# Compréhension de programmes : Analyse de l'architecture de GIMP

Groupe 1

Vendredi 9 Novembre 2018

## Table des matières

# 1 Introduction

Pour réaliser notre projet GIMP nous avons formé un groupe de 5 membres qui sont :
- PANCHALINGAMOORTHY Gajenthran
- LOKO Loïc
- MOTAMED SALEHI Amin
- BOUCHIHA Anas
- GHOUIBI Ghassen.

Ce devoir sur GIMP était réparti en 3 étapes : l'organisation de l'archive (avec la description des répertoires principaux qui ne sont pas du code source), l'organisation du code (avec la description et la justification de celle-ci des répertoires de code) et enfin le schéma de l'arborescence (avec notamment l'utilisation de la commande `tree`).
Nous tenons à préciser avant de passer aux sections suivantes que les commandes seront en gras (`commande`) et que le nom des fichiers et des dossiers sera encadré (*fichier*)

# 2 Organisation de l'archive

Dans cette partie-là, nous décrirons de manière brève les différents répertoires qui ne sont pas du code source. Chaque sous-partie contiendra un répertoire ou un ensemble de répertoires traitant d'un même sujet.

## 2.1 Auteurs

Les fichiers *AUTHORS*, *authors.xml*, *authors.dtd* et *authors.xsl* correspondent à la liste de l'ensemble des auteurs/contributeurs du logiciel (les gens qui ont codé mais également ceux qui ont documenté).

## 2.2 Licence

Le fichier *LICENSE* contient les différentes licences du programme. En réalité ce fichier là ne traite que la GNU General Public License (GNU GPL) en nous renvoyant vers le fichier *COPYING* dont on va parler dans la sous-partie suivante.

## 2.3 Copie

Le fichier *COPYING* contient la licence publique générale GNU (GNU GPL) qui permet de fixer les conditions légales de distribution d'un logiciel libre du projet GNU (source : Wikipedia).

## 2.4 Journal de modifications

Les fichiers *ChangeLog* ou encore *ChangeLog.pre...* (où les symboles "..." correspondent à une suite de caractère) sont des journaux de modifications ou plus communément appelés changelog. Il s'agit d'une liste des différentes modifications. On y trouvera les commits (git) des différentes modifications avec le nom de l'auteur (de la modification), la date, le titre de la modification et évidemment les modifications apportées dans les fichiers (insertions et suppressions).
Les *ChangeLog.pre...* sont des pré-versions de *ChangeLog*.

## 2.5 Hacking

Le fichier *HACKING* comporte des détails concernant le hack du logiciel GIMP (des informations sur les prérequis pour pouvoir utiliser GIMP, la manière dont procédera la compilation, les patchs ou encore l'accès à GIMP via git).

## 2.6 Installation

Le fichier *INSTALL* nous apporte des informations sur la manière dont on doit installer GIMP sur notre machine en nous énumérant les différentes étapes à suivre par exemple.

## 2.7 News

Les fichiers *NEWS* ou encore *NEWS.pre...* (où les symboles "..." correspondent à une suite de caractère) décrivent les nouveautés, les changements ou encore les corrections apportées dans les différents répertoires (et également en général) de GIMP entre chaque version.
Les *NEWS.pre...* sont des pré-versions de *NEWS* qui elle, se base sur la version stable de GIMP.

## 2.8 Readme

Le fichier *README* (qu'on pourrait traduire par lisezmoi en français) contient de nombreuses informations concernant les versions du logiciel, les site Web faisant référence au logiciel (la page officiel : http ://www.gimp.org/ par exemple), customisation...
Il est conseillé de lire ce fichier avant même de commencer à toucher au logiciel, pour avoir des informations (que faire ? où se diriger ? sur quel fichier se diriger ?) sur la manière de l'installer et de l'utiliser.

## 2.9 Configuration

Le fichier *configure* permet de configurer l'application dans différentes machines. Il met en relation *Makefile.in* ou encore *Makefile.am*. En effet, *configure* génèrera un *Makefile* depuis le *Makefile.in* qui lui-même a été généré par le *Makefile.am* en utilisant automake et autoconf (source : stackoverflow.com).

# 3 Organisation du code source

Désormais, nous devons nous occuper des répertoires de code qui constituent une grande partie du projet. Pour cela, nous avons décidé d'abord de commencer à lire la documentation et donc de nous diriger vers les dossiers *devel-docs* et *docs* dans le but d'avoir des renseignements sur les autres répertoires avant même de les avoir traités. Chercher la documentation n'était pas un exercice difficile. Le nom des dossiers concernant la documentation est explicite (car les dossiers contiennent le mot "docs"), ce qui nous demande très peu d'efforts pour la recherche.

## 3.1 docs

Le dossier *docs* contient la documentation plutôt tourné pour un utilisateur commun qu'un développeur car il contient des informations et des explications sur ce qu'est GIMP (définition, le but de GIMP), sur les différentes activités réalisables dans GIMP en tant qu'utilisateur ou encore sur les différentes fonctionnalités que proposent GIMP (en tant qu'outil d'édition et de retouche d'image). Néanmoins, il n'est pas déconseillé pour un développer de s'y aventurer !
Comme indiqué précédemment, le contenu de ce dossier n'était pas difficile à deviner : son nom est explicite et la majorité des fichiers présents dans le dossier est écrit dans le langage humain (en anglais).

## 3.2 devel-docs

Le dossier *devel-docs* contient la documentation plutôt destinée pour les développeurs (d'où le nom "devel"). Il donne des informations qui seront potentiellement utiles aux développeurs en décrivant les différents répertoires de code que l'on peut trouver dans GIMP (par exemple les lib... ou encore le dossier *app*) mais également les conventions que doivent adoptées les développeurs (par exemple les inclusions dans le fichier *includes.txt*).
On se doutait du contenu de *devel-docs* (le nom est explicite et le dossier contenait plusieurs fichiers textes) mais le fichier *README* nous a également expliqué les fichiers/dossiers présents dans *devel-docs*.
Le dossier *devel-docs* comporte également des sous-dossiers que nous allons décrire de manière brève car nous les verrons dans les prochaines sections.

