Système de gestion de fichiers (SGF)

Introduction

Le but de ce projet est de simuler le fonctionnement d'un système de gestion de fichiers. Ce système de gestion de fichiers est la partie du système d'exploitation qui permet de manipuler des fichiers et des répertoires à l'aide d'un ensemble d'opérations.

Première partie

1 Système de gestion de fichiers

1.1 Exemple

L'exemple ci-dessous décrit un répertoire racine nommé / avec deux sous-répertoires Rep1 et Rep2. Le répertoire Rep2 contient un fichier et un sous-répertoire. Le répertoire Rep21 est vide.

```
/
|
|--------- Rep1
|         |--------- fic1.txt
|         |--------- fic2.ads
|         |--------- Rep11
|                   |------- fic2.txt
|--------- Rep2
           |--------- fic4.c
           |--------- Rep21
```

On parle de hiérarchie de répertoires et de fichiers.

1.2 Opérations

Dans la suite, le terme "répertoire courant" correspond au répertoire en cours d'utilisation. De plus, l'accès à un fichier (ou un répertoire) se fait par un nom relatif (../../rep1/rep2/rep3/fich1) ou un nom absolu (/rep1/rep2/rep3/fich1).

Dans un SGF, il est possible d'effectuer les traitements suivants :

1. la création d'un fichier dans le répertoire courant
2. l'affichage du contenu d'un fichier
3. l'affichage de la liste des répertoires et fichiers du répertoire courant
4. le changement de répertoire courant vers un autre répertoire
5. la copie d'un fichier du répertoire courant dans un autre répertoire
6. la suppression d'un répertoire ou d'un fichier dans le répertoire courant
7. la suppression de tous les fichiers et répertoires du répertoire courant (et ceci de manière récursive)
8. le changement de nom d'un répertoire ou d'un fichier dans le répertoire courant
9. la recherche de tous les fichiers du répertoire courant qui contiennent une chaîne de caractères donnée
10. la recherche de tous les fichiers du répertoire courant et de ses sous-répertoires qui contiennent une chaîne de caractères donnée

1.3 Représentation interne

Pour ce projet, nous nous intéressons à la représentation d'un ensemble de fichiers et/ou de répertoires à l'aide d'une structure d'arbre n-aire. Nous implanterons ensuite quelques opérations de manipulation des fichiers et (ou) des répertoires parmi celles mentionnées ci-dessus. Les informations ci-dessous vous permettront de bien définir vos structures de données :

1. La racine de l'arbre représentant le SGF définit le répertoire racine. Les nœuds de l'arbre sont des répertoires.
   Chaque répertoire, racine comprise, est caractérisé par les informations suivantes :
   – un nom de répertoire,
   – un ensemble de fichiers
   – un ensemble de répertoires fils.
   – le père du répertoire
   En effet, le parcours du SGF nécessite l'accès au répertoire père (excepté pour la racine).
2. L'ensemble des fichiers d'un répertoire sera représenté par la structure de données de votre choix. Chaque fichier a un nom (au plus 10 caractères lettre ou chiffre ou _) et une extension de trois caractères (lettre). Pour ce projet on supposera que tous les fichiers sont simulés par des chaînes de caractères (ou tableau de caractères) de taille inférieure ou égale à 100.
3. Le SGF sera donc représenté par un arbre n-aire (c'est-à-dire un arbre dont les nœuds peuvent avoir un nombre quelconque de fils) (Figure 1).

