A Collection of Delphi/Lazarus Tips & Tricks

December 23, 2019

The code has not been extensively tested and might not work in every version of Delphi or Lazarus. The samples usually do not incorporate modern syntax features developed after Delphi 7.

Contents

1 Controls

1.1 Customized hints (tooltips)

First method:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Screen.HintFont.Name := 'Arial';
  Screen.HintFont.Size := 14;
  Application.HintColor := clBlue; //background color
  Application.HintPause := 0;
end;
```

Second method:

```
...
type
  TMyHintWindow = class(THintWindow)
    constructor Create(AOwner: TComponent); override;
  end;
...

constructor TMyHintWindow.Create(Owner: TComponent);
begin
  inherited Create(Owner);
  Canvas.Font.Name := 'Arial';
  Canvas.Font.Size := 14;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.ShowHint := false;
  HintWindowClass := TMyHintWindow;
  Application.ShowHint := true;
end;
```

1.2 Implementing a control array

```
...
const
  Count = 5;

type
  TForm1 = class(TForm)
    //event handler for OnCreate
    procedure FormCreate(Sender: TObject);
  public
    Edits: array[0..Count-1] of TEdit;
    procedure EditChange(Sender: TObject);
  end;
...

procedure TForm1.FormCreate(Sender: TObject);
var
  i: integer;
begin
  for i := 0 to Count-1 do begin
    Edits[i] := TEdit.Create(Self);
    with Edits[i] do begin
      Parent := Self;
      SetBounds(10, 10 + i*50, 200, 40);
      Tag := i; //each control should remember its index
      OnChange := EditChange; //same event handler for all controls
    end;
  end;
end;

procedure TForm1.EditChange(Sender: TObject);
var
  i: integer;
begin
  i := TEdit(Sender).Tag;
  ShowMessage(Format('Edit[%d] has been changed',[i]));
end;
```

1.3 Show controls' hints on statusbar

First method:

```
Button1.ShowHint := true;
StatusBar1.AutoHint := true;
```

Second method:

```
type
  TForm1 = class(TForm)
    //event handler for OnCreate
    procedure FormCreate(Sender: TObject);
```

```
  private
    procedure ShowHint(Sender: TObject);
...

procedure TForm1.ShowHint(Sender: TObject);
begin
  StatusBar1.SimpleText := Application.Hint;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.OnHint := ShowHint;
end;
```

1.4 Disable the popup menu of textbox

```
//create a handler for OnContextPopup event of TMemo
procedure TForm1.Memo1ContextPopup(Sender: TObject;
  MousePos: TPoint; var Handled: Boolean);
begin
  Handled := true;
end;
```

1.5 Hot-tracking controls

The purpose is to highlight the control which receives the focus.

First method:

```
type
  TMyEdit = class(TEdit)
  protected
    procedure DoEnter; override;
    procedure DoExit; override;
  end;
...

procedure TMyEdit.DoEnter;
begin
  Color := clAqua;
  inherited;
end;

procedure TMyEdit.DoExit;
begin
  Color := clWindow;
  inherited;
end;
```

Second method:

```
type
```

```
  TMyLabel = class(TLabel)
  protected
    procedure WndProc(var AMessage : TMessage); override;
  end;
...

procedure TMyLabel.WndProc(var AMessage: TMessage);
begin
  if (AMessage.Msg = CM_MOUSEENTER) then
    Color := clAqua
  else if (AMessage.Msg = CM_MOUSELEAVE) then
    Color := clWindow;
  inherited;
end;
```

1.6 Obtaining the character index, line number and line length of textbox

`Memo1.Perform(EM_LINEFROMCHAR, nCaret, 0)` returns zero-based index of the line which contains zero-based caret poisition `nCaret`. If `nCaret` is -1 then the index of the current line is returned.

`Memo1.Perform(EM_LINEINDEX, nLine, 0)` returns caret position at the beginning of the line `nLine`. If `nLine` is -1 then the beginning of the current line is returned.

`Memo1.Perform(EM_LINELENGTH, nCaret, 0)` returns the length of the line which contains caret position `nCaret`.

`Memo1.Perform(EM_CHARFROMPOS, 0, MAKELPARAM(x,y))` returns character index in the low-order word and the line index in the high-order word corresponding to the pixel position (x,y) in memo's client area.

1.7 Setting textbox margins

The following code sets the left and right margin of `Memo1` to `wLeft` and `wRight` pixels, respectively:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Memo1.Perform(EM_SETMARGINS, EC_LEFTMARGIN or EC_RIGHTMARGIN,
                MAKELONG(wLeft,wRight));
end;
```

1.8 How to clear all textboxes in a form

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i : integer;
begin
  for i := 0 to ComponentCount-1 do
    if Components[i] is TEdit then
      (Components[i] as TEdit).Text := '';
end;
```

2 Mouse

2.1 Check whether a mouse button is being pressed

```
...
uses
  Windows;
...
function IsBtnPressed(button: integer): boolean;
//"button" can be either VK_LBUTTON, VK_MBUTTON, or VK_RBUTTON
begin
  result := (GetAsyncKeyState(button) and $8000) = $8000;
end
```

2.2 Capture Maximize/Minimize/Close button clicks

```
...
type
  TForm1 = class(Form)
  ...
  public
    procedure WMSysCommand(var Msg: TWMSysCommand); message WM_SYSCOMMAND;
...

procedure TForm1.WMSysCommand(var Msg: TWMSysCommand);
begin
  if (Msg.CmdType = SC_MINIMIZE) or (Msg.CmdType = SC_MAXIMIZE) or
      (Msg.CmdType = SC_CLOSE) then
    ...
  else
    inherited; //or DefaultHandler(Msg)
end;
```

2.3 Simulate mouse activity

```
uses Windows;
  ...

SetCursorPos(Form1.Left + 50, Form1.Top + 50);
mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);
mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0);
```

2.4 Capture mouse clicks at statusbar

```
uses
  Windows, Messages, Forms, Classes, SysUtils, Controls, ComCtrls, Commctrl;

type
  TForm1 = class(TForm)
    StatusBar1: TStatusBar;
    procedure FormCreate(Sender: TObject);
```

```
  private
    procedure WMNotify(var Msg: TWMNotify); message WM_NOTIFY;
  end;
...

procedure TForm1.FormCreate(Sender: TObject);
var
  i: integer;
begin
  for i := 0 to 3 do
    with StatusBar1.Panels.Add do begin
      Width := 100;
      Text := Format('Panel %d',[i]);
    end;
end;

procedure TForm1.WMNotify(var Msg: TWMNotify);
var
  pNMM: PNMMouse;
  ctrl: TWinControl;
  hWindow: HWND;
  iPanel: integer;
begin
  inherited;
  pNMM := PNMMouse(Msg.NMHdr);
  hWindow := pNMM^.hdr.hwndFrom;
  ctrl := FindControl(hWindow);
  iPanel := pNMM^.dwItemSpec;

  case pNMM^.hdr.code of
    NM_CLICK:
      Caption := Format('%s was clicked at panel %d',[ctrl.Name,iPanel]);
    NM_DBLCLK:
      Caption := Format('%s was double-clicked at panel %d',[ctrl.Name,iPanel]);
    NM_RCLICK:
      Caption := Format('%s was right-clicked at panel %d',[ctrl.Name,iPanel]);
    NM_RDBLCLK:
      Caption := Format('%s was right-double-clicked at panel %d',[ctrl.Name,iPanel
         ]);
  end;
end;
```

3 Forms

3.1 Create forms with rounded corners

```
procedure TForm1.FormCreate(Sender: TObject);
var
  rgn: HRGN;
begin
```

```
  GetWindowRgn(Handle, rgn);
  DeleteObject(rgn);
  // set the radius of corners to 100px
  rgn:= CreateRoundRectRgn(0, 0, Width, Height, 100, 100);
  SetWindowRgn(Handle, rgn, TRUE);
end;
```

3.2 Show/hide titlebar of a form

```
...
uses
  Forms, Windows;
...
procedure ShowTitlebar(AForm: TForm; bShow: boolean);
var
  style: longint;
begin
  with AForm do begin
    if BorderStyle = bsNone then exit;
    style := GetWindowLong(Handle, GWL_STYLE);
    if bShow then begin
      if (style and WS_CAPTION) = WS_CAPTION then exit;
      case BorderStyle of
      bsSingle, bsSizeable:
        SetWindowLong(Handle, GWL_STYLE, style or WS_CAPTION or WS_BORDER);
      bsDialog:
        SetWindowLong(Handle, GWL_STYLE, style or WS_CAPTION or DS_MODALFRAME or
           WS_DLGFRAME);
      end;
    end else begin
      if (style and WS_CAPTION) = 0 then exit;
      case BorderStyle of
      bsSingle, bsSizeable:
        SetWindowLong(Handle, GWL_STYLE, style and (not(WS_CAPTION)) or WS_BORDER);
      bsDialog:
        SetWindowLong(Handle, GWL_STYLE, style and (not(WS_CAPTION)) or
           DS_MODALFRAME or WS_DLGFRAME);
      end;
    end;
    SetWindowPos(Handle, 0, 0, 0, 0, 0, SWP_NOMOVE or SWP_NOSIZE or SWP_NOZORDER or
       SWP_FRAMECHANGED or SWP_NOSENDCHANGING);
  end;
end;
```

3.3 Disable (gray out) the Close-button of a form

```
procedure TForm1.FormCreate(Sender: TObject);
var
  hSysMenu: HMENU;
begin
  hSysMenu := GetSystemMenu(Handle, false);
```

```
  EnableMenuItem(hSysMenu, SC_CLOSE, MF_BYCOMMAND or MF_GRAYED);
end;
```

3.4 Set a form to stay on top of all other (non-topmost) windows

```
...
uses
  Forms, Windows;
...
procedure SetTopmost(form: TForm; topmost: boolean);
begin
  if topmost then
    SetWindowPos(form.Handle, HWND_TOPMOST, 0, 0, 0, 0, SWP_NOMOVE or SWP_NOSIZE)
  else
    SetWindowPos(form.Handle, HWND_NOTOPMOST, 0, 0, 0, 0, SWP_NOMOVE or SWP_NOSIZE);
end;
```

3.5 Disable resizing and movement of a form by user

```
uses
  Windows, Forms;

type
  TForm1 = class(TForm)
  ...
    procedure WMNCHitTest(var Msg: TWMNCHitTest); message WM_NCHITTEST;
  end;
...

procedure TForm1.WMNCHitTest(var Msg: TWMNCHitTest);
begin
  inherited;
  with Msg do
    if Result in [HTCAPTION, HTLEFT, HTRIGHT, HTBOTTOM, HTBOTTOMRIGHT, HTBOTTOMLEFT,
        HTTOP, HTTOPRIGHT, HTTOPLEFT] then
      Result := HTNOWHERE;
end;
```

3.6 Detect when a form has been minimized, maximized or restored

```
TForm1 = class(TForm)
  ...
  procedure WMSize(var Msg: TWMSize); message WM_SIZE;
end;
...

procedure TForm1.WMSize(var Msg: TWMSize);
begin
  inherited; // otherwise TForm1.Resize is not fired
  if Msg.SizeType = SIZE_MAXIMIZED then
    ...
```

```
  else if Msg.SizeType = SIZE_RESTORED then
    ...
  else if Msg.SizeType = SIZE_MINIMIZED then
    ...
end;
```

3.7 Detect the movement of a form

```
uses
  Windows, Forms;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
  ...
    procedure WMMove(var Msg: TWMMove); message WM_MOVE;
  end;
...

procedure TForm1.WMMove(var Msg: TWMMove);
begin
  Edit1.Text := IntToStr(Left);
  Edit2.Text := IntToStr(Top);
end;
```

3.8 Make a form with adjustable transparency

```
uses
  Windows, Forms, Classes, Controls, ComCtrls;

type
  TForm1 = class(TForm)
    TrackBar1: TTrackBar;
    ...
    procedure TrackBar1Change(Sender: TObject);
  end;
...

procedure SetTransparent(hWnd: longint; value: byte);
// opaque: value=255; fully transparent: value=0
var
  style: integer;
begin
  style := GetWindowLong(hWnd, GWL_EXSTYLE);
  if value < 255 then  begin
    style := style Or WS_EX_LAYERED;
    SetWindowLong(hWnd, GWL_EXSTYLE, style);
    SetLayeredWindowAttributes(hWnd, 0, value, LWA_ALPHA);
  end else begin
    style := style xor WS_EX_LAYERED;
```

```
    SetWindowLong(hWnd, GWL_EXSTYLE, style);
  end;
end;

procedure TForm1.TrackBar1Change(Sender: TObject);
begin
  SetTransparent(Handle, TrackBar1.Position);
end;
```

3.9 Drag a form by clicking in its client area

First method:

```
type
  TForm1 = class(TForm)
    Label1: TLabel;
    procedure FormMouseDown(Sender: TObject; Button: TMouseButton; Shift:
       TShiftState; X, Y: Integer);
    procedure Label1MouseDown(Sender: TObject; Button: TMouseButton; Shift:
       TShiftState; X, Y: Integer);
  end;
...

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift:
   TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Perform(WM_SYSCOMMAND, $f012, 0)
end;

procedure TForm1.Label1MouseDown(Sender: TObject; Button: TMouseButton; Shift:
   TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  Perform(WM_SYSCOMMAND, $f012, 0)
end;
```

Second method:

```
uses
  Windows, Forms;

type
  TForm1 = class(TForm)
    Label1: TLabel;
  public
    procedure WMNCHitTest(var Msg: TWMNCHitTest); message WM_NCHITTEST;
  end;
...

procedure TForm1.WMNCHitTest(var Msg: TWMNCHitTest);
begin
```

```
  inherited;
  with Msg do
    if Result = HTCLIENT then Result := HTCAPTION;
end;
```

3.10 Resize and move forms without border or title

Create a form and set its BorderStyle to bsNone.

```
type
  TForm1 = class(TForm)
    procedure FormMouseDown(Sender: TObject; Button: TMouseButton; Shift:
       TShiftState; X, Y: Integer);
  end;
...

implementation

const
  SNAPSIZE = 20; //how many pixels are considered for snapping region
  SC_DRAGMOVE = $f012;
  SC_LEFTSIZE = $f001;
  SC_RIGHTSIZE = $f002;
  SC_UPSIZE = $f003;
  SC_UPLEFTSIZE = $f004;
  SC_UPRIGHTSIZE = $f005;
  SC_DNSIZE = $f006;
  SC_DNLEFTSIZE = $f007;
  SC_DNRIGHTSIZE = $f008;

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton; Shift:
   TShiftState; X, Y: Integer);
begin
  ReleaseCapture;
  if Y < SNAPSIZE then begin
    if X < SNAPSIZE then
      Perform(WM_SYSCOMMAND, SC_UPLEFTSIZE, 0)
    else if X > (ClientWidth - SNAPSIZE) then
      Perform(WM_SYSCOMMAND, SC_UPRIGHTSIZE, 0)
    else
      Perform(WM_SYSCOMMAND, SC_UPSIZE, 0)
  end else if Y > (ClientHeight - SNAPSIZE) then begin
    if X < SNAPSIZE then
      Perform(WM_SYSCOMMAND, SC_DNLEFTSIZE, 0)
    else if X > (ClientWidth - SNAPSIZE) then
      Perform(WM_SYSCOMMAND, SC_DNRIGHTSIZE, 0)
    else
      Perform(WM_SYSCOMMAND, SC_DNSIZE, 0)
  end else begin
    if X < SNAPSIZE then
```

```
      Perform(WM_SYSCOMMAND, SC_LEFTSIZE, 0)
    else if X > (ClientWidth - SNAPSIZE) then
      Perform(WM_SYSCOMMAND, SC_RIGHTSIZE, 0)
    else
      Perform(WM_SYSCOMMAND, SC_DRAGMOVE, 0)
  end
end;
```

4 Application

4.1 Minimize application to taskbar

A common error is attempting to minimize the form (`WindowStyle := wsMinimized`), rather than the application itself.

```
...
procedure TForm1.Button1Click(Sender: TObject);
begin
  Application.Minimize;
end;
```

4.2 A centralized handling of error messages

```
type
//create unique index for each error message
  TErrMsgID = (emOutOfMemory, emFileNotFound, emDivZero, emCantOpen);

const
  ErrMsgFmt : array[Low(TErrMsgID)..High(TErrMsgID)] of string = (
    'Out of memory',
    'File not found: %s',
    'Division by zero',
    'Cannot open file: %s'
  );

//array of corresponding dialog buttons
  ErrMsgBtns : array[Low(TErrMsgID)..High(TErrMsgID)] of TMsgDlgButtons = (
    mbOKCancel,
    [mbOK],
    [mbOK],
    [mbRetry,mbCancel]
  );


function ErrMsg(id: TErrMsgID; params: array of const): integer; overload;
//possible parameters (filename, etc) are contained
//in the open array parameter "params"
begin
```

```
  result := MessageDlg(
    Format(ErrMsgFmt[id], params), //formatted message
    mtError, //dialog type
    ErrMsgBtns[id], //buttons
    0
  );
end;

function ErrMsg(id: TErrMsgID): integer; overload;
begin
  result := ErrMsg(id, []);
end;

//examples of use
procedure TForm1.Button1Click(Sender: TObject);
begin
  ErrMsg(emDivZero)
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  resp : integer;
begin
  resp := ErrMsg(emCantOpen, ['myfile.txt'])
end;
```

4.3 Set system wide Hot Key for your application

```
uses
  Windows;

type
  TForm1 = class(Form)
  ...
  private
    FHotkeyID: integer;
    procedure WMHotkey(var Msg: TWMHotkey); message WM_HOTKEY;
...

implementation

procedure TForm1.FormCreate(Sender: TObject);
begin
  FHotkeyID := GlobalAddAtom('MyAppHotKey1');
  if not RegisterHotkey(Handle, FHotkeyID, MOD_ALT or MOD_SHIFT, VK_F9) then
    ShowMessage('Unable to assign Alt-Shift-F9 as hotkey');
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
```

```
  //when application terminates, remove the hot key
    UnRegisterHotkey(Handle, FHotkeyID);
    GlobalDeleteAtom(FHotkeyID);
  end;

  procedure TForm1.WMHotkey(var Msg: TWMHotkey);
  begin
    if Msg.HotKey = FHotkeyID then with Application do begin
      if IsIconic(Handle) then Restore;
      BringToFront;
    end;
  end;
```

4.4 Prompt user before closing application

Handle the `OnCloseQuery` event of the main form to detect and control the closing of an application.

```
  procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
  begin
    if MessageDlg('Exit program?', mtQuestion, mbOKCancel, 0) = mrCancel) then
      CanClose := false;  //by default CanClose = true
  end;
```

5 Common dialogs

5.1 Modify the controls in common dialogs

```
  ...
  uses
    Windows, Forms, Dialogs, Dlgs;
  ...
  //create handler for OnShow event of common dialog
  procedure TForm1.OpenDialog1Show(Sender: TObject) ;
  var
    hwnd: THandle;
  begin
    hwnd := GetParent(OpenDialog1.Handle);
    //change title of OK button, "Open" --> "Do Open"
    SetDlgItemText(hwnd, IDOK, PChar('&Do Open'));
    //hide the "Open as read-only" check box
    ShowWindow(GetDlgItem(hwnd,chx1),SW_HIDE);
    //these are the IDs of remaining controls:
    //IDCANCEL: the CANCEL button
    //stc3, stc2: the two label controls, "File name" and "Files of type"
    //edt1, cmb1: the edit control and combo box next to the labels
    //cmb2: combo box that allows to select the drive or folder
    //stc4: label for the cmb2 combo
    //lst1: list box that displays the contents of selected drive or folder
  end;
```

5.2 Repositioning common dialogs

```
//create handler for OnShow event of common dialog
procedure TForm1.OpenDialog1Show(Sender: TObject);
var
  hwnd: THandle;
  rect: TRect;
  dlgWidth, dlgHeight: integer;
begin
  hwnd := GetParent(OpenDialog1.Handle);
  GetWindowRect(hwnd, rect);
  dlgWidth := rect.Right-rect.Left;
  dlgHeight := rect.Bottom-rect.Top;
  MoveWindow(hwnd, Left+(Width-dlgWidth) div 2, Top+(Height-dlgHeight) div 2,
     dlgWidth, dlgHeight, true);
  Abort;
end;
```

5.3 Translate common dialogs

The following simple approach can be used to translate any resource strings in application, incl. the button captions and title bar texts of common dialogs as defined in the unit `Consts`.

