

Techniques de parcours de dossiers et de sous dossiers, récursif ou linéaire ?

par [Cédric Chatelain](#)

Date de publication : 10/03/2006

Dernière mise à jour :

En cherchant le meilleur moyen de gérer les sources Delphi de mes applications (dossiers de production et de développement) je me suis penché sur le parcours de dossiers et de sous dossiers. J'ai alors testé deux méthodes : linéaire et récursive. Les différences entre les deux, même avec un volume de données restreint comme celui que j'ai à gérer, sont sensibles.

Merci

1 - Introduction

2 - Le parcours linéaire des dossiers et sous dossiers

3 - Le parcours des dossiers et sous dossiers en utilisant une fonction récursive

4 - Conclusion

Merci

Tout d'abord je souhaite remercier [Nono40](#) pour sa QR sur le parcours des éléments d'un dossier. J'ai en effet basé mon travail sur ces informations et sur le code en exemple dans [cette QR](#) : "[Comment lister les fichiers d'un répertoire ?](#)"

Je souhaite aussi remercier [axel-erator](#) pour ses bons conseils en récursivité ainsi que [Pedro](#) pour les corrections.

1 - Introduction

Je gère, entre autre, 6 applications en Delphi. Pour chacune d'elles j'ai 2 projets : celui en cours de développement et la version en production. Cela permet bien sur, d'apporter des corrections en production sans gêner (s'il y en a) les évolutions en cours de développement. Le souci c'est que ces projets ont une arborescence assez complexe, sont composés de plusieurs paquets et que souvent dans les fichiers DPR on ne trouve plus de chemins relatifs mais des chemins en dur. Je travaille encore en Delphi 5, peut-être les versions supérieures s'en sortent t-elles mieux à ce sujet ?

Toujours est il que pour passer les évolutions d'un dossier de dev au dossier de prod ce n'est pas toujours simple. Il faut copier les sources, mais seulement les fichiers .pas, .dfm et autres sources (en dehors des .bpg, .dpr, .dof et autres .cfg). J'ai donc décidé d'automatiser cette tâche. Le principe est simple. On part d'un dossier dit de dev et d'un dossier dit de prod. On parcourt tous les sous dossiers et fichiers du dossiers de dev et on doit suivre les mêmes chemins relatifs à partir de celui de prod. On peut alors repérer si des fichiers ont été ajoutés ou modifiés.

Dans les deux cas j'ai préparé les classes suivantes :

```
type
  TFichierAMigrer = class(TPersistent)
  private
    name : string;
    directory : string;
    directorydest : string;
    size : int64;
    time : _FILETIME;
    todo : string;
    status : string;
    gridline : integer;
  public
    property Nom : string read name write name;
    property Source : string read directory write directory;
    property Destination : string read directorydest write directorydest;
    property Taille : int64 read size write size;
    property TimeStampModif : _FILETIME read time write time;
    property Action : string read todo write todo;
    property Statut : string read status write status;
    property LigneDansLeTableau : integer read gridline write gridline;
  end;

type
  TStructureDossiers = class(TPersistent)
  private
    level : integer;
    name : string;
    directory : string;
  public
    property niveau : integer read level write level;
    property Nom : string read name write name;
    property Emplacement : string read directory write directory;
  end;
```

2 - Le parcours linéaire des dossiers et sous dossiers

Le principe est simple (en ce qui concerne la conception), c'est pour ça que j'ai commencé par cette méthode. Le but est de balayer le dossier de départ, puis tous ses sous dossiers, puis les sous dossiers de niveau 2 et ainsi de suite jusqu'à ce qu'on ne trouve plus de sous dossier.

