



**RAPPORT DU PROJET LONG
DE PROGRAMMATION IMPERATIVE
LANGAGE : Ada
SYSTEME DE GESTION DE FICHIERS
(S.G.F).**

**Fait par : EL MAHI Moncef.
Encadré par : Mme CHAMBON Sylvie.**

23 mai 2014

Sommaire

Introduction	3
Spécifications générales	3
Cahier des charges	3
I Navigation	4
1 Le système de gestion de fichiers.	5
1.1 Exemple de système de gestion de fichiers.	5
1.2 Opérations possibles dans un système de gestion de fichiers.	5
2 Paquetage générique p-arbren	6
2.1 Spécifications du paquetage générique p-arbren	6
2.1.1 Représentation d'un arbre n-aire	6
2.1.2 Différentes fonctions et procédures du paquetage générique p-arbren	6
2.2 Conception du paquetage générique p-arbren	7
2.2.1 Choix du type	7
2.2.2 Structuration des données génériques	7
2.2.3 Définition des fonctions du paquetage générique p-arbren	8
2.2.4 Raffinages de certaines fonctions et procédures	12
2.3 Validation du paquetage générique p-arbren	16
2.3.1 Tests	16
2.3.2 Conclusion 1	17
3 Paquetage p_sys	18
3.1 Spécifications du paquetage p_sys	18
3.1.1 Définition des fonctionnalités du paquetage p_sys	18
3.2 Conception du paquetage p_sys	18
3.2.1 Choix du type	18
3.2.2 Définition des fonctions du paquetage p_sys	19
3.2.3 Raffinages de certaines fonctions et procédures	22
3.3 Validation du paquetage p_sys	26
3.3.1 Interface de ce système	26
3.3.2 Tests	26
3.3.3 Limites de mon paquetage	30
3.3.4 Conclusion 2	30
II Le contrôle d'accès	31
4 Le contrôle d'accès	32
4.1 Définition	32

5	Paquetage <code>p_sys_users</code>	33
5.1	Spécifications du paquetage <code>p_sys_users</code>	33
5.2	Conception du paquetage <code>p_sys_users</code>	34
5.2.1	Choix du type et des structures de données	34
5.2.2	Définition des fonctions du paquetage <code>p_sys_users</code>	35
5.2.3	Raffinages de certaines fonctions et procédures	39
5.3	Validation du paquetage <code>p_sys_users</code>	41

Introduction

Le système de gestion de fichiers permet à des utilisateurs la manipulation, ainsi que le contrôle d'accès des fichiers et des répertoires, ce qui le rend la partie la plus utilisée dans un système d'exploitation.

Le but de ce projet est alors de simuler le fonctionnement d'un système de gestion de fichiers.

Spécifications générales

Dans ce paragraphe, je vais reprendre les différentes conceptions des objets manipulés au cours de ce projet :

Les répertoires : c'est un arbre n-aire décrivant une certaine hiérarchie et dont chaque noeud représente un répertoire qui contiendrait le nom du répertoire ainsi que les différents fichiers du répertoire.

Les fichiers : Identifiés par leurs noms, leurs extensions ainsi que leurs contenus. (Nom : chaîne de caractère de taille inférieure ou égale à 10, extension : chaîne de 3 caractères, contenu : chaîne de caractère de taille inférieure ou égale à 100).

Les utilisateurs : Chaque utilisateur est identifié par un numéro, a le droit de créer des fichiers, ce qui le rend propriétaire du fichier et qui lui permet d'octroyer les droits ou de lecture ou d'écriture (des fichiers qu'il a créé) à tous les autres utilisateurs. Il est initialement propriétaire d'un seul répertoire.

Les droits : On distingue principalement deux droits : Le droit de lecture (Consultation du contenu des fichiers) et le droit d'écriture (Modification du contenu des fichiers).

Cahier des charges

Les deux principaux critères du cahier des charges sont les suivants :

- Nous devons écrire un algorithme qui réalise toutes les fonctions demandées tout en prenant en compte la complexité temporelle et spatiale (But : rendre un algorithme de complexité optimale).
- Ces fonctions ainsi que les arguments qu'elles prennent doivent être compréhensibles par l'utilisateur afin de faciliter l'utilisation du programme et la collecte des données (arguments) sur un terminal.

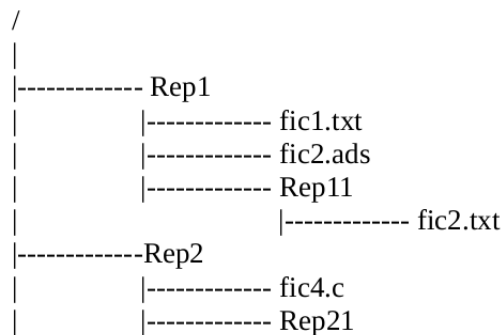
Première partie

Navigation

1 Le système de gestion de fichiers.

1.1 Exemple de système de gestion de fichiers.

L'exemple ci dessous nous montre 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.



1.2 Opérations possibles dans un système de gestion de fichiers.

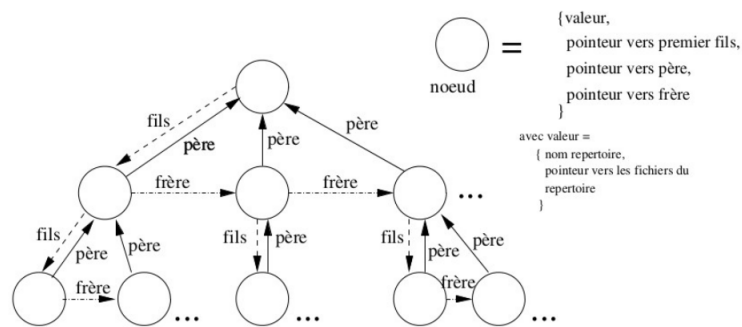
Les opérations possibles dans un système de gestion de fichiers sont :

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.

2 Paquetage générique p-arbren

2.1 Spécifications du paquetage générique p-arbren

2.1.1 Représentation d'un arbre n-aire



2.1.2 Différentes fonctions et procédures du paquetage générique p-arbren

1. fonction An_Vide (a : (*Donnée*) arbren) retourne boolean ;
2. fonction An_Creer_Vide retourne arbren ;
3. fonction An_Valeur (a : (*Donnée*) arbren) retourne T ;
4. fonction An_Est_Feuille (a : (*Donnée*) arbren) retourne boolean ;
5. fonction An_Creer_Feuille (nouveau : (*Donnée*) T) retourne arbren ;
6. fonction An_pere (a : (*Donnée*) arbren) retourne arbren ;
7. fonction An_fils (a : (*Donnée*) arbren ; numero : (*Donnée*) entier) retourne arbren ;
8. fonction An_Frere (a : (*Donnée*) arbren ; numero : (*Donnée*) entier) retourne arbren ;
9. procédure An_Afficher (a : (*Donnée*) arbren) ;
10. fonction An_Rechercher (a : (*Donnée*) arbren ; data : (*Donnée*) T) retourne arbren ;
11. fonction An_Est_Racine (a : (*Donnée*) arbren) retourne boolean ;
12. procédure An_Inserer_Fils (a : (*Donnée/Résultat*) arbren ; a_ins : (*Donnée/Résultat*) arbren) ;
13. procédure An_Inserer_Frere (a : (*Donnée/Résultat*) arbren ; a_ins : (*Donnée/Résultat*) arbren) ;
14. fonction An_Nombre_Fils (a : (*Donnée*) arbren) retourne entier ;
15. procédure An_Supprimer_fils (a : (*Donnée/Résultat*) arbren ; numero : (*Donnée*) entier) ;
16. procédure An_Supprimer_Frere (a : (*Donnée/Résultat*) arbren ; numero : in entier) ;

2.2 Conception du paquetage générique p_arbren

2.2.1 Choix du type

A partir de l'implantation choisie pour un arbre n aire représenté dans le schéma précédent, on peut définir le type arbre n-aire comme suit :

Type *arbren* :

```

1 type noeud;
2 type arbren est access noeud;
3 type noeud est record
4     val:T;
5     premier_fils: arbren;
6     frere: arbren;
7     pere: arbren;
8 end record;
```

2.2.2 Structuration des données génériques

Le type T

Le type T est déclaré comme un type privé. C'est un paramètre générique du paquetage p_arbren, c'est à dire qu'il va être substitué par différents types en fonction du contexte de l'utilisation. Dans la suite du projet, on verra que pour valider le paquetage p_arbren par exemple, on va affecter à T le type entier.

```

1 generic
2 Type T est privé;
```

Procédure générique Ecrire

La procédure générique écrire prend un paramètre de type T. Elle sera définie pour chaque type que l'on affecte à T.

```

1 generic
2 with procédure écrire(a : (*Donnée*) T) ;
```

Exemple dans le cas où T est un entier :

```

1 procedure écrire_int(a : (*Donnée *) entier) est
2 Début
3 écrire( " |-----" ) ;
4 écrire(a,2) ;
5 fin écrire_int ;
```

Procédure générique Egalité

La procédure générique égalité sert à comparer deux éléments de type T. On la définira systématiquement au moment de l'instanciation du paquetage générique p_arbren.

```

1 with fonction egalite( a : (*Donnée*) T ; b (*Donnée*) T) retourne Boolean ;
```


Exemple dans le cas où T est un entier :

```

1  function egalite_int( a : (*Donnée*) entier ; b : (*Donnée*) entier)
2  retourne boolean est :
3  Début
4  retourne(a=b) ;
5  fin egalite_int ;

```

2.2.3 Définition des fonctions du paquetage générique p-arbren

le paquetage p-arbren nous permettra de développer les fonctions permettant de manipuler les arbres n aires.

Spécifications du paquetage p-arbren.ads

```

1  generic
2  type T is private;
3  with procedure ecrire (a: in T);
4  with function egalite(a: in T; b: in T) return Boolean;
5
6  package p-arbren is
7
8  type arbren is private;
9
10 — exception
11 arbre_vide: exception;
12 no_pere: exception;
13 no_nieme_fils: exception;
14 no_frere: exception;
15
16 — Fonction An_Vide
17 — Semantique: Detecter si un arbre n-aire est vide ou non
18 — Parametres: a: arbren (D)
19 — Type retour: boolean (vaut vrai si l'arbre n-aire est vide)
20 — pre : aucune
21 — post: retourne true si l'arbre est vide
22 — exeption : aucune
23 function An_Vide (a: in arbren) return boolean ;
24
25
26 — Fonction An_Creer_Vide
27 — Semantique: Creer un arbre n-aire vide
28 — Parametres: aucun
29 — pre : aucune
30 — post: on obtient true si on applique la fonction An_Vide
31 — exeption : aucune
32 function An_Creer_Vide return arbren ;
33
34
35 — Fonction An_Valeur
36 — Semantique: Retourner la valeur rangee a la racine d'un arbre n-aire
37 — Parametres: a: arbren (D)
38 — Type retour: T
39 — pre : aucune
40 — post: aucune
41 — exeption : arbre vide
42 function An_Valeur (a: in arbren) return T ;

```

```

43
44
45 — Fonction An_Est_Feuille
46 — Semantique: Indiquer si un arbre n-aire est une feuille (pas de fils)
47 — Parametres: a: arbren (D)
48 — Type retour: boolean (vaut vrai si l'arbre n-aire n'a pas de fils)
49 — pre : aucune
50 — post : retourne vrai si l'arbre a n'a pas de fils
51 — exeption : arbre vide
52 function An_Est_Feuille (a: in arbren) return boolean ;
53
54
55 — Fonction An_Creer_Feuille
56 — Semantique: Creer un arbre n-aire avec une valeur mais sans fils ,
57 —           ni frere , ni pere
58 — Parametres: nouveau: T (D)
59 — Type retour: arbren
60 — pre : aucune
61 — post : en appliquant An_Est_Feuille cela nous ernvoit vrai
62 — exeption : aucune
63 function An_Creer_Feuille (nouveau: in T) return arbren ;
64
65
66 — Fonction An_Pere
67 — Semantique: Retourner l'arbre n-aire pere d'un arbre n-aire
68 — Parametres: a: arbren (D)
69 — Type retour: arbren
70 — pre : aucune
71 — post : aucune
72 — exeption : arbre vide
73 function An_pere (a: in arbren) return arbren ;
74
75
76 — Fonction An_Fils
77 — Semantique: Retourner le nieme fils de a
78 —           le numero 1 est le premier fils
79 — Parametres: a: arbren (D)
80 —           numero: integer (D)
81 — Type retour: arbren
82 — pre : numero >= 1
83 — post : aucune
84 — exeption : arbre vide
85 function An_fils (a: in arbren; numero : IN INTEGER) return arbren ;
86
87
88 — Fonction An_Frere
89 — Semantique: Retourner le nieme Frere d'un arbre n-aire
90 —           le numero 1 est le premier frere
91 — Parametres: a: arbren (D)
92 —           numero: integer (D)
93 — Type retour: arbren
94 — pre : numero >= 1
95 — post : aucune
96 — exeption : arbre vide
97 function An_Frere (a: in arbren; numero : IN INTEGER) return arbren ;
98

```

```

99
100 — Procédure An_Afficher
101 — Semantique: Afficher le contenu complet d'un arbre n-aire
102 — Parametres: a: arbren (D)
103 — pre : aucune
104 — post : aucune
105 — exeption : arbre vide
106 procedure An_Afficher (a: in arbren) ;
107
108
109 — Fonction An_Rechercher
110 — Semantique: Recherche la premiere occurrence d'une valeur dans un
111 — arbre n-aire. Retourne l'arbre n-aire dont la valeur est racine
112 — si elle est trouvee, un arbre n-aire vide sinon
113 — Parametres: a: arbren (D), data: T (D)
114 — Type retour: arbren
115 — pre : aucune
116 — post : aucune
117 — exeption : arbre vide
118 function An_Rechercher (a: in arbren; data: in T ) return arbren ;
119
120
121 — Fonction An_Est_Racine
122 — Semantique: Indiquer si un arbre n-aire est sans pere
123 — Parametres: a: arbren (D)
124 — Type retour: boolean (vaut vrai si l'arbre n-aire n a pas de pere)
125 — pre : aucune
126 — post : retourne vrai si a n'a pas de pere
127 — exeption : arbre vide
128 function An_Est_Racine (a: in arbren) return boolean ;
129
130
131 — Procédure An_Changer_Valeur
132 — Semantique: Changer la valeur rangee a la racine d'un arbre n-aire
133 — Parametres: a: arbren (D), nouveau: T (D)
134 — pre : aucune
135 — post : valeur de a changee par nouveau
136 — exeption : arbre vide
137 procedure An_Changer_Valeur (a:in out arbren; nouveau:in T ) ;
138
139
140 — Procédure An_Inserer_Fils
141 — Semantique: Insérer un arbre naire sans frere en position de premier
142 — fils d'un arbre n-aire a. L'ancien fils de a devient alors le
143 — premier frere de l'arbre n-aire insere.
144 — Parametres: a: arbren (D), a_ins: arbren (D)
145 — pre : aucune
146 — post : arbre avec un nouveau fils
147 — exeption : arbre vide
148 procedure An_Inserer_Fils (a:in out arbren; a_ins:in out arbren) ;
149
150
151 — Procédure An_Inserer_Frere
152 — Semantique: Insérer un arbre naire sans frere en position de premier
153 — frere d'un arbre n-aire a
154 — Parametres: a: arbren (D), a_ins: arbren (D)

```

```

155 — pre : aucune
156 — post : arbre avec un nouveau frere
157 — exeption : arbre vide
158 procedure An_Inserer_Frere (a:in out arbren; a_ins: in out arbren) ;
159
160
161 — fonction An_Nombre_Fils: retourne le nombre de fils au premier
162 — niveau d'un arbre
163 — parametres: a arbre n-aire
164 — post-conditions: si l'arbre est null, on retourne 0
165 function An_Nombre_Fils ( a: in arbren ) return integer;
166
167
168 — Procedure An_Supprimer_fils
169 — Semantique: Supprime le nieme fils d'un arbre n-aire
170 — Parametres: a: arbren (D)
171 —          numero: integer (D)
172 — pre : aucune
173 — post : arbre avec un fils en moins si numero est inferieur au nbr
174 — de fils
175 — exeption : arbre vide
176 procedure An_Supprimer_fils (a:in out arbren; numero : IN INTEGER) ;
177
178
179 — Procedure An_Supprimer_frere
180 — Semantique: Supprime le nieme frere d'un arbre n-aire
181 — Parametres: a: arbren (D)
182 —          numero: integer (D)
183 — pre : aucune
184 — post : arbre avec un fils en moins si numero est inferieur au nbr
185 — de frere
186 — exeption : arbre vide
187 procedure An_Supprimer_Frere ( a :in out arbren ; numero: in integer);
188
189
190 private — tous ces types sont privés.
191
192
193 — Operations sur un arbre n-aire (de type arbren)
194 — L information rangée dans le noeud est de type T
195
196 type noeud;
197 type arbren is access noeud;
198 type noeud is record
199     val:T;
200     premier_fils: arbren;
201     frere: arbren;
202     pere: arbren;
203 end record;
204
205 end p-arbren;

```

2.2.4 Raffinages de certaines fonctions et procédures

Par souci d'efficacité, je ne décomposerai que les fonctions qui sont un peu complexe, j'éviterai ainsi de raffiner les fonctions et les procédures dont l'algorithme est licite.

An_rechercher :

R0 :

-
- Fonction An_Rechercher
 - Sémantique : Recherche de la première occurrence d'une valeur dans un 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
 - pre : aucune
 - post : aucune
 - exeption : arbre vide
-

fonction An Rechercher ((*D*) A : Arbren ; (*D*) Data :T) retourne Arbren

R1 :

- Rechercher l'elt data et ensuite arrêter la recherche.
 - Retourner l'arbre contenant l'élément data.
-

R2 :

–cas terminal
si a est vide **alors**
 on lève l'expection
sinon
 si a.all.val correspond à data **alors**
 on retourne a
 –cas general
 sinon
 –rechercher l'elt data et arreter juste après
 –on fait appel a une procédure qui prend en paramètre un booléen qui nous permettra de continuer ou arreter les recherches au cas où data n'existe pas.
 continue ← vrai – on initialise le booléen qui indique que la recherche continue à vrai
 arb rech ← null – on initialise l'arbre recherché à l'arbre vide – on retourne un seul arbre contenant la valeur data sinon null.
 An Recherche Aux(a,data,arb rech,continue)
 –retourner l'arbre contenant l'elt data
 retourne arb rech
 fin si
fin si

Résolution de An_Rechercher_Aux :**R0 :**

-
- An Rechercher Aux : parcourt l'arbre tant que le boolean qu'elle prend en parametre est vrai (i.e : l'elt recherché n'est pas retrouvé ,et une fois l'elt retrouvé dans un arbre,on garde ce dernier dans un paramètre du meme type.
 - paramètres : a :arbren (*D*), Data :T (*D*), arb rech :arbren (*R*),
 - continue :boolean (*D/R*)
 - précondition : aucune
 - postcondition : si data retrouvé alors arb rech prend une valeur , sinon il ne prend aucune valeur.
 - exeption : aucune
-

Procedure An Rechercher Aux((**D**)a : arbren ; (**D**) data :T ; (**D/R**) continue : boolean ; (**R**) arb rech :arbren)

R1 :

- Vérifier si l'arbre est vide (cas terminal).
 - Vérifier si on a égalité entre la valeur du noeud courant est l'élément recherché auquel cas le boolean devient négatif.
 - Vérifier si le booléen est positif auquel cas, on continue la recherche dans la lignée des fils d'abord et ensuite dans celle des frères.
-

R2 :

- Cas terminal
 - Si** a non vide **alors**
 - on ne fait rien
 - Vérifier si égalité
 - sinon si** a.all.val correspond à data **alors**
 - continue ← faux
 - arb rech ← a
 - sinon**
 - verifie si boolean vrai, puis on recherche parmi les fils d'abord
 - si** continue=vrai **alors**
 - appel récursif sur le 1er fils
 - An Rechercher Aux (Arb.All.Fils,Data,Arb Rech,Continue)
 - sinon** rien
 - fin si**
 - verifie si boolean est toujours vrai, ensuite on recherche parmi les frères
 - si** continue=vrai **alors**
 - appel récursif sur le frère
 - An Rechercher Aux (Arb.All.Frere,Data,Arb Rech,Continue)
 - sinon** rien
 - fin si**
 - fin si**
-

An_Inserer_Fils :**R0 :**

-
- Procedure An Inserer Fils
 - Semantique : Insérer un arbre n-aire sans frere en position de premier fils d'un arbre n-aire a. L'ancien fils de a devient alors le premier frere de l'arbre n-aire insere.
 - Parametres : a : arbren (D), a ins : arbren (D)
 - pre : aucune
 - post : arbre avec un nouveau fils
 - exeption : arbre vide
-

procedure An Inserer Fils ((*D*)A :Arbren;(*D*) A Ins :Arbren)

R1 :

- Vérifier si l'arbre à insérer est vide.
 - Définir 1er fils comme étant le frère de a ins.
 - Relier a ins au pere de a.
 - Insérer a ins comme étant 1er fils.
-

R2 :

– cas terminal
si a est vide **alors**
 on lève l'expection arbre vide.
sinon
 – si l'arbre à inserer est vide alors rien
 si a ins est vide **alors**
 rien
 sinon
 inter ins ← a ins
 Inter ← a
 – définir 1er fils comme étant le frère de a ins
 inter ins.all.frere ← inter.all.fils
 – relier a ins au pere de a
 inter ins.all.pere ← inter
 –insérer a ins comme étant 1er fils
 inter.all.fils ← inter ins
 fin si
fin si

An_Supprimer_Frere :**R0 :**

-
- Procedure An Supprimer frere
 - Semantique : Supprime le nieme frere d'un arbre n-aire
 - Parametres : a : arbren (*D*) / numero : integer (*D*)
 - pre : aucune
 - post : arbre avec un fils en moins si numero est inferieur au nbre de frere
 - exeption : arbre vide
-

procedure An Supprimer Frere ((*D*)A : Arbren ;(*D*)Numero :entier)

R1 :

- Traiter cas terminal.
 - Supprimer le n-ème frère.
 - Relier les autres parties de l'arbre pour ne pas perdre d'élèments.
-

R2 :

– cas terminal
si a est vide **alors**
 on lève l'expection arbre vide
sinon
 si Numero=1 **alors**
 inter1 \leftarrow A
 – on relie les arbres par un lien de fraternité direct
 inter1.all.frere \leftarrow inter1.all.frere.all.frere
 sinon
 – on cherche l'arbre le (n-1)eme frere
 inter1 \leftarrow Inter1 :=AnFils(A, Numero – 1)
 – on prend le (n+1)eme frere
 inter2 \leftarrow Inter2 :=Inter1.All.Frere.All.Frere
 – on supprime le nieme frère
 inter1.all.frere.all.pere \leftarrow null
 inter1.all.frere.All.frere \leftarrow null
 – on relie les 2 arbres se trouvant à ces cotés
 pour garder les autres elts de l'arbre initial
 inter1.all.frere \leftarrow Inter2
 fin si
fin si

2.3 Validation du packaging générique p_arbren

2.3.1 Tests

```

|----- 0
  |-----15
    |-----67
      |-----27
        |-----12
          |----- 9
            |-----18
              |-----15
                |----- 7
                  |-----94
                    |-----87
                      |----- 2
                        |-----74
                          |----- 1
                            |-----12
                              |----- 3
                                |-----17
                                  |-----77
                                    |-----32
                                      |----- 5
                                        |-----18
                                          |----- 6

