# The Lazarus IDE: How the program is being built

# 1 General ideas

The program uses as the programs in itself the LCL or Lazarus Component Library, which can be seen as the replacement of the VCL or Visual Component Library in Delphi/Kylix. Also, some extra file extensions are necessary: the .lfm-file or Lazarus Form, the .lpr-file or Lazarus Project file, the .lrs or Lazarus Resource file.

# 2 How the program starts

## 2.1 The program lazarus.pp

The program starts from the file lazarus.pp. In that file, you can find between *begin* and *end.* how the application starts.

The program is a typical LCL program. As lots of you will know how such an application starts, we describe these details in Appendix A.

Generally speaking, the command-line options are parsed, the Splash form is shown (afterwards removed), and four Forms are created: MainIDE, MessagesView, FindReplaceDlg and FindInFilesDlg. Interesting is also the inheritance scheme of TMainIDE: TObject -> TCustomForm -> TForm -> TMainIDEBar -> TMainIDE


## 2.2 main.pp: the TMainIDE class and TMainIDE.Create

As you would scroll through main.pp, you would notice that the unit is built around the one big class TMainIDE. TMainIDE compromises as it name implies, the main pieces of the IDE and all other pieces are indirectly related to it. So, the constructor of TMainIDE can be seen as the real start of the program.

After the command-line options are parsed and the config path is initialized, the global options are loaded into their data structures:

- EnvironmentOptions (TEnvironmentOptions) from the unit EnvironmentOpts contains general environment options.
- EditorOptions (TEditorOptions) from the unit editoroptions contains options for the unit editor.
- CodeToolOpts (TCodeToolOpts) from the unit codetoolsoptions contains code tool options.
- InputHistories (TInputHistories) will keep track of what has already been done, so it can be undone, ...

After that ToolStatus, which sets the IDE mode, is set into editable mode. (other modes will include compiling mode, I presume)

InitCodeToolBoss initializes the CodeToolBoss, which is the frontend for the codetools. The frontend for the codetools still needs to be developed, but the procedure already sets a basic set of compiler macros.

Thereafter, the MainIDE form is built and positioned.

**Fig. 1: Speed Buttons and Component Notebook**

```
 if LazarusResources.Find(ClassName)=nil then begin
    SetupMainMenu; // Creates menu as in fig 1
    SetupSpeedButtons; // see fig1
    SetupComponentNoteBook; // see fig 1
    ConnectMainBarEvents;
    SetupHints; //Setting up the Hints window, which shows text when you pass somewhere by.  It's
actually a timer being set up
  end;

  DebugBoss:=TDebugManager.Create(Self);
  DebugBoss.ConnectMainBarEvents;
```

--> Starting the debug manager: box

```
  LoadMenuShortCuts;
  SetupComponentTabs;
  SetupOutputFilter;
  SetupCompilerInterface;
  SetupObjectInspector;
  SetupFormEditor;
  SetupSourceNotebook;
  SetupTransferMacros;
  SetupControlSelection;


 SetupStartProject;
```

**The Project unit**

It contains the TProject and TUnitInfo classes. They keep information about the project and its associated units.

**The LazConf unit**

It manages OS specific configuration paths,... and is used in LoadGlobalOptions above.

**The CompReg unit**

All units viable to the IDE must register themselves. The Compreg unit has the classes through which that happens. These classes are the TRegisteredComponent, TRegisteredComponentList

**The MsgView unit**

The TMessageview class (=TForm) is responsible for displaying the PPC386 compiler messages. It's a little form you can see while compiling.

**The IDEComp unit**

Contains TIDEComponent.

**The LazarusIDEStrConst unit**

All strings are put into constants in this units.  Can be handy if you want to make a version in another language.

**The Compiler unit**

The TCompiler class has the methods to start the compiler and let it compile the program.

