Projet 6

Henry Ledoux

21 Décembre 2001

Problème

Le problème posé est constitué de deux étapes. En premier lieu il s'agit de créer les classes nécessaires à l'élaboration d'une structure semblable à un systeme de fichier, composée d'un répertoire racine pouvant contenir un ou plusieurs répertoires, chacun de ces répertoires peut en contenir d'autres, mais chacun peut contenir un ou plusieurs fichiers, eux-mêmes contiennent une liste de chaine de caracteres.

L'implémentation de cette structure doit se faire à l'aide de listes liées, de la manière suivante :

- Un système de fichier unique contenant un répertoire seul (répertoire racine).
- Chaque répertoire est composé d'une liste simple circulaire de sous-répertoire, d'une liste simple circulaire de fichiers et d'un nom
- Chaque fichier est composé d'une liste doublement liée circulaire de chaines de caractères et d'un nom.

Il est bien entendu nécessaire de créer les méthodes permettant la manipulation du systeme de fichier (création de répertoires, fichiers et ajout de chaines en début et fin de fichier).

Dans la seconde partie du problème, on demande d'écrire deux méthodes pour le systeme de fichier : la premiere doit afficher tous les fichiers et répertoires dont le nom complet[1] contient une chaine donnée en paramètrla seconde doit afficher tous les fichiers dont le contenu (une des chaines de caracteres de la liste) contient une chaine donnée en paramètre.

Structure

Le système de fichier contient un et un seul répertoire (la racine), spécifié par *Directory *root* , et un pointeur vers le répertoire courant (dans lequel on ajoutera les fichiers grace a *InsereFileInCur()* ou encore les répertoires via *InsereRepInCur()*).

Chaque répertoire, contient une liste de répertoires désignée par *Directory* sub* contenant au moin l'élément vide[2] (qui lui ne contient pas de liste de répertoire, le pointeur sub vaut NULL), un pointeur *parent* pointant vers le répertoire parent (ceci afin de faciliter le parcours de la hiérarchie) ; et bien entendu un pointeur *next* pointant vers le répertoire suivant dans le meme niveau de sous-répertoire. Il contient également une liste (simple ciculaire) de fichiers (désignée par *File* first_file*.

Les fichiers, contiennent une liste circulaire doublement liée de chaines de caractères (*String* str_list*)

Chacun de ces éléments contient un nom (simple chaine de caractere) utilisé poru la manipulation de ceux-ci.

Le schéma ci-dessous illustre un exemple d'un tel systeme.

Description des méthodes

void File : :AddStringT(char contenu[]) et *void File : :AddStringB(char contenu[])* permettent respectivement d'ajouter une chaine de caractère en début ou en fin de fichier.

void FileSystem : :Cdd(char nom[]) modifie le répertoire courant vers le répertoire spécifié par nom (non simple, et non le nom complet).

void FileSystem : :InsereRepInCur(char nom[]) et *void InsereFileInCur(char nom[])* permettent d'insérer respectivement un répertoire ou un fichier dans le répertoire courant.

void FileSystem : :SearchName(char chain_a_chercher[]) et *void SearchStr(char chaine_a_chercher[])* recherchent tous les fichiers dont le nom (pour SearchName) ou le contenu (pour SearchStr) contient la chaine_a_chercher.

void FileSystem : :DispCur() affiche le répertoire courant a l'écran.

[1] Le nom complet est l'ensemble du chemin d'accès et du nom du fichier.

[2] Premier élément de la liste, qui n'est pas considéré ou élément "bidon"

Exemple d'execution

Voici un exemple d'utilisation du programme :

```
void main()
{
    FileSystem* fsys;
    fsys = new FileSystem();

    fsys->InsereRepInCur("essai1"); // Cree un repertoire
    fsys->Cdd("essai1"); // Se place dans le repertoire essai1
    fsys->InsereRepInCur("essai2");
    fsys->Cdd("essai2");
    fsys->InsereRepInCur("essai3");
    fsys->Cdd("essai3");
    fsys->InsereRepInCur("boul");
    fsys->OpenFile("boul")->AddStringT("bilili"); // Ajoute une chaine en édbut de fichier
    fsys->Cdd("..");
    fsys->InsereRepInCur("essai4");
    fsys->Cdd("essai4");
    fsys->InsereFileInCur("testfile");
    fsys->InsereFileInCur("mooh");
    fsys->OpenFile("testfile")->AddStringT("ijlikiklazersf");
    fsys->OpenFile("testfile")->AddStringB("milili");
    cout << "Recherche le mot il dans le contenu des fichiers" << endl;
    fsys->SearchStr("il");

    cout << "Recherche les fichiers dont le nom contient oh" << endl;
    fsys->SearchName("oh");

    cout << "Affiche toute la éhirarchie" << endl;

    fsys->SearchName("");
}
```