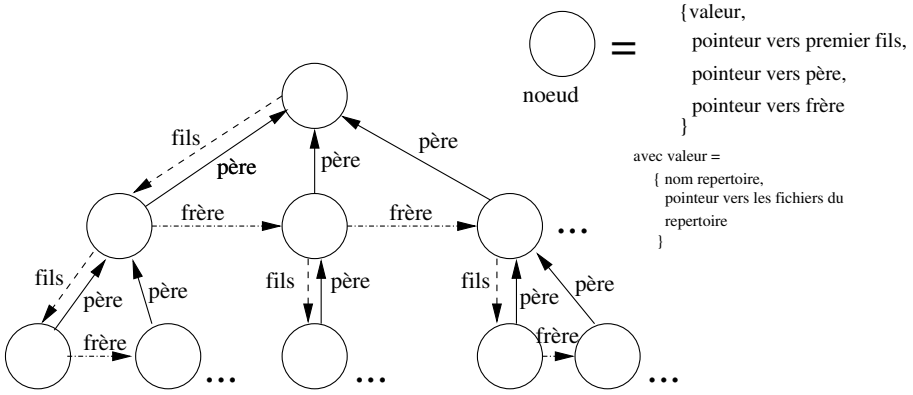

FIGURE 1 – L'implantation choisie d'un arbre n-aire.

2 Travail demandé : navigation

2.1 Paquetage Arbre-naire

On demande d'implanter et de tester un paquetage 'p_arbren' **générique** permettant de manipuler des arbres n-aires de type quelconque dont la spécification est donnée en annexe.

2.2 Paquetage 'p_sys'

On désire créer un paquetage 'p_sys' permettant de définir et de manipuler le SGF. On définira uniquement les fonctionnalités 1, 2, 3, 4, 5, 6, 8 et 10.

1. Spécifier le paquetage 'p_sys'
2. Implanter le paquetage 'p_sys' en instanciant le paquetage 'p_arbren'
3. Ecrire un programme de test des fonctionnalités définies dans 'p_sys', sous forme d'un menu.

Deuxième partie

3 Le contrôle d'accès

3.1 Définition

On souhaite définir un contrôle d'accès à un SGF. Pour cela, on supposera l'existence d'un ensemble fixé d'utilisateurs du SGF (identifiés par des numéros). Dans cette partie, un utilisateur ne pourra exécuter les opérations implantées dans la partie 1 que s'il en a le droit.

– On définit plusieurs types de droits sur un fichier (ou un répertoire) : le droit de lecture, et le droit en écriture. Ainsi, on pourra associer des droits de lecture ou des droits d'écriture, à un utilisateur, pour un fichier (ou un répertoire) donné.
– Un utilisateur est dit propriétaire des fichiers (répertoires) qu'il crée. Il a sur ces fichiers, tous les droits.
– Tout utilisateur qui a un droit sur un fichier (ou un répertoire) peut l'octroyer à un ou plusieurs autres utilisateurs.
– Tout utilisateur qui a octroyé un droit à un (ou plusieurs) utilisateur(s) U (U_i) peut le (les) lui retirer. Dans ce cas, le droit sera aussi retiré à tous ceux qui l'avaient acquis de U (U_i) et ainsi de suite.

Chaque utilisateur crée des fichiers (répertoires) organisés hiérarchiquement et situés sous la racine / . Chaque utilisateur possède une et une seule hiérarchie fixée de répertoires et de fichiers (voir dessin ci-dessous).

```
/
|---- Rep-U1
|     |
|        ...
|---- Rep-U2
|     |
|        ...
  ...

|---- Rep-Un
      |
         ...
```

dans lequel le répertoire Rep-U_i est la propriété de l'utilisateur U_i.

Remarque : Les utilisateurs U_i n'ont aucun droit sur la racine /.

En résumé, les commandes associées à cette gestion du contrôle d'accès aux fichiers sont :
– l'octroi de droits (r ou w) à un ou plusieurs utilisateurs u_1, u_2, $\ldots u_k$ sur un ou plusieurs fichiers (ou répertoires) f_1, f_2, $\ldots$, f_k du répertoire courant
– le retrait de droits (r ou w) à un ou plusieurs utilisateurs u_1, u_2, $\ldots u_k$ sur un ou plusieurs fichiers (ou répertoires) f_1, f_2, $\ldots$, f_k du répertoire courant

4 Travail demandé

1. Spécifier un paquetage p_sys_users permettant de représenter, en plus des fichiers et répertoires, les droits des utilisateurs. Il faudra redéfinir les opérations de la première partie qui nécessitent d'être modifiées pour prendre en compte la politique de sécurité induite par les règles définies ci-dessus.
2. Implanter ce paquetage en ADA.
3. Ecrire un programme de test des nouvelles fonctionnalités définies dans 'p_sys_users', sous forme d'un menu.