**The UnitEditor unit**

This unit builds the TSourceNotebook (=form) that the editors are held on.  It also has a class that controls the editors (TSourceEditor)

# 3 Sorting units from main directory by function

## Dialogs/Forms

aboutfrm.pas
buildlazdialog.pas
codetoolsdefines.pas
codetoolsdefpreview.pas
exttooldialog.pas
exttooleditdialog.pas
findinfilesdlg.pas
inputfiledialog.pas
macropromptdlg.pas
patheditordlg.pas
sysvaruseroverridedlg.pas
codetemplatedialog.pp
customformeditor.pp/formeditor.pp
findreplacedialog.pp
msgview.pp
newprojectdlg.pp
splash.pp
uniteditor.pp (TSourceNoteBook)
unitinfodialog.pp
viewunit_dlg.pp

## Defines/Constants and Definitions

ideoptiondefs.pas
lazarusidestrconsts.pas
projectdefs.pas

## Data structures/Information/Options

codetoolsoptions.pas
miscoptions.pas
runparamsopts.pas
compileroptions.pp

editoroptions.pp
environmentopts.pp
keymapping.pp
lazconf.pp
projectopts.pp

### Managers

basedebugmanager.pas/debugmanager.pas
compiler.pp
project.pp
compreg.pp/idecomp.pp
wordcompletion.pp
inputhistory.pas

### Help for others/Other

editdefinetree.pas
outputfilter.pas
ideprocs.pp
global.pp
transfermacros.pp

### Starters

mainbar.pas
lazarus.pp
main.pp

# 4 Units from subdirs

### The designer directory

I don't understand the true meaning of the designer directory, but as it name implies, it contents is designer stuff.  And it's mainly about the form and editor stuff.
It consists of:
- The Object Inspector with the propedits for editing all the properties
- A lot of abstract ancestors for units from the parent dir
- dialogs scalecompsdlg and sizecompsdlg
- the controlselection unit for keeping track of the controls being selected.

### The debugger directory

All debugging tools are there: the dialogs for breakpoints, callstack, watches, debugger, locals. You could probably open the forms with the form editor.

# Appendix A: Structure of a standard LCL-program

The first unit you will need is the Forms unit. This unit contains the classes TForm, TCustomForm, TApplication, TScreen, THintWindow. It also contains the TApplication Application. Application.Initialize tells the library to take the appropriate control over the library.

Maybe, it's not bad to look at the Lazarus project itself and start describing what each unit stands for:

```
uses
{$IFDEF IDE_MEM_CHECK}
  MemCheck,
{$ENDIF}
  Forms,
  Splash,
  Main,
  MainBar,
  MsgView,
  FindReplaceDialog,
  FindInFilesDlg,
  aboutfrm;
```

Forms is a unit out of the LCL, so can be compared to the Delphi Forms unit. It contains the code to start an application which uses the LCL and contains the classes TForm, TCustomForm, TApplication, TScreen, THintWindow.

Application is a TApplication and Application.Initialize will start that application and take control over event-handling and such.

The rest of the units are from Lazarus itself. The Splash unit contains the splashing form with the Lazarus symbol appearing when you start the Lazarus IDE. The other units (except Main/Mainbar) are also for forms and are here to create them within the application.

MainBar contains the TMainIDEBar class and main.pp the TMainIDE class.

The TMainIDE has the following inheritance scheme:

TObject -> TCustomForm -> TForm -> TMainIDEBar -> TMainIDE

I copied the code out of lazarus.pp and wrote some extra comments in it:

```
begin
  Application.Initialize; {Initializes the application as an LCL-application and lets LCL take control}
  TMainIde.ParseCmdLineOptions; {TMainIDE from MainIDE, parsing the command-line options}

  // Show splashform
  SplashForm := TSplashForm.Create(nil);
  with SplashForm do begin
    Show;
    Paint;
  end;
  Application.ProcessMessages; // process splash paint message

  Application.CreateForm(TMainIDE, MainIDE); //MainIDE is a variable out of TMainIDEBar
  {$IFDEF IDE_MEM_CHECK}
  CheckHeap('TMainIDE created');
  {$ENDIF}
```

```
{Creating those forms associated with the units}
  Application.CreateForm(TMessagesView, MessagesView);//MessagesView will be a variable out of the unit MsgView

  Application.CreateForm(TLazFindReplaceDialog, FindReplaceDlg);
  Application.CreateForm(TLazFindInFilesDialog, FindInFilesDialog);
  SplashForm.StartTimer;
  Application.Run;
  SplashForm.Free;

  writeln('LAZARUS END');
end.
```