Pour cela on se sert de 2 tableaux dynamiques :

- dans le 1er on met les dossiers que l'on trouve lors de l'exploration des dossiers du niveau en cours.
- dans le second on a les dossiers du niveau en question

En pratique cela donne ceci :

Parcours des dossiers

```
function TForm1.traitedossier_(fsource : string; fdestination : string; fchemin : string):integer;
var
  Info : TSearchRec;
  Info2 : TSearchRec;
  extension : string;
  i,j,k : integer;
  action : string;
  time1 : tfiletime;
  time2 : tfiletime;
  textel, texte2 : tstrings;
  comparemethode : string;
begin
  result := 0;

  If FindFirst(fsource+fchemin+'*.*',faAnyFile,Info)=0 Then
  Begin
    Repeat
      { Les fichiers sont affichés dans ListBox1 }
      { Les répertoires sont affichés dans ListBox2 }
      If Not((Info.Attr And faDirectory)=0) then begin
        // traitement des répertoires
        // on met les infos dans le tableau des répertoires trouvés, il seront traités plus tard
        if (trim(Info.FindData.cFileName) <> '.') and (trim(Info.FindData.cFileName) <> '..') then
        begin
          result := result + 1;
          nbdossierslus := nbdossierslus + 1;
          setlength(tabdossierslus,nbdossierslus);
          TabDossiersLus[nbdossierslus-1] := TStructureDossiers.Create;
          TabDossiersLus[nbdossierslus-1].niveau := 99;
          TabDossiersLus[nbdossierslus-1].Nom := Info.FindData.cFileName;
          TabDossiersLus[nbdossierslus-1].Nom := TabDossiersLus[nbdossierslus-1].Nom + '\';
          TabDossiersLus[nbdossierslus-1].Emplacement := fchemin;
        end;
      end Else begin
        // traitement des fichiers
        action := 'none';
        //extension :=
        uppercase(copy((Info.FindData.cFileName),(length(trim(Info.FindData.cFileName))-2),3));
        extension := uppercase(gettext(Info.FindData.cFileName));
        if copy((extension),1,1) <> '~' then begin
          for i := 0 to nbextensions - 1 do begin
            if extension = uppercase(tabextension[i]) then begin
              // on cherche si le fichier existe dans le dossier prod
              If FindFirst(fdestination+fchemin+Info.FindData.cFileName,faAnyFile,Info2)=0 Then
              begin
                comparemethode := 'date';
                // si l'extension indique un fichier de type texte la comparaison se fera sur
                le contenu
                if nbttextfilesext > 0 then begin
                  for j := 1 to nbttextfilesext do begin
                    if extension = uppercase(tabtextfilesext[j-1]) then begin
```

Parcours des dossiers

```

                                comparemethode := 'contenu';
                                break;
                                end;
                                end;
                                end;
                                end;
                                if comparemethode = 'date' then begin
                                // si l'extension est prise en compte et que le fichier a été modifié il sera
                                traité
                                time1 := Info.FindData.ftLastWriteTime;
                                time2 := Info2.FindData.ftLastWriteTime;
                                if (Time1.dwHighDateTime <> Time2.dwHighDateTime) or
                                (Info.FindData.nFileSizeLow <> Info2.FindData.nFileSizeLow)
                                then
                                action := 'Remplace';
                                end else begin
                                // pour les fichiers texte on compare le contenu
                                textel := TStringList.Create;
                                texte2 := TStringList.Create;
                                textel.LoadFromFile(fsource + fchemin + info.FindData.cFileName);
                                texte2.LoadFromFile(fdestination + fchemin + Info2.FindData.cFileName);
                                if textel.Count <> texte2.Count then begin
                                action := 'Remplace';
                                end else begin
                                for k := 1 to textel.Count do begin
                                if textel.Strings[k-1] <> texte2.Strings[k-1] then begin
                                action := 'Remplace';
                                break;
                                end;
                                end;
                                end;
                                end;
                                textel.Free;
                                texte2.Free;
                                end;
                                end else begin
                                // si l'extension est prise en compte et que le fichier n'est pas présent en
                                repertoire de prod il sera traité
                                action := 'Ajout';
                                end;
                                FindClose(Info2);
                                if action <> 'none' then begin
                                // initialisation des données concernant le fichier et de l'affichage
                                //nbfichiers := nbfichiers + 1;
                                end;
                                end;
                                end;
                                end;
                                end;
                                { Il faut ensuite rechercher l'entrée suivante }
                                Until FindNext(Info) <> 0;
                                { Dans le cas où une entrée au moins est trouvée il faut }
                                { appeler FindClose pour libérer les ressources de la recherche }
                                FindClose(Info);
                                End;
                                end;

