Projet Huffman

0.1

Généré par Doxygen 1.8.11

Table des matières

Chapitre 1

A_FAIRE

- Make propre
- Générer document Doxygen
- Fermez tous les trucs ouverts avec mattézob logiciel
- Fichier README avec la manière d'utilisation, nos bugs résiduels, et les fichiers testés
- Rapprt de 1 a 2 pages (fonctionnement, test, bug, structure fichier compréser, choix technique)
- Améliorer l'affichage du compresseur

Chapitre 2

Huffman

Le contrôle continu est soumis à la règle du max , il n'est donc pas indispensable mais est fortement recommandé !

Les démos sur machine du projet « Code de Huffman » auront lieu fin décembre. Si vous désirez y participer, le binôme (groupe de deux étudiants) doit :

- envoyer à Michel Meynard (meynard@lirmm.fr) et Pierre Pompidor (pompidor@lirmm.fr), une archive de vos codes 2 JOURS AVANT LA SOUTENANCE ;
- en retour, nous vous enverrons un créneau plus précis de votre passage.

Veuillez dans votre archive (nommée par les noms des participants au projet), créer un petit fichier README qui indiquera l'état d'avancement de votre projet (et par exemple les bogues résiduels). Par ailleurs, TOUS les participants au projet devront être présents (les absents ne seront sinon pas notés).

Le jour de la soutenance, les participants doivent venir avec :

- un micro rapport papier de 1 à 2 pages décrivant les choix importants effectués ;
- les listings (fichiers sources) documentés (doxygen) ;
- des réponses aux questions indiquées dans le CC ;

Chaque soutenance durera 12 à 15 minutes ! C'est extrêmement rapide, aussi soyez certain d'avoir les exécutables prêts à fonctionner sur votre compte Unix ainsi que les fichiers exemples. Une recompilation du projet durant la soutenance n'est pas envisageable !

une démonstration de l'application durant 5 minutes (utilisant des fichiers fournis par les examinateurs) ; — des réponses précises des 2 étudiants aux questions posées durant 5 minutes ; — un micro rapport papier de 1 à 2 pages décrivant les choix importants effectués ; — des listings papiers commentés (doxygen)

Quel est le nombre maximum de caractères (char) différents ? — Comment représenter l'arbre de Huffman ? Si l'arbre est implémenté avec des tableaux (fg, fd, parent), quels sont les indices des feuilles ? Quelle est la taille maximale de l'arbre (nombre de noeuds) ? — Comment les caractères présents sont-ils codés dans l'arbre ? — Le préfixe du fichier compressé doit-il nécessairement contenir l'arbre ou les codes des caractères ou bien les deux (critère d'efficacité) ? — Quelle est la taille minimale de ce préfixe (expliquer chaque champ et sa longueur) ? — Si le dernier caractère écrit ne finit pas sur une frontière d'octet, comment le compléter ? Comment ne pas prendre les bits de complétion pour des bits de données ? — Le décompresseur doit-il reconstituer l'arbre ? Comment ?

Le projet est décomposé en deux programmes C : huf.c le programme de compression utilisé selon la syntaxe suivante : $huf source dest où source est un fichier quelconque et où dest est le nom du fichier généré par compression à la Huffman du fichier source . Cette compression doit afficher les informations suivantes sur la sortie standard : — liste des caractères et de leur probabilité d'apparition ; — arbre de Huffman (tableaux fg fd parent ou par indentation) ; — affichage des codes de chaque caractère (codeChar(E)=010100) ; — longueur moyenne de codage ; — taille originelle de la source, taille compressée et gain en pourcentage comme dans l'exemple suivant : Taille originelle : 5194 ; taille compressée : 3761 ; gain : 27.6% ! dehuf.c le programme de décompression utilisé selon la syntaxe suivante : $dehuf dest où dest est un fichier compressé à la Huffman. Le fichier décompressé sera envoyé directement sur la sortie standard.

Chapitre 3

Index des structures de données

3.1 Structures de données

Liste des structures de données avec une brève description :

Chapitre 4

Index des fichiers

4.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

Chapitre 5

Documentation des structures de données

5.1 Référence de la structure noeud

Noeud d'un arbre binaire.

`#include <defNoeud.h>`

Champs de données

- unsigned int **pere**
- unsigned int **fg**
- unsigned int fd
- double frequences

5.1.1 Description détaillée

Noeud d'un arbre binaire.