```

```

menu
1- Detecter si un arbre n aire est vide ou pas
2- Creer un arbre n aire vide
3- Retourner la valeur rangee a la racine d'un arbre n aire
4- Indiquer si un arbre n aire est une feuille (Pas de fils)
5- Creer un arbre n aire avec une valeur mais sans fils ni frere ni pere
6- Retourner l arbre n aire pere d un arbre n aire
7- Retourner le n ieme fils de a
8- Retourner le n ieme frere d un arbre n aire
9- Afficher le contenu complet d'un arbre n aire
10- Rechercher la premiere occurence d une valeur dans un arbre n aire
11- Indiquer si un arbre n aire est sans pere
12- Changer la valeur rangee a la racine d un arbre n aire
13- Insérer un arbre n aire sans frere en position de premier fils d un arbre n aire a
14- Insérer un arbre n aire sans frere en position de premier fils d un arbre n aire a
15- Supprimer le n ieme fils d un arbre n aire
16- Supprimer le n ieme frere d un arbre n aire
0- Pour sortir du programme

```

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 1
l arbre n est pas vide

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 2
l'arbre vide est cree

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 3
La valeur stockee dans la racine de cet arbre n aire est: 0

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 4
l'arbre n est pas une feuille, il a par consequent au moins un fils

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 5
Quelle est la valeur que vous voulez stocker dans votre arbre n aire (sans fils, ni pere ni frere) 9
la feuille que vous voulez représenter est: |----- 9

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 6

raised TEST_P_ARBREN.TEST1.NO_PERE : p_arbren.adb:108 instantiated at test_p_arbren.adb:12

```

Veuillez choisir une fonctionnalite parmi celles qui figurent sur le menu 7
Quel est le numero du fils de l arbre a que vous voulez retourner
2
|----- 7
|-----94
|-----87
|----- 2
|-----74
|----- 1
|-----12

```

```

Veuillez choisir une fonctionnalite parmi celles qui figurent sur le menu 10
Quel est l element que vous voulez rechercher dans l arbre? 3
|----- 3
|-----17
|-----77
|-----32
|----- 5
|-----18
|----- 6

```

```

Veuillez choisir une fonctionnalite parmi celles qui figurent sur le menu 15
Quel est le numero du fils que vous voulez supprimer? 1
|----- 0
|----- 7
|-----94
|-----87
|----- 2
|-----74
|----- 1
|-----12
|----- 3
|-----17
|-----77
|-----32
|----- 5
|-----18
|----- 6

```

```

Veuillez choisir une fonctionnalite parmi celles qui figurent sur le menu 16
Quel est le numero du frere que vous voulez supprimer? 1
raised TEST_P_ARBREN.TEST1.NO_FRERE : p_arbren.adb:462 instantiated at test_p_arbren.adb:12

```

2.3.2 Conclusion 1

Après avoir fait les tests, on peut conclure que le paquetage générique `p_arbren` marche bien, donc on peut faire appel à ses fonctions dans le nouveau paquetage `p_sys` en effectuant une instantiation, ceci nous évitera de redéfinir à nouveau les mêmes fonctionnalités.

On pourra aussi exploiter le paquetage sur les listes chaînées fait précédemment en TP en le transformant en un paquetage générique, ceci nous évitera de reprogrammer à nouveau ses fonctions.

3 Paquetage p_sys

3.1 Spécifications du paquetage p_sys

3.1.1 Définition des fonctionnalités du paquetage p_sys

- Créer un fichier dans le répertoire courant.
- Afficher le contenu d'un fichier.
- Afficher la liste des répertoires et des fichiers du répertoires courant.
- Changer le répertoire courant vers un autre répertoire.
- Copier un fichier du répertoire courant dans un autre répertoire.
- Supprimer un répertoire dans le répertoire courant.
- Supprimer un fichier dans le répertoire courant.
- Renommer un répertoire dans le répertoire courant.
- Renommer un fichier dans le répertoire courant.
- Rechercher tous les fichiers du répertoire courant et de ses sous répertoires qui contiennent une chaîne de caractères donnée.

3.2 Conception du paquetage p_sys

3.2.1 Choix du type

Un fichier est défini par son nom (chaîne de caractère dont la taille maximale est de 10 caractères), son contenu (taille maximale de 100 caractères), son extension (3 caractères) et pointe vers le fichier qui le suit, si c'est le dernier de la liste, il pointe vers null. Le type choisi pour représenter un fichier est le suivant :

```
1 type fichiers;  
2 type fich is access fichiers;  
3 type fichiers is record  
4   nom_du_fichier: string(1..kmax);  
5   extension: string(1..emax);  
6   contenu_du_fichier: string(1..cmax);  
7   suivant: fich;  
8 end record;
```

Un répertoire est un noeud de l'arbre n-aire, il est défini par son nom (chaîne de caractères de taille maximale égale à 10 caractères), il contient des sous répertoires qui sont ses fils (structure d'arbre n-aire) et une liste de fichiers. Le type choisi pour représenter un répertoire est alors le suivant :

```

1 type repertoire is recordtant que
2   nom_du_repertoire: string(1..kmax);
3   fichiers_du_repertoire: fich;
4 end record;
```

3.2.2 Définition des fonctions du paquetage p_sys

le paquetage p_sys va contenir les fonctions gérant les fichiers dans l'arbre n-aire.

Spécifications du paquetage p_sys.ads

```

1 with p_arbren;
2
3 package p_sys is
4
5
6 Nmax: constant Integer:=100;
7 kmax: integer :=10; -- taille maximale du nom.
8 emax: integer :=3; -- taille de l'extension.
9 cmax: integer :=100; -- taille maximale du contenu.
10
11
12 -- Déclaration du type fichier.
13 type fichiers;
14 type fich is access fichiers;
15 type fichiers is record
16   nom_du_fichier: string(1..kmax);
17   extension: string(1..emax);
18   contenu_du_fichier: string(1..cmax);
19   suivant: fich;
20 end record;
21
22
23 type repertoire is record
24   nom_du_repertoire: string(1..kmax);
25   fichiers_du_repertoire: fich;
26 end record;
27
28
29 -- SPECIFICATION
30
31
32 -- la procedure generique ecrire_txt.
33 procedure ecrire_txt(a: in repertoire);
34
35
36 -- La fonction générique égalité.
37 function egalite_repertoire(a: in repertoire; b: in repertoire) return
38 boolean;
39
40
41 package p_arbren_test is new p_arbren(repertoire , ecrire_txt ,
```

```

42 egalite_repertoire);
43 use p_arbren_test;
44
45
46
47 —creer: cree un fichier dans le repertoire courant
48 —param : a:arbren (*D*), nom_fich:string(1..Kmax) (*D*) ,
49 —        contenu_fich:string(1..Nmax) (*D*)
50 —pre:taille du nom_fich inferieur a Kmax,contenu_fich inferieur a 100
51 —post : aucune
52 —exemption : aucune
53 procedure creer_fichier(a0: in out arbren;nom_fichier: in string;
54 contenu_fichier: in string; extension_du_fichier: in string);
55
56
57 —contenu_fichier : affiche le contenu d un fichier
58 —param : fichiers (*D*)
59 —pre : aucune
60 —post :aucune
61 —exemption: fichier inexistant
62 function contenu_fichier(f: in fich) return string;
63
64
65 — 3' Pour l'affichage de la liste des repertoires et fichiers du
66 — repertoire courant.
67 — Semantique: afficher la liste des repertoires et fichiers du
68 — repertoire courant.
69 — Parametres: v valeur (dans laquelle est stocke le nom du repertoire
70 — courant ainsi que les fichiers du repertoire).
71 — Pre-conditions: Le repertoire n est pas le repertoire vide.
72 — Post-conditions: La liste des repertoires et fichiers du repertoire
73 — courant est affichee.
74 procedure Afficher_rep_fich(a: in arbren);
75
76
77 — 4' Pour le changement d'un repertoire courant vers un autre
78 — repertoire.
79 — Semantique: changer le repertoire courant vers un autre repertoire.
80 — Parametres: Le repertoire courant(in/out valeur) et le repertoire
81 — cible (in valeur)
82 — Pre-conditions: Le repertoire courant n esst pas le repertoire vide.
83 — Post conditions: le repertoire courant bascule vers le repertoire
84 — cible.
85 function changer_repertoire(arbre_total: in arbren; nom_rep_cible:
86 in string)
87 return arbren;
88
89
90 function recherche_fichier(a: in arbren; nom: string; ext: string)
91 return fich;
92
93
94 —copie:copier un fichier du repertoire actuel vers un autre repertoire
95 — param : a:arbren (*D*), nom_rep_nouveau: string(1..Kmax) (*D*),
96 —        nom:string(1..Kmax) (*D*).
97 — pre : taille des chaines de caracteres inferieur a Kmax

```

```

98  — post : copie du fichier s'il existe sinon rien
99  — exeption : fichier inexistant
100 function copie_fichier(a0:in arbren;a:in arbren;nom_fichier:in string;
101   nom_rep_d : in string ;ext: in string) return arbren;
102
103
104 procedure supprimer_fichier (a: in arbren; nom : in string;
105   ext: in string);
106
107
108 —supprime : supprime un repertoire ou un fichier dans le repertoire
109 — courant
110 — param : a:arbren (*D*), nom_elt: string(1..Kmax) (*D*)
111 — pre : aucune
112 — post : supprime l'elt sauf s'il n'existe pas
113 — exeption : arbre vide
114 procedure supprimer_repertoire(a: in out arbren;numero1: in integer);
115
116
117 —renomme : change le nom d'un repertoire du repertoire courant
118 — param : a:arbren (*D*),
119 —         nom_nouveau:string(1..Kmax) (*D*)
120 — pre : taille des chaines de caracteres inferieur a Kmax
121 — post : renomme l'elt s'il existe
122 — exeption : arbre vide
123 procedure changer_nom_rep (a: in out arbren; nouveau_nom : in string);
124
125
126 procedure renommer_fichier(a0: in out arbren;
127   nom_repertoire_courant: in string; nom_fichier_a_renommer : in string;
128   nouveau_nom_fichier : in string; ext1: in string; ext2: in string);
129
130
131 —contient:verifie si une chaine de caractere est contenu dans une
132 — autre chaine .
133 — param : chaine et sa taille :string+integer , la chaine recherchee
134 —         et sa taille :string+integer
135 — pre : aucune
136 — post : retourne vrai si elle y est
137 — exeption : aucune
138 FUNCTION Contient (Chaine_Contenu:IN String;L_Contenu: IN Integer;
139   Chaine_Rech: IN String;L_Rech:IN Integer) RETURN Boolean;
140
141
142 —recherche_fichiers:recherche les fichiers du repertoire courant et de
143 —
144 —         ses sous-repertoires qui contiennent une chaine
145 —         de caracteres donnee
146 — param : a: arbren (*D*), chaine:string (*D*), J:integer (*D*)
147 — pre : aucune
148 — post : retourne les noms des fichiers ou rien sinon
149 — exeption : aucune
150 PROCEDURE Recherche_Fichiers(A:IN Arbren ; Chaine:IN String ;
151   J : IN Integer );
152
153 end p_sys;

```

3.2.3 Raffinages de certaines fonctions et procédures

Copie

R0 :

-
- copie : copier un fichier du repertoire actuel vers un autre repertoire
 - param : a :arbren (*D*), nom rep nouveau : string(1..Kmax) (*D*) / nom :string(1..Kmax) (*D*).
 - pre : taille des chaines de caracteres inferieur a Kmax
 - post : copie du fichier s'il existe sinon rien
 - exeption : aucune
-

procedure Copie((*D*)A :Arbren; (*D*)Nom Rep Nouveau :String; (*D*)Nom :String)

R1 :

- Rechercher le fichier à copier.
 - Changer de répertoire.
 - Créer fichier du meme nom et contenu.
-

R2 :

inter le 1er fichier du repertoire courant

- rechercher le fichier à copier

tant que inter/=null **et alors** egalite(inter.all.Nom,Nom) **faire**

Inter ← Inter.All.Suiv

fin tant que

si inter=null **alors**

rien (fichier inexistant)

sinon

- on change de repertoire vers le repertoire destination(rep courant) de la copie.

changement(a,nom rep nouveau,rep courant)

- on cree un fichier avec le meme nom et contenu dans ce repertoire.

creer(rep courant,inter.all.nom,inter.all.contenu)

fin si

Supprimer fichier**R0 :**

-
- supprimer : supprime un repertoire ou un fichier dans le repertoire courant
 - param : a :arbren (*D*), nom elt : string(1..Kmax) (*D*)
 - pre : aucune
 - post : supprime le repertoire sauf s'il n'existe pas
 - exeption : arbre vide
-

procedure Supprime((*D/R*)A :Arbren ; (*D*)Nom Elt :String)

R1 :

- Rechercher l'élément à supprimer parmi les fichiers. - Rechercher (au cas où ce n'est pas un fichier) parmi les sous-répertoires du repertoire courant. - Finir par supprimer l'élément dans les deux cas.