### 3.2.1 app

Le sous-dossier *devel-docs/app* qui veut dire "Application" fait référence au dossier *app* (nous allons le décrire dans une prochaine section) situé au même niveau que le *devel-docs* dans l'arborescence de GIMP. Le sous-dossier se contente de lister ce que l'on peut voir dans le dossier *app* (fonctions, types, sections...). Pour cela, nous avons pris une fonction du fichier *devel-docs/app/app-sections.txt* et vérifié à l'aide d'un `grep` si il existait bien cette même fonction dans le dossier *app*.

### 3.2.2 lib...

Il existe 8 sous-dossiers *devel-docs/lib...* qui sont tous organisés de la même manière et font tous références aux dossiers *lib...* correspondants à leur nom situé au même niveau que *devel-docs* dans l'arborescence de GIMP. Les sous-dossiers sont tous formés de la même manière et décrivent les fonctions présentes dans les dossiers *lib...*

### 3.2.3 tools

Le sous-dossier *devel-docs/tools* fait référence au dossier *tools* (nous allons le décrire dans une prochaine section) situé au même niveau que le *devel-docs* dans l'arborescence de GIMP. Le sous-dossier se contente de décrire les différents outils présents dans le programme comme par exemple l'outil "capture d'écran". Le *README.shooter* nous a grandement facilité la tâche.

## 3.3 cursors

Le dossier *cursors* contient des images (.png) correspondantes aux différentes formes que peut prendre notre curseur que l'on peut apercevoir dans GIMP (chacune représentant une fonctionnalité dans GIMP. Par exemple, la loupe servant à zoomer). Le nom du dossier, le contenu du dossier (des images en grande majorité donc concerne l'affichage) mais également notre expérience en tant qu'utilisateur sur des logiciels d'édition et de retouche d'image nous ont permis d'arriver à ce raisonnement.

## 3.4 ressources

Le dossier *data* regroupe les différentes ressources dont le logiciel a besoin pour le code. Vu qu'il s'agit d'outil d'édition et de retouche d'image, le logiciel aura besoin de palettes ( *palettes* , de motifs *patterns*, de dégradés ( *gradients* ), des brosses/pinceaux ( *brushes* ), des dynamiques de la brosses ( *dynamics* ), des raccourcis/astuces ( *tips* ), ou encore des tags/noms pour chaque ressource ( *tags* ).
Pour comprendre, notre expérience nous a également beaucoup aidé. Mais nous avons également cherché les extensions qui se trouvaient dans les sous-dossiers de *data* (.ggr, .gpl ...) afin de mieux comprendre l'utilité du dossier. Le noms des sous-dossiers nous a également permis à reconnaitre le dossier *data*.

## 3.5 desktop

Le dossier *desktop* possède 7 sous-dossiers qui contiennent chacun une image correspondant au logo de GIMP. Le nom de chaque sous-dossier indique la dimension de l'image qu'il contienne.
En lisant le nom du dossier ("desktop" qui veut dire bureau en français) mais également le fichier *gimp.desktop.in.in* (où on voit le mot "icon"), il semblerait que ce dossier corresponde à l'icône du logiciel sur le Bureau.

## 3.6 etc

Le dossier *etc* contient les fichiers de configuration et les paramètres du programme (tous les fichiers se terminant par "rc"). Il permet de conserver les données de l'utilisateur, conserver sa réalisation/ses activités entre deux sessions GIMP, ou encore la gestion de la souris et du clavier.
Ce dossier était plus complexe à comprendre, car nous avons très peu d'indice

nous permettant de savoir le contenu du dossier. Du coup, nous étions obligés de lire tous les fichiers du dossier afin de nous faire une idée. De plus, tous les fichiers se terminent par "rc" donc nous nous sentions obligés de réaliser des recherches pour comprendre la définition de ce mot-clé. Le "rc" permet de dire que ce sont des fichiers qui sont exploités dès que l'application GIMP se lance.

## 3.7 libgimp

Le dossier *libgimp* correspond à la librairie GIMP contenant les fonctions principales qui seront utiles pour la création d'un plug-in GIMP. Il possède l'ensemble des énumérations et définitions du code, des fonctions utiles pour les plug-ins, des fonctions qui vont manipuler les images et leurs propriétés, la manipulations des outils (ressources dont on avait parlé dans une des sections précédentes) ou encore les interactions qu'il peut y avoir dans le logiciel (selection d'outil par exemple).
Pour décrire ce dossier, nous nous sommes aidés du dossier *devel-docs* afin d'avoir une description concise des différents fichiers du dossier (et plusieurs de *index.html* qui se situe dans le sous-dossier *devel-docs/libgimp*.
Cependant, pour en être sûr, nous avons voulu vérifier le contenu du dossier *libgimp*. Nous remarquons, dans le nom des fichiers, de fortes relations avec ce que nous avons dit précédemment. Par exemple :
- pour la manipulations des images et de leurs propriétés, nous avons *gimpimage**, *gimpdraw**...
- pour la manipulation des outils, nous avons *gimpbrushes**, *gimpfont**, *gimpgradient**...
- pour les interactions, nous avons *gimpselection**, *gimpselect**...

## 3.8 libgimpbase

Le dossier *libgimpbase* correspond à la librairie GIMP qui va contenir toutes sortes de fichiers (et donc fonctions) qui n'ont pas forcément des points communs entre eux mais seront utiles pour le fonctionnement du logiciel. On y trouve des énumérations des types de données (mélange, masque, dégradé...), des macros et des constantes qui vérifieront les capacités du logiciel, des fonctions qui géreront les rectangles ou encore les limites de certains types de données ou constantes.
Pour décrire ce dossier, nous nous sommes aidés du dossier *devel-docs* afin d'avoir une description concise des différents fichiers du dossier. Mais nous

avons néanmoins vérifié si il y avait une corrélation entre les deux en vérifiant les différents fichiers de *libgimpbase*.

