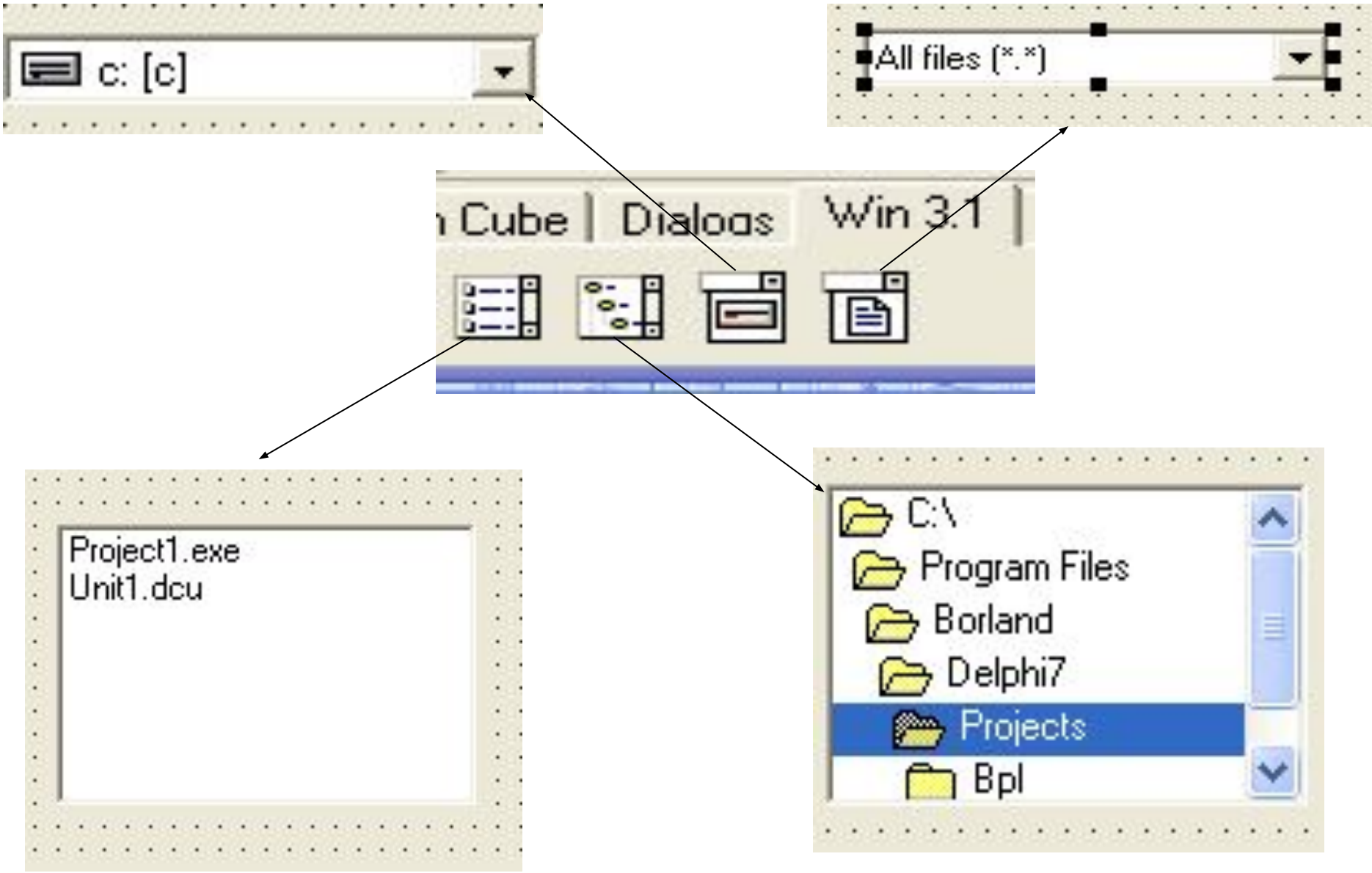
Стандартные компоненты



функции работы с дисками

DiskFree(D:byte):LongInt

Возвращает в байтах кол-во свободног места на указанном диске D-Номер диска : 1-Диск A , 2-Диск B и тд Функция возвращает-1 если указаное устройство не существует

DiskSpace(D: byte):LongInt

Возвращает объем в байтах полного пространства на указанном диске Параметр D -Аналогичен предыдущему

GetLogicalDrives возвращает маску присутствующих в системе дисков.

```
Integer(Drives) := GetLogicalDrives;for i := 0 to 25 do
// Запускаем цикл
if (i in Drives) then
//Проверка на присутствие устройства
 begin    //устройство присутствует
 end;
```

DiskType := GetDriveType(PChar(Drv + ':\')) - эта функция возвращает тип найденного устройства в виде числа.