R2 :

–cas terminal
si a est vide **alors**
 on lève exeption
sinon
 inter est le 1er fichier du repertoire courant.
 –rechercher l'élément à supprimer parmi les fichiers
 tant que Inter/=NULL **et alors** egalite(Inter.All.Nom,Nom Elt) **faire**
 inter ← inter.all.suiv
 fin tant que
 si inter/=NULL **alors**
 –on supprime le fichier (car il existe)
 Inter ← Inter.All.Suiv
 sinon
 – Rechercher (au cas où ce n'est pas un fichier) parmi les sous-répertoires du repertoire courant.
 –on supprime un sous repertoire
 i ← 1 –on initialise le compteur.
 –rechercher (au cas où ce n'est pas un fichier) parmi les sous-répertoires du repertoire courant
 tant que le ième fils n'est pas vide **et alors** egalite(An Valeur((An fils(a,i))).all.nom,nom elt)**faire**
 i ← i+1
 fin tant que
 si le ième fils est vide **alors**
 ecrire("il n'y'a pas d'elt à supprimer")
 sinon
 –on supprime le ième fils
 An Supprimer Fils(A,I)
 fin si
 fin si
fin si

appartient :

La fonction appartient est une fonction auxiliaire qui va être utilisée dans la fonction Rechercher chaîne de caractère contenue dans un fichier du répertoire courant ou de ses sous répertoires.

R0 :

-
- contient : vérifie si une chaîne de caractère est contenue dans une autre chaîne .
 - param : chaîne et sa taille : string+integer , la chaîne recherchée et sa taille :string+integer
 - pre : aucune
 - post : retourne vrai si elle y est
 - exception : aucune
-

fonction Contient ((*D*)Chaîne Contenu :String;(*D*)
L_Contenu :entier; (*D*)Chaîne Rech :String;(*D*)L_Rech :entier) retourne Booleen

R1 :

- compter le nombre d'éléments similaires entre les deux chaînes.
 - Vérifier si la chaîne contenu correspond du premier coup.
 - Décaler sinon la recherche d'un caractère (le premier).
-

R2 :

-cas terminal
si L Contenu L Rech **alors**
 retourner faux
sinon
 i ← 1 -on initialise le compteur
 -compter le nombre d'éléments similaires entre les deux chaînes
 tant que Chaîne Contenu(i)=Chaîne Rech(i) **faire**
 i ← i+1
 fin tant que
 -vérifier si la chaîne contenu correspond du premier coup
 si i ≥ L Rech **alors**
 retourner vrai
 sinon
 -on decale d'une position(1 carac.) pour rechercher a nouveau
 pour M de 2 à L Contenu **faire**
 Chaîne2(M) ← Chaîne Contenu(M)
 fin pour
 -ajuster la chaîne2
 pour M de 1 à L Contenu-1 **faire**
 Chaîne2(M) ← Chaîne2(M+1)
 fin pour
 -on recherche dans chaîne2
 retourne Contient(Chaîne2,L Contenu-1,Chaîne Rech,L Rech)
 fin si
fin si

Rechercher_fichiers**R0 :**

-
- rechercher fichiers : recherche les fichiers du repertoire courant et de ses sous-repertoires qui contiennent une chaine de caractères donnée.
 - param : a : arbren (*D*), chaine :string (*D*), J :integer (*D*).
 - pre : aucune.
 - post : retourne le nom des fichiers ou rien sinon.
 - exeption : aucune.
-

procedure Recherche Fichiers((**D**)A :Arbren ; (**D**)Chaine :String ; (**D**)J :entier)

R1 :

- Afficher le nom des fichiers du répertoire courant contenant la chaine de caractères.
 - Afficher le nom des fichiers des sous-répertoire contenant la chaine de caractères.
-

R2 :

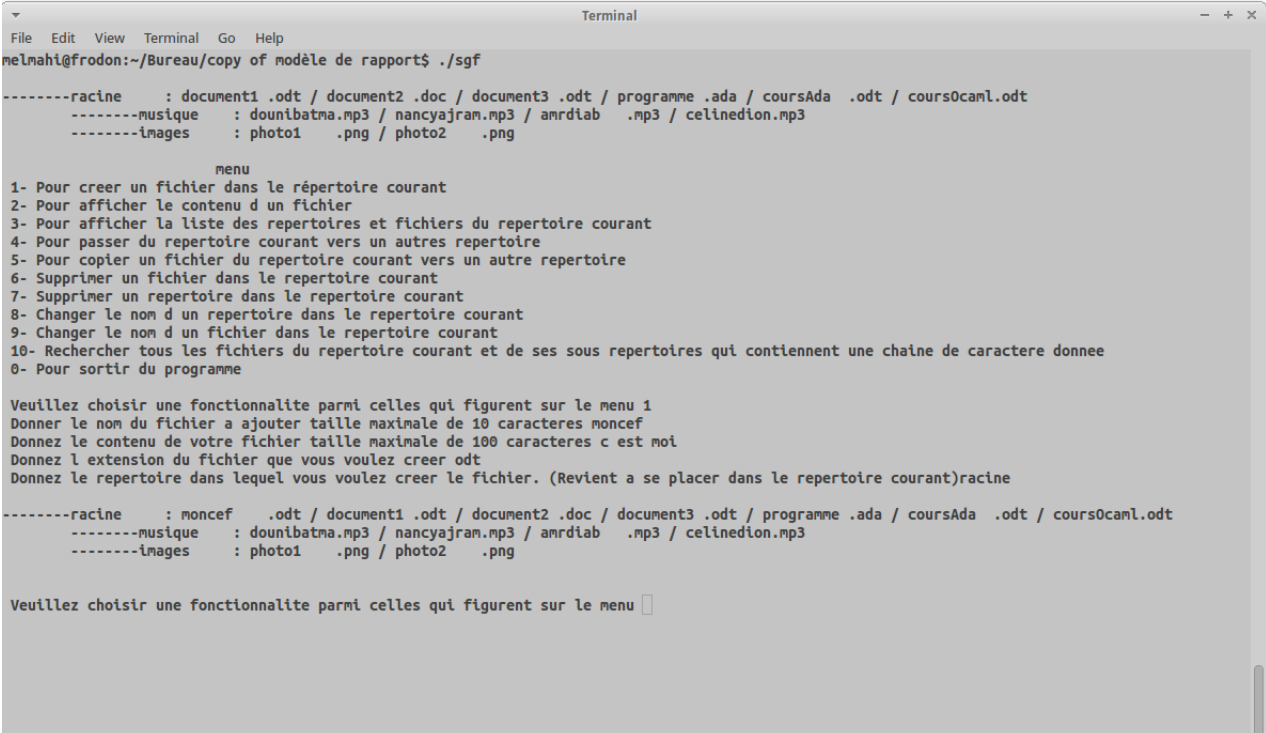
–cas terminal
 inter est le 1er fichier du répertoire courant
 –on affiche le nom des fichiers contenant la chaine de caractere
tant que Inter/=NULL **faire**
 si Contient(Inter.All.Contenu,Nmax,Chaine,J) **alors**
 Ecrire(Inter.All.Nom)
 sinon
 rien
 fin si
 Inter ← Inter.All.Suiv
fin tant que
 –Afficher le nom des fichiers des sous-repertoire contenant la chaine
 I ← 1 –on initialise le compteur
tant que le ième fils est non vide **faire**
 Arbre ← An Fils(a,i)
 –on recherche dans les fichiers du ième fils
 Recherche Fichiers(Arbre,Chaine,J)
 I ← I+1 –on incrémente le compteur pour passer au fils suivant
fin tant que

3.3 Validation du paquetage p_sys

3.3.1 Interface de ce système

Dans cette partie je vais détailler l'idée de la conception de l'interface de ce système. Comme cela est précisé dans l'énoncé du projet, nous devons permettre à l'utilisateur l'accès aux différentes fonctions et procédures implantées dans les paquetages précédents de manière conviviale et facile, donc le choix s'est naturellement porté sur un menu présentant toutes les fonctionnalités de ce système.

3.3.2 Tests



```

Terminal
File Edit View Terminal Go Help
melmahif@frodon:~/Bureau/copy of modèle de rapport$ ./sgf

-----racine      : document1.odt / document2.doc / document3.odt / programme.ada / coursAda .odt / coursOcanl.odt
-----musique     : dounibatma.mp3 / nancyajram.mp3 / anrdiab .mp3 / celinedion.mp3
-----images      : photo1 .png / photo2 .png

      menu
1- Pour creer un fichier dans le repertoire courant
2- Pour afficher le contenu d un fichier
3- Pour afficher la liste des repertoires et fichiers du repertoire courant
4- Pour passer du repertoire courant vers un autres repertoire
5- Pour copier un fichier du repertoire courant vers un autre repertoire
6- Supprimer un fichier dans le repertoire courant
7- Supprimer un repertoire dans le repertoire courant
8- Changer le nom d un repertoire dans le repertoire courant
9- Changer le nom d un fichier dans le repertoire courant
10- Rechercher tous les fichiers du repertoire courant et de ses sous repertoires qui contiennent une chaine de caractere donnee
0- Pour sortir du programme

Veuillez choisir une fonctionnalite parmi celles qui figurent sur le menu 1
Donner le nom du fichier a ajouter taille maximale de 10 caracteres moncef
Donnez le contenu de votre fichier taille maximale de 100 caracteres c est moi
Donnez l extension du fichier que vous voulez creer odt
Donnez le repertoire dans lequel vous voulez creer le fichier. (Revient a se placer dans le repertoire courant)racine
-----racine      : moncef .odt / document1.odt / document2.doc / document3.odt / programme.ada / coursAda .odt / coursOcanl.odt
-----musique     : dounibatma.mp3 / nancyajram.mp3 / anrdiab .mp3 / celinedion.mp3
-----images      : photo1 .png / photo2 .png

Veuillez choisir une fonctionnalite parmi celles qui figurent sur le menu

```

```

Terminal
File Edit View Terminal Go Help
-----
Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 1
Donner le nom du fichier a ajouter taille maximale de 10 caracteres moncef
Donnez le contenu de votre fichier taille maximale de 100 caracteres c est moi
Donnez l extension du fichier que vous voulez creer odt
Donnez le repertoire dans lequel vous voulez creer le fichier. (Revient a se placer dans le repertoire courant)racine
-----racine      : moncef      .odt / document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique     : dounibatma.mp3 / nancyajram.mp3 / amrdiab .mp3 / celinedion.mp3
-----images      : photo1      .png / photo2      .png

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 2
Quel est le nom du repertoire courant ou vous voulez vous placer? racine
Quel est le nom du fichier pour lequel vous voulez afficher le contenu? programme
Quel est l'extension du fichier pour lequel vous voulez afficher le contenu? ada
*****
Le contenu de votre fichier est:
with adatextio use ada textio          begin      instructions end programme
Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 3
Quel est le nom de votre repertoire courant ?racine
La liste des répertoires et des fichiers du repertoire courant est:
-----racine      : moncef      .odt / document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique     : dounibatma.mp3 / nancyajram.mp3 / amrdiab .mp3 / celinedion.mp3
-----images      : photo1      .png / photo2      .png

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 3
Quel est le nom de votre repertoire courant ?images
La liste des répertoires et des fichiers du repertoire courant est:
-----images      : photo1      .png / photo2      .png

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 4
Quelle est le nom du repertoire cible que vous voulez atteindre? images
-----images      : photo1      .png / photo2      .png

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu

```

```

Terminal
File Edit View Terminal Go Help
-----
La liste des répertoires et des fichiers du repertoire courant est:
-----racine      : moncef      .odt / document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique     : dounibatma.mp3 / nancyajram.mp3 / amrdiab .mp3 / celinedion.mp3
-----images      : photo1      .png / photo2      .png

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 3
Quel est le nom de votre repertoire courant ?images
La liste des répertoires et des fichiers du repertoire courant est:
-----images      : photo1      .png / photo2      .png

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 4
Quelle est le nom du repertoire cible que vous voulez atteindre? images
-----images      : photo1      .png / photo2      .png

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 5
Donnez le nom du repertoire contenant le fichier que vous voulez copier. (revient a se placer dans le repertoire courant) racine
Donnez le nom du fichier que vous voulez copier moncef
Donnez l extension du fichier que vous voulez copier odt
Quel est le nom du repertoire ou vous voulez copier le fichier ? musique
-----racine      : moncef      .odt / document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique     : moncef      .odt / dounibatma.mp3 / nancyajram.mp3 / amrdiab .mp3 / celinedion.mp3
-----images      : photo1      .png / photo2      .png

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 6
Dans quel repertoire vous voulez supprimer votre fichier?racine
Quel est le nom du fichier que vous souhaitez supprimer ? moncef
Quel est l extension du fichier que vous souhaitez supprimer ? odt
-----racine      : document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique     : moncef      .odt / dounibatma.mp3 / nancyajram.mp3 / amrdiab .mp3 / celinedion.mp3
-----images      : photo1      .png / photo2      .png

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu

```

```

Terminal
File Edit View Terminal Go Help
-----racine      : moncef      .odt / document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique     : moncef      .odt / dounibatma.mp3 / nancyajram.mp3 / amrdiab .mp3 / celinedion.mp3
-----images      : photo1      .png / photo2      .png

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 6
Dans quel repertoire vous voulez supprimer votre fichier? racine
Quel est le nom du fichier que vous souhaitez supprimer ? moncef
Quel est l'extension du fichier que vous souhaitez supprimer ? odt

-----racine      : document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique     : moncef      .odt / dounibatma.mp3 / nancyajram.mp3 / amrdiab .mp3 / celinedion.mp3
-----images      : photo1      .png / photo2      .png

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 7
Quel est le nom du repertoire ou vous voulez vous placer? (repertoire courant) racine
Quel est le numero du repertoire que vous voulez supprimer? 2

-----racine      : document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique     : moncef      .odt / dounibatma.mp3 / nancyajram.mp3 / amrdiab .mp3 / celinedion.mp3

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 8
Quel est le nom du repertoire courant? (cad le repertoire pour lequel vous voulez changer de nom) racine
Quel le nouveau nom que vous voulez attribuer à ce repertoire ? maximum 10 caracteres melmahi
-----melmahi     : document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique     : moncef      .odt / dounibatma.mp3 / nancyajram.mp3 / amrdiab .mp3 / celinedion.mp3

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 9
Quel est le nom du repertoire dans lequel vous voulez vous placer? racine
Quel est le nom du fichier que vous voulez renommer ?
Quelle est l'extension du fichier que vous voulez renommer
Quel est le nouveau nom que vous voulez donner a ce fichier ?
nouvelle extension
le repertoire cible n'existe pas

raised P_SYS.P_ARBREN_TEST.ARBRE_VIDE : p_arbren.adb:28 instantiated at p_sys.ads:41
melmahi@frodon:~/Bureau/copy of modèle de rapport$

```

```

Terminal
File Edit View Terminal Go Help
-----racine      : document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique     : dounibatma.mp3 / nancyajram.mp3 / amrdiab .mp3 / celinedion.mp3
-----images      : photo1      .png / photo2      .png

menu
1- Pour creer un fichier dans le repertoire courant
2- Pour afficher le contenu d un fichier
3- Pour afficher la liste des repertoires et fichiers du repertoire courant
4- Pour passer du repertoire courant vers un autres repertoire
5- Pour copier un fichier du repertoire courant vers un autre repertoire
6- Supprimer un fichier dans le repertoire courant
7- Supprimer un repertoire dans le repertoire courant
8- Changer le nom d un repertoire dans le repertoire courant
9- Changer le nom d un fichier dans le repertoire courant
10- Rechercher tous les fichiers du repertoire courant et de ses sous repertoires qui contiennent une chaine de caractere donnee
0- Pour sortir du programme

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 8
Quel est le nom du repertoire courant? (cad le repertoire pour lequel vous voulez changer de nom) racine
Quel le nouveau nom que vous voulez attribuer à ce repertoire ? maximum 10 caracteres melmahi
-----melmahi     : document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique     : dounibatma.mp3 / nancyajram.mp3 / amrdiab .mp3 / celinedion.mp3
-----images      : photo1      .png / photo2      .png

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu 9
Quel est le nom du repertoire dans lequel vous voulez vous placer? melmahi
Quel est le nom du fichier que vous voulez renommer ? document1
Quelle est l'extension du fichier que vous voulez renommer? odt
Quel est le nouveau nom que vous voulez donner a ce fichier ? salut
nouvelle extensiontxt
-----melmahi     : salut      .txt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique     : dounibatma.mp3 / nancyajram.mp3 / amrdiab .mp3 / celinedion.mp3
-----images      : photo1      .png / photo2      .png

Veillez choisir une fonctionnalite parmi celles qui figurent sur le menu

```

```

Terminal
File Edit View Terminal Go Help

-----racine      : document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique    : dounibatma.mp3 / nancyajram.mp3 / anrdiab .mp3 / celinedion.mp3
-----images     : photo1 .png / photo2 .png

      menu
1- Pour creer un fichier dans le repertoire courant
2- Pour afficher le contenu d un fichier
3- Pour afficher la liste des repertoires et fichiers du repertoire courant
4- Pour passer du repertoire courant vers un autres repertoire
5- Pour copier un fichier du repertoire courant vers un autre repertoire
6- Supprimer un fichier dans le repertoire courant
7- Supprimer un repertoire dans le repertoire courant
8- Changer le nom d un repertoire dans le repertoire courant
9- Changer le nom d un fichier dans le repertoire courant
10- Rechercher tous les fichiers du repertoire courant et de ses sous repertoires qui contiennent une chaine de caractere donnee
0- Pour sortir du programme

Veuillez choisir une fonctionnalite parmi celles qui figurent sur le menu 8
Quel est le nom du repertoire courant? (cad le repertoire pour lequel vous voulez changer de nom) racine
Quel le nouveau nom que vous voulez attribuer à ce repertoire ? maximum 10 caracteres melmahi
-----melmahi    : document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique    : dounibatma.mp3 / nancyajram.mp3 / anrdiab .mp3 / celinedion.mp3
-----images     : photo1 .png / photo2 .png

Veuillez choisir une fonctionnalite parmi celles qui figurent sur le menu 9
Quel est le nom du repertoire dans lequel vous voulez vous placer? melmahi
Quel est le nom du fichier que vous voulez renommer ? document1
Quelle est l'extension du fichier que vous voulez renommerodt
Quel est le nouveau nom que vous voulez donner a ce fichier ? salut
nouvelle extensiontxt
-----melmahi    : salut .txt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique    : dounibatma.mp3 / nancyajram.mp3 / anrdiab .mp3 / celinedion.mp3
-----images     : photo1 .png / photo2 .png

Veuillez choisir une fonctionnalite parmi celles qui figurent sur le menu

```

```

Terminal
File Edit View Terminal Go Help

-----racine      : document1 .odt / document2 .doc / document3 .odt / programme .ada / coursAda .odt / cours0caml.odt
-----musique    : dounibatma.mp3 / nancyajram.mp3 / anrdiab .mp3 / celinedion.mp3
-----images     : photo1 .png / photo2 .png

      menu
1- Pour creer un fichier dans le repertoire courant
2- Pour afficher le contenu d un fichier
3- Pour afficher la liste des repertoires et fichiers du repertoire courant
4- Pour passer du repertoire courant vers un autres repertoire
5- Pour copier un fichier du repertoire courant vers un autre repertoire
6- Supprimer un fichier dans le repertoire courant
7- Supprimer un repertoire dans le repertoire courant
8- Changer le nom d un repertoire dans le repertoire courant
9- Changer le nom d un fichier dans le repertoire courant
10- Rechercher tous les fichiers du repertoire courant et de ses sous repertoires qui contiennent une chaine de caractere donnee
0- Pour sortir du programme

Veuillez choisir une fonctionnalite parmi celles qui figurent sur le menu 10
Quel est le nom du repertoire dans lequel vous voulez vous placez afin de chercher un certain contenu racine
Quelle est la chaine que vous cherchez
cours
La liste des fichiers contenant cette chaine de caracteres est
document1
document2
document3

Veuillez choisir une fonctionnalite parmi celles qui figurent sur le menu 10
Quel est le nom du repertoire dans lequel vous voulez vous placez afin de chercher un certain contenu musique
Quelle est la chaine que vous cherchez
Parol
La liste des fichiers contenant cette chaine de caracteres est
dounibatma
nancyajram
anrdiab
celinedion

Veuillez choisir une fonctionnalite parmi celles qui figurent sur le menu

```

3.3.3 Limites de mon paquetage

La fonction `recherche_fichiers` qui cherche les fichiers du répertoire courant et de ses sous répertoires contenant une chaîne de caractère donnée ne renvoie que les fichiers du répertoire courant contenant cette chaîne de caractères. (j'ai pourtant fait une boucle me permettant de parcourir les fils du répertoire courant).

3.3.4 Conclusion 2

Après avoir effectué les tests, je peux affirmer que toutes les fonctions et procédures fonctionnent normalement.

Deuxième partie

Le contrôle d'accès

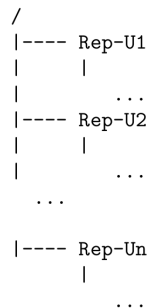
4 Le contrôle d'accès

4.1 Définition

On souhaite définir un contrôle d'accès à un système de gestion de fichiers. Pour cela, on supposera l'existence d'un ensemble fixe d'utilisateurs du SGF (Chaque utilisateur est identifié par un numéro). Dans cette partie, un utilisateur ne pourra exécuter les opérations implantées en 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). dans lequel le répertoire Rep- U_i est le répertoire de l'utilisateur U_i .



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

5 Paquetage `p_sys_users`

5.1 Spécifications du paquetage `p_sys_users`

Résumé des commandes du paquetage

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 sur un ou plusieurs fichiers (ou répertoires) du répertoire courant.
- Le retrait de droits (r ou w) à un ou plusieurs utilisateurs sur un ou plusieurs fichiers (ou répertoires) du répertoire courant.