## 3.9 libgimpcolor

Le dossier *libgimpcolor* correspond à la librairie contenant les fonctions qui vont mettre en relations les couleurs (la gestion des couleurs) comme par exemple les couleurs RGB, HSV, CMYK, l'interpolation ou encore les conversions de couleurs.
Pour décrire ce dossier, nous nous sommes aidés du dossier *devel-docs* afin d'avoir une description concise des différents fichiers du dossier. Mais nous avons néanmoins vérifié si il y avait une corrélation entre les deux en vérifiant les différents fichiers de *libgimpcolor*.
Par exemple :
- pour les différentes définitions des couleurs, on a *gimphsv*, *gimprgb*, *gimpcmyk*
- pour la gestion/modification des couleurs, on a *gimprgb-parse*, *gimpbiliniear*

## 3.10 libgimpconfig

Le dossier *libgimpconfig* correspond à la librairie contenant les fonctions qui vont traiter à propos de la configuration du logiciel. On y trouve le code de sérialisation, de désérialisation, des macros permettant d'installer les propriétés de configuration ou encore des fonctions qui envoient les informations de configurations dans un fichier.
Pour décrire ce dossier, nous nous sommes aidés du dossier *devel-docs* afin d'avoir une description concise des différents fichiers du dossier. Mais nous avons néanmoins vérifié si il y avait une corrélation entre les deux en vérifiant les différents fichiers de *libgimpconfig*.
Par exemple :
- pour la sérialisation et la désérialisation, on a *gimpconfig-serialize*, *gimpconfig-deserialize*
- pour la gestion d'erreur et l'écriture dans un fichier, on a respectivement *gimpconfig-error*, *gimpconfig-writer*

## 3.11 libgimpmath

Le dossier *libgimpmath* correspond à la librairie qui va contenir des expressions mathématiques. On y trouve essentielement des matrices ou des vecteurs.
Pour décrire ce dossier, nous nous sommes aidés du dossier *devel-docs* afin

d'avoir une description concise des différents fichiers du dossier. Mais nous avons néanmoins vérifié si il y avait une corrélation entre les deux en vérifiant les différents fichiers de *libgimpmath*.
Par exemple :
- pour les matrices, on a *gimpmatrix*
- pour les vecteurs, on a *gimpvector*

## 3.12 libgimpmodule

Le dossier *libgimpmodule* correspond à la librairie qui va traiter les modules. On y trouve essentiellement des fonctions qui vont implémenter des modules ou encore des fonctions de tous les modules et leur chemin.
Pour décrire ce dossier, nous nous sommes aidés du dossier *devel-docs* afin d'avoir une description concise des différents fichiers du dossier. Mais nous avons néanmoins vérifié si il y avait une corrélation entre les deux en vérifiant les différents fichiers de *libgimpmodule*.
Par exemple :
- pour les implémentations, on a *gimpmodule*
- pour la recherche du chemin, on a *gimpmoduledb*

## 3.13 libgimpthumb

Le dossier *libgimpthumb* correspond à la librairie qui va traiter les vignettes. On y trouve la fonction qui va créer un objet GimpThumbnail qui représentera la vignette (les propriétés de ma vignette seront : la date de modification, les dimensions de ma vignette, le type de mon image ou encore la taille), des fonctions qui vont modifier/utilser cette objet, les erreurs possibles provenant de cette objet...
Pour décrire ce dossier, nous nous sommes aidés du dossier *devel-docs* afin d'avoir une description concise des différents fichiers du dossier. Mais nous avons néanmoins vérifié si il y avait une corrélation entre les deux en vérifiant les différents fichiers de *libgimpthumb*.
Par exemple :
- pour créer et gérer l'objet GimpThumbnail , on a *gimpthumbnail*
- pour modifier/utiliser l'objet, on a *gimpthumb-utils*
- pour la gestion d'erreurs, on a *gimpthumb-error*

## 3.14 libgimpwidgets

Le dossier *libgimpwidgets* correspond à la librairie qui va traiter l'ensemble des widgets (gadgets/vignettes). Ces widgets vont nous servir à zoomer, cliquer sur un button qui va ensuite nous permettre de déclencher une action, d'interagir avec l'utilisateur, de choisir une couleur depuis la palettes de couleurs...
Pour décrire ce dossier, nous nous sommes aidés du dossier *devel-docs* afin d'avoir une description concise des différents fichiers du dossier. Mais nous avons néanmoins vérifié si il y avait une corrélation entre les deux en vérifiant les différents fichiers de *libgimpwidgets*.
Par exemple :
- pour le widget "button", on a *gimpbutton*
- pour le widget pour la recherche de fichier, on a *gimpfileentry*
- pourla le widget "couleur", on a *gimpcolorscales*

## 3.15 m4macros

Le dossier *m4macros* contient des macros. On y trouve toute sorte de fichier comme des fichiers qui vérifie si le compilateur détecte bien un ensemble de flag, vérifie si le programme a bien la possibilité de créer une extension python...
Ce dossier était assez court, il suffisait donc de lire le contenu de chaque fichier pour se faire une idée. Nous sommes également allés chercher sur Internet, la signification de l'extension .m4 (langage de traitement de macros).

## 3.16 menus

Le dossier *menus* contient des fichiers .xml qui traitent des informations présentes dans les menus. Dans les menus, on peut donc trouver les brosses et l'édition de celle-ci (*brush-editor-menu*), les motifs (*palettes-menu.xml*), les images, la gestion des erreurs, les curseurs...
Comme le dossier *m4macros* le dossier est assez court, il suffit donc de lire un par un le contenu de chaque fichier.

## 3.17 modules

Le dossier *modules* contient des modules du programme notamment l'interaction avec le programme permettant de modifier/gérer la couleur .
Pour ce dossier, il ne suffisait pas que de regarder le nom des fichiers mais également leur contenu. Les mots clefs "input", "controller", "selector" et "dis-

play" permettent d'expliquer notre raisonnement car le "controller" s'agit des actions que l'on va réaliser sur un de notre périphérique, "input" s'agit de la réponse que reçoit le programme de notre périphérique, "selector" concerne l'action réalisée dans le programme suite à la réponse reçue du "input" et enfin le "display" concerne l'affichage.