```

Cette fonction parcourt un dossier donné. Elle vérifie si les fichiers du dossier "dev" ont besoin d'être ajoutés / copiés dans le dossier prod et garde dans un tableau les dossiers qu'elle trouve pour qu'ils soient parcourus plus tard. Elle est appelée dans un boucle depuis la procédure que voici :

gestion des appels à la fonction de parcours

```

procedure TForm1.traitedossiers (psource : string; pdestination : string);
var
longueur : integer;
chemin : string;
finished : boolean;
nbrep : integer;
lnbdossiersatraiter : integer;
i : integer;
begin
    StatusBar1.SimpleText := 'Scan en cours';
    if NbTabFichiers > 0 then begin
        for i := 1 to NbTabFichiers do begin

```

gestion des appels à la fonction de parcours

```

        freeandnil(TabFichiers[i-1]);
    end;
    StringGrid1.RowCount := 2;

    StringGrid1.Cells[0,1] := '';
    StringGrid1.Cells[1,1] := '';
    StringGrid1.Cells[2,1] := '';
    StringGrid1.Cells[3,1] := '';

    NbTabFichiers := 0;
end;

longueur := length(psource);
if (copy(psource,longueur,1) <> '\') then psource := psource + '\';

longueur := length(pdestination);
if (copy(pdestination,longueur,1) <> '\') then pdestination := pdestination + '\';
nbdossierslus := 0;
finished := false;

nbdossierslus := 0;

//initialisations var pour le 1er dossier à traiter : le dossier source
chemin := '';
lnbdossiersatraiter := 0;
//début du scan des fichiers de dev...
//scan du 1er dossier (racine)
nbrep := traite dossier_(psource, pdestination, chemin);
if nbdossierslus > 0 then begin
    while finished = false do begin
        //la liste des répertoires trouvés est passée dans le tableau des répertoire à traiter
        for i := 1 to nbdossierslus do begin
            lnbdossiersatraiter := lnbdossiersatraiter + 1;
            SetLength(tabdossiersatraiter,lnbdossiersatraiter);
            tabdossiersatraiter[lnbdossiersatraiter-1] := TStructureDossiers.Create;
            tabdossiersatraiter[lnbdossiersatraiter-1].niveau := TabDossiersLus[i-1].niveau;
            tabdossiersatraiter[lnbdossiersatraiter-1].Nom := TabDossiersLus[i-1].Nom;
            tabdossiersatraiter[lnbdossiersatraiter-1].Emplacement :=
TabDossiersLus[i-1].Emplacement;
        end;
        //on vide le tableau des dossiers lus
        for i := 1 to nbdossierslus do begin
            freeandnil(TabDossiersLus[i-1]);
        end;
        setlength(TabDossiersLus,0);
        nbdossierslus := 0;
        //on va traiter les dossiers du tableau des dossiers à traiter et récupérer des
enregistrement dans les dossiers lus
        for i := 1 to lnbdossiersatraiter do begin
            chemin := tabdossiersatraiter[i-1].Emplacement + tabdossiersatraiter[i-1].Nom;
            nbrep := traite dossier_(psource, pdestination, chemin);
        end;
        //on efface les dossiers du tableau dossiers à traiter
        for i := 1 to lnbdossiersatraiter do begin
            freeandnil(tabdossiersatraiter[i-1]);
        end;
        setlength(tabdossiersatraiter,0);
        lnbdossiersatraiter := 0;
        //si on n'a pas trouvé de dossiers le traitement est fini
        if nbdossierslus = 0 then
            finished := true;
    end;
end;

    StatusBar1.SimpleText := 'Scan fini';
end;