5.2 Conception du paquetage p_sys_users

5.2.1 Choix du type et des structures de données

Les types ainsi que les structures de données pour lesquels j'ai opté dans ce paquetage afin de rajouter les notions de droits et d'utilisateurs sont :

```

1  — types necessites pour ce paquetage
2  Nmax:CONSTANT Integer:=100;
3  Kmax:CONSTANT Integer:=14;
4
5  — Les types de droits , j'ai choisi une énumération.
6  type droit is (lecture ,écriture);
7
8  — j'ai fixé dans mon exemple le nombre d'utilisateurs a 3
9  type utilisateur is (u1,u2,u3);
10
11 — liste chainee d'utilisateur
12 type util;
13 type p_util is access util
14 type util is record
15     user: utilisateur;
16     next: p_util;
17 end record;
18
19 — fichiers d un repertoire
20 type fichiers_users;
21 type fich_users is access fichiers;
22 type fichiers_users is record
23     nom_du_fichier: string(1..kmax);
24     extension: string(1..emax);
25     contenu_du_fichier: string(1..cmax);
26     droit_lec: p_util;
27     droit_ecr: p_util;
28     proprietaire: utilisateur;
29     suivant: fich;
30 end record;
31
32 — repertoires
33 type repertoire_users
34 type prep is record repertoire_users;
35 type repertoire_users is record
36     Nom; string(1..kmax);
37     proprietaire: utilisateur;
38     droit_lec: p_util;
39     droit_ecr: p_util;
40     fichier_util: fichiers_users;
41 end record;
42
43 —exceptions
44
45 repertoire_null :EXCEPTION ;
46 Fichier_Inexistant:EXCEPTION;
47 Pas_De_Fichier:EXCEPTION ;

```

5.2.2 Définition des fonctions du paquetage p_sys_users

Le paquetage p_sys_users est une modification du paquetage précédent p_sys. Il s'agit de rajouter des conditions d'accès aux fichiers (et répertoires).

```

1  — instantiation du paquetage generique
2  PROCEDURE Ecrire_Rep(Fich:IN Repertoire_Users) ;
3  FUNCTION Correspond_Rep(Data1:IN Repertoire_Users;
4  Data2:IN Repertoire_Users) RETURN Boolean ;
5  PACKAGE P_Arbren_Users IS NEW
6  P_Arbren(Repertoire_Users ,Ecrire_Rep , Correspond_Rep);
7  USE P_Arbren_Users ;
8
9  —fonctions du paquetage
10
11 —egalite : verifie si 2 chaines de caracteres sont egales
12 —param : chaine1 :string (*D*), chaine2:string (*D*)
13 —pre : aucune
14 —post : renvoie vrai si les chaines sont egales
15 —exemption : aucune
16 FUNCTION Egalite(Chaine1:IN String;Chaine2: IN String) RETURN Boolean;
17
18
19 —insere_liste : insere un elt dans une liste chainee
20 —param : u:utilisateur (*D*), l:liste_util (*D/R*)
21 —pre : aucune
22 —post: liste avec un elt en plus
23 —exemption : aucune
24 PROCEDURE Insere_Liste(U:IN Utilisateur;L:IN OUT Liste_Util ) ;
25
26
27 —appartient_liste : teste si un elt appartient a la liste
28 —param : u: utilisateur (*D*), l: liste_util (*D*)
29 —pre : aucune
30 —post : retourne vrai si elt appartient a la liste
31 —exemption : aucune
32 FUNCTION Appartient_Liste(U:IN Utilisateur ;L:IN Liste_Util)
33 RETURN Boolean ;
34
35
36 —supprime_liste : supprime un elt de la liste chainee
37 —param : u:utilisateur(*D*), l:liste_util (*D/R*)
38 —pre : aucune
39 —post : liste avec elt en moins
40 —exemption : aucune
41 PROCEDURE Supprime_Liste(U:IN Utilisateur ; L :IN OUT Liste_Util) ;
42
43
44 —octroyer_rep : donne a un utilisateur le droit sur un repertoire
45 —param : a:arbren (*D*), nom_rep:string (*D*), u_proprio:utilisateur ,
46 —      (*D*), u: utilisateur (*D*),d:droit (*D*)
47 —pre : aucune
48 —post : aucune
49 —exemption : aucune
50 PROCEDURE Octroyer_Rep(A:IN Arbren ;Nom_Rep:IN String ;U_Proprio:
51 IN Utilisateur ; U:IN Utilisateur;D:IN Droit);
52

```

```

53
54 —octroyer_fich : donne a un utilisateur le droit sur un fichier
55 —param : param : a:arbren (*D*), nom_rep:string (*D*),
56 — u_proprio:utilisateur(*D*), u: utilisateur (*D*),d:droit (*D*)
57 —pre : aucune
58 —post: aucune
59 —exeption: aucune
60 PROCEDURE Octroyer_Fich(A:IN Arbren ;Nom_Fich:IN String ;
61 U_Proprio:IN Utilisateur ; U:IN Utilisateur;D:IN Droit);
62
63
64 —retirer_rep : retire a un utilisateur le droit sur un repertoire
65 —param : param : a:arbren (*D*), nom_rep:string (*D*),
66 — u_proprio:utilisateur(*D*), u: utilisateur (*D*),d:droit (*D*)
67 —pre : aucune
68 —post : aucune
69 —exeption : aucune
70 PROCEDURE Retirer_Rep(A:IN Arbren ;Nom_Rep:IN String ;
71 U_Proprio:IN Utilisateur ; U:IN Utilisateur;D:IN Droit);
72
73
74 —retirer_fich : retire a un utilisateur le droit sur un fichier
75 —param : param : a:arbren (*D*), nom_rep:string (*D*),
76 — u_proprio:utilisateur(*D*), u: utilisateur (*D*),d:droit (*D*)
77 —pre : aucune
78 —post : aucune
79 —exeption : aucune
80 PROCEDURE Retirer_Fich(A:IN Arbren ;Nom_Fich:IN String ;
81 U_Proprio:IN Utilisateur ; U:IN Utilisateur;D:IN Droit);
82
83
84 —creer_users: cree un fichier dans le repertoire courant
85 —param : a:arbren (*D*), nom_fich:string(1..Kmax) (*D*) ,
86 — contenu_fich:string(1..Nmax) (*D*),u_p:utilisateur(*D*)
87 —pre:taille du nom_fich inferieur a Kmax, contenu_fich inferieur a 100
88 —post : aucune
89 —exeption : aucune
90 PROCEDURE Creer_Users(A: IN Arbren ;Nom_Fich: IN String ;
91 Contenu_Fich :IN String;UP:IN Utilisateur );
92
93
94 —afficher_users : affiche le contenu d'un fichier
95 —param : fichiers (*D*),u_courant :utilisateur(*D*)
96 —pre : aucune
97 —post :aucune
98 —exeption : si le fichier n'existe pas (pas_de_fichier)
99 PROCEDURE Afficher_Contenu_aux_Users(Fich:IN Fichiers_Users;
100 U_Courant:IN Utilisateur ) ;
101
102
103 —afficher_nieme_contenu_users : affiche le contenu du nieme fichier
104 — param : a:arbren (*D*), nom_fich:string(1..Kmax) (*D*),
105 — u_courant :utilisateur(*D*).
106 — pre : aucune
107 — post : aucune
108 — exeption : aucune

```

```

109 PROCEDURE Afficher_Contenu_Users(A:IN Arbren ; Nom_fich: IN string;
110 U_Courant:IN Utilisateur ) ;
111
112
113 —afficher_nom_fichiers_users : affiche le nom de tous les fichiers du
114 — repertoire courant.
115 — param : a: arbren (*D*), u_courant : utilisateur(*D*)
116 — pre : aucune
117 — post : aucune
118 — exeption : arbre vide
119 PROCEDURE Afficher_Nom_Fichiers_Users(A:IN Arbren;
120 U_Courant:IN Utilisateur );
121
122
123 —afficher_nom_repertoire_users : affiche le nom du repertoire courant
124 — param : a:arbren (*D*), u_courant : utilisateur(*D*)
125 — pre : aucune
126 — post : aucune
127 — exeption : arbre vide
128 PROCEDURE Afficher_Nom_Repertoire_Users(A:IN Arbren;
129 U_Courant:IN Utilisateur);
130
131
132 —afficher_tous_users : affiche la liste des repertoires et fichiers
133 — du repertoire courant.
134 —param : a:arbren (*D*), u_courant : utilisateur(*D*)
135 —pre : aucune
136 —post : aucune
137 —exeption : arbre vide (i.e repertoire vide)
138 PROCEDURE Afficher_Tous_Users(A:IN Arbren;U_Courant:IN Utilisateur ) ;
139
140
141 —changement_users : change le repertoire vers un autre repertoire
142 — param : a: arbren (*D*), nom_rep:string(1..Kmax) (*D*),
143 — rep_courant: arbren (*R*), u_courant :utilisateur(*D*)
144 — pre : taille des chaines de caracteres inferieur a Kmax
145 — post:on change de repertoire s'il existe ,sinon on reste
146 — dans l'actuel.
147 — exeption : arbre vide
148 PROCEDURE Changement_Users (A: IN Arbren;Nom_Rep:IN String;
149 Rep_Courant:OUT Arbren;U_Courant:IN Utilisateur) ;
150
151
152 —copie_users : copier un fichier du repertoire actuel vers un autre
153 — repertoire
154 — param : a:arbren (*D*), nom_rep_nouveau: string(1..Kmax) (*D*),
155 — nom:string(1..Kmax) (*D*),u_courant :utilisateur(*D*)
156 — pre : taille des chaines de caracteres inferieur a Kmax
157 — post : copie du fichier s'il existe sinon rien
158 — exeption : fichier inexistant
159 PROCEDURE Copie_Users(A: IN Arbren ; Nom_Rep_Nouveau: IN String ;
160 Nom : IN String;U_Courant:IN Utilisateur ) ;
161
162
163 — supprime_users : supprime un repertoire ou un fichier dans
164 — le repertoire courant

```

```

165 — param : a:arbren (*D*), nom_elt: string(1..Kamx) (*D*),
166 — u_courant :utilisateur(*D*)
167 — pre : aucune
168 — post : supprime l'elt sauf s'il n'existe pas
169 — exeption : arbre vide
170 PROCEDURE Supprime_Users(A:IN OUT Arbren ; Nom_Elt:IN String ;
171 U_Courant:IN Utilisateur ) ;
172
173
174 —renomme_users : change le nom d'un repertoire ou d'un fichier dans
175 — le repertoire courant.
176 — param : a:arbren (*D*), nom_anc:string(1..Kmax) (*D*),
177 —         nom_nouveau:string(1..Kmax) (*D*), u_courant :utilisateur(*D*)
178 — pre : taille des chaines de caracteres inferieur a Kmax
179 — post : renomme l'elt s'il existe
180 — exeption : arbre vide
181 PROCEDURE Renommer_Users(A: IN Arbren ; Nom_Anc: IN String ;
182 Nom_Nouveau : IN String;U_Courant:IN Utilisateur ) ;
183
184
185 —contient_users : verifie si une chaine de caractere est contenu dans
186 — une autre chaine .
187 — param : chaine et sa taille :string+integer , la chaine recherchee
188 —         et sa taille :string+integer
189 — pre : aucune
190 — post : retourne vrai si elle y est
191 — exeption : aucune
192 FUNCTION Contient_Users (Chaine_Contenu:IN String;L_Contenu:IN Integer ;
193 Chaine_Rech: IN String;J:IN Integer) RETURN Boolean ;
194
195
196 —recherche_fichiers_users : recherche les fichiers du repertoire
197 — courant et de ses sous-repertoires qui contiennent une chaine
198 — de caracteres donnee
199 — param : a: arbren (*D*), chaine:string (*D*), J:integer (*D*)
200 —         u_courant :utilisateur(*D*)
201 — pre : aucune
202 — post : retourne les noms des fichiers concernees , rien sinon
203 — exeption : aucune
204 PROCEDURE Recherche_Fichiers_Users(A:IN Arbren ; Chaine:IN String ;
205 J : IN Integer ;U_Courant: IN Utilisateur) ;
206

```

5.2.3 Raffinages de certaines fonctions et procédures

Octroyer Rep :

R0 :

-
- octroyer rep : donne à un utilisateur le droit sur un repertoire
 - param : a :arbren (*D*), nom rep :string (*D*), u proprio :utilisateur(*D*), u : utilisateur (*D*), d :droit (*D*).
 - pre : aucune
 - post : aucune
 - exeption : aucune
-

procedure Octroyer Rep((*D*)A :Arbren;(*D*)Nom Rep :String;(*D*)U Proprio :Utilisateur; (*D*)U :Utilisateur;(*D*) D :Droit)

R1 :

- Rechercher le repertoire nom_rep.
 - Vérifier si celui qui donne le droit est propriétaire.
 - Ajouter l'utilisateur dans la liste (r ou w), selon le droit octroyé.
-

R2 :

```

– on recherche le repertoire de nom :nom rep
rep ← new enreg r
Rep.All.Nom ← Nom Rep
Inter ← An Rechercher(A,Rep)
– verifier si u proprio est bien le proprietaire du repertoire
si proprietaire different du u proprio alors
    rien
sinon
    – ajouter l'utilisateur dans la liste (r ou w), selon le droit octroyé.
    – dans le cas droit de lecture
    si droit d'écriture alors
        si u appartient à la liste des utilisateurs ayant droit d'écriture alors
            rien
        sinon
            on insère u dans cette liste
        fin si
    –dans le cas droit d'écriture
    sinon
        si u appartient à la liste des utilisateurs ayant droit de lecture alors
            rien
        sinon
            on insère u dans cette liste
        fin si
    fin si
fin si

```

Retirer fich :**R0 :**

– retirer fich : retire a un utilisateur le droit sur un fichier
 – param : param : a :arbren (*D*), nom rep :string (*D*), u proprio :utilisateur, (*D*),
 u : utilisateur (*D*), d :droit (*D*).
 –pre : aucune
 –post : aucune
 –exemption : aucune

procedure Retirer Fich((*D*)A :Arbren;(*D*)Nom fich :String; (*D*)U Proprio :Utilisateur;
 (*D*)U :Utilisateur;(*D*) D :Droit);

R1 :

- Rechercher le fichier nom fich.
 - Vérifier si celui qui retire le droit est propriétaire.
 - Supprimer l'utilisateur de la liste (r ou w), selon le droit retiré.