Sortie écran :

```
Recherche le mot il dans le contenu des fichiers
/essai1/essai2/essai4/testfile
Recherche les fichiers dont le nom contient oh
/essai1/essai2/essai4/mooh
Affiche toute la hiérarchie
/essai1/
/essai1/essai2/
/essai1/essai2/essai4/mooh
/essai1/essai2/essai4/testfile
/essai1/essai2/essai4/
/essai1/essai2/essai3/
/essai1/essai2/essai3/boul/
```

Le code

```
#include <iostream.h>
#include <string.h>
class String;
class File;
class Directory;
class FileSystem;
// Cette fonction verifie si la chaine str2 est inclue dans str1.
bool check_if_inc(char str1[], char str2[])
{
   int i=0, j=0, sl = strlen(str2);
```

```
   while (str1[i] != '\0' && j < sl)
   {
      if (str1[i] == str2[j])
         ++j;
      else
      {
         i -= j;
         j = 0;
      }
      ++i;
   }
   return (sl == j);
}

// Classe d'element de liste de string
class String
{
   public:
      friend File;
      friend FileSystem;
   private:
      String(char ninfo[], String* nnext, String* nprev);
      String();
      String* next;
      String* prev;
      char* info;
};

// Classe fichier. (Chaque fichier contient une liste de String)
class File
{
   public:
      friend Directory;
      friend FileSystem;
      void AddStringT(char content[]);
      void AddStringB(char content[]);
   private:
      File(char nname[], File* nnext);
      File();
      File* next;
      String* str_list;
      char* name;
};

// Classe repertoire (Chaque repertoire contient une liste de repertoire et une liste de fichier)
class Directory
{
   public:
      friend FileSystem;
   private:
      Directory(char nname[], Directory* nnext, Directory* nparent);
      Directory(Directory* nparent);
      void NewFile(char nname[]);
      void NewDir(char nname[]);
      File* first_file;
      Directory* next;
      Directory* sub;
      Directory* parent;
      char* name;
};

// Classe filesystem (N'est pasé élment d'une liste,
```

```
// elle contient un érpertoire: le repertoire racine.
class FileSystem
{
   public:
      FileSystem();
      void Cdd(char gtname[]);
      void InsereRepInCur(char nname[]);
      void InsereFileInCur(char nname[]);
      void SearchName(char sname[]);
      void SearchStr(char sname[]);
      void DispCur();
      void DispRep(Directory* tbd);
      File* OpenFile(char oname[]);
      Directory* root;
      Directory* current;
};

// Constructeur pour les chaines de caracteres: en parametre: (contenu, suivant, precedent)
String::String(char ninfo[], String* nnext, String* nprev)
{
    info = new char[strlen(ninfo)];
    strcpy(info,ninfo);
    next=nnext;
    prev=nprev;
};

// Constructeur pour l'element bidon.
String::String()
{
      info = NULL;
}

// Constructeur de fichier: en parametre (nom, fichier_suivant)
File::File(char nname[], File* nnext)
{
   str_list = new String;
   str_list->next = str_list;
   str_list->prev = str_list;
   next = nnext;
   name = new char[strlen(nname)];
   strcpy(name,nname);
}

// Constructeur pour lé'élment bidon des listes de fichiers.
File::File()
{
   name = NULL;
}

// Methode pour ajouter une chaine en debut de fichier
void File::AddStringT(char content[])
{
   str_list->next = new String(content,str_list->next,str_list);
   str_list->next->next->prev = str_list->next;
}

// Idem pour ajouter en fin de fichier
void File::AddStringB(char content[])
{
   str_list->prev = new String(content,str_list,str_list->prev);
   str_list->prev->prev->next = str_list->prev;
}
```

```
// Constructeur pour les repertoires: en param: (nom, rep_suivant, rep_parent)
Directory::Directory(char nname[], Directory* nnext, Directory* nparent)
{
   first_file = new File();
   first_file->next = first_file;
   name = new char[strlen(nname)];
   strcpy(name,nname);
   sub = new Directory(this);
   sub->next=sub;
   next = nnext;
   parent = nparent;
}

// Constructeur pour l'element bidon des listes de repertoires.
Directory::Directory(Directory* nparent)
{
   sub = NULL;
   first_file = NULL;
   name = NULL;
   parent = nparent;
}

// Cree un fichier dans le repertoire
void Directory::NewFile(char nname[])
{
   first_file->next = new File(nname,first_file->next);
}

// Cree un sous repertoire dans le repertoire
void Directory::NewDir(char nname[])
{
   sub->next = new Directory(nname, sub->next, this);
}

// Constructeur du systeme de fichier
FileSystem::FileSystem()
{
   root = new Directory("root", NULL, NULL);
   root->next=root;
   current = root;
}

// Va dans le sous repertoire dont le nom est dans gtname
void FileSystem::Cdd(char gtname[])
{
   Directory* curs = current->sub;
   if (strcmp("..",gtname) != 0)
