Examen de contrôle continu du cours de pratique de la programmation

Arash Habibi & Frédéric Vivien

Jeudi 21 mars, 8H00–10H00

Documents autorisés

Les seuls documents autorisés pendant cette épreuve sont :

1. Les sujets de TP distribués cette année (dont le sujet 3 : GNU make).
2. Le supplément de cours distribué au TP 2.
3. Les notes de cours (manuscrites).
4. Les pages webs :
   – `http://icps.u-strasbg.fr/~vivien/Enseignement/PPP-2001-2002/`
   – `http://www-info.u-strasbg.fr/~habibi/PPP/02.GccMakefile.html`
   à l'exclusion de toute autre.

Préparation des sources à compiler

Dans votre compte vous trouverez une archive `SujetExamen.tgz`. Pour la décompresser, exécutez les commandes suivantes :

```
gunzip SujetExam.tgz
tar -xf SujetExam.tar
```

Vous obtiendrez de la sorte un répertoire `SujetExamen` contenant trois sous-répertoires : `LancerRayon`, `LibrariePpm` et `PpmMaker`.

Remarque importante

Dans tout cet examen, vous allez écrire des makefiles. En tout rigueur il vous faudrait spécifier, dans le prérequis des règles, les éventuelles dépendances entre fichiers sources (« .c ») et fichiers d'entêtes (« .h »). Par souci de simplicité, vous ignorerez ces possibles dépendances et vous ne ferez *jamais* figurer de fichier d'entêtes dans le prérequis des règles.

1 Compilation de la librairie `PpmMaker`

Les makefiles demandés dans cette partie seront construits dans le sous-répertoire `PpmMaker` du répertoire `SujetExamen`.

1.1 Compilation simple : `Makefile1`

Écrivez un fichier makefile `Makefile1` permettant de générer l'exécutable `main` à partir des fichiers sources `PpmMaker1.c`, `PpmMaker2.c` et `main.c`, et satisfaisant les contraintes suivantes (contraintes classiques sur les makefiles) :

1. Pendant le processus de fabrication de l'exécutable, chaque fichier source devra être compilé au plus une fois.

2. L'exécutable devra être regénéré à chaque fois qu'un des fichiers sources aura été modifié ; et seul les fichiers sources modifiés devront être recompilés.
3. La fabrication des fichiers objets à partir des fichiers sources devra faire l'objet d'une et une seule règle.
4. Les répétitions doivent être évitées par l'utilisation de variables.

Rappel : pour lancer `make` en utilisant la cible `MaCible` du fichier de makefile `MonJoliMakefile`, il faut utiliser l'option `-f` comme suit :

```
make -f MonJoliMakefile MaCible
```

1.2 Génération d'une librairie : `Makefile2`

Écrivez un fichier `Makefile2` incluant les règles de `Makefile1` et qui, en plus, fabriquera la librairie `PpmMaker` :

1. La librairie est fabriquée à partir des fichiers `PpmMaker1.o` et `PpmMaker2.o`.
2. Le makefile comportera une règle pour fabriquer la librairie statique `libPpmMaker.a`.
3. Le makefile comportera une règle pour fabriquer la librairie dynamique `libPpmMaker.so`.

1.3 Compilation à partir d'une librairie : `Makefile3`

Dans `Makefile1`, l'exécutable `main` était généré à partir des fichiers sources `PpmMaker1.c`, `PpmMaker2.c` et `main.c`. Maintenant que nous disposons de la librairie `PpmMaker`, un tel exécutable peut être généré en utilisant uniquement le fichier source `main.c` et la librairie `PpmMaker`.

Écrivez un makefile `Makefile3`, incluant toutes les règles de `Makefile2`, et permettant de générer :

1. Un exécutable appelé `main-static` obtenu par liaison statique à partir du fichier source `main.c` et de la librairie `PpmMaker`.
2. Un exécutable appelé `main-dynamic` obtenu par liaison dynamique à partir du fichier source `main.c` et de la librairie `PpmMaker`.

Rappel : pour que le compilateur `gcc` lie statiquement un exécutable aux librairies, il faut utiliser l'option `-static` ; pour que les liaisons soient dynamiques, il faut utiliser l'option `-shared` (ou ne pas indiquer d'options de ce type).

1.4 Nettoyage et installation de la librairie : `Makefile4`

Écrivez un fichier `Makefile4` contenant, en plus de toutes les règles de `Makefile3` :

1. Une règle permettant de générer les deux librairies (`libPpmMaker.a` et `libPpmMaker.so`) et les trois exécutables (`main`, `main-static` et `main-dynamic`).
2. Une règle `clean` permettant de supprimer tous les fichiers objets.

   Faites **très attention**, en utilisant la commande `rm`, à ne pas supprimer par erreur certains de vos makefiles, ou d'autres fichiers importants...
3. Une règle `veryclean` permettant de supprimer tous les fichiers objets, tous les exécutables et toutes les librairies.
4. Une règle `install` qui copiera :
   – les trois fichiers exécutables dans le répertoire `SujetExamen/LibrairiePpm/bin` ;
   – les deux fichiers librairies dans le répertoire `SujetExamen/LibrairiePpm/lib` ;
   – le fichiers d'entête dans le répertoire `SujetExamen/LibrairiePpm/include`.
   Vous effectuerez les copies au moyen de la commande `cp` :

   ```
   cp fichier1 fichier2 ... fichiern répertoire
   ```

   copie les fichiers `fichier1` à `fichiern` dans le répertoire `répertoire`.

2 Compilation de `LancerRayon`

Les makefiles demandés dans cette partie seront construits dans le sous-répertoire `LancerRayon/src` de `SujetExamen`.

2.1 Compilation simple : `Makefile5`

Écrivez un fichier `Makefile5`, satisfaisant les mêmes contraintes que `Makefile1` (cf. paragraphe 1.1), et permettant de construire un exécutable `main` à partir de tous les fichiers sources du répertoire `src` (`Couleur.c`, `main.c`, `Objet.c`, `Phong.c`, `Plan.c`, `Rayon.c`, `Scene.c`, `Source.c`, `Sphere.c`, `Triangle.c` et `Vecteur.c`) **et de la librairie** `PpmMaker` installée dans le répertoire `LibrairiePpm`. Pour construire cet exécutable, le compilateur aura *entre autres* besoin d'utiliser les fichiers d'entêtes stockés dans le sous-répertoire `include` de `LancerRayon`.

2.2 Génération automatique de la liste des fichiers objets : `Makefile6`

Écrivez un fichier `Makefile6` ayant les mêmes caractéristiques que `Makefile5` mais dans lequel la seule liste de fichiers explicite sera celle des fichiers sources : vous devez donc générer ici la liste des fichiers objets à partir de la liste des fichiers sources.

2.3 Génération automatique de la liste des fichiers sources : `Makefile7`

Écrivez un fichier `Makefile7` ayant les mêmes caractéristiques que `Makefile6` mais dans lequel nous n'aurez écrit aucune énumération de fichiers : votre makefile devra donc générer automatiquement la liste des fichiers sources présents dans le répertoire.

2.4 Compilation conditionnelle : `Makefile8`

Écrivez un fichier `Makefile8`, contenant toutes les règles de `Makefile7`, et qui, *suivant qu'une variable* `STATIC` *vaudra* `YES` *ou* `NO`, liera statiquement ou dynamiquement l'exécutable à la librairie `PpmMaker`.

Vous indiquerez au moyen de commentaires dans le makefile s'il est possible d'indiquer, à l'invocation du makefile, si l'on désire une liaison statique ou dynamique.

Vous ajouterez à `Makefile8` des règles `clean` et `veryclean` qui supprimeront : les fichiers objets (règle `clean`) ; les fichiers objets et l'exécutable (règle `veryclean`).

Rappel : toute ligne d'un makefile commençant par le caractère « # » est un commentaire.

3 Compilation plus propre de `LancerRayon` : `Makefile9`

Le makefile demandé dans cette partie sera construit dans le sous-répertoire `LancerRayon` du répertoire `SujetExamen`. Tout le travail fait dans la partie précédente se situait dans le sous-répertoire `src` de `LancerRayon`. Nous voulons ici un makefile `Makefile9`, situé dans le répertoire `LancerRayon`, effectuant le même travail que `Makefile8`, mais stockant les fichiers objets dans le sous-répertoire `obj` de `LancerRayon` (et non plus dans `src`) et stockant l'exécutable obtenu dans le sous-répertoire `bin` de `LancerRayon` (et non plus dans `src`).

4 Compilation globale : `Makefile10`

Le makefile demandé dans cette partie sera construit dans le répertoire `SujetExamen`. Il devra appeler les « bons » makefiles des sous-répertoires `PpmMaker` et `LancerRayon`. Il devra comporter les règles suivantes :

1. Une règle `compilation` qui compile et installe la librairie `PpmMaker`, puis qui compile l'exécutable de `LancerRayon`.
2. Une règle `clean` qui appelle les règles `clean` sur les sous-répertoires `PpmMaker` et `LancerRayon`, et une règle `veryclean` qui appelle les règles `veryclean` sur les mêmes sous-répertoires.
3. Une règle `all` qui effectue d'abord les compilations, puis qui élimine les fichiers objets (mais pas les exécutables). Cette règle devra être la règle lancée par défaut (c'est-à-dire si aucune règle n'est indