Licence d'Informatique – Sorbonne Université
2I012-2018fev – Environnement de développement sous Linux
Devoir Make 2018

Le devoir est à rendre pour au plus tard **le 14 mai 2018 à minuit** selon les modalités fixées par votre adjoint de TME (moodle, courrier électronique) et sous la forme d'une archive tar-compressée de nom *numero_etudiant*`.tgz` et contenant un répertoire `jpeg-6b` avec les fichiers demandés dans les questions qui suivent.

L'archive à rendre aura la structure suivante :

```
% tar tvfz numero_etudiant.tar.gz
jpeg-6b/Makefile
jpeg-6b/src/Makefile
jpeg-6b/tests/Makefile
```

Introduction

Le but de ce devoir est de construire le logiciel `jpeglib`. Il vous est demandé d'écrire les fichiers `Makefile`, qui sont utilisés pour construire la bibliothèque `libjpeg.a`, ainsi que cinq utilitaires, réaliser des tests et installer le logiciel dans le système. Il n'est pas utile de comprendre le fonctionnement de la bibliothèque ni même de savoir ce qu'est une image JPEG.

Les fichiers `Makefile` ne se baseront sur aucune règle implicite par défaut ni aucune valeur par défaut de variable prédéfinie. Vous pourrez vérifier vous-même que cette contrainte est bien respectée en appelant la commande `make` avec l'option `-r`.

Vous trouverez sur le site Moodle, section "Devoir Make", l'archive tar-gzippée `https://moodle-sciences.upmc.fr/moodle/draftfile.php/4156/user/draft/325185917/jpeg-6b.tar.gz` qui contient une version du logiciel adaptée pour ce devoir. Après décompression de l'archive, on trouve un répertoire `jpeg-6b` qui contient plusieurs fichiers et 5 répertoires (`src`, `tests`, `jconfig`, `doc` et `man`). Cette archive est très différente de celle qu'on peut trouver sur le site officiel du logiciel. En effet, les fichiers ont été regroupés dans les 5 répertoires listés ci-dessus, et cela afin de mettre un peu de lisibilité. Les fichiers non utilisés par le devoir sont regroupés dans le répertoire `jconfig` (il contient diverses configurations du logiciel pour différents systèmes et architectures).

Dans la suite de l'énoncé, on suppose que l'on se place à la base du répertoire `jpeg-6b`.

1 Production de la bibliothèque `libjpeg.a`

La bibliothèque `libjpeg.a` est produite à partir des fichiers objet (d'extension `.o`) obtenus par compilation des fichiers sources C suivant trouvés dans le répertoire `src` :

```
jcapimin.c jcapistd.c jccoefct.c jccolor.c jcdctmgr.c jchuff.c
jcinit.c jcmainct.c jcmarker.c jcmaster.c jcomapi.c jcparam.c
jcphuff.c jcprepct.c jcsample.c jctrans.c jdapimin.c jdapistd.c
jdatadst.c jdatasrc.c jdcoefct.c jdcolor.c jddctmgr.c jdhuff.c
jdinput.c jdmainct.c jdmarker.c jdmaster.c jdmerge.c jdphuff.c
```

```
jdpostct.c jdsample.c jdtrans.c jerror.c jfdctflt.c jfdctfst.c
jfdctint.c jidctflt.c jidctfst.c jidctint.c jidctred.c jquant1.c
jquant2.c jutils.c jmemmgr.c jmemnobs.c
```

Pour produire les fichiers objets, nous utiliserons les options `-O2 -I.` du compilateur C.

Les fichiers sources dépendent d'un certain nombre de fichiers en-tête d'extension `.h` présents dans le répertoire `src` ainsi que du fichier `jconfig.h`, absent dans le répertoire `src`. Ce fichier décrit des caractéristiques essentielles du système pour construire la bibliothèque. Comme il dépend du système, ce fichier est fabriqué par l'utilisateur, à l'aide de la commande `ckconfig`, laquelle est construite par compilation directe du fichier source `ckconfig.c`. Ainsi, pour produire `jconfig.h`, il faut enchaîner la suite de commandes suivante :

```
% gcc ckconfig.c -o ckconfig
% ./ckconfig
Configuration check for Independent JPEG Group's software done.

I have written the jconfig.h file for you.

You should use makefile.ansi as the starting point for your Makefile.
% ls jconfig.h
jconfig.h
```

Enfin, la bibliothèque `libjpeg.a` est produite en rassemblant les fichiers objets avec un appel adéquat à la commande `ar`.

Livraison : Écrire le fichier `src/Makefile` qui fabrique `src/libjpeg.a`. Ce fichier a les contraintes suivantes :

- il utilise les variables `CC`, `AR`, `RANLIB`, `CPPFLAGS`, `CFLAGS` et `ARFLAGS` ;
- il utilise une règle implicite pour produire les fichiers `.o` ;
- il utilise une variable `LIB_SRC` qui contient la liste des fichiers source C qui permettent de produire la bibliothèque ;
- les dépendances entre les fichiers sont calculées par une cible virtuelle `depend`, et à l'aide du compilateur C, qui fabrique un fichier `.depend` inclus par le `Makefile` ;
- il produit le fichier `jconfig.h` qui dépend de `ckconfig.c`. L'absence du fichier `jconfig.h` doit provoquer une erreur de `make` ;
- l'appel `make -r libjpeg.a` produit la bibliothèque.

2 Production des cinq utilitaires

Dans le répertoire `src`, on trouvera les codes sources pour produire les cinq utilitaires suivants :

- la commande `cjpeg`, produite à partir des sources suivants :

  ```
  cjpeg.c rdppm.c rdgif.c rdtarga.c rdrle.c rdbmp.c rdswitch.c cdjpeg.c
  ```

  et de la bibliothèque `libjpeg.a` ;
- la commande `djpeg`, produite à partir des sources suivants :

  ```
  djpeg.c wrppm.c wrgif.c wrtarga.c wrrle.c wrbmp.c rdcolmap.c cdjpeg.c
  ```

  et de la bibliothèque `libjpeg.a`,

— la commande `jpegtran`, produite à partir des sources suivants :

```
jpegtran.c rdswitch.c cdjpeg.c transupp.c
```

et de la bibliothèque `libjpeg.a` ;

— des deux commandes `rdjpgcom` et `wrjpgcom` produites respectivement à partir de `rdjpgcom.c` et `wrjpgcom.c`.

Les fichiers objets sont produits à partir des fichiers sources avec les mêmes options de compilation que ceux de la section précédente.

Livraison : Compléter le fichier `src/Makefile` de façon à ce qu'il produise les 5 utilitaires mentionnés ci-dessus et avec les contraintes suivantes :

— la cible `depend` sera modifiée pour qu'elle produise les dépendances des utilitaires `djpeg`, `cjpeg` et `jpegtran` ;

— les fichiers sources des utilitaires `djpeg`, `cjpeg` et `jpegtran` seront indiqués dans les variables `CJPEG_SRC`, `DJPEG_SRC` et `JPEGTRAN_SRC` et ces variables utilisées pour la production des commandes ;

— les fichiers objets intermédiaires sont produits à l'aide de la même règle implicite décrite dans la question précédente ;

— la cible virtuelle `all` doit être aussi la cible par défaut qui produit bibliothèque et commandes.

3 Test du logiciel

Le logiciel est testé à travers le bon fonctionnement des commandes `cjpeg` (encodeur JPEG) et `djpeg` (décodeur JPEG). Ces deux commandes lisent ou écrivent des images JPEG à partir d'autres formats d'image tels que BMP, et PNM.

Le répertoire `tests` contient une image source JPEG (`testorig.jpg`) et le résultat de conversion vers les formats BMP et PNM, JPEG, et pour ce dernier une option dite "JPEG progressif". Ces images font office de référence, le principe du test est que les commandes `cjpeg` et `djpeg` produisent les mêmes images que les références. Pour comparer deux fichiers binaires, on utilise la commande système `cmp`.