```
uses
  Windows, Consts;
...
procedure HookResourceString(const ResStr: PResStringRec; const NewStr: PChar);
var
  Old: DWORD;
begin
  VirtualProtect(ResStr, SizeOf(ResStr^), PAGE_EXECUTE_READWRITE, Old) ;
  ResStr^.Identifier := Integer(NewStr) ;
  VirtualProtect(ResStr, SizeOf(ResStr^), Old, Old) ;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  HookResourceString(@SMsgDlgWarning, 'Hoiatus');
  HookResourceString(@SMsgDlgError, 'Viga');
  HookResourceString(@SMsgDlgInformation, 'Info');
  HookResourceString(@SMsgDlgConfirm, 'Kinnitus');
  HookResourceString(@SMsgDlgYes, 'Jah');
  HookResourceString(@SMsgDlgNo, 'Ei');
  HookResourceString(@SMsgDlgCancel, 'Loobu');
  HookResourceString(@SMsgDlgAbort, 'Katkesta');
  HookResourceString(@SYesButton, 'Jah');
  HookResourceString(@SNoButton, 'Ei');
  HookResourceString(@SCloseButton, 'Sulge');
  HookResourceString(@SAbortButton, 'Katkesta');
end;
```

6 Files

6.1 Determine the size of a file without opening it

Return value −1 indicates that the file does not exist.

```
uses
  SysUtils;
...
function FileSizeByName(const filename: string): integer;
var
  sr: TSearchRec;
begin
  result := -1;

  if (Pos(filename,'*') <> 0) or (Pos(filename,'?') <> 0) or
      (FindFirst(filename, faAnyFile, sr) <> 0) then
    Exit;

  if (sr.Attr and faDirectory) = 0 then
    result := sr.Size;

  FindClose(sr);
end;
```

6.2 Check a filename (or any string) against a pattern containing wildcard characters

Use the `MatchesMask` routine.

7 System

7.1 Get a list of system fonts

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ComboBox1.Items.Assign(Screen.Fonts);
end;
```

7.2 Append items into system menu

```
type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    procedure WMSysCommand(var Msg: TWMSysCommand); message WM_SYSCOMMAND;
  end;
...
```

```
const
  ID_ABOUT = WM_USER + 1;

procedure TForm1.FormCreate(Sender: TObject);
var
  hSysMenu: HMENU;
begin
  hSysMenu := GetSystemMenu(Handle, false);
  AppendMenu(hSysMenu, MF_SEPARATOR, 0, nil);
  AppendMenu(hSysMenu, MF_STRING, ID_ABOUT, PChar('&About...'));
end;

procedure TForm1.WMSysCommand(var Msg: TWMSysCommand);
begin
  if Msg.CmdType = ID_ABOUT then
    AboutForm.ShowModal
  else
    inherited;  //or DefaultHandler(Msg)
end;
```

7.3 Drag and drop files from Windows Explorer

```
type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    procedure WMDropFiles(var Msg: TWMDROPFILES); message WM_DROPFILES;
  end;
...

procedure TForm1.FormCreate(Sender: TObject);
begin
  //inform Windows that this window accepts dropped files
  DragAcceptFiles(Handle, true);
end;

procedure TForm1.WMDropFiles(var Msg: TWMDROPFILES);
var
  buf: array[0..MAX_PATH] of char;
  filecount, i: integer;
  filename: string;
begin
  filecount := DragQueryFile(Msg.Drop, $FFFFFFFF, nil, 0);
  for i := 0 to Pred(filecount) do begin
    DragQueryFile(Msg.Drop, i, @buf, sizeof(buf));
    filename := string(buf);
    ...  //do something with file
  end;
  DragFinish(Msg.Drop);
end;
```

7.4 Send a file to Recycle bin

```
uses
  ShellApi;
...
function RecycleFile(const filename: string): boolean;
var
  foStruct: TSHFileOpStruct;
begin
  with foStruct do begin
    wnd := 0;
    wFunc := FO_DELETE;
    pFrom := PChar(filename+#0#0);
    pTo := nil;
    fFlags:= FOF_ALLOWUNDO or FOF_NOCONFIRMATION or FOF_SILENT;
    fAnyOperationsAborted := false;
    hNameMappings := nil;
  end;
  Result := SHFileOperation(foStruct) = 0;
end;
```

7.5 Open a file/URL using its associated application

The result is the same as if the file were double-clicked in Windows Explorer.

```
uses
  ShellApi;
...

function OpenFile(filename: string; directory: string = nil; params: string = nil):
   boolean;
begin
  result := ShellExecute(Application.Handle, 'open', PChar(filename), directory,
     params, SW_SHOWNORMAL) >= 32;
end;
```

The `directory` parameter specifies the default directory for the shell operation. If **nil**, your application's current directory is used. If `filename` specifies an executable file, then `params` should contain the command line parameters to be passed to the application.

If the return value of `ShellExecute` is greater than `32`, the application was executed successfully. Other return values are error codes indicating the reason of failure. For instance, error code `31` means that there is no application associated with the given filename extension.

If the filename represents a URL, it should be fully qualified (e.g., starting with `http://`) and double-quotes should be escaped (replaced with `%22`).

In Lazarus, a cross-platform solution is implemented by routines `OpenURL` and `OpenDocument`, contained in the unit `LCLIntf`.