## 3.18 po

Le dossier *po* correspond à la traduction dans différentes langues des fichiers du code GIMP. Dans chaque fichier de *po*, nous avons le chemin du fichier que nous voulons traduire, le texte original puis le texte traduit dans la langue indiqué sur le nom du fichier.
Pour ce dossier, l'extension .po nous a permis de savoir qu'il s'agissait de traduction et que le nom des fichiers correspondaient à la langue. Par exemple, *de.po* pour une traduction en allemand (de = deutsch = allemand).
Pour la suite des fichiers commençant par "po", nous allons les voir dans les prochaines sous-sections sans forcément rentrer dans les détails (vu qu'il s'agit de la même chose que *po*).

### 3.18.1 po-libgimp

Le dossier *po-libgimp* correspond à la traduction dans différentes langues des fichiers situés dans le répertoire *libgimp* comme son nom l'indique.

### 3.18.2 po-plug-in

Le dossier *po-plug-in* correspond à la traduction dans différentes langues des fichiers situés dans le répertoire *plug-ins* comme son nom l'indique.

### 3.18.3 po-python

Le dossier *po-python* correspond à la traduction dans différentes langues des fichiers situés dans le répertoire *plug-ins/pygimp*. Cette fois-ci nous avons pas de nom de dossier comportant le mot "python". Du coup, il faut donc ouvrir un des fichiers .po, et voir le chemin des textes qui sont traduits. Ici, les textes originaux viennent tous du répertoire *plug-ins/pygimp*.

### 3.18.4 po-script-fu

Le dossier *po-script-fu* correspond à la traduction dans différentes langues des fichiers situés dans le répertoire *plug-ins/script-fu*.

### 3.18.5 po-tips

Le dossier *po-tips* correspond à la traduction dans différentes langues des fichiers situés dans le répertoire *data/tips* comme son nom l'indique.

## 3.19 themes

Le dossier *themes* décrit le thème du programme c'est-à-dire la disposition et le style de chaque image (la manière dont il se présente dans le programme). Le dossier possède également 2 sous-dossiers, un qui correspond au thème par défaut : *Default* et un autre dossier qui correspond à un thème où le style des images sera différent (taille des images moins important par exemple) : *Small*.
Pour comprendre ce fichier, nous avons lu le fichier *gtkrc*. Dans ce fichier, on insère le fichier que l'on souhaite afficher et la manière dont on veut l'afficher (style) à savoir sa taille, l'espace entre les différentes images, les marges, le relief, les bordures... Ce fichier existe sous le même dans les dossiers *Default* et *Small* (évidemment avec le style des images qui est différent).

## 3.20 build

Le dossier *build* permet de créer le programme pour les possesseurs de Windows. Il génère les versions des ressources pour les possesseurs de Windows.
Les seules pistes que nous avons eu sont les commentaires (`# Version resources for Microsoft Windows`) et les inclusions (`#include <winver.h>` qui permet de savoir la version du système en cours sous Windows).

## 3.21 tools

Le dossier *tools* présente différentes sortes d'outils où chacun a des fonctions différentes. Par exemple, les vérifications .def par rapport à leur librairie respective (*tools*), convertir en svg (*gimppath2svg*) ou encore créer un noyau (*kernelgen*). Il comporte également un dossier *pdbgen*.
Le nom du dossier n'est pas totalement explicite, du coup ce fut un peu difficile de comprendre le dossier car il fallait s'intéresser à tous les fichiers présents dans le dossier.

## 3.22 plug-ins

Le dossier *plug-ins* comporte les différents plug-ins du projet. Le dossier est divisé en 34 sous-dossiers chacun des sujets plus ou moins différents que nous allons traiter dans les sous-sections suivantes. Pour chacun des sous-dossiers du dossier *plug-ins*, nous nous sommes principalement aidés soit du nom des fichiers, soit du nom des sous-dossiers ou soit du code que l'on survolait afin de comprendre le contenu du sous-dossier (notamment les fichiers .h pour les prototypes).

### 3.22.1 color-rotate

Le dossier *color-rotate* permet de faire une rotation des couleurs c'est-à-dire qu'elle va pouvoir échanger deux intervalles de couleurs.
Les commentaires et le nom du dossier "color-rotate" ont permis de trouver la réponse. Les fonctions de certains fichiers et les macrons ("RADIUS") nous ont fait rester sur cette idée également.

### 3.22.2 common

Le dossier *common* comporte l'ensemble des plug-ins les plus communs (les plus courants). Il possède de nombreux fichiers qui peuvent être catégorisés : par exemple, nous avons les plug-ins s'occupant de la gestion des fichiers ( *file-** ), la gestion de la couleur ( *color-** ), la gestion des outils ( *smooth-palette* ...
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.3 file-bmp

Le dossier *file-bmp* s'occupe de la gestion des images en format bitmap à savoir l'accès avec ( *bmp-read* ) ou encore l'écriture avec ( *bmp-write* ).
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.4 file-compressor

Le dossier *file-compressor* permet de gérer la compression des fichiers.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.5 file-faxg3

Le dossier *file-faxg3* s'occupe de la gestion des images en format faxg3. Les fonctions de (*faxg3* nous donnent des indications sur le contenu du dossier notamment la fonction `load_image`.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.6 file-fits

Le dossier *file-fits* s'occupe de la gestion des images en format flexible image transport system (fits). Les fonctions de *file-fits* nous donnent des indications sur le contenu du dossier notamment les fonctions `load_image`, `save_image` ou encore `create_new_image`.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.7 file-fli

Le dossier *file-fli* s'occupe de la gestion des vidéos/animations en format fli et flc. Les fonctions de *file-fli* nous donnent des indications sur le contenu du dossier notamment les fonctions `load_image`, `save_image` ou encore `fli_read_lc` pour les compressions.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.8 file-ico

Le dossier *file-ico* s'occupe de la gestion des images/icônes en format ico. Les fichiers *ico-load* ou encore *ico-save* nous donnent des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.9 file-jpeg