R2 :

– on recherche le fichier du nom :nom fich
 inter est le 1er fichier du repertoire courant.
tant que inter/= null **et alors** egalite(inter.all.nom,nom fich) **faire**
 inter ← inter.all.suiv
fin tant que
si inter=null **alors**
 on leve l'exemption fichier inexistant
sinon
 –verifier si u proprio est bien le proprietaire du repertoire
 si proprietaire different du u proprio **alors**
 rien
 sinon
 –supprimer l'utilisateur de la liste (r ou w), selon le droit retiré.
 –dans le cas droit d'écriture
 si droit d'écriture **alors**
 si u appartient à la liste des utilisateurs ayant droit d'écriture **alors**
 on supprime u de cette liste
 sinon
 rien
 fin si
 –dans le cas droit de lecture
 sinon
 si u appartient à la liste des utilisateurs ayant droit de lecture **alors**
 on supprime u de cette liste
 sinon
 rien
 fin si
 fin si
fin si

5.3 Validation du paquetage p_sys_users

Après avoir fait la spécification et les raffinages correspondant au paquetage p_sys_users (qui est une extension du paquetage p_sys), j'ai pas pu coder les fonctions et les procédures en langage Ada par manque de temps, c'est pourquoi j'ai pas pu effectuer des tests sur ce paquetage.

Conclusion

Ce projet demande une vraie maîtrise de programmation impérative, parceque d'une part, on a les notions les plus importantes (pointeurs, énumérations, ...) et d'autre part la méthode des raffinages qui retrouve toute son importance pour une bonne conception des différents paquetages. Ceci m'a permis de consolider mes connaissances et de prendre utilité de la grande utilité de ce langage.

Annexes

Annexe 1 : Code de p_arbren.adb

Code de p_arbren.adb

```
1 with text_io; use text_io;
2 with ada.integer_text_io; use ada.integer_text_io;
3 with ada.float_text_io; use ada.float_text_io;
4
5 package body p_arbren is
6
7   -- Fonction An_Vide
8   function An_Vide (a: in arbren) return boolean is
9   begin
10      -- si a=null alors elle retourne true sinon false.
11      return (a=null);
12   end An_vide;
13
14
15   -- Fonction An_Creer_Vide
16   function An_Creer_Vide return arbren is
17   begin
18      return null;
19   end An_Creer_vide;
20
21
22   -- Fonction An_Valeur
23   function An_Valeur (a: in arbren) return T is
24
25   begin
26      return (a.all.val);
27   exception
28   when constraint_error => raise arbre_vide;
29   end An_valeur;
30
31
32   -- Fonction An_Est_Feuille
33   function An_Est_Feuille (a: in arbren) return boolean is
34   begin
35      return (a.all.premier_fils=null);
36   exception
37   when constraint_error => raise arbre_vide;
38   end An_Est_Feuille ;
39
40
41
```

```

42 — Fonction An_Creer_Feuille
43 function An_Creer_Feuille (nouveau: in T) return arbren is
44 a: arbren;
45
46 Begin
47
48 a:= new noeud; — on lui donne de la place.
49 a.all.val:= nouveau;
50 a.all.premier_fils:=null;
51 a.all.frere:=null;
52 a.all.pere:=null;
53
54 return a;
55
56 end An_Creer_Feuille ;
57
58
59 — Fonction An_Pere
60 function An_Pere ( a: in arbren ) return arbren is
61
62 begin
63
64 if An_Vide(a) then
65     raise arbre_vide;
66 else if a.all.pere=null then
67     raise no_pere;
68     else
69         return a.all.pere;
70     end if;
71 end if;
72
73 end An_pere;
74
75
76 — Fonction An_Fils
77
78 function An_Fils ( a: in arbren; numero: in integer ) return arbren is
79
80 arb_aux: arbren;
81 arb_aux2: arbren;
82
83 begin
84 arb_aux2:= new noeud;
85
86 if An_Vide(a) then
87     raise arbre_vide;
88     else
89         arb_aux:=a.all.premier_fils;
90         for i in 1..numero-1 loop
91             arb_aux:=arb_aux.all.frere;
92         end loop;
93 end if;
94
95 arb_aux2.all.frere:=null;
96 arb_aux2.all.pere:=null;
97 arb_aux2.all.val:=arb_aux.all.val;

```

```

98 arb_aux2.all.premier_fils:=arb_aux.all.premier_fils;
99
100 return arb_aux2;
101 exception
102     when constraint_error => raise no_nieme_fils;
103 end An_Fils;
104
105
106 — Fonction An_Frere
107
108 function An_Frere ( a: in arbren; numero: in integer ) return arbren is
109
110     arbre_aux: arbren;
111     arbre_aux2: arbren;
112     begin
113         arbre_aux:= new noeud;
114         arbre_aux2:= new noeud;
115
116         if An_Vide(a) then
117             raise arbre_vide;
118         else
119             arbre_aux:=a;
120             for i in 1..numero loop
121                 arbre_aux:=arbre_aux.all.frere;
122             end loop;
123         end if;
124         arbre_aux2.all.val:=arbre_aux.all.val;
125         arbre_aux2.all.premier_fils:=arbre_aux.all.premier_fils;
126         arbre_aux2.all.frere:=null;
127         arbre_aux2.all.pere:=null;
128
129         return arbre_aux2;
130     exception
131         when constraint_error => raise no_frere;
132     end An_Frere;
133
134
135 procedure An_Afficher_Aux (a: in arbren; decal:in integer) is
136     begin
137         if a /= null then
138             — decaler de decal espaces
139             for i in 1..decal loop
140                 put(" ");
141             end loop;
142             — ecrire la valeur du noeud
143             ecrire(a.all.val);
144             new_line;
145             — afficher le fils
146             An_Afficher_Aux(a.all.premier_fils ,decal+8);
147             — afficher le frere
148             An_Afficher_Aux(a.all.frere ,decal);
149         else
150             — rien
151             null;
152         end if;
153     end An_Afficher_Aux;

```

```

154
155
156
157 procedure An_Afficher (a: in arbren) is
158 begin
159     if a = null then null;
160     else
161         An_Afficher_Aux(a,0);
162     end if;
163 end An_Afficher;
164
165
166 — Procédure auxiliaire:
167 —procedure An_Afficher_Aux(a: in arbren ; espace: in string ) is
168 —begin
169 —    if a = null then null;
170 —    else
171 —        put(espace);
172 —        ecrire(a.all.val);
173 —        An_Afficher_Aux(a.all.premier_fils ,espace& "      |");
174 —        An_Afficher_Aux(a.all.frere ,espace);
175 —    end if;
176 —
177 —end An_Afficher_Aux;
178
179 —Procedure An_Afficher
180 —Semantique: Afficher le contenu complet d un arbre n-aire
181 — Parametres: a: arbren (D)
182 —procedure An_Afficher (a: in arbren) is
183 —Begin
184 —if An_Vide(a) then null;-- raise arbre_vide;
185 —else
186 —    put ("|");
187 —    An_Afficher_Aux(a,"");
188 —end if;
189 —end An_Afficher;
190
191
192
193 — Fonction An_Rechercher
194 function An_Rechercher(a : in arbren; data : in T) return arbren is
195     p : arbren;
196     begin
197         p:= new noeud;
198
199         if a = Null or else egalite(a.all.val ,data) then
200             return a;
201         else
202             p := An_Rechercher(a.all.frere , data);
203
204             if p /= Null then
205                 return p;
206             else
207                 return An_Rechercher(a.all.premier_fils , data);
208             end if;
209         end if;

```

```

210   end An_Rechercher;
211
212
213 — Fonction An_Est_Racine
214 function An_Est_Racine (a: in arbren) return boolean is
215 begin
216
217   return (a.all.pere=null);
218
219   exception
220     when constraint_error => raise arbre_vide;
221 end An_Est_Racine ;
222
223
224 — Procedure An_Changer_Valeur
225 procedure An_Changer_Valeur (a:in out arbren; nouveau:in T) is
226
227   begin
228
229     a.all.val:=nouveau;
230   exception
231     when constraint_error => raise arbre_vide;
232 end An_Changer_Valeur ;
233
234
235 — Procedure An_Inserer_Fils
236 procedure An_Inserer_Fils (a:in out arbren; a_ins: in out arbren) is
237
238
239   Begin
240
241   — d'abord il y a l'exception si l'arbre est nul
242   if a= null then raise arbre_vide;
243   — sinon il faut traiter le cas où l'arbre à insérer est vide.
244   else if a_ins= null then null;
245     — sinon si l'arbre n'a pas de fils.
246     else if a.all.premier_fils=null
247       then a.all.premier_fils := a_ins;
248         — sinon dans le cas général.
249     else a_ins.all.frere := a.all.premier_fils;
250         a_ins.all.pere := a;
251         a.all.premier_fils := a_ins;
252     end if;
253   end if;
254 end if;
255 end An_Inserer_Fils ;
256
257
258 — Procedure An_Inserer_Frere
259 procedure An_Inserer_Frere (a:in out arbren; a_ins:in out arbren) is
260
261   Begin
262
263   — SI l'arbre est nul
264   if a=null then raise arbre_vide;
265   — sinon si l'arbre à insérer est nul alors on fait rien;

```



```

266     else if a_ins = null then null;
267     -- sinon si a n'a pas de pere alors on fait intervenir
268         -- l'exception no_pere.
269     else if a.all.pere = null then raise no_pere;
270         -- sinon si l'arbre a n'a pas de frere
271         else if a.all.frere=null then a.all.frere:=a_ins;
272             a_ins.all.pere:=a.all.pere;
273         -- sinon dans le cas general
274         else a_ins.all.frere:= a.all.frere;
275             a.all.frere:=a_ins;
276             a_ins.all.pere:=a.all.pere;
277         end if;
278     end if;
279 end if;
280 end if;
281
282 end An-Inserer_Frere ;
283
284
285 -- fonction An_Nombre_Fils:
286 function An_Nombre_Fils( a: in arbren ) return integer is
287
288 n:integer;
289 arb_aux:arbren;
290
291 begin
292
293 if An_Vide(a) then
294     raise arbre_vide;
295 else if a.all.premier_fils /= null then
296     n:=1;
297     arb_aux:=a.all.premier_fils;
298     while arb_aux.all.frere /= null loop
299         n:=n+1;
300         arb_aux:=arb_aux.all.frere;
301     end loop;
302     return n;
303 else
304     return 1;
305 end if;
306 end if;
307
308 end An_Nombre_Fils;
309
310
311 -- Procedure An_Supprimer_fils
312 procedure An_Supprimer_Fils(a : in out arbren; numero : in integer) is
313 begin
314     if a = Null then -- arbre vide
315         raise Arbre_vide;
316     elsif a.all.premier_fils = Null then -- arbre sans fils
317         raise no_nieme_fils;
318     elsif numero = 1 then -- n=1
319         a.all.premier_fils := a.all.premier_fils.all.frere;
320     else
321         begin

