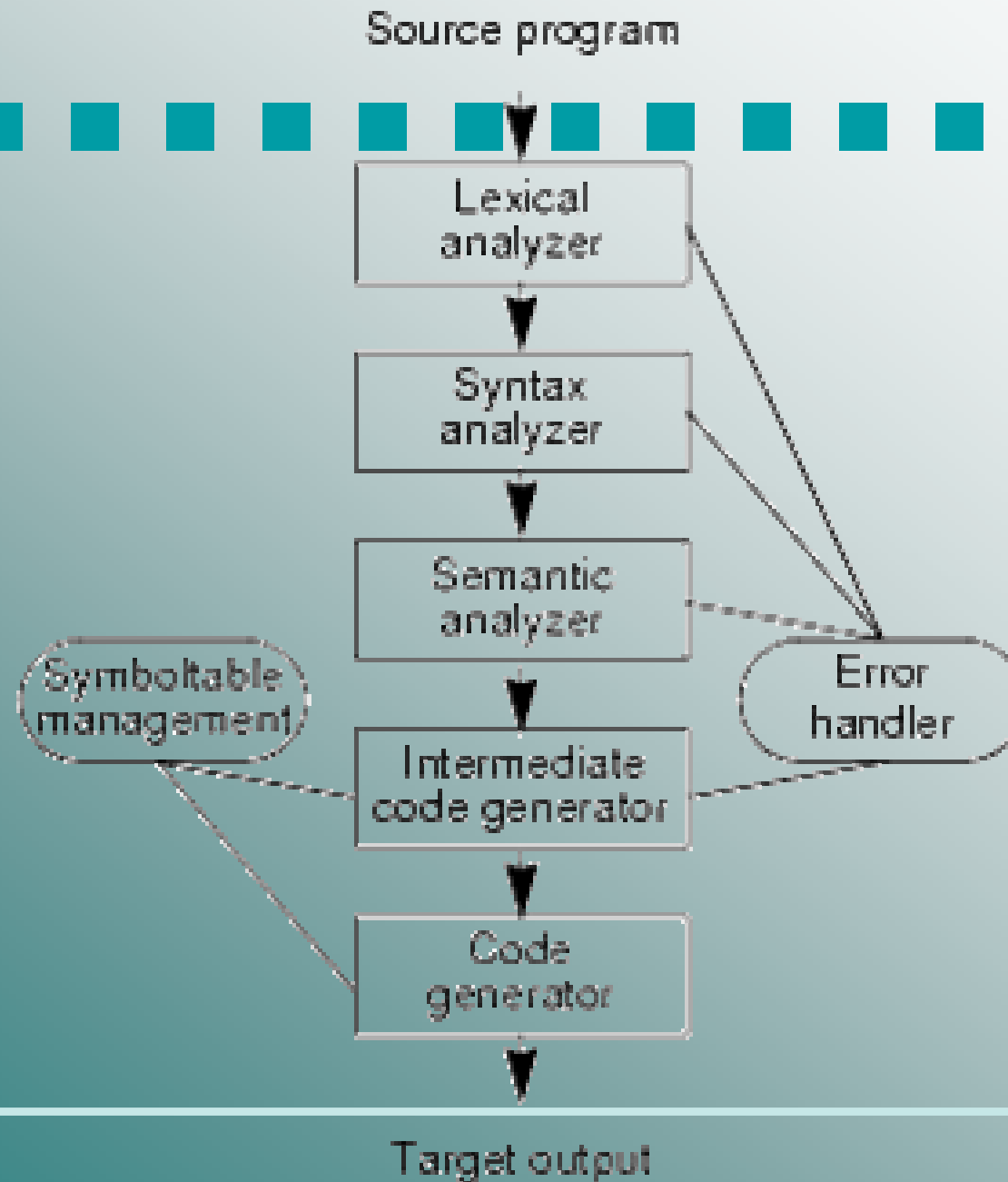
- 
- When we update a DLL (change function's implementation), we simply compile it, export some new routines and ship the new version. All the applications using this DLL will still work (unless, of course, you've removed existing exported routines).
  - On the other hand, when updating a package, you cannot ship a new version of your package without also updating the executable. This is why we cannot use a unit compiled in D9 in a D10 project unless we have the unit's source; the compiler checks version information of DCU's and decides whether an unit has to be recompiled so any package you provide for your application must be compiled using the same Delphi version used to compile the application.
  - Note: You cannot provide a package written in D10 to be used by an application written in D9.

# Tasks of a Compiler

- 
- Tokenising (Lexer) → Syntax analysis (Parser) → Semantic analysis → Translation → Code Generator
  - Memory Management
  - Exception Handling and Preprocessing
  - Symbol Table Management
  - Call Convention
  - Data alignment and Types
  - Name Mangling (more later on)
  - RTL (Run Time Library)
    - - to Common
    - - to Sys
    - - to Win

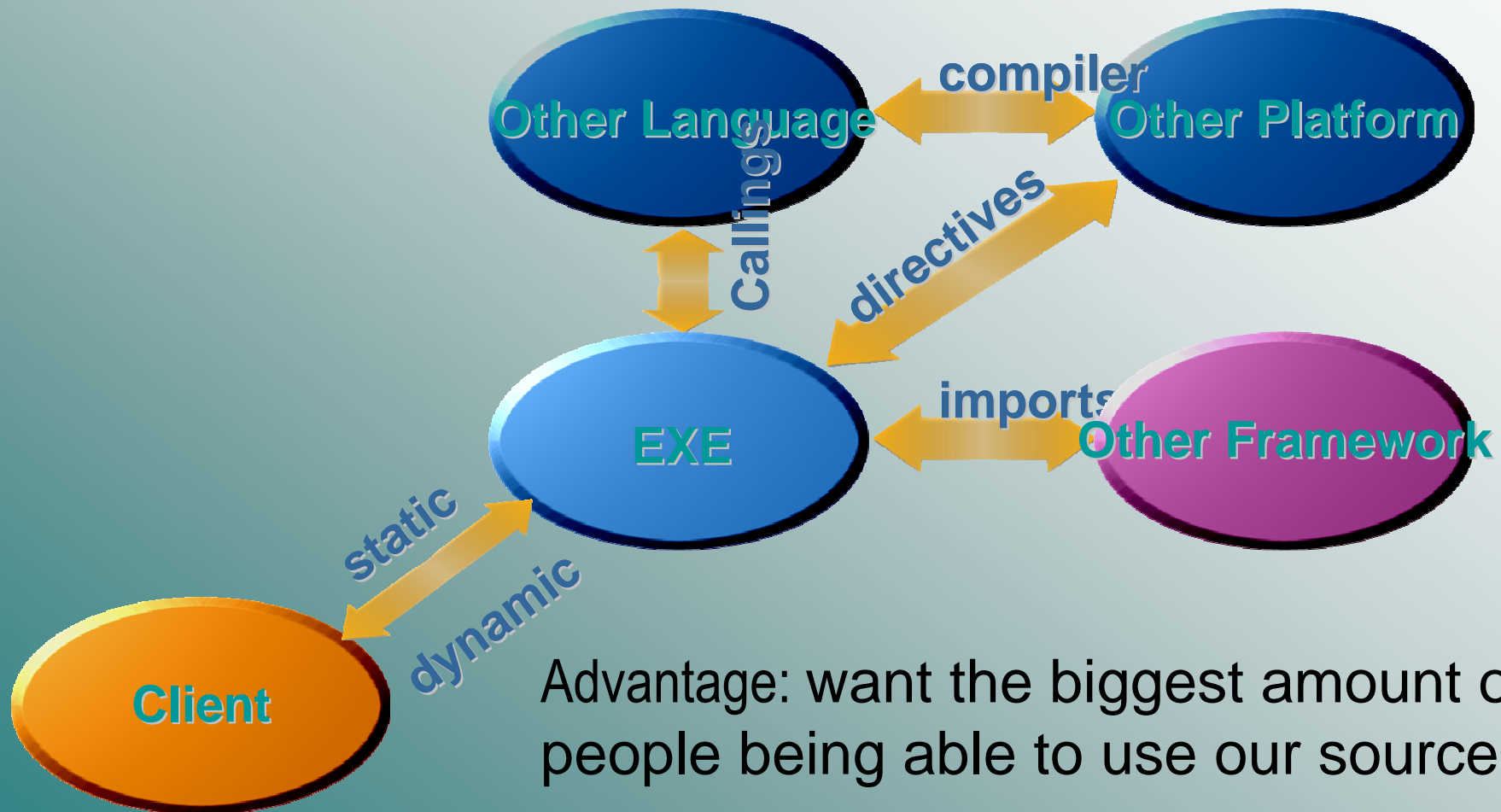
# Compiler View

EKON 13




# Compiler Multi Environment

EKON 13



Advantage: want the biggest amount of people being able to use our source



Delphi uses a mixed memory model, but it is very close to the "C" large model. The defaults are:

- Methods are far
- Procedures in an interface section are far
- Procedures only used in an implementation section are near
- Heap data and pointers in general (including class instances) are far
- Global variables are near (DS based)
- Procedure parameters and local variables are near (SS based)
- Procedures declared FAR or EXPORT are far
- Virtual memory tables are far for the new class model and near for the old


Code Ex.: FastMM Project




# Get your Options first

The first decision we should make is:

- with Ctr-O-O or in a \*.inc file ? → ex.
- Build Debug Version like (R+,I+,Q+)
- (when Q also R!)
- \*.bdsproj : Borland Developer Studio **Project File**. Successor of the .dof file holding compiler options etc. Also used for opening a project. **.dproj Project File**.  
D2007 Replaces bdsproj file.  
Set the target builds (Debug and Release)

- 
- Directives as \*.inc file available ({`$IdCompilerDirectivs.inc`}, {`$IdVers.inc`}).  
Example of SysInit (no inc file):
  - {`$H+,I-,R-,S-,O+,W-`}
  - {`$WARN SYMBOL_PLATFORM OFF`}
  - {`$WARN UNSAFE_TYPE OFF`}



Leave the `{$O+}` (or `{$Optimization On}`) compiler directive on. With this directive, Delphi compiler produces more efficient code. Sometimes, debugging is more difficult with optimization enabled: you can't set a breakpoint on a statement if the optimizer determines that the statement serves no purpose. All optimizations performed by the compiler are guaranteed not to alter the meaning of a program ;).

All binary modules which use the Delphi RTL (CLX) must be built with the same version of the RTL runtime package.

- *Directive \$A controls the alignment of fields in record types and class structures.*
- *In status {\$A1} or {\$A-} fields don't get an alignment. All records and structures of classes will be packed.*
- type  
TTeststruct = record  
    iVar : Integer; { 4 Byte }  
    dVar : double; { 8 Byte }  
    bVar : boolean; { 1 Byte }  
    sVar : Array[1..50] of char; { n \* 1 Byte }  
end;
- Using **packed** in Delphi slows data access and, in the case of a character array, affects type compatibility !



The `$IfOpt` compiler directive is a meta-directive - it tests for the + or - state of a single character compiler directive.

```
{$IfOpt H+}
```

```
  ShowMessage('Longstrings are set on');
```

```
{$EndIf}
```

It's useful to report on directive settings at the start of a program, when testing.

The `$V` directive controls type checking on short strings passed as variable parameters. → ex.


The `$B` directive tells Delphi whether to continue a multi argument boolean expression evaluation when the result is known before evaluation completes. → ex.

# syntax options example I

```

{$B-} //complete boolean evaluation shows cpu asm
function testrestbool(i: integer; mystring: shortstring): boolean;
begin
  if (i>0) or (mystring <> "") then result:= true else result:= false;
end;
```

{\$B-}		{\$B+}	
85DB	test ebx,ebx	803C2400	cmp byte ptr [esp],\$00
7F06	jnle \$0045dbb0	0F95C0	setnz al
803C2400	cmp byte ptr [esp],\$00	85DB	test ebx,ebx
7404	jz \$0045dbb4	0F9FC2	setnle dl
B001	mov al,\$01	0AC2	or al,dl
EB02	jmp \$0045dbb6	7404	jz \$0045dbba
		B001	mov al,\$01
		EB02	jmp \$0045dbbc



The Extended Syntax {\$X+} compiler directive determines whether Delphi includes a number of Pascal language extensions or not.

1. Treating functions as procedures
2. Using **Result** in functions
3. Treating Char arrays as strings


The \$T directive controls the types of pointer values generated by the @ operator and the compatibility of pointer types.

Assignable (writable) typed constants \$J are constant between function calls: to avoid using global variables!