Voici 6 appels des utilitaires `cjpeg`, `djpeg` et `jpegtran` qui produisent six images à partir de cinq images source. L'option `-outfile` donne le nom de l'image à produire.

```
% ../src/djpeg -dct int -ppm -outfile testout.ppm testorig.jpg
% ../src/djpeg -dct int -bmp -colors 256 -outfile testout.bmp\
testorig.jpg
% ../src/cjpeg -dct int -outfile testout.jpg testimg.ppm
% ../src/djpeg -dct int -ppm -outfile testoutp.ppm testprog.jpg
% ../src/cjpeg -dct int -progressive -opt -outfile testoutp.jpg testimg.ppm
% ../src/jpegtran -outfile testoutt.jpg testprog.jpg
```

Il est demandé d'écrire les 6 règles explicites qui fabriquent les 6 images `testout.ppm`, `testout.bmp`, `testout.jpg`, `testoutp.ppm`, `testoutp.jpg` et `testoutt.jpg`.

Les appels suivant permettent de vérifier que les images produites par les règles précédentes sont correctes :

```
% cmp testimg.ppm testout.ppm
% cmp testimg.bmp testout.bmp
% cmp testimg.jpg testout.jpg
% cmp testimg.ppm testoutp.ppm
```

```
% cmp testimgp.jpg testoutp.jpg
% cmp testorig.jpg testoutt.jpg
```

Il vous est demandé d'écrire une cible virtuelle `check` qui provoque la fabrications des 6 images et les tests de vérification sur ces images.

Livraison : Écrire le fichier `tests/Makefile` selon les contraintes suivantes :

- il contient les règles explicites demandées,
- la cible virtuelle `check` réalise les 6 tests (et donc produit 6 images).

4 Intendance

4.1 Fichiers src/Makefile et tests/Makefile

Ajouter à `src/Makefile` les deux cibles virtuelles suivantes :

- `clean` qui supprime tous les fichiers produits par la cible `all`,
- `install` qui réalise les opérations suivantes :
  - la bibliothèque `libjpeg.a` est copiée dans `$(DESTDIR)/lib`;
  - les fichiers `jconfig.h`, `jpeglib.h`, `jerror.h` et `jpegint.h` sont copiés dans `$(DESTDIR)/include`;
  - les commandes `cjpeg`, `djpeg`, `jpegtran`, `rdjpgcom` et `wrjpgcom` sont copiées dans `$(DESTDIR)/bin`.

  La variable `DESTDIR` sera initialisée à `/usr/local` et les répertoires qui n'existent pas devront être créés.

Ajouter à `tests/Makefile` la cible virtuelle `clean` qui supprime les fichiers créés par la cible `check`.

4.2 `Makefile` principal

Créer à la racine du répertoire `jpeg-6b` un fichier `Makefile` qui a les contraintes suivantes :

- il délègue les cibles virtuelles `clean` et `install` aux deux autres `Makefile`, et la cible `all` à `src/Makefile`;
- la cible `all` doit déléguer également la cible `check` à `tests/Makefile`, ce qui signifie qu'un appel `make all` à la racine du logiciel déclenche la production du logiciel suivi de son test ;
- il transmet la variable `DESTDIR` quand c'est nécessaire et sa valeur est par défaut est `/usr/local` ;
- en plus de déléguer sa cible, la cible `install` réalise les opérations suivantes :
  - Les fichiers d'extension `.1` du répertoire `man` sont copiés dans `$(DESTDIR)/share/man/man1`;
  - Les fichiers d'extension `.doc` du répertoire `doc` sont copiés dans `$(DESTDIR)/share/doc/jpeg-6b` et seront compressés à l'aide de la commande `gzip`;
  - Les fichiers `README` et `change.log` seront copiés dans le répertoire `$(DESTDIR)/share/jpeg-6b`.

On veillera à ce que le comportement lors de l'exécution des commandes `make all`, `make clean` et `make install`, soit insensible à la présence de fichiers de même nom dans les répertoires des `Makefile`.