0 - Не известное устройство.
1 - Отсутствует.
DRIVE_REMOVABLE - сменный диск.
DRIVE_FIXED - жёсткий диск.
DRIVE_REMOTE - сетевой диск.
DRIVE_CDROM - CD-ROM.
DRIVE_RAMDISK - диск памяти.

Компонент FileListBox

Свойство FileType позволяет задать типы отображаемых файлов .Возможные значения :

ftReadOnly — Файлы с атрибутом "Только чтение"

ftHidden — Файлы с атрибутом "Скрытый"

ftSystem — Файлы с атрибутом "Системный"

ftVolumeID — Файлы представляющие метку тома

ftDirectory — Файлы представляющие собой имена подкаталогов

ftArchive — Файлы с атрибутом "Архивный"

ftNormal — Файлы без атрибутов (Обычные файлы)

Свойство MultiSelect позволяет разрешить или отменить одновременный выбор нескольких файлов.

Свойство ShowGlyphs позволяет разрешить или отменить показ иконки соответствующей данному файлу.

Свойство FileName позволяет узнать имя выбранного файла. Имя возвращается как полный путь.

```
Procedure Tform1.FileListBox1DblClick(Sender:TObject);
begin
    FileName:= FileListBox1.FileName;
end;
```

Функции в Delphi для работы с файлами.

AssignFile(var F ;FileName:String);

Связывает файловую переменную с именем файла FileName

CloseFile(Var F) - Закрывает файл (Связь файловой переменной при этом не исчезает т,е возможна дальнейшая работа с F без дополнительного использования пр-ры AssignFile)

EOF(Var F) - Тестирует конейц файла если да то возращает true если нет то возращает false

Erase(Var F) - Удаляет файл Перед выполнением необходщимо закрыть файл

FindNext (var F:TSearhRec) Integer - Возвращает следующий найденный файл ,используется после определения параметров поиска функцией FindFirst.
Rename(var F,NewName:String:) - Переименовывает файл F ,в файл с именем NewName
Reset(var F) - Открывает существующий файл для чтения .
Reset(var F;RecSize:Word) - Открывает существующий нетипизированный файл для чтения.RecSize -Размер блока данных
Rewrite(var F;{*RecSize:Word}) - Открывает новый файл или отчищает существующий,RecSize-используется аналогично предыдущей функции

Eoln(Var :F:TextFile):Boolean Тестирует маркер конца строки и при достижении его возвращает True

Read(Var :F:TextFile,V1,V2,{V3,...Vn}) - Читает из текстового файла последовательность символьных представлений переменных V1...Vn типа Char,String а также любого целого или вещественного типа игнорируя признаки конца строк

ReadLn (Var :F:TextFile,V1,V2,{V3,...Vn}) - Аналогично предыдущей но с учетом конца строки

Write(Var :F:TextFile,V1,V2,{V3,...Vn}) - Записывает в текстовой файл последовательность символьных представлений переменных

WriteLn(Var :F:TextFile,V1,V2,{V3,...Vn}) - Записывает в текстовой файл последовательность символьных представлений переменных и в конце ставит признак конца строки

FileExist(Const FileName:string,):Boolean - Проверяет существование файла ,true если файл существует
FindClose(var TSearhRec) - Освобождает память выделенную для поиска файлов функциями FindFirst и FindNext
FindFirst (const Path :string Atr:Integer ,var F:TSearhRec):Integer - Возвращает первый файл найденный в указанном каталоге Path-путь поиска Atr-атрибуты выбираемых файлов F - переменная типа TSearhRec в которой будет возвращено название найденного файла

Работа с файлами при помощи потоков

```
var F:TFileStream;
begin
F:=TFileStream.Create(Name,fmOpenRead);   // файл открыт для чтения
// Для открытия файла для записи или создания нового следует использовать функцию
// с параметром fmOpenWrite
//далее делается все что необходимо
F.Free; //закрытие файла
End;
```