Noeud est une petite structure comportant les indices de son noeud père et de ses noeuds fils (droit et gauche). Lui est également associé la fréquence d'apparation dans un fichier du caractère associé à ce noeud.

Noeud est une petite structure comportant les indices de son noeud père et de ses noeuds fils. Lui est également associé la fréquence d'apparation dans un fichier du caractère associé à ce noeud

5.1.2 Documentation des champs

5.1.2.1 unsigned int noeud : :fd

Indices des noeuds père, fils gauche et fils droit

5.1.2.2 double noeud : :frequences

Fréquence d'apparation du caractère lié au noeud

La documentation de cette structure a été générée à partir du fichier suivant :
- compresseur/include/defNoeud.h

Chapitre 6

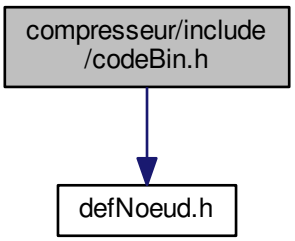
Documentation des fichiers

6.1 Référence du fichier compresseur/include/codeBin.h

Déclaration de la fonction qui va générer les codes binaires.

```
#include "defNoeud.h"
```

Graphe des dépendances par inclusion de codeBin.h :



Fonctions

— char ** CodeBin (Noeud *arbre, int racine)
Génère les codes binaires.

6.1.1 Description détaillée

Déclaration de la fonction qui va générer les codes binaires.

6.1.2 Documentation des fonctions

6.1.2.1 char** CodeBin (Noeud * *arbre,* int *racine*)

Génère les codes binaires.

Ces codes binaires sont associés à chaque caractère présent dans le fichier d'origine.

Paramètres

arbre	Arbre
racine	Indice de la racine de l'Arbre

Renvoie

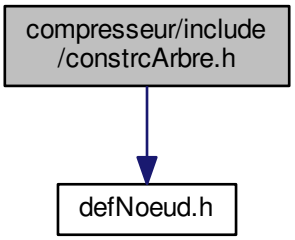
Index Tableau associant les codes binaires aux différents caractères

6.2 Référence du fichier compresseur/include/constrcArbre.h

Déclaration de la fonction qui va construire l'arbre.

```
#include "defNoeud.h"
```

Graphe des dépendances par inclusion de constrcArbre.h :



Fonctions

— int ConstrcArbre (double ∗tab_frequence, Noeud ∗arbre)
Construction de l'arbre.

6.2.1 Description détaillée

Déclaration de la fonction qui va construire l'arbre.

6.2.2 Documentation des fonctions

6.2.2.1 int ConstrcArbre (double ∗ *tab_frequence,* Noeud ∗ *arbre*)

Construction de l'arbre.

Paramètres

tab_frequence	Tableau des fréquences de répartition des caractères
arbre	Arbre préalablement initialisé

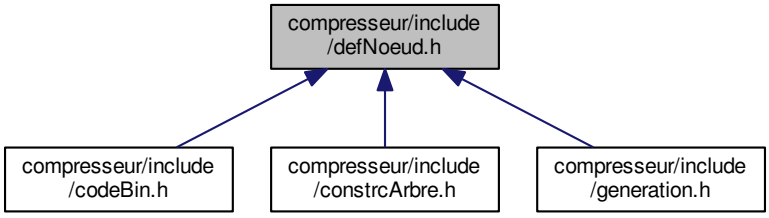
Renvoie

Ind_racine Indicide de la racine de l'arbre

6.3 Référence du fichier compresseur/include/defNoeud.h

Définit la structure Noeud.

Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Structures de données

— struct noeud
Noeud d'un arbre binaire.

Définitions de type

— typedef struct noeud Noeud
Noeud d'un arbre binaire.

6.3.1 Description détaillée

Définit la structure Noeud.

6.3.2 Documentation des définitions de type

6.3.2.1 typedef struct noeud Noeud

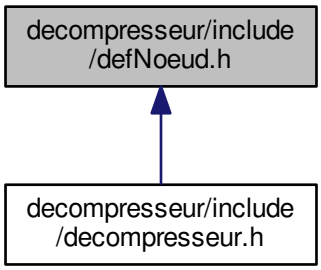
Noeud d'un arbre binaire.

Noeud est une petite structure comportant les indices de son noeud père et de ses noeuds fils (droit et gauche). Lui est également associé la fréquence d'apparation dans un fichier du caractère associé à ce noeud.