Remarque : vous avez la possibilité de spécifier et d'implanter tout paquetage supplémentaire qui vous semblera nécessaire à la réalisation du programme.

5 Conseils et documents à rendre

- La conception du programme sera faite en utilisant la méthode des raffinages. Les raffinages seront écrits en langage algorithmique. Le programme sera écrit en ADA. D'autre part, vous devez respecter les normes de programmation élaborées pendant les séances de CTD et disponibles sur moodle.
- La notation de votre projet tiendra fortement compte du respect des normes de programmation et de la qualité de vos raffinages.
- Des séances de TP sont prévues à l'emploi du temps avec la présence d'un enseignant qui pourra répondre à vos questions. Vous pouvez aussi poser des questions sur le forum de moodle.
- Vous devez rendre :
  - Un rapport au format PDF qui comportera :
    - un sommaire,
    - une introduction présentant brièvement le sujet,
    - les choix de conception que vous avez effectués et leur justification et toute explication permettant au correcteur de comprendre votre travail,
    - les différents raffinages,
    - une conclusion avec les limites de votre programme, les difficultés rencontrées, une estimation du temps passé sur ce projet, ...
  - une archive avec les sources ADA (pas les binaires !)
- Les documents numériques sont à envoyer par mail à votre enseignant de TP pour le vendredi 24 janvier à 18 h.