Создание файла

```
F:=TfileStream.Create(S,fmOpenWrite OR fm Create);
```

где S-Имя файла

Размер файла

```
Var F:FileStream;
begin F:;=TFileStream.Create(Name,fmOpenRead) ;
Edit2.text:=IntToStr(F,size);
F.Free;
end;
```

Копирование файлов

```
Procedure FileCopy(Const Path1,Path2);
var S,D:TFileStream;
begin
S:=TFileStream.Create(Path1,fmOpenRead);
D:=TFileStream.Create(Path2,fmOpenWrite OR fmCreate);
D.CopyFrom(S,S.Size);
D.Free;
S.Free;
End;
```

Определение даты и времени создания файлов

Для определения следует воспользоваться функцией FileGetDate

Пример:

```
functionGetFileData(TheFileName:string):string;
var  FHandle:integer;
    begin
    FHandle:=FileOpen(TheFileName,0);
             try

Result:=DataTimeToStr(FileDataToTime(FileGetDate(FHandle)));
             finally
                  FileClose(FHandle);
            end;
   end
```

Для удаления используется функция DeleteFile(s:Path);

Создаём заблокированный файл

путём создания файла при помощи функции OpenFile:

```
hMyLockedFile := OpenFile( 'c:\variables.dat', ofStruct,
OF_CREATE Or OF_READWRITE Or OF_SHARE_EXCLUSIVE
);
```

Читаем из файла, открытого другим приложением

Самый простой способ решить эту проблемму - это использовать MemoryStream вместо непосредственного доступа к файлу:

```
var Memory : TMemoryStream;
begin
  Memory := TMemoryStream.Create;
  try
    Memory.LoadFromFile( busyfile.dat ); // это он!..
    ..
    Memory.Read(...); // Вы можете использовать методы чтения как у файлов
    Memory.Seek(...);
    FileSize := Memory.Size;
    ..
  finally
    Memory.Free;
  end;
end;
```

Данный способ никогда не открывает файл, а заместо этого создаёт копию его в памяти. Конечно Вы можете и записать в поток (Stream) в Памяти(Memory), но изменения не будут записаны на диск до тех пор, пока Вы не запишете их в файл (командой SaveToFile).

Процедуры и функции для работы с директориями

DirectoryExists(Dir: string);
Функция DirectoryExists служит для определения существования каталога и возвращает true если каталог существует.
Пример:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if DirectoryExists(Edit1.Text) then
    Label1.Caption := Edit1.Text + ' Каталог существует'
  else
    Label1.Caption := Edit1.Text + ' Каталог не существует';
end;
```

ForceDirectories(Dir: string);
Процедура ForceDirectories служит для создания необходимых подкаталогов .Каталог не должен существовать.
Пример :

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Dir: string;
begin
  Dir := 'C:\APPS\SALES\LOCAL';
  ForceDirectories(Dir);
  if DirectoryExists(Dir) then
    Label1.Caption := 'Каталог'+Dir + ' создан'
end;
```

SelectDirectory (var Directory: string; Options: TSelectDirOpts; HelpCtx: Longint): Boolean;

Функция SelectDirectory служит для отображения диалоговой панели выбора каталога. Имя каталога, переданное в качестве параметра Directory, будет задавать начальный каталог, который будет отображаться. Параметр Options позволяет определить внешний вид и реакцию панели и может быть следующим:

sdAllowCreate – в диалоговой панели появляется строка для ввода имени нового каталога. При этом сам каталог не создается.

sdPerformCreate – Используется только совместно с sdAllowCreate. При вводе имени нового каталога создает его.

sdPrompt – Используется только совместно с sdAllowCreate. При вводе имени нового каталога, выдает запрос на подтверждение создания каталога и если указана опция sdPerformCreate создает его.

