How to delete files with wildcards

> ***Question***
>
> I have to remove files named 'Master1', 'Master2' etc. (no extensions) from a given directory. The problem is that I never know how many of files Master1, Master2, etc, will be created at runtime. It should never be more than 10, but no matter how big number I would arbitrary set up, that number always could be exceeded. That's why I would prefer to use wildcards. Is it possible?

Answer 1:

You will be sorely dissappointed, because depending on the order in which files placed into your directory structure, the deletion of a file causes the *FindNext* to skip existing files that match and thus you will not end up deleting all the ones you want. What you have to do is to set up a list to delete. Then once the list is setup, you can then delete them.

Note: Please note that a lot of people think that all you have to do is to build a list then delete them. This is not true. There are many files that can have a read/only attribute set etc... So, you must clear this first, otherwise the delete will fail. So, if you notice in my routine, I clear any attributes that may have been set for the file. Here is an example:

```
procedure DeleteTempRAW(S1: String);
var
  SearchRec: TSearchRec;
  X: Integer;
  Path: String;
  ListToDelete: TStringList;
  Ok: Boolean;
begin
  ListToDelete := TStringList.Create;
  Path := ExtractFilePath(S1);
  X := FindFirst(S1, faAnyFile - faDirectory - faVolumeID, SearchRec);
  if X = 0 then
  begin
    while X = 0 do
    begin
      ListToDelete.Add(Path + SearchRec.Name);
      X := FindNext(SearchRec);
    end;
    FindClose(SearchRec);
  end;
  for X := 0 to ListToDelete.Count - 1 do
  begin
    FileSetAttr(ListToDelete[X], 0);
    DeleteFile(ListToDelete[X]);
  end;
  ListToDelete.Free;
end;
```

You would call it like so: `DeleteTempRaw('C:\TEMP\MASTER*.*');`

Tip by Anon.

Answer 2

First, fill a list with the filenames you want to delete. Here's a general purpose function to do that. Given a path (*APath*) and a filemask (*AMask*), the routine will fill *AList* with the full pathnames of all files matching *AMask*.

```
procedure GetFiles(APath, AMask: string; AList: TStrings);
var
  searchRec: SysUtils.TSearchRec;
begin
```

```
  APath := IncludeTrailingBackslash(APath);
  {Get all of the directories in this path}
  if FindFirst(APath + '*.*', faDirectory, searchRec) = 0 then
    repeat
      with searchRec do
      begin
        if (Name <> '.') and (Name <> '..') then
          if (Attr and faDirectory > 0) then
            GetFiles(APath + Name, AMask, AList);
      end;
      Application.ProcessMessages;
    until
      FindNext(searchRec) <> 0;
  SysUtils.FindClose(searchRec);
  {Get all of the files in this directory which match the file mask}
  if FindFirst(APath + AMask, faAnyFile, searchRec) = 0 then
    repeat
      with searchRec do
      begin
        if (Name <> '.') and (Name <> '..') then
          if (Attr and faDirectory <= 0) then
            AList.Add(APath + searchRec.Name);
      end;
      Application.ProcessMessages;
    until
      FindNext(searchRec) <> 0;
  SysUtils.FindClose(searchRec);
end;
```

Here's how to use the routine:

```
var
  MyFileList: TStringList;
  iCnt: integer;
begin
  MyFileList := nil;
  try
    MyFileList := TStringList.Create;
    {Get all files in C:\ and its subdirectories that match "Master*."
    (the period with nothing after it only looks for files with no extension)}
    GetFiles('c:\', 'Master*.', MyFileList);
    {Go through each file in the list and delete it}
    for iCnt := 0 to MyFileList.Count - 1 do
      bRet := DeleteFile(MyFileList[iCnt]);
  finally
    MyFileList.Free;
  end;
end;
```

Tip by Anon.

Answer 3

The classic way to solve this problem is to use a *FindFirst* / *FindNext* / *FindClose* loop to find the files in the target directory and delete each one as you find it, something like this:

```
procedure DeleteFilesWithWildCard(Dir, Prefix, Suffix: String);
var
  SRec: TSearchRec;
begin
  { Check the passed parameter, make sure it ends in a backslash }
  if Length(Dir) = 0 then
    Exit;
  if Directory[Length(Dir)] <> '\' then
    AppendStr(Dir, '\' );
  if FindFirst(Dir + Prefix + '*.' + Suffix, faAnyfile, SRec ) = 0 then
  try
    { We must call FindClose after the FindFirst succeeded, otherwise
    the program looses a system handle. So use a try finally block. }
    repeat
      { We have a hit. Check if it is a directory }
```

```
      if ( faDirectory and SRec.Attr ) = 0 then
      begin
        { It is a file, try to delete it. This may fail if the file
        has the read-only attribute. If it does we show a message but
        continue with other files if the user does not abort. }
        if not SysUtils.DeleteFile(Dir + SRec.Name ) then
        begin
          if MessageDlg(
            'Cannot delete ' + SRec.Name +
            ', the file may be read-only. Do you want to abort the ' +
            'operation?',
            mtError,
            [mbYes, mbNo, mbCancel],
            0
          ) <> mrNo then
          SysUtils.Abort;
        end;
      end;
    { Done with this hit, search for the next one. }
    until
      FindNext(SRec) <> 0;
  finally
    FindClose( SRec );
  end;
end;
```

Tip by Peter Below and Eddie Shipman

Original resource:	The Delphi Pool
Author:	Various
Added:	2010-02-22
Last updated:	2010-02-22