7.6 Monitor the changes of clipboard's content

```
uses
```

```
  Windows, Forms, Clipbrd;

type
  TForm1 = class(TForm)
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
  public
    procedure WMDrawClipboard(var Msg: TWMDrawClipBoard); message WM_DRAWCLIPBOARD;
  end;
...

procedure TForm1.FormCreate(Sender: TObject);
begin
  SetClipboardViewer(Handle);
end;

procedure TForm1.WMDrawClipboard(var Msg: TWMDrawClipBoard);
begin
  if Clipboard.HasFormat(CF_BITMAP) then
    Image1.Picture.Assign(Clipboard);
end;
```

7.7 Listing system's drives in a listbox

```
uses
  Windows;

function GetVolumeName(Root: string): string;
var
  VolumeSerialNumber : DWORD;
  MaximumComponentLength : DWORD;
  FileSystemFlags : DWORD;
  VolumeNameBuffer : array[0..255] of char;
begin
  GetVolumeInformation(PChar(Root),
    VolumeNameBuffer,
    sizeof(VolumeNameBuffer),
    @VolumeSerialNumber,
    MaximumComponentLength,
    FileSystemFlags,
    nil,
    0);
  result := string(VolumeNameBuffer);
end;

function DiskInDrive(DriveLetter: char): boolean;
var
  ErrMode: word;
begin
  if DriveLetter in ['a'..'z'] then Dec(DriveLetter, $20);
```

```
  if not (DriveLetter in ['A'..'Z']) then
    raise Exception.Create('Not a valid drive letter');
  ErrMode := SetErrorMode(SEM_FAILCRITICALERRORS);
  try
    result := (DiskSize(Ord(DriveLetter) - $40) <> -1)
  finally
    SetErrorMode(ErrMode);
  end;
end;

procedure TMainForm.FormCreate(Sender: TObject);
var
  DriveLetter: char;
begin
  for DriveLetter := 'a' to 'z' do
    case GetDriveType(PChar(DriveLetter + ':\')) of
      DRIVE_REMOVABLE, DRIVE_FIXED, DRIVE_CDROM:
        if DiskInDrive(DriveLetter) then
          ListBox1.Items.Add(UpperCase(DriveLetter) + ': (' + GetVolumeName(
              DriveLetter + ':\') + ')')
        else
          ListBox1.Items.Add(UpperCase(DriveLetter) + ':');
    end;
end;
```

7.8 Using WinHelp API

Implementing your own "What's this?" button

```
uses
  Windows;
...

procedure TForm1.btnHelpClick(Sender: TObject);
begin
  PostMessage(Handle, WM_SYSCOMMAND, SC_CONTEXTHELP, 0);
end;
```

7.9 Prevent a screensaver to kick in as long as your application is running

```
uses
  Windows;
...

procedure AppMessage(var Msg: TMsg; var Handled: Boolean);
begin
  if (Msg.Message = WM_SYSCOMMAND) and ((Msg.wParam = SC_SCREENSAVE) or (Msg.wParam
     = SC_MONITORPOWER)) then
    Handled := true;
end;
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.OnMessage := AppMessage;
end;
```

7.10 Installing a font on the fly

Assume that a font file `MyFont.ttf` resides in application directory.

```
uses
  Windows, Forms, SysUtils;
...

procedure TForm1.FormCreate(Sender: TObject);
begin
  AddFontResource(PChar(ExtractFilePath(Application.ExeName) + '\MyFont.ttf'));
  SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  RemoveFontResource(PChar(ExtractFilePath(Application.ExeName) + '\MyFont.ttf'));
  SendMessage(HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
end;
```

7.11 Launching Control Panel applets from your program

```
function RunControlPanelApplet(sAppletFileName : string) : integer;
begin
  result := WinExec(PChar('rundll32.exe shell32.dll,Control_RunDLL ' +
     sAppletFileName), SW_SHOWNORMAL);
//access.cpl: Accessibility Options
//appwiz.cpl: Add/Remove Programs
//desk.cpl: Display Properties
//intl.cpl: Regional Settings
//joy.cpl: Joystick Settings
//main.cpl: Mouse Settings
//mmsys.cpl: Multimedia Settings
//modem.cpl: Modems
//sysdm.cpl: System Properties
//timedate.cpl: Time/Date Settings
end;
```

7.12 Retrive the path of the Windows directory

```
uses
  Windows, StrUtils;
...
```

```
function GetWinDir: string;
var
  buffer: array[0..MAX_PATH] of char;;
begin
  GetWindowsDirectory(buffer, MAX_PATH);
  result := string(buffer);
  if RightStr(result,1) <> '\' then result := result + '\';
end;
```

8 Text and graphics

8.1 Drawing rotated text

```
uses
  Windows, Graphics;
...
procedure AngleTextOut(Canvas: TCanvas; Angle, X, Y: integer; Text: string);
var
  LogFont: TLogFont;
  hOldFont, hNewFont: HFONT;
begin
  GetObject(Canvas.Font.Handle, SizeOf(LogFont), Addr(LogFont));
  LogFont.lfEscapement := Angle * 10;
  LogFont.lfOrientation := Angle * 10;
  hNewFont := CreateFontIndirect(LogFont);
  hOldFont := SelectObject(Canvas.Handle, hNewFont);
  ACanvas.TextOut(X, Y, Text);
  hNewFont := SelectObject(Canvas.Handle, hOldFont);
  DeleteObject(hNewFont);
end;
```