   {
      curs->name = new char[strlen(gtname)];
      strcpy(curs->name, gtname);
      do {
         curs = curs->next;
      } while(strcmp(curs->name, gtname) != 0);
      delete[] current->sub->name;
      current->sub->name = NULL;
      if (curs != current->sub)
         current = curs;
   }
   else
   {
      current = current->parent;
```

```
    }
}

// Insere un repertoire dans le rep courant.
void FileSystem::InsereRepInCur(char nname[])
{
    current->NewDir(nname);
}

// Affiche le repertoire courant.
void FileSystem::DispCur()
{
    DispRep(current);
}

// Affiche le nom complet d'un repertoire donne
void FileSystem::DispRep(Directory* tbd)
{
    char ret[256] = "";
    int tlen, rlen=0;
    Directory* pos = tbd;
    while(pos != root)
    {
        tlen=strlen(pos->name);
        for (int i=rlen+1; i>=0; --i)
        {
            ret[i + tlen + 1] = ret[i];
        }
        rlen+=tlen;
        strncpy(ret + 1, pos->name, tlen);
        ret[0] = '/';
        pos = pos->parent;
    }
    cout << ret << endl;
}

// Ajoute un fichier dans le repertoire courant
void FileSystem::InsereFileInCur(char nname[])
{
    current->NewFile(nname);
}

// Renvoie l'adresse du fichier nome oname
File* FileSystem::OpenFile(char oname[])
{
    File* pf = current->first_file;
    pf->name = new char[strlen(oname)];
    strcpy(pf->name ,oname);
    do {
        pf = pf->next;
    } while(strcmp(pf->name, oname) != 0);
    delete[] current->first_file->name;
    current->first_file->name = NULL;
    if (pf == current->first_file)
        pf = NULL;
    return pf;
}

// Affiche tous les fichiers dont le nom complet contient sname
void FileSystem::SearchName(char sname[])
{
    Directory* x = root->sub->next;
```

```
    File* ptf;
    while(x != root)
    {
        ptf = x->first_file;
        if (x->first_file != NULL)
        {
            char ret[256] = "/";
            int tlen, rlen=1;
            Directory* pos = x;
            while(pos != root)
            {
                tlen=strlen(pos->name);
                for (int i =rlen+1; i >=0; --i)
                {
                    ret[i + tlen + 1] = ret[i];
                }
                rlen+=tlen+1;
                strncpy(ret + 1, pos->name, tlen);
                ret[0] = '/';
                pos = pos->parent;
            }
            do
            {
                ptf = ptf->next;
                if (ptf->name != NULL)
                    for (int q = 0; q <= strlen(ptf->name); ++q)
                    {
                        ret[rlen+q] = ptf->name[q];
                    }

                if (check_if_inc(ret,sname))
                {
                    cout << ret << endl;
                }
                ret[rlen] = '\0';
            } while(ptf->name != NULL);
        }
        if (x->sub == NULL)
        {
            x = x->parent;
            x = x->next;
        }
        else
        {
            x = x->sub->next;
        }

    }
}

// Affiche tous les fichiers dont le contenu contient sname
void FileSystem::SearchStr(char sname[])
{
    Directory* x = root->sub->next;
    File* ptf;
    while(x != root)
    {
        ptf = x->first_file;
        if (x->first_file != NULL)
        {
            char ret[256] = "/";
            int tlen, rlen=1;
```

```
            Directory* pos = x;
            while(pos != root)
            {
               tlen=strlen(pos->name);
               for (int i =rlen+1; i >=0; --i)
               {
                  ret[i + tlen + 1] = ret[i];
               }
               rlen+=tlen+1;
               strncpy(ret + 1, pos->name, tlen);
               ret[0] = '/';
               pos = pos->parent;
            }
            do
            {
               ptf = ptf->next;

               if (ptf->name != NULL)
               {
                  String* stra = ptf->str_list;
                  for (int q = 0; q <= strlen(ptf->name); ++q)
                  {
                     ret[rlen+q] = ptf->name[q];
                  }
                  do
                  {
                     stra = stra->next;
                     if (stra->info != NULL && check_if_inc(stra->info,sname))
                     {
                        cout << ret << endl;
                     }
                  } while (stra->info != NULL);
                  ret[rlen] = '\0';
               }
            } while(ptf->name != NULL);
         }
         if (x->sub == NULL)
         {
            x = x->parent;
            x = x->next;
         }
         else
         {
            x = x->sub->next;
         }

      }
}
```