- Controls what run-time checking code is generated. If such a check fails, a run-time error is generated. → ex. Stack overflow
- Compiler compiler example:  
{`$R 'langFile1ekon.res' 'langFile1ekon.rc'`}
- Command line compiler also with runtime checks  
`dcc32 [options] filename [options]`
- Build your own compiler stack machine:  
maXbox <http://www.softwareschule.ch/maxbox.htm>



# Which runtime checks ?

- 
- Range checking
    - Checks the results of enumeration and subset type operations like array or string lists within bounds
  - I/O checking
    - Checks the result of I/O operations
  - Integer overflow checking
    - Checks the result of integer operations (no buffer overrun)
  - Missing: Object method call checking
    - Check the validity of the method pointer prior to calling it (more later on).

# Range checking example



```
{ $R+ }
```

```
  SetLength(Arr,2);
```

```
  Arr[1]:= 123;
```

```
  Arr[2]:= 234;
```

```
  Arr[3]:= 345;
```

```
{ $R- }
```

Delphi (sysutils.pas) throws the `ERangeError` exception → ex.

# I/O checking example

The \$I compiler directive covers two purposes! Firstly to include a file of code into a unit. Secondly, to control if exceptions are thrown when an API I/O error occurs.

{\$I+} default generates the EInOutError exception when an IO error occurs. {\$I-} does not generate an exception. Instead, it is the responsibility of the program to check the IO operation by using the IOResult routine.


```

{$i-}
  reset(f,4);
  blockread(f,dims,1);
{$i+}
  if ioresult<>0 then begin
```

# Include Files (parameters)

- The `$I` parameter directive instructs the compiler to include the named file in the compilation. In effect, the file is inserted in the compiled text right after the `{$I filename}` directive. The default extension for filename is `.pas`. If filename does not specify a directory path, then, in addition to searching for the file in the same directory as the current module, unit recompiles if file newer.
- To specify a filename that includes a space, surround the file name with single quotation marks: `{$I 'My file'}`.
- → ex.: maXbox pascal script

# Buffer overflow of strings



The Delphi compiler hides the fact that the string variable is a heap pointer to the structure but setting the memory in advance is advisable:

```
//Check against Buffer overflow
```

```
var Source, Dest: PChar;
```

```
begin
```

```
    Source:= aSource;
```

```
    Dest:= @FData[FBufferEnd];
```


```
    if BufferWriteSize < Count then
```


```
        raise EFIFOStream.Create('Buffer over-run.');
```

```
var buffer: array[0..25] of Char;
```

```
buffer:= 'this is too long and so on';
```

```
    SetString(mystring, buffer, sizeof(buffer)); → ex. of result
```

- 
- When overflow checking is turned on (the \$Q compiler directive), the compiler inserts code to check that CPU flag and raise an exception if it is set. → ex.
  - The CPU doesn't actually know whether it's added signed or unsigned values. It always sets the same overflow flag, no matter of types A and B. The difference is in what code the compiler generates to check that flag.
  - In Delphi ASM-Code a call @IntOver is placed.



The cause of diff. problems is the not standardised "name mangling" of different compilers, which decorates the signature of an method to guarantee overloading. So the VC++ compiler (linker) puts some information about types and parameters on the entry point which the caller doesn't know.

Decorated names were originally created to allow C++ to work with legacy linkers (might not understand uppercase/lowercase, namespaces, class names, and overloading). In practice these "decorated names" are still around for reasons of compatibility.

# Prevent Name Mangling

You can work with an index instead of a name in a .def file and export section (depends on your signature)

C++:

```
LIBRARY mxlump_dll
```

```
EXPORTS
```

```
FunctionName1 @1
```

```
FunctionName1 @2
```

```
ProcedureName1 @3
```

Solution: Set an alias in the Delphi external declaration:

```
function CreateIncome2: CIncome; stdcall; external
```

```
'income.dll'
```

```
    name '_CreateIncome';
```



# Prevent Name Mangling II

You can work in a block to prevent name mangling:

macro **NoMangle** means 'extern "C"'

```
extern "C"
```

```
{ __declspec(dllexport) CIncome *CreateIncome();  
  void __EXPORT_TYPE SayHello2(); }
```

```
NoMangle long DLL_IMPORT_EXPORT csp2GetDeviceId(char  
szDeviceId[8], long nMaxLength);
```

```
//pascal
```

```
function csp2GetDeviceId(szDeviceId: PChar; nMaxLength: Longint):
```

```
Longint; stdcall; external 'csp2.dll' name 'csp2GetDeviceId';
```

```
var myBuffer: array [0..7] of Char;
```

```
begin
```

```
  csp2GetDeviceId(@myBuffer[0], SizeOf(myBuffer));
```

- Add in a file „compilerdef.inc“ the options D-,L-,Y-,C- . (switches off all debug information and asserts too):

```
accObj:= TAccount.createAccount(FCustNo, std_account);
```

- ```
assert(aTrans.checkOBJ(accObj),'bad condition with OBJ');
```
- Use Assert {\$C+} as a debugging check to test that conditions implicit assumed to be true are never violated (pre- and postconditions). → ex.