6.4 Référence du fichier decompresseur/include/defNoeud.h

Définit la structure Noeud.

Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Structures de données

— struct noeud
Noeud d'un arbre binaire.

Définitions de type

— typedef struct noeud Noeud
Noeud d'un arbre binaire.

6.4.1 Description détaillée

Définit la structure Noeud.

6.4.2 Documentation des définitions de type

6.4.2.1 typedef struct noeud Noeud

Noeud d'un arbre binaire.

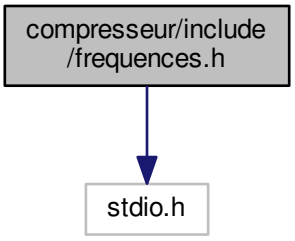
Noeud est une petite structure comportant les indices de son noeud père et de ses noeuds fils. Lui est également associé la fréquence d'apparation dans un fichier du caractère associé à ce noeud

6.5 Référence du fichier compresseur/include/frequences.h

Définit la fonction de génération des fréquences.

```
#include <stdio.h>
```

Graphe des dépendances par inclusion de frequences.h :



Fonctions

— double ∗ CalculFrequencesCaractere (FILE ∗fichier)
Fonction de calcul de la répartition des fréquences dans un fichier.

6.5.1 Description détaillée

Définit la fonction de génération des fréquences.

6.5.2 Documentation des fonctions

6.5.2.1 double∗ CalculFrequencesCaractere (FILE ∗ *fichier*)

Fonction de calcul de la répartition des fréquences dans un fichier.

Paramètres

fichier	Fichier où les fréquences des caractères doivent être calculées

Renvoie

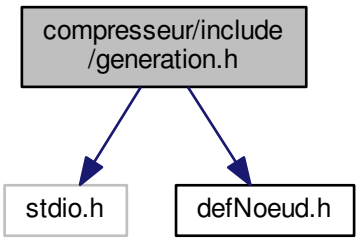
tab_frequence Un tableau contenant les fréquences des caractères contenus dans le fichier. L'indice d'une case correspond au code ASCII du caractère correspondant

6.6 Référence du fichier compresseur/include/generation.h

Déclaration de la fonction de génération du fichier compressé

```
#include <stdio.h>
#include "defNoeud.h"
```

Graphe des dépendances par inclusion de generation.h :



Fonctions

— void Generation (char ∗∗index, FILE ∗entre, FILE ∗sortie, int racine, Noeud ∗arbre)
Ecrit dans le fichier "compressé" les codes binaires des caractères.

6.6.1 Description détaillée

Déclaration de la fonction de génération du fichier compressé

6.6.2 Documentation des fonctions

6.6.2.1 void Generation (char ∗∗ *index,* FILE ∗ *entre,* FILE ∗ *sortie,* int *racine,* Noeud ∗ *arbre*)

Ecrit dans le fichier "compressé" les codes binaires des caractères.

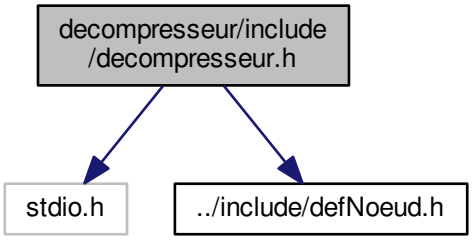
Paramètres

index	Tableau de codes binaires associés aux différents caractères
entre	Fichier d'entrée
sorti	Fichier de sortie (fichier compressé)
racine	Indice de la racine de l'arbre
arbre	Arbre

6.7 Référence du fichier decompresseur/include/decompresseur.h

```
#include <stdio.h>
#include "../include/defNoeud.h"
```

Graphe des dépendances par inclusion de decompresseur.h :



Macros

— #define ERR_FORMAT 1
Définit ERR_FORMAT = 1.

Fonctions

— int Decompression (FILE ∗Entree, FILE ∗Sortie)
Fonction permettant de décompresser un fichier.

6.7.1 Description détaillée

Déclaration de la fonction de décompression

6.7.2 Documentation des fonctions

6.7.2.1 int Decompression (FILE ∗ *Entree,* FILE ∗ *Sortie*)

Fonction permettant de décompresser un fichier.

Paramètres

Entree	Fichier compressé d'entrée
Sortie	Fichier de sortie décompressé

Renvoie

Retourne ERR_FORMAT si ce n'est pas un fichier qui peut être décompressé, retourne 0 sinon

Index