```

On amorce la boucle en pré-remplissant la liste des dossiers à traiter avec le répertoire "racine" dev. Puis on boucle sur la liste "tabdossiersatraiter" pour scanner leur contenu. Tous les dossiers trouvés sont référencés dans la liste "TabDossiersLus". En fin de boucle on efface le contenu de "tabdossiersatraiter", on passe le contenu de "TabDossiersLus" dans "tabdossiersatraiter" et on recommence jusqu'à ce qu'on ne trouve plus aucun sous

dossier à traiter.

3 - Le parcours des dossiers et sous dossiers en utilisant une fonction récursive

C'est un collègue qui m'a conseillé de le faire. Il est vrai que ce n'est pas ma spécialité, ni mes habitudes. Personnellement, j'ai fait mon expérience en cobol sur main frame avant de faire du Delphi, et en cobol on a davantage l'habitude des méthodes linéaires... Mais c'est vrai que la récursivité allège le code.

Parcours des dossiers appelé en récursif

```

procedure TForm1.traitedossiers (psource : string; pdestination : string);
var
  longueur : integer;
  chemin : string;
  finished : boolean;
  status : integer;
  lnbddossiersatraitier : integer;
  i : integer;
begin
  StatusBar1.SimpleText := 'Scan en cours';
  initfichiers;

  longueur := length(psource);
  if (copy(psource, longueur, 1) <> '\') then psource := psource + '\';

  longueur := length(pdestination);
  if (copy(pdestination, longueur, 1) <> '\') then pdestination := pdestination + '\';
  finished := false;

  //initialisations var pour le 1er dossier à traiter : le dossier source
  chemin := '';
  lnbddossiersatraitier := 0;
  //début du scan des fichiers de dev...
  //scan du 1er dossier (racine), les sous dossier seront traités récursivement
  status := traitedossier_(psource, pdestination, chemin);

  StatusBar1.SimpleText := 'Scan fini';
end;

```

Là on voit clairement l'allègement du code. On n'appelle plus qu'une fois la fonction récursive (bien sur il y aura autant d'appels récursifs que de sous dossiers à scanner). On ne gère plus la boucle d'appels et on n'a plus à jongler avec les deux tableaux dynamiques de dossiers lus ou à traiter. Voyons la fonction en elle même.

Initialisation et déclenchement de la fonction récursive

```

function TForm1.traitedossier(fsource : string; fdestination : string; fchemin : string):integer;
Var
  Info : TSearchRec;
  Info2 : TSearchRec;
  extension : string;
  i,j,k : integer;
  action : string;
  time1 : tfiletime;
  time2 : tfiletime;
  textel, texte2 : tstrings;
  comparemethode : string;
  chemin : string;
  status : integer;
begin
  result := 0;

  If FindFirst(fsource+fchemin+'*.*', faAnyFile, Info)=0 Then
  Begin
    Repeat
      { Les fichiers sont affichés dans ListBox1 }
      { Les répertoires sont affichés dans ListBox2 }
      If Not((Info.Attr And faDirectory)=0) then begin
        // traitement des répertoires
        // on appelle la fonction en récursif
        if (trim(Info.FindData.cFileName) <> '.') and (trim(Info.FindData.cFileName) <> '..') then
          begin
            nbrepert := nbrepert + 1;
            setlength(tabrepertoires, nbrepert);
            tabrepertoires[nbrepert-1] := fdestination+fchemin+Info.FindData.cFileName+'\';