Le dossier *file-jpeg* s'occupe de la gestion des images en format jpeg. Les fichiers *jpeg-load* ou encore *jpeg-save* nous donnent des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.10 file-psd

Le dossier *file-psd* s'occupe de la gestion des images en format jpeg. Les fichiers *psd-load* ou encore *psd-save* nous donnent des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.11 file-uri

Le dossier *file-uri* s'occupe de la gestion des fichiers en format uri. Les fonctions de *uri* nous donnent des indications sur le contenu du dossier notamment les fonctions `load_image` et `save_image`.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.12 file-xjt

Le dossier *file-xjt* s'occupe de la gestion des fichiers en format xjt. Le fichier *README* nous donne des indications sur le contenu du dossier : le plugin permet de charger et sauvegarder les images en RGB et en GRAY.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.13 flame

Le dossier *flame* décrit le plug-in réalisant un "cosmic recursive fractal flames" qui est une visualisation mathématiques . Le fichier *README* nous donne des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.14 fractal-explorer

Le dossier *fractal-explorer* décrit le plug-in réalisant un explorateur de fractales qui est un filtre. Le dossier *examples* et le nom du dossier nous donnent des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.15 gfig

Le dossier *gfig* décrit le plug-in dessinant des figures géométriques telles que la ligne, le cercle, la courbes ou encore la spirale. Le fichier *README* nous donne des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.16 gimpressionist

Le dossier *gimpressionist* décrit le plug-in créant un effet de peinture naturel. Le fichier *README* nous donne des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.17 gradient-flare

Le dossier *gradient-flare* décrit le plug-in permettant d'obtenir une image avec une source de lumière éblouissante tout en réalisant des dégradés. Le fichier *README* nous donne des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.18 help

Le dossier *help* décrit le plug-in permettant de solliciter l'aide du programme (on peut également écrire notre requête afin de potentiellement avoir le résultat qu'on recherche par exemple). Les fonctions de *help* nous donnent des indications sur le contenu du dossier notamment les fonctions `query` pour les requêtes et `run` pour l'exécution de la requête ou encore, les macros `#define GIMP_HELP_DEFAULT_DOMAIN "http://www.gimp.org/help"` (redirection vers le site Web de GIMp) et `#define GIMP_HELP_DEFAULT_LOCALE "en"` (les aides seront en anglais par defaut).
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.19 help-browser

Le dossier *help-browser* décrit le plug-in permettant de solliciter l'aide du programme mais cette fois-ci sous un navigateur. On aperçoit notamment l'ajout de fonction comme `gimp_throbber_action_connect_proxy` dans le

fichier *gimpthrobberaction* qui est propre au réseau ou encore `help_browser_show_help` dans *help-browser*.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.20 ifs-compose

Le dossier *ifs-compose* décrit le plug-in créant un système de fonctions fractales itératives. Le fichier *README* nous donne des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) pour savoir ce que le dossier fait.

### 3.22.21 imagemap

Le dossier *imagemap* décrit le plug-in créant une carte cliquable.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) qui nous a grandement aidé pour ce plug-in pour savoir ce que le dossier fait.

### 3.22.22 lighting

Le dossier *lighting* décrit le plug-in qui réalise un effet d'éclairage sur l'image en question. Le fichier *README* nous donne des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) qui nous a grandement aidé pour ce plug-in pour savoir ce que le dossier fait.

### 3.22.23 map-object

Le dossier *map-object* décrit le plug-in qui place une image dans un objet. Le fichier *README* nous donne des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) qui nous a grandement aidé pour ce plug-in pour savoir ce que le dossier fait.

### 3.22.24 maze

Le dossier *maze* décrit le plug-in qui réalise un labyrinthe. Les fichiers *README* et *maze-algoithms* (correspondant à l'algorithme du labyrinthe) nous donnent des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) qui nous a grandement aidé pour ce plug-in pour savoir ce que le dossier fait.

### 3.22.25 metadata

Le dossier *metadata* décrit le plug-in qui concerne les metadonnées.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) qui nous a grandement aidé pour ce plug-in pour savoir ce que le dossier fait.

### 3.22.26 pagecurl

Le dossier *pagecurl* décrit le plug-in qui donnent l'impression de tourner la page. Les images (avec probablement la zone en blanc pour l'image en question et l'autre zone pour l'autre page) nous donnent des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) qui nous a grandement aidé pour ce plug-in pour savoir ce que le dossier fait.

### 3.22.27 print

Le dossier *print* décrit le plug-in qui s'occupe de l'affichage d'une surface.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) qui nous a grandement aidé pour ce plug-in pour savoir ce que le dossier fait.

### 3.22.28 pygimp

Le dossier *pygimp* décrit des plug-ins réalisés en Python. Le sous-dossier fbox*pygimp/plug-ins* nous donne des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) qui nous a grandement aidé pour ce plug-in pour savoir ce que le dossier fait.

### 3.22.29 script-fu

Le dossier *script-fu* décrit le plug-in implémentant le langage Scheme. Le fichier *README* nous donne des indications sur le contenu du dossier.

### 3.22.30 selection-to-path

Le dossier *selection-to-path* décrit le plug-in permettant de passer une selection vers un chemin. Le fichier *README* nous donne des indications sur le contenu du dossier.

### 3.22.31 twain

Le dossier *twain* décrit le plug-in permettant d'utiliser un scanneur. Le fichier *README* et l'image *gimp-twainnous* (représentant un scanneur) donnent des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) qui nous a grandement aidé pour ce plug-in pour savoir ce que le dossier fait.

### 3.22.32 winsnap

Le dossier *winsnap* décrit le plug-in permettant de réaliser une capture d'écran. Le fichier *winsnap* et notamment les mots-clefs "snapshot" et "capture" dans ce fichier donnent des indications sur le contenu du dossier.
Il suffit simplement de lire le nom des fichiers (voire les commentaires en début de page) qui nous a grandement aidé pour ce plug-in pour savoir ce que le dossier fait.