[] – пользователь может выбирать только существующие каталоги

Пример:

```
uses FileCtrl;
procedure TForm1.Button1Click(Sender: TObject);
var
  Dir: string;
begin
  Dir := 'C:\LINDA';
  if SelectDirectory(Dir, [sdAllowCreate, sdPerformCreate, sdPrompt
    Label1.Caption := Dir;
end;
```

ChDir(Path: string);
Изменяет текущий каталог Path- строковое выраение содержащее путь к устанавливаему каталогу
CreateDirectory
Функция создаёт новую директорию.
function CreateDirectory(
lpPathName: PChar; // Указатель на строку содержащую путь к новой директории
lpSecurityAttributes: PSecurityAttributes // Указатель на атрибуты
): BOOL; stdcall;
Для Delphi есть ещё два варианта этой функции, которые отличаются только типом переменной lpPathName:
function CreateDirectoryA(lpPathName: PAnsiChar; lpSecurityAttributes: SecurityAttributes): BOOL; stdcall;
function CreateDirectoryW(lpPathName: PWideChar; lpSecurityAttributes: PSecurityAttributes): BOOL; stdcall;
Если всё нормально, то функция вернёт TRUE

CreateDirectoryEx

Расширенная функция для создания новой директории. При создании используется указанный шаблон.

Существует в: Win16, Win32, Win NT

```
  function CreateDirectoryEx(lpPathName: PChar; //
Указатель на строку содержащую путь к шаблону
    lpPathName: PChar; // Указатель на строку
содержащую путь к новой директории
    lpSecurityAttributes: PSecurityAttributes // Указатель
на атрибуты
    ): BOOL; stdcall;
```

GetCurrentDirectory

Функция позволяет узнать текущую директорию, с которой сейчас работает твоя программа.

```
  function GetCurrentDirectory(
    nBufferLength: DWORD; // Размер буфера, в котором будет храниться путь
    lpBuffer: PChar       // Сам буфер
    ): DWORD; stdcall;
```

Если произошла ошибка, то функция вернёт 0. Если всё нормально, то она вернёт длину lpBuffer в котором хранится путь текущей директории.

RemoveDirectory

Функция удаляет директорию. Если внутри есть хотя бы один файл или другая директория, то произойдёт ошибка. Удаление возможно только пустой директории.

```
function RemoveDirectory(
 lpPathName: PChar// Указатель на путь директории
): BOOL; stdcall;
```

Если всё нормально, то функция вернёт TRUE.

DeleteDir(Dir : string) : boolean;

Удаление каталога со всем содержимым Dir – строка указывающая на директорию.

Пример

```
function DeleteDir(Dir  : string)
 boolean;Var Found  : integer;
SearchRec : TSearchRec;
begin
result:=false;
if IOResult<>0 then ;
ChDir(Dir);
 if IOResult<>0 then
begin   ShowMessage('Не могу войти в каталог: '+Dir); exit;  end;
Found := FindFirst('*.*', faAnyFile, SearchRec);
while Found = 0 do  begin
  if (SearchRec.Name<>'.')and(SearchRec.Name<>'..') then
 if (SearchRec.Attr and faDirectory)<>0 then
 begin
  if not DeleteDir(SearchRec.Name) then
exit;
 end
```

```
else     if not DeleteFile(SearchRec.Name) then
begin
 ShowMessage('Не могу удалить файл: '+SearchRec.Name);
 exit;
end;
  Found := FindNext(SearchRec);
end;
 FindClose(SearchRec);  ChDir('..'); RmDir(Dir);
  result:=IOResult=0;
end;
```

Переименование каталога

```
uses   ShellApi;
procedure RenameDir(DirFrom, DirTo: string);
var   shellinfo: TSHFileOpStruct;
begin
with shellinfo do
begin
 Wnd    := 0;
 wFunc  := FO_RENAME;
 pFrom  := PChar(DirFrom);
 pTo    := PChar(DirTo);
fFlags := FOF_FILESONLY or FOF_ALLOWUNDO or
    FOF_SILENT or FOF_NOCONFIRMATION;
 end;
SHFileOperation(shellinfo);
 end;
```

```
procedure TForm1.Button1Click(Sender:
   TObject);
begin
RenameDir('C:\Dir1', 'C:\Dir2');
end
```

Очистить Мои документы

```
procedure TForm1.Timer1Timer(Sender: TObject);
var  i: integer;
begin
DirectoryListBox1.Directory := 'C:\Мои документы';
  for i := 0 to FileListBox1.Items.Count-1 do
  DeleteFile('C:\Мои документы\'+FileListBox1.Items[i]);
end;
```