8.2 Save graphics to a bitmap file or a Windows Metafile

```
uses
  Graphics;
...

procedure SaveToBitmapFile(const AFilename: TFilename);
var
  bmp: TBitmap;
begin
  bmp := TBitmap.Create;
  bmp.Width := 400;
  bmp.Height := 400;
  with bmp.Canvas do begin
    //drawing commands
  end;
  bmp.SaveToFile(AFilename);
  bmp.Free;
end;
```

```
procedure SaveToMetafile(const AFilename: TFilename);
var
  emf: TMetafile;
  mfc: TMetafileCanvas;
begin
  emf := TMetafile.Create;
  emf.Enhanced := true; // save as Enhanced Metafile
  emf.Width := 400;
  emf.Height := 400;
  mfc := TMetafileCanvas.Create(emf, 0);
  with mfc do begin
    //drawing commands
  end;
  mfc.Free;

    try
    emf.SaveToFile(AFilename);
  finally
    emf.Free;
  end;
end;
```

8.3 Save the screenshot of a control to a bitmap file

Based on the file extension, the code chooses either PNG or BMP file format.

```
uses
  SysUtils, Graphics;
...
procedure SaveControlToBitmapFile(const AFilename: TFilename);
var
  Bitmap: TFPImageBitmap;
begin
  if LowerCase(ExtractFileExt(AFilename)) = '.png' then
    Bitmap := TPortableNetworkGraphic.Create
  else
    Bitmap := TBitmap.Create;

  Bitmap.Width := ClientWidth;
  Bitmap.Height := ClientHeight;
  Bitmap.Canvas.CopyRect(ClientRect, Canvas, ClientRect);

  try
    Bitmap.SaveToFile(AFilename);
  finally
    Bitmap.Free;
  end;
end;
```

8.4 Avoid flickering in graphics programming

There are four ways to reduce flickering:

1. Use the `DoubleBuffered` property of `TWinControl` descendants: set `DoubleBuffered := true;`.
2. If your control is not transparent, include `csOpaque` in `ControlStyle`: `ControlStyle := ControlStyle + [csOpaque];`.
3. Handle the `WM_ERASEBKGND` Windows message and set `Msg.Result := 1;` in the handler.
4. Use off-screen bitmaps (like double-buffering, but works for any control)

Let's see an example of the last method:

```
uses
  Graphics;
...

procedure TForm1.Paint(Sender: TObject);
var
  bmp: TBitmap;
begin
  bmp := TBitmap.Create;
  bmp.Width := ClientWidth;
  bmp.Height := ClientHeight;
  //now draw on bmp.Canvas as you would do on TForm1.Canvas
  with bmp.Canvas do begin
    ...
  end;
  Canvas.Draw(0, 0, bmp);
  bmp.Free;
end;
```

8.5 Changing text alignment

By default, `TCanvas.TextOut(X,Y,Text)` puts the upper left corner of `Text` at (X,Y).

First method:

```
uses
  Windows;
...
TForm1.Paint(Sender: TObject);
begin
  SetTextAlign(Canvas.Handle, TA_CENTER or TA_BOTTOM);
  Canvas.TextOut(100,100,'Hello World');
  //other possibilities are:
  //TA_BASELINE, TA_BOTTOM, TA_TOP,
  //TA_LEFT, TA_RIGHT
end;
```

Second method:

```
...
TForm1.Paint(Sender: TObject);
var
  Text: string;
begin
  Text := 'Hello World';
  with Canvas do
    TextOut(100 - TextWidth(Text) div 2, 100 - TextHeight(Text), Text);
end;
```

8.6 Drawing transparent text

The backround of the text drawn onto a canvas is filled with the current brush.

```
Canvas.Brush.Style := bsClear;
Canvas.TextOut(100,100,'Hello World');
```

9 Math

9.1 Converting between decimal, binary, and hexadecimal representation of a number

```
uses
  StrUtils, SysUtils;

function DecToBin(N: int64): string;
var
  i: integer;
  neg: boolean;
begin
  if N = 0 then begin
    result := '0';
    exit
  end;

  SetLength(result, SizeOf(N)*8);
  neg := N < 0;
  N := Abs(N);
  i := 1;
  while N <> 0 do begin
    if N and 1 = 1 then
      result[i] := '1'
    else
      result[i] := '0';
    N := N shr 1;
    Inc(i);
  end;
  if neg then begin
```

```
    result[i] := '-';
    Inc(i);
  end;
  Delete(result, i, length(result));
  result := ReverseString(result);
end;

function DecToHex(N: int64): string;
begin
  if N < 0 then
    result := '-' + Format('%0x', [Abs(N)])
  else
    result := Format('%0x', [N]);
end;
```

10 Internet

10.1 Find if you are connected to the Internet

```
uses IWinInet;
...

procedure CheckConnection;
var
  dwFlags: DWORD;
begin
  if InternetGetConnectedState(@dwFlags, 0) then begin
    if (dwFlags and INTERNET_CONNECTION_MODEM)=INTERNET_CONNECTION_MODEM then
      ShowMessage('Connected through modem')
    else if (dwFlags and INTERNET_CONNECTION_LAN) = INTERNET_CONNECTION_LAN then
      ShowMessage('Connected through LAN')
    else if (dwFlags and INTERNET_CONNECTION_PROXY) = INTERNET_CONNECTION_PROXY then
      ShowMessage('Connected through Proxy')
    else if (dwFlags and INTERNET_CONNECTION_MODEM_BUSY) =
       INTERNET_CONNECTION_MODEM_BUSY then
      ShowMessage('Modem is busy');
  end else
    ShowMessage('Offline');
end;
```