## 3.23 app

Le dossier *app* (qui veut dire "Application") est, selon nous, le plus important des dossiers. On peut le considérer comme l'équivalent de *src* pour Gnuplot (même si le dossier *src* de Gnuplot concentre beaucoup plus de fichiers que *app*. Les fonctionnalités sont beaucoup plus dispatchées sur GIMP). Ce dossier comporte 21 sous-dossiers que nous allons traiter dans les prochaines sous-sections.
On remarque que la plupart des sous-dossiers contiennent le **-types* ou **-enums* ou les deux qui donnent des informations (brèves) sur le contenu du dossier en question. Cependant, cela nous limite sur la description du dossier mais ça reste suffisant dans le cadre du devoir.

### 3.23.1 actions

Le dossier *actions* décrit les différentes actions que peut réaliser l'utilisateur dans le programme. Par exemple, *brushes-actions* explique les différentes manipulations possibles avec la brosse, *gradient-editor-action* rassemble les actions réalisables par l'utilisateur avec les dégradés. Il y a également des actions pour les interactions, pour les commandes, le mouvement des curseurs, des images, la sauvegarde.
Il suffit simplement de lire le nom des fichiers pour savoir ce que le dossier fait et survoler les fichiers .h pour voir les prototypes des fonctions et être sûr du contenu de chaque fichier.

### 3.23.2 base

Le dossier *base* décrit la "base" de notre programme. Le nom du dossier "base" ne nous disait pas grand chose, du coup nous nous sentions obligés de lire les fichiers qu'il contient. Ce dernier rassemble des fichiers traitant la manipulation des couleurs *color-** et de ses propriétés (saturation, luminosité, contraste...) ( *lut-funcs* et *hue-saturation*, des pixels *pixel-** ou encore la gestion de la toile *tile**. Cependant, nous ne pouvons pas donner de nom particulier à cet ensemble de fichier. Peut-être qu'il se nomme "base" car ces fichiers correspondent à la base du programme (c'est-à-dire que sans ces codes là le programme n'aurait aucun sens).

### 3.23.3 composite

Le dossier *composite* décrit le compositing qui consiste à mélanger plusieurs images afin de réaliser des effets spéciaux. Néanmoins, pour en être sûr, nous avons vérifié le contenu fichier *gimp-composite*, plus précisément la fonction `gimp_composite_mode_astext` introduisant un `switch` où chaque option correspond des effets que l'on peut appliquer sur un ensemble d'image (par exemple le `GIMP_COMPOSITE_XOR` ne s'applique pas sur une seule image mais sur plusieurs.
Le fait d'avoir des utilisateurs de logiciel d'édition et de retouche d'image dans notre groupe, nous facilite la tâche dans certains dossiers du projet.

### 3.23.4 config

Le dossier *config* décrit la configuration des différents composants de l'application (plug-in, display, core, base...).

Nous nous sommes contentés de lire les noms des fichiers et le nom du fichier pour arriver à cette conclusion.

### 3.23.5 core

Le dossier *core* correspond au noyau du programme, il traite les informations concernant les images, le contexte, les outils, le nom de chaque élément présent... Pour
Le nom du dossier a été d'une grande utilité mais cela reste assez vague pour expliquer le contenu de ce dernier. Donc nous avons décidé de lire les fichiers *core-enums* et *core-types* qui regroupent l'ensemble des énumérations et des objets utilisés dans le dossier.

### 3.23.6 dialogs

Le dossier *dialogs* regroupe des fenêtres qui contiennent un outil et ses options. L'ensemble des fenêtres se nomme dock. Lorsqu'on a exploré les fichiers *dialogs* et *dialogs-constructors*, nous avons remarqué plusieurs la présence du mot-clé "dock" ; ce qui nous a permis de nous rassurer.
Le fait d'avoir des utilisateurs de logiciel d'édition et de retouche d'image dans notre groupe, nous facilite la tâche dans certains dossiers du projet.

### 3.23.7 display

Le dossier *display* regroupe tous les fichiers qui concerne l'affichage. On y trouve l'affichage de la toile, la gestion de la toile, l'affichage du curseur, l'affichage des icônes/images, l'affichage des outils. Les fichiers *display-types*, *display-enums* nous ont servi pour connaître l'ensemble des structures (objets) et des énumérations utilisés dans ce dossier.
Par exemple, dans *display-types*, on sait que l'on va utiliser les structures `GimpCanvas*` pour l'affichage et la gestion de la toile, `GimpDisplay*` pour l'affichage des différents éléments ou encore `GimpCursorView` pour le curseur.

### 3.23.8 file

Le dossier *file* correspond à la gestion des fichiers. On y trouve l'accès au fichier *file-open*, la sauvegarde *file-save* ou encore l'exploitation d'un fichier *file-utils*.
Il suffit simplement de lire le nom des fichiers pour savoir ce que le dossier fait et survoler les fichiers .h pour voir les prototypes des fonctions et être sûr du contenu de chaque fichier.

### 3.23.9 gegl

Le dossier *gegl* décrit la manière dont va être traitée l'image. On y trouve des modifications qui concerne la luminosité *gimpbrightnesscontrastconfig*, la saturation *gimpdesaturateconfig* ou encore la coloration *gimpcolorbalanceconfig*. Nous nous sommes contentés de lire le nom des fichiers dans le dossier pour comprendre le contenu du dossier *gegl*.

### 3.23.10 menus

Le dossier *menus* décrit les menus présents dans notre applications et les interactions avec eux. On y trouve les images présents dans la barre de menu (*image-menu*) ou encore les différents outils mis à notre disposition dans la barre de menu (*tool-option-menu*).
Il suffit simplement de lire le nom des fichiers qui nous a grandement aidé pour pour savoir ce que le dossier fait et lire également le noms de certaines fonctions. Mais comme la plupart des sous-dossiers de *app*, je suis allé voir le fichier **-types*. Cependant celui ne possède rien, si ce n'est qu'une inclusion du fichier *action/action-type*. Du coup, nous sommes allés voir le contenu de ce fichier, et nous remarquons la présence d'`enum` de différentes manières de sauvegardes comme on peut en retrouver dans les menus.