# Conditional Directives example

- Does the code functionality could be achieved by using different API's?
- function MakeTempFilename: string;
- begin
- {\$IFDEF LINUX}
- result:= tempnam(NIL, 'Indy'); {do not localize}
- {\$ELSE}
- SetLength(Result, MAX\_PATH + 1);
- GetTempFileName(PChar(ATempPath), 'Indy', 0, PChar(result));
- result:= PChar(result);
- {\$ENDIF}
- end;

# Call a Client platform independent

EKON 13

```
begin
```

```
  {$IFDEF LINUX}
```

```
  dllhandle:= dlopen(PChar(s2), RTLD_LAZY);
```

```
  {$ELSE}
```

```
  dllhandle:= LoadLibrary(Pchar(s2));
```

```
  {$ENDIF}
```

```
  if dllhandle = {$IFDEF LINUX} NIL {$ELSE} 0 {$ENDIF} then
```


```
    {$IFDEF LINUX}
```

```
    p.Ext1:= dlsym(dllhandle, pchar(copy(s, 1, pos(#0, s)-1)));
```

```
    {$ELSE}
```

```
    p.Ext1:= GetProcAddress(dllhandle, pchar(copy(s, 1, pos(#0, s)-1)));
```

```
    {$ENDIF}
```

- 
- 1.) Zur Sicherheit die beiden DCUs system.dcu und sysinit.dcu aus dem Lib-Verzeichnis sichern.
  - 2.) In der Datei system.pas die Compileroptionen D-,C-,L-,Y- kontrollieren. (you should have debug information)
  - 3.) Shell starten und in Verzeichnis (C:\Programme\CodeGear\RAD Studio\5.0\bin) wechseln.
  - 4.) Das Kommando ausführen  
dcc32 -m -y -z ..\source\win32\rtl\sys\system.pas und die Units system + sysinit werden compiliert.
  - 5.) Die neuen dcu Dateien aus dem Source in das lib Verzeichnis schieben

# Solution: Raise an exception to log

EKON 13

```
EStackOverflow = class(EExternal) → ex.  
  end deprecated;
```

```
{ $Q+ }  
  try  
    b1 := 255;  
    inc(b1);  
    showmessage(inttostr(b1));  
  //show silent exception  
  except  
    on E: Exception do begin  
      //ShowHelpException2(E);  
      LogOnException(NIL, E);  
    end;  
  end;
```

# Missing example: Check Object

Proposal to Runtime Checks

```
function TTrans.checkOBJ(aObject: TObject): boolean;
var str: string;
i: integer;
begin
  result:= false;
  if aObject= NIL then exit;
  try
    str:=ansiUppercase(aObject.classname);
    if str= "" then exit;
    for i:= 1 to length(str) do
      if not (str[i]in['0'..'9','A'..'Z','_']) then exit;
    aObject.classType;
    if aObject.InstanceSize < 1 then exit;
    aObject.Classnamels('TObject');
    result:= aObject.ClassNamels(aObject.Classname);
  except
    exit;
  end;
end;
```

# Test at last: Is this runtime error or exception handling ?

- `function IsDate(source: TEdit): Boolean;`
- `begin`
- `try`
- `StrToDate(TEdit(source).Text);`
- `except`
- `on EConvertError do`
- `result:= false;`
- `else`
- `result:= true;`
- `end;`
- `end;`



- DCC32 compiler with JHPNE as option will generate C++ headers (with .hpp extension) for your units!

```
DCC32 -JHPHN -N D:\DEV\PPPT -O D:\DEV\PPPT -U  
D:\COMPONENTS\SC2_2\securePtBase.pas
```

- rundll32 income.dll '\_SayHello2' //for short tests
- Dependency Viewer shows the inside of exe and dll's:

<http://delphi-jedi.org/Jedi:CODELIBJCL>

- CPU View in debugger → Ctr Alt C

Now: Delphi <=7 Compiler Virus in SysConst.dcu !!

## Questions and hope answers ? max@kleiner.com