Annexes

Le fichier spécification du paquetage *pnaire* :

```
-----------------------------------------------------------------------
-- Opérations sur un arbre n-aire (de type arbren)
-- L'information rangée dans le noeud est de type T
-- NB : Les PRE et POST conditions ne sont volontairement
-- pas précisées ici. De même pour les EXCEPTIONS.
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-- Fonction An_Vide
-- Sémantique: Détecter si un arbre n-aire est vide ou non
-- Paramètres: a: arbren (D)
-- Type retour: booléen (vaut vrai si l'arbre n-aire est vide)
function An_Vide (a: in arbren) return boolean ;
-----------------------------------------------------------------------

-----------------------------------------------------------------------
-- Fonction An_Creer_Vide
-- Sémantique: Créer un arbre n-aire vide
-- Paramètres: /
function An_Creer_Vide return arbren ;
-----------------------------------------------------------------------

-----------------------------------------------------------------------
```

```
-- Fonction An_Valeur
-- Sémantique: Retourner la valeur rangée à la racine d'un arbre n-aire
-- Paramètres: a: arbren (D)
-- Type retour: T
function An_Valeur (a: in arbren) return T ;
-------------------------------------------------------------------------

-------------------------------------------------------------------------
-- Fonction An_Est_Feuille
-- Sémantique: Indiquer si un arbre n-aire est une feuille (pas de fils)
-- Paramètres: a: arbren (D)
-- Type retour: booléen (vaut vrai si l'arbre n-aire n'a pas de fils)
function An_Est_Feuille (a: in arbren) return boolean ;
-------------------------------------------------------------------------

-------------------------------------------------------------------------
-- Fonction An_Creer_Feuille
-- Sémantique: Créer un arbre n-aire avec une valeur mais sans fils,
-- ni frère, ni père
-- Paramètres: nouveau: T (D)
-- Type retour: arbren
function An_Creer_Feuille (nouveau: in T) return arbren ;
-------------------------------------------------------------------------

-------------------------------------------------------------------------
-- Fonction An_Pere
-- Sémantique: Retourner l'arbre n-aire père d'un arbre n-aire
-- Paramètres: a: arbren (D)
-- Type retour: arbren
function An_pere (a: in arbren) return arbren ;
-------------------------------------------------------------------------

-------------------------------------------------------------------------
-- Fonction An_Fils
-- Sémantique: Retourner le nieme fils de a
    -- le numero 1 est le premier fils
-- Paramètres: a: arbren (D)
--             numero: integer (D)
-- Type retour: arbren
function An_fils (a: in arbren; numero : IN INTEGER) return arbren ;
-------------------------------------------------------------------------

-------------------------------------------------------------------------
-- Fonction An_Frere
-- Sémantique: Retourner le nieme  Frere d'un arbre n-aire
    -- le numero 1 est le premier frere
-- Paramètres: a: arbren (D)
--             numero: integer (D)
-- Type retour: arbren
function An_Frere (a: in arbren; numero : IN INTEGER) return arbren ;
-------------------------------------------------------------------------

-------------------------------------------------------------------------
-- Procédure An_Afficher
-- Sémantique: Afficher le contenu complet d'un arbre n-aire
-- Paramètres: a: arbren (D)
procedure An_Afficher (a: in arbren) ;
-------------------------------------------------------------------------

-------------------------------------------------------------------------
-- Fonction An_Rechercher
-- Sémantique: Recherche la première occurrence d'une valeur dans un
-- arbre n-aire. Retourne l'arbre n-aire dont la valeur est racine
-- si elle est trouvée, un arbre n-aire vide sinon
-- Paramètres: a: arbren (D), data: T (D)
-- Type retour: arbren
function An_Rechercher (a: in arbren; data: in T) return arbren ;
-------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------
-- Fonction An_Est_Racine
-- Sémantique: Indiquer si un arbre n-aire est sans père
-- Paramètres: a: arbren (D)
-- Type retour: booléen (vaut vrai si l'arbre n-aire n'a pas de père)
function An_Est_Racine (a: in arbren) return boolean ;
-----------------------------------------------------------------------

-----------------------------------------------------------------------
-- Procédure An_Changer_Valeur
-- Sémantique: Changer la valeur rangée à la racine d'un arbre n-aire
-- Paramètres: a: arbren (D), nouveau: T (D)
procedure An_Changer_Valeur (a:in arbren; nouveau:in T) ;
-----------------------------------------------------------------------

-----------------------------------------------------------------------
-- Procédure An_Inserer_Fils
-- Sémantique: Insérer un arbre n-aire sans frère en position de premier
-- fils d'un arbre n-aire a. L'ancien fils de a devient alors le premier
-- frère de l'arbre n-aire inséré.
-- Paramètres: a: arbren (D), a_ins: arbren (D)
procedure An_Inserer_Fils (a:in arbren; a_ins:in arbren) ;
-----------------------------------------------------------------------

-----------------------------------------------------------------------
-- Procédure An_Inserer_Frere
-- Sémantique: Insérer un arbre n-aire sans frère en position de premier
-- frère d'un arbre n-aire a
-- Paramètres: a: arbren (D), a_ins: arbren (D)
procedure An_Inserer_Frere (a:in arbren; a_ins:in arbren) ;
-----------------------------------------------------------------------

-----------------------------------------------------------------------
-- Procédure An_Supprimer_fils
-- Sémantique: Supprime le nieme fils d'un arbre n-aire
-- Paramètres: a: arbren (D)
--             numero: integer (D)
procedure An_Supprimer_fils (a:in arbren; numero : IN INTEGER) ;
-----------------------------------------------------------------------

-----------------------------------------------------------------------
-- Procédure An_Supprimer_frere
-- Sémantique: Supprime le nieme frere d'un arbre n-aire
-- Paramètres: a: arbren (D)
--             numero: integer (D)
procedure An_Supprimer_Frere (a:in arbren; numero : IN INTEGER) ;
-----------------------------------------------------------------------
```