11 Registry

11.1 Make a program launch at Windows startup

```
uses Registry;
...

procedure RunOnStartup(title, command : string; runOnce : boolean = false);
var
```

```
  section : string;
  reg  : TRegIniFile;
begin
  if runOnce then
    section := 'Software\Microsoft\Windows\CurrentVersion\RunOnce'
  else
    section := 'Software\Microsoft\Windows\CurrentVersion\Run';

  reg := TRegIniFile.Create( '' );
  reg.RootKey := HKEY_LOCAL_MACHINE;
  reg.WriteString(section, title, command);
  reg.Free;
end;

procedure DisableRunOnStartup(title : string; runOnce : boolean = false);
var
  section : string;
  reg  : TRegIniFile;
begin
  if runOnce then
    section := 'Software\Microsoft\Windows\CurrentVersion\RunOnce'#0
  else
    section := 'Software\Microsoft\Windows\CurrentVersion\Run'#0;

  reg := TRegIniFile.Create( '' );
  reg.RootKey := HKEY_LOCAL_MACHINE;
  reg.DeleteKey(section, title);
  reg.Free;
end;
```

In `RunOnStartup` routine, the title of the program can be arbitrary, but is needed to delete the entry. Parameter `command` is the fully qualified path and executable name of the program with possible command line parameters. Settings parameter `runOnce` to `true` makes the program launch just once (after the next boot up of Windows). Once it's launched, Windows will delete the entry from Registry.

12 Object Pascal language

12.1 Creating routines that can accept variable number of parameters

The following function creates a string representation of each element passed to it and concatenates the results into a single string.

```
uses
  SysUtils;
...

function MakeStr(A: array of const): string;
//A is a variant open array
//'array of const' is equivalent to 'array of TVarRec'
//the type of data held by TVarRec is indicated by VType field
var
```

```
  i: integer;
begin
  result := '';
  for i := Low(A) to High(A) do
    with A[i] do
      case VType of
        vtInteger:    result := result + IntToStr(VInteger);
        vtBoolean:    result := result + BoolToStr(VBoolean);
        vtChar:       result := result + VChar;
        vtExtended:   result := result + FloatToStr(VExtended^);
        vtString:     result := result + VString^;
        vtPChar:      result := result + VPChar;
        vtObject:     result := result + VObject.ClassName;
        vtClass:      result := result + VClass.ClassName;
        vtAnsiString: result := result + string(VAnsiString);
        vtCurrency:   result := result + CurrToStr(VCurrency^);
        vtVariant:    result := result + string(VVariant^);
        vtInt64:      result := result + IntToStr(VInt64^);
      end;
end;
```

13 Miscellany

13.1 Delphi's equivalent of the VB's App object

Go to Project | Options and fill in the fields in Application page and Version info page.

```
unit VbApp;

interface

type
  TApp = class
  protected
    //if the constructor is declared as protected then the
    //class can be instantiated only within this unit
    constructor Create;
  public
    Title: string;
    Path: string;
    ExeName: string;
    HelpFile: string;
    FileDescription: string;
    FileVersion: string;
    ProductName: string;
  ...
  end;

var
```

```
  App: TApp;

implementation

uses SysUtils, StrUtils, Forms, Windows, Dialogs;

type
  PLongInt = ^Longint;

constructor TApp.Create;
var
  nBytes, nInfoLen: DWORD;
  psBuffer: PChar;
  pntValue: Pointer;
  iLangID, iCharSetID: Word;
  strID: string;
begin
  ExeName := Application.ExeName;
  Path := IncludeTrailingPathDelimiter(ExtractFilePath(ExeName));
  Title := Application.Title;
  HelpFile := Application.HelpFile;

  nBytes := GetFileVersionInfoSize(PChar(ExeName), nBytes);
  if nBytes = 0 then begin
    ShowMessage('No version information available!');
    exit;
  end;
  psBuffer := AllocMem(nBytes + 1);
  GetFileVersionInfo(PChar(ExeName), 0, nBytes, psBuffer)
  VerQueryValue(psBuffer, '\VarFileInfo\Translation', pntValue, nInfoLen)
  iLangID := LoWord(PLongint(pntValue)^);
  iCharSetID := HiWord(PLongint(pntValue)^);
  strID := Format('\StringFileInfo\%.4x%.4x\', [iLangID, iCharSetID]);

  if VerQueryValue(psBuffer, PChar(strID + 'FileDescription'), pntValue, nInfoLen)
     then
    FileDescription := AnsiReplaceStr(PChar(pntValue), '\n', #13#10);
  if VerQueryValue(psBuffer, PChar(strID + 'FileVersion'), pntValue, nInfoLen) then
    FileVersion := PChar(pntValue);
  if VerQueryValue(psBuffer, PChar(strID + 'ProductName'), pntValue, nInfoLen) then
    ProductName := PChar(pntValue);
  ...
  FreeMem(psBuffer, nBytes);
end;

initialization

App := TApp.Create;

finalization
```

```
App.Free;

end.
```

Index