### 3.23.11 paint

Le dossier *paint* décrit tous les outils disponibles pour la retouche d'image. On y trouve les brosses (*gimpbrushcore*) ou encore la gomme (*gimperaser*). Comme la plupart des sous-dossiers de *app*, le nom est explicite donc on sait à l'avance le contenu du dossier. Mais nous sommes quand même allés lire le fichier *paint-types* et *paint-enums* pour connaître les objets et les énumérations de ce dossier. Dans *paint-types*, les structures `GimpBrush` (brosse), `GimpInk` (encre ou stylo à plume) ou encore `GimpEraser` (gomme) font bien références aux outils.

### 3.23.12 paint-funcs

Le dossier *paint-funcs* décrit toutes les fonctionnalités des outils disponibles pour la retouche d'image. On y trouve *paint-funcs* qui comprend les différentes fonctions qui gèrent les outils. Dans le fichier, `paint-funcs`, on a des fonctions qui s'occupe du traitements des pixels (échange de de pixels, modification de pixels...), de colorier des régions, de créer des courbes...

### 3.23.13 pdb

Le dossier *pdb* comporte les procédures pdb qui vont être stocker dans les bases de données des procédures (comme on l'a vu dans le dossier *tools/pdbgen*). Si on regarde le fichier *pdb-types*, on constate que les structures traitent des procédures (`GimpProcedure`, `GimpPluginProcedure` ou encore `GimpTemporaryProcedure`. On remarque également que la majorité des fichiers .c contiennent le mot "cmds", "pdb" ou "procedure".

### 3.23.14 plug-in

Le dossier *plug-in* explique la manière dont on va traiter les plu-gins du programme. Le nom du dossier est clair pour qu'on comprenne son contenu. Dans le fichier, *plug-in-types* comporte les structures qui représentent la gestion, le debug, la définition des plug-in : `GimpPluGin`, `GimpPluGinDebug` ou encore `GimpPlugGinDef`...

### 3.23.15 tests

Le dossier *tests* regroupe l'ensemble des tests du programme. Le nom du dossier est clair pour qu'on comprenne son contenu. Il suffit également de lire le nom des fichiers pour connaître le sujet des tests. Par exemple, *test-tools* correspond aux tests des différents éléments dans le dossier *tools* ou encore *test-xcf* correspond aux vérifications des différents éléments dans le dossier *xcf*.

### 3.23.16 text

Le dossier *text* s'occupe de la gestion des textes (zone de texte, police). Le nom du dossier est clair pour qu'on comprenne son contenu. Par exemple, *text-types* regroupe les structures traitant la liste des police, la gestion de police, les textes, les zones de textes.

### 3.23.17 tools

Le dossier *tools* s'occupe de la gestion des outils dans l'application (différents de *paint* qui, lui, s'occupe des outils concernant les retouche d'image). Le nom du dossier est clair pour qu'on comprenne son contenu. Par exemple, dans *tools-types*, on retrouve les outils de retouche d'image `GimpPaintTool`, la brosse `GimpBrushTool` ou encore les outils contrôlant les couleurs `GimpColorTool`.

#### 3.23.18 vectors

Le dossier *vectors* s'occupe de la gestion des vecteurs dans l'application. Le nom du dossier est clair pour qu'on comprenne son contenu. Par exemple, dans *tools-types*, `GimpVectors` correspond à la représentation d'un vecteur. De plus, la définition des vecteurs fait référence aux vecteurs présents dans le dossier *libgimpmath* que nous avons vu dans une section précédente, comme on le voit dans *gimpvectors.c* (`#include "libgimpmath/gimpmath.h"`).

#### 3.23.19 widgets

Le dossier *widgets* s'occupe de la gestion des widgets dans l'application. Le nom du dossier est clair pour qu'on comprenne son contenu. On remarque que les widgets peuvent être coupés en plusieurs catégories. Par exemple, dans *widgets-types*, on a des structures qui s'occupent des widgets concernant les docks `GimpDock`, les modifications d'image `GimpColorMapEditor`, les menus `GimpAction`...

#### 3.23.20 xcf

Le dossier *xcf* s'occupe de la gestion des fichiers en format xcf (comme nous l'avons vu sur le dossier *devel-docs*). On comprend assez facilement le role du dossier en lisant le nom des fichiers. Par exemple, `xcf-load` permet de charger un fichier xcf, `xcf-save` permet de sauvegarder dans un fichier xcf ou encore `xcf-write` permet d'écrire dans un fichier xcf.

# 4 Schéma de l'arborescence

Enfin, nous pouvons créer suite à cela une arborescence de ce projet en catégorisant et en trouvant les points communs des différents fichiers. Pour cela, nous allons nous aider de la commande `tree`. Vu que nous ne voulons pas traiter les sous-dossiers, nous allons écrire : `tree -L 1`.
Il existe plusieurs manières de trier nos 23 dossiers et 43 fichiers. Nous vous présentons la notre.

## 4.1 bleu : informations pour les utilisateurs communs

Les fichiers/dossiers en bleu représentent les différents fichiers que les utilisateurs peuvent consulter sans difficulté : il n'y a pas de code, simplement des informations sur le logiciel utilisé comme la description du logiciel, la manière de l'installer ou encore les différentes licences. Nous aurions pu

également couper ce fichier en plusieurs catégories en mettant d'un côté le *docs* (en l'associant avec *devel-docs* pour la documentation) et de l'autre les autres fichiers mais nous avons décidé de procéder de cette manière.

## 4.2 gris : libraries / modules / plug-ins

Nous avons décidé de mettre les libraries, les modules et les plug-ins ensemble car ce sont des fonctions qui sont implémentées de manière séparée du noyau. Néanmoins nous aurions pu les séparer car malgré le fait qu'ils partagent ce point commun, il existe de grosses différences entre eux. Par exemple, un plug-in est un ensemble de fichiers qui peut être intégré à un programme spécifique pour une application spécifique (pas forcément utilisable dans d'autres applications) tandis qu'une librairie est une ensemble de fichiers qui peut être intégré à plusieurs programme pour plusieurs applications différentes (il n'est pas spécifique à un programme).