```

Initialisation et déclenchement de la fonction récursive

```

        status := traite dossier(fssource,fdestination,fchemin+Info.FindData.cFileName+'\');
    end;
end Else begin
    // traitement des fichiers
    action := 'none';
    //extension :=
uppercase(copy((Info.FindData.cFileName),(length(trim(Info.FindData.cFileName))-2),3));
extension := uppercase(gettext(Info.FindData.cFileName));
    if copy((extension),1,1) <> '~' then begin
        for i := 0 to nbextensions - 1 do begin
            if extension = uppercase(tabextension[i]) then begin
                If FindFirst(fdestination+fchemin+Info.FindData.cFileName,faAnyFile,Info2)=0 Then
begin
                    comparemethode := 'date';
                    // si l'extension indique un fichier de type texte la comparaison se fera sur
le contenu
                    if nbtextfilesext > 0 then begin
                        for j := 1 to nbtextfilesext do begin
                            if extension = uppercase(tabtextfilesext[j-1]) then begin
                                comparemethode := 'contenu';
                                break;
                            end;
                        end;
                    end;
                    if comparemethode ='date' then begin
                        // si l'extension est prise en compte et que le fichier a été modifié il sera
traité
                        time1 := Info.FindData.ftLastWriteTime;
                        time2 := Info2.FindData.ftLastWriteTime;
                        if (Time1.dwHighDateTime <> Time2.dwHighDateTime) or
                            (Info.FindData.nFileSizeLow <> Info2.FindData.nFileSizeLow)
                        then
                            action := 'Remplace';
                        end else begin
                            // pour les fichiers texte on compare le contenu
                            textel := TStringList.Create;
                            texte2 := TStringList.Create;
                            textel.LoadFromFile(fssource + fchemin + info.FindData.cFileName);
                            texte2.LoadFromFile(fdestination + fchemin + Info2.FindData.cFileName);
                            if textel.Count <> texte2.Count then begin
                                action := 'Remplace';
                            end else begin
                                for k := 1 to textel.Count do begin
                                    if textel.Strings[k-1] <> texte2.Strings[k-1] then begin
                                        action := 'Remplace';
                                        break;
                                    end;
                                end;
                            end;
                            textel.Free;
                            texte2.Free;
                        end;
                    end else begin
                        // si l'extension est prise en compte et que le fichier n'est pas présent en
repertoire de prod il sera traité
                        action := 'Ajout';
                    end;
                    FindClose(Info2);
                    if action <> 'none' then begin
                        // initialisation des données concernant le fichier et de l'affichage
                        //nbfichiers := nbfichiers + 1;
                    end;
                end;
            end;
        end;
    end;
end;
end;

{ Il faut ensuite rechercher l'entrée suivante }
Until FindNext(Info)<>0;

{ Dans le cas où une entrée au moins est trouvée il faut }
{ appeler FindClose pour libérer les ressources de la recherche }
FindClose(Info);
End;
end;

```


4 - Conclusion

A l'écriture comme à l'utilisation la récursivité fait gagner du temps. Moins de code, moins de temps dans les boucles et dans la gestion des tableaux dynamiques, moins de création et de destruction d'instances d'objets et moins d'espace mémoire à utiliser c'est tout bénéfique. Toutefois un effet de bord m'a surpris :

Avec la méthode linéaire je scannais tous les dossiers et sous-dossiers par "niveaux". Résultat, au moment de valider la copie de fichiers, la création (si besoin) de dossiers dans le répertoire cible ne posait pas de soucis car je relisais les données dans l'ordre de parcours, j'avais donc en premier les informations sur les fichiers dans les dossiers de niveaux moindre.

Avec la méthode récursive, on parcourt d'abord toute l'arborescence qu'on peut, puis on traite les fichiers et on revient ensuite progressivement vers le dossier de départ pour repartir dans une autre branche de l'arborescence. Il est donc nécessaire de garder dans un coin la trace des dossiers et sous dossiers que l'on a parcourus.

```
tabrepertoires[nbrepert-1] := fdestination+fchemin+Info.FindData.cFileName+'\\';
status := traitedossier(fsource,fdestination,fchemin+Info.FindData.cFileName+'\\');
```

On obtient donc un tableau facilement utilisable pour vérifier et recréer au besoin toute l'arborescence dans le dossier cible.