# Show Application Title, Version, and Company

From Free Pascal wiki

Jump to navigationJump to search

| **English (en)** | русский (ru) |

## Overview

### Windows

From its earliest versions, Microsoft Windows had the capability of storing version information etc. in an executable file. Support for that was added to Delphi around v3, and was partly functional in Lazarus 0.9.24.

### Linux

There is no implicit provision in the ELF file format for version numbers, copyright and so on, but Lazarus stores this in resource sections. Reading the information back at runtime relies on FPC units, below is some useful information.

### macOS and iOS

On macOS, Lazarus file version information can be stored in two locations:

- the MACH-O executable (like on Windows, Linux); see below
- the application bundle (if present) in a plist. See Mac Show Application Title, Version, and Company for details.

## Implementations

### FPC 3.0+

Implementation in FPC 3.0.x using fcl-res: see announcement in User Changes 3.0.

The code below shows how to get executable info out of:

- .exe/.dll/.ocx files (Windows format)
- Lazarus-compiled ELF executables (Linux)
- Lazarus-compiled MACH-O executables (macOS)

```
program printfileinfo;
```

```
{
  Displays file version info for
- Windows PE executables
- Linux ELF executables (compiled by Lazarus)
- macOS MACH-O executables (compiled by Lazarus)
  Runs on Windows, Linux, macOS
}

{$mode objfpc}{$H+}
{$ifdef mswindows}{$apptype console}{$endif}
uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes,sysutils
  // FPC 3.0 fileinfo reads exe resources as long as you register the
appropriate units
  , fileinfo
  , winpeimagereader {need this for reading exe info}
  , elfreader {needed for reading ELF executables}
  , machoreader {needed for reading MACH-O executables}
  ;

var
  FileVerInfo: TFileVersionInfo;

{$R *.res}

begin
  FileVerInfo:=TFileVersionInfo.Create(nil);
  try
    FileVerInfo.ReadFileInfo;
    writeln('Company: ',FileVerInfo.VersionStrings.Values['CompanyName']);
    writeln('File description:
',FileVerInfo.VersionStrings.Values['FileDescription']);
    writeln('File version:
',FileVerInfo.VersionStrings.Values['FileVersion']);
    writeln('Internal name:
',FileVerInfo.VersionStrings.Values['InternalName']);
    writeln('Legal copyright:
',FileVerInfo.VersionStrings.Values['LegalCopyright']);
    writeln('Original filename:
',FileVerInfo.VersionStrings.Values['OriginalFilename']);
    writeln('Product name:
```

```
',FileVerInfo.VersionStrings.Values['ProductName']);
    writeln('Product version:
',FileVerInfo.VersionStrings.Values['ProductVersion']);
  finally
    FileVerInfo.Free;
  end;
end.
```

## Implementations using legacy FPC 2.6.x

Uses fcl-res; contribution via the mailing list: [1]

Use this like

```
uses
  resource, versiontypes, versionresource;

 FUNCTION resourceVersionInfo: STRING;

 (* Unlike most of AboutText (below), this takes significant activity at run-
*)
 (* time to extract version/release/build numbers from resource information
*)
 (* appended to the binary.
*)

 VAR     Stream: TResourceStream;
         vr: TVersionResource;
         fi: TVersionFixedInfo;

 BEGIN
   RESULT:= '';
   TRY

 (* This raises an exception if version info has not been incorporated into
the  *)
 (* binary (Lazarus Project -> Project Options -> Version Info -> Version
*)
 (* numbering).
*)

     Stream:= TResourceStream.CreateFromID(HINSTANCE, 1, PChar(RT_VERSION));
     TRY
       vr:= TVersionResource.Create;
       TRY
         vr.SetCustomRawDataStream(Stream);
         fi:= vr.FixedInfo;
         RESULT := 'Version ' + IntToStr(fi.FileVersion[0]) + '.' +
IntToStr(fi.FileVersion[1]) +
                ' release ' + IntToStr(fi.FileVersion[2]) + ' build ' +
IntToStr(fi.FileVersion[3]) + eol;
         vr.SetCustomRawDataStream(nil)
       FINALLY
         vr.Free
       END
     FINALLY
       Stream.Free
     END
   EXCEPT
   END
 END { resourceVersionInfo } ;
```

Using vinfo: [2] and [3]

## Related tips

### SVN/Git/Hg/Mercurial revision

Use $(lazarusdir)/tools/svn2revisioninc to get revision number (from a subversion, git or mercurial repository) into a file revision.inc, e.g. something like:

```
// Created by Svn2RevisionInc
const RevisionStr = '43594';
```

## Unix-only hacks

These work with Linux on various platforms, and probably with Solaris provided that the GNU-derived utilities are installed.

### Getting Subversion revision information as a program-accessible string

Put this into Project options -> Compilation -> Execute before -> Command:

```
/bin/sh -c "echo -n C`svnversion -n`C |tr A-G %-+ >project_svnrevision.inc"
```

Note quote and backtick positions. The tr is converting C into another layer of quotes which is necessary for things to work as required.

Put this into the program:

```
(*$IFDEF UNIX   *)
        rev= (*$I project_svnrevision.inc *) ;
(*$ELSE         *)
        rev= 'unimplemented';
(*$ENDIF        *)
```

Note that that has to be a string, since the revision number will have a non-numeric suffix if the project has been updated since it was last committed.

### Renaming the final executable to include platform and timestamp

Put this into Project options -> Paths -> Unit output directory:

```
lib/$(TargetCPU)-$(TargetOS)
```

Put this into Project options -> Paths -> Target file name:

```
UnyokedBackend-$(TargetCPU)-$(TargetOS)-$(LCLWidgetType)
```

Make sure that "Apply conventions" is ticked (checked). This might vary slightly according to IDE and compiler/linker versions.

Put this into Project options -> Compilation -> Execute after -> Command:

```
 /bin/sh -c "mv
libunyokedbackend-$(TargetCPU)-$(TargetOS)-$(LCLWidgetType).so
UnyokedBackend-$(TargetCPU)-$(TargetOS)-$(LCLWidgetType).`date +%F.%R`.so"
```

That should be a single line. Note quote and backtick positions.

## See also

- TVersionResource

Retrieved from "http://wiki.freepascal.org/index.php?title=Show_Application_Title,_Version,_and_Company&oldid=139378"