## 4.3 jaune : informations pour les développeurs

Les fichiers/dossiers en jaune concernent plutôt les développeurs car ils rassemblent les changelogs (journaux de modifications), les nouveautés entre chaque versions et la documentation pour les développeurs. Encore une fois, nous aurions pu séparer les *ChangeLog* et les *NEWS* car ils ne traitent pas les informations de la même manière. *ChangeLog* traite toutes les modifications et offre plus de détails sur les modifications réalisées (date, auteur(s), titre). Quant à *NEWS*, il traite les informations de manière condensée.

## 4.4 blanc : auteurs

Les fichiers en blanc correspondent aux informations à propos des auteurs/contributeurs. Nous avons hésité à les associer avec les fichiers en bleu car il s'agit d'informations dont un utilisateur commun peut consulter mais également avec les fichiers en jaune car cela concerne les développeurs.

## 4.5 marron : aspect visuel

Les fichiers/dossiers en marron abordent l'aspect visuel du programme, et regroupe en grande partie les images du programme.

## 4.6 rouge : m4macros et tools

Les fichiers/dossiers en rouge traitent à propos de l'optimisation du programme : les vérifications, les détections, le développement en Python, pdb... On retrouve notamment le dossier *m4macros* (avec les fichiers m4) et *tools* (avec les fichiers .c et .py et surtout .pdb).

## 4.7 cyan : traductions

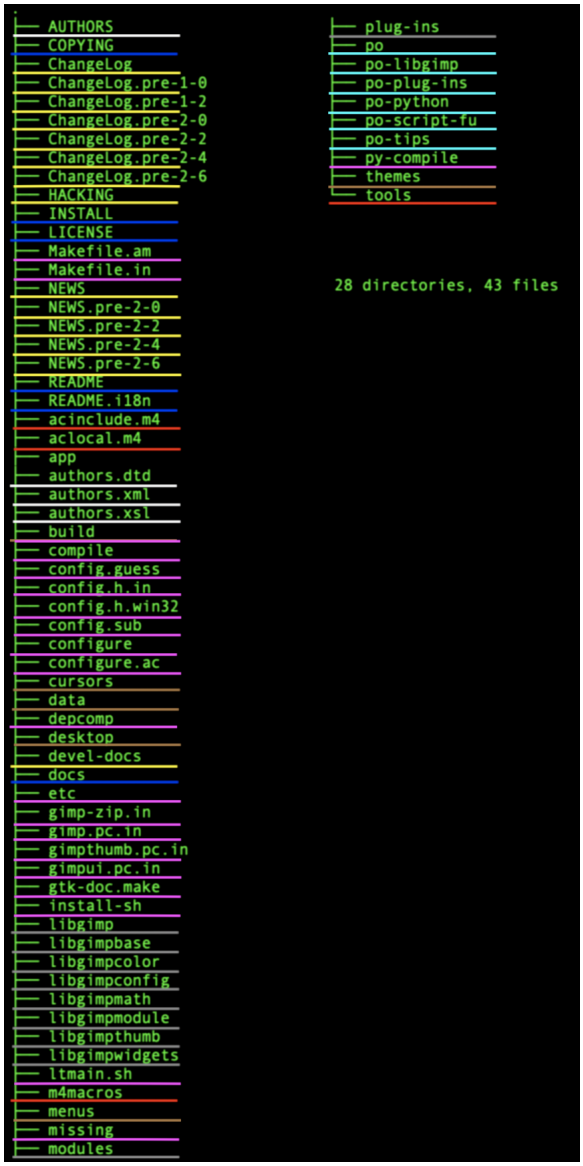
Les dossiers en cyan correspondent aux traductions des différents fichiers et donc il s'agit des fichiers dont le nom comporte le mot "po". On aurait pu peut-être les mettre avec "les informations pour les développeurs" car ce sont des données qui peuvent être utiles pour un développeur, pour mieux se repérer dans le code.

## 4.8 violet : configurations/compilations/scripts

Les fichiers/dossiers en violet regroupent les scripts, les configurations ou encore les compilations. On y trouve de nombreux fichiers exécutables mais également des fichiers .in ainsi que des Makefile (même si les Makefile ne doivent pas être dans cette catégorie, ils aboutissent à réaliser ce genre de fichier donc nous nous sommes permis de les mettre là).

## 4.9 app

Nous considérons *app* comme un dossier à part comme nous l'avions dit précédemment. Du coup, pour rester cohérent avec nos propos, nous avons choisi de le laisser sans groupe. Vu qu'il fait appel à plusieurs dossiers, il peut être lié à de nombreux répertoires de code.


.
├── AUTHORS
├── COPYING
├── ChangeLog
├── ChangeLog.pre-1-0
├── ChangeLog.pre-1-2
├── ChangeLog.pre-2-0
├── ChangeLog.pre-2-2
├── ChangeLog.pre-2-4
├── ChangeLog.pre-2-6
├── HACKING
├── INSTALL
├── LICENSE
├── Makefile.am
├── Makefile.in
├── NEWS
├── NEWS.pre-2-0
├── NEWS.pre-2-2
├── NEWS.pre-2-4
├── NEWS.pre-2-6
├── README
├── README.i18n
├── acinclude.m4
├── aclocal.m4
├── app
├── authors.dtd
├── authors.xml
├── authors.xsl
├── build
├── compile
├── config.guess
├── config.h.in
├── config.h.win32
├── config.sub
├── configure
├── configure.ac
├── cursors
├── data
├── depcomp
├── desktop
├── devel-docs
├── docs
├── etc
├── gimp-zip.in
├── gimp.pc.in
├── gimpthumb.pc.in
├── gimpui.pc.in
├── gtk-doc.make
├── install-sh
├── libgimp
├── libgimpbase
├── libgimpcolor
├── libgimpconfig
├── libgimpmath
├── libgimpmodule
├── libgimpthumb
├── libgimpwidgets
├── ltmain.sh
├── m4macros
├── menus
├── missing
├── modules
├── plug-ins
├── po
├── po-libgimp
├── po-plug-ins
├── po-python
├── po-script-fu
├── po-tips
├── py-compile
├── themes
└── tools
28 directories, 43 files