```

```

322      -- Supprime le (n-1)ième frère du premier fils
323      An_Supprimer_Frere(a.all.premier_fils , numero-1);
324      exception
325          when no_frere => raise no_nieme_fils;
326      end;
327  end if;
328  end An_Supprimer_Fils;
329
330
331  -- Procedure An_Supprimer_frere
332  procedure An_Supprimer_Frere(a : in out arbren; numero : in integer) is
333      temp : arbren;
334      p : integer;
335  begin
336      if a = Null then -- arbre vide
337          raise ARBRE_VIDE;
338      else
339          -- Parcours des frères
340
341          -- Initialisation
342          temp := a;
343          p := 1;
344
345          -- Parcours
346          while temp.all.frere /= Null and p < numero loop
347              temp := temp.all.frere;
348              p := p+1;
349          end loop;
350          -- temp.all.frere = Null or p=n
351
352          -- Fin de Parcours des frères
353
354          -- Supprimer le nième frère s'il existe , sinon lever une exception
355          if p=numero and temp.all.frere /= Null then
356              temp.all.frere := temp.all.frere.all.frere;
357          else
358              raise no_frere;
359          end if;
360      end if;
361  end An_Supprimer_Frere;
362
363
364
365
366  end p-arbren;

```

Annexe 2 : Code de p_sys.adb

Code de p_sys.adb

```

1  with text_io; use text_io;
2  with ada.integer_text_io; use ada.integer_text_io;
3  with ada.float_text_io; use ada.float_text_io;
4  with p_arbren;
5
6  Package body p_sys is
7
8
9  procedure lire_chaine(chaine: out string; l_ch: out integer; l_max:
10 in integer) is
11 begin
12   Get_line(chaine, l_ch);
13   for i in l_ch+1 .. l_max loop
14     chaine(i):= ' ';
15   end loop;
16 end lire_chaine;
17
18
19 — la procedure ecrire_txt:
20 procedure ecrire_txt(a: in repertoire) is
21 a_aux: fich;
22 decal: integer;
23 begin
24 decal:=4;
25 a_aux:= new fichiers;
26 put("———"); put(a.nom_du_repertoire);
27 — si le repertoire ne contient pas de fichiers.
28   if a.fichiers_du_repertoire=null then null;
29   else
30 — sinon si le repertoire contient des fichiers.
31   put(" : ");
32 — ecrire le nom du premier fichier
33 put(a.fichiers_du_repertoire.all.nom_du_fichier);
34 put(" .");
35 put(a.fichiers_du_repertoire.all.extension);
36 a_aux:=a.fichiers_du_repertoire.all.suivant;
37 —ecrire le nom des suivants
38 while a_aux/=null loop
39   put (" / ");
40   put(a_aux.all.nom_du_fichier);
41   put(" .");
42   put(a_aux.all.extension);
43   a_aux:=a_aux.all.suivant;
44 end loop;
45   end if;
46 end ecrire_txt;
47
48
49 — la fonction egalite valeur
50 function egalite_repertoire(a: in repertoire; b: in repertoire)
51 return boolean is
52
```

```

53 begin
54 return a.nom_du_repertoire=b.nom_du_repertoire;
55 end egalite_repertoire;
56
57
58 — La première fonctionnalité: insérer un fichier dans le répertoire
59 — courant.
60 procedure creer_fichier(a0: in out arbren; nom_fichier: in string;
61 contenu_fichier: in string; extension_du_fichier: in string) is
62 v: repertoire;
63 f: fich;
64 begin
65 f:= new fichiers;
66 v:=an_valeur(a0);
67
68 f.all.nom_du_fichier:=nom_fichier;
69 f.all.contenu_du_fichier:= contenu_fichier;
70 f.all.extension := extension_du_fichier;
71 f.all.suivant:=v.fichiers_du_repertoire;
72 v.fichiers_du_repertoire:=f;
73 An_changer_valeur(a0,v);
74
75 end creer_fichier;
76
77
78 function contenu_fichier(f: in fich) return string is
79
80 begin
81 put_line("*****");
82 put_line("Le contenu de votre fichier est:");
83 return(f.all.contenu_du_fichier);
84 end contenu_fichier;
85
86
87
88 procedure Afficher_rep_fich(a: in arbren) is
89 begin
90 an_afficher(a);
91 end Afficher_rep_fich;
92
93
94 function changer_repertoire(arbre_total: in arbren;
95 nom_rep_cible: in string) return arbren is
96 f: fich;
97 a: arbren;
98 v: repertoire;
99 rep_cour: repertoire;
100 begin
101 f:= new fichiers;
102
103 v.nom_du_repertoire:=nom_rep_cible;
104 v.fichiers_du_repertoire:= null;
105
106 a:=An_rechercher(arbre_total,v);
107 if an_vide(a) then
108 put("Le repertoire cible n'existe pas");

```

```

109     else rep_cour:= An_valeur(a);
110     end if;
111 —Afficher_rep_fich(a);
112 return a;
113 end changer_repertoire;
114
115
116 function recherche_fichier(a: in arbren; nom: string; ext: string)
117 return fich is
118 f: fich;
119 f1: fich;
120 v: repertoire;
121 begin
122 v:=An_valeur(a);
123 f:= new fichiers;
124 f1:=new fichiers;
125 f:=v.fichiers_du_repertoire;
126
127 while f.all.nom_du_fichier /= nom
128 and then f.all.suivant/=null loop
129     f:=f.all.suivant;
130 end loop;
131 if f/=null and then f.all.nom_du_fichier=nom
132 and then f.all.extension=ext then
133     f1.all.nom_du_fichier:=f.all.nom_du_fichier;
134     f1.all.extension:=f.all.extension;
135     f1.all.contenu_du_fichier:=f.all.contenu_du_fichier;
136     f1.all.suivant:=null;
137     else put_line("le_fichier_n'existe_pas_ds_le_repertoire_source");
138     end if;
139 return f1;
140 end recherche_fichier;
141
142
143 function copie_fichier(a0:in arbren;a:in arbren;nom_fichier:in string;
144 nom_rep_d : in string ;ext: in string) return arbren is
145
146 f: fich;
147 c: arbren;
148
149 begin
150 f:=new fichiers;
151 f := recherche_fichier (a, nom_fichier , ext);
152 c:=changer_repertoire (a0, nom_rep_d);
153 creer_fichier(c,f.all.nom_du_fichier ,f.all.contenu_du_fichier ,
154 f.all.extension);
155 new_line;
156 return c;
157 end copie_fichier ;
158
159
160 procedure supprimer_fichier(a:in arbren;nom:in string;ext:in string) is
161 —f: fich;
162 p : fich;
163 Begin
164 p:=new fichiers;

```

```

165 p := an_valeur(a).fichiers_du_repertoire;
166 if An_Vide (a) then raise arbre_vide;
167 else
168   while p.all.nom_du_fichier /= nom and then p/= null loop
169 —and then p.all.suivant.all.suivant/=null loop
170     p := p.all.suivant;
171   end loop;
172 — si le fichier n est pas trouve
173 if p.all.nom_du_fichier/=nom and then p.all.extension/=ext
174 then put( "_fichier_inexistant_" );
175 — sinon si le fichier est l avant dernier:
176 else if p.all.nom_du_fichier=nom and then p.all.extension=ext
177 and then p.all.suivant/=null
178 and then p.all.suivant.all.suivant=null
179 then p.all.nom_du_fichier:=p.all.suivant.all.nom_du_fichier;
180 p.all.contenu_du_fichier:=p.all.suivant.all.contenu_du_fichier;
181 p.all.extension:=p.all.suivant.all.extension;p.all.suivant:=null;
182 — sinon si c est dernier
183 else if p.all.nom_du_fichier=nom and then p.all.extension=ext
184 and then p.all.suivant=null then p.all.nom_du_fichier:="_____";
185 p.all.contenu_du_fichier:="_____"&"_____"&"_____"&
186 "_____"&"_____"&"_____"&"_____"&"_____"&
187 "_____"&"_____";p.all.extension:="___";p:=null;
188
189   else
190     p.all.nom_du_fichier := p.all.suivant.all.nom_du_fichier;
191     p.all.contenu_du_fichier:=p.all.suivant.all.contenu_du_fichier;
192     p.all.extension:=p.all.suivant.all.extension;
193     p.all.suivant:=p.all.suivant.all.suivant;
194   end if;
195   end if;
196   end if;
197 end if;
198 exception
199 when constraint_error => put ( "_fichier_inexistant_" );
200
201 end supprimer_fichier ;
202
203
204 procedure supprimer_repertoire(a: in out arbren; numero1: in integer) is
205
206 begin
207   An_supprimer_fils(a, numero1);
208 end supprimer_repertoire;
209
210
211 procedure changer_nom_rep (a:in out arbren; nouveau_nom : in string) is
212 b: arbren;
213 c: repertoire;
214 begin
215 b:=a;
216 c:= An_valeur(b);
217 c.nom_du_repertoire := nouveau_nom;
218
219 an_changer_valeur (a,c) ;
220

```

```

221 end changer_nom_rep;
222
223
224 procedure renommer_fichier(a0: in out arbren ;
225 nom_repertoire_courant: in string; nom_fichier_a_renommer : in string;
226 nouveau_nom_fichier : in string; ext1: in string; ext2: in string) is
227
228 f: fich;
229 c: arbren;
230 begin
231
232
233 f:= new fichiers;
234 c:= changer_repertoire(a0,nom_repertoire_courant);
235 f:=recherche_fichier(c,nom_fichier_a_renommer,ext1);
236 creer_fichier(c,nouveau_nom_fichier,f.all.contenu_du_fichier,ext2);
237 supprimer_fichier (c,nom_fichier_a_renommer,ext1);
238
239 end renommer_fichier;
240
241
242
243 -----RECHERCHE FICHIERS-----
244
245 FUNCTION Contient (Chaine_Contentu:IN String;L_Contentu: IN Integer ;
246 Chaine_Rech: IN String;L_Rech:IN Integer) RETURN Boolean IS
247     I:Integer ;
248     Chaine2:String(1..Nmax) ;
249     BEGIN
250     --verifier si chaine_rech a plus de caractere auquel on retourne faux
251     IF L_Contentu<L_Rech THEN
252         RETURN False ;
253     ELSE
254         I:=1;
255         --calculer le nbre d'elts sont en commun entre
256         -- les 2 chaines dans la meme position
257         WHILE Chaine_Contentu(I)=Chaine_Rech(I) LOOP
258             I:=I+1 ;
259         END LOOP ;
260         IF I>=L_Rech THEN
261             RETURN True ;
262
263         --verifier à nouveau en decalant la recherche d'un caractere
264         ELSE FOR M IN 2..L_contenu LOOP
265             Chaine2(M):= Chaine_Contentu(M) ;
266         END LOOP ;
267         FOR M IN 1..L_Contentu-1 LOOP
268             Chaine2(M):= Chaine2(M+1) ;
269         END LOOP;
270         RETURN Contient(Chaine2,L_Contentu-1,Chaine_Rech,L_Rech);
271     END IF ;
272     END IF ;
273
274     END Contient ;
275
276 PROCEDURE Recherche_Fichiers(A:IN Arbren ; Chaine:IN String ;

```

```

277 J : IN Integer ) IS
278     Arbre:Arbren ;
279     I:Integer ;
280     Inter:fich ;
281 BEGIN
282
283     Inter:=An_Valeur(A).fichiers_du_repertoire ;
284     WHILE Inter/=NULL LOOP
285         — afficher tous les fichiers qui dans leur contenu il y'a la
286 — chaine recherchee
287         IF Contient(Inter.All.Contenu_du_fichier ,Nmax,Chaine,J) THEN
288             Put(Inter.All.Nom_du_fichier) ;
289             new_line;
290             ELSE
291                 —sinon rien
292                 NULL ;
293             END IF ;
294             Inter:=Inter.All.Suivant ;
295         END LOOP ;
296         I:=1;
297         —on relance la recherche dans les fichiers des sous repertoire
298         WHILE An_Nombre_Fils(A)<I and then NOT(An_Vide(An_Fils(A,I))) LOOP
299             Arbre:=An_Fils(A,I) ;
300             Recherche_Fichiers(Arbre,Chaine,J) ;
301             I:=I+1;
302         END LOOP;
303
304     END Recherche_Fichiers ;
305
306
307 end p_sys ;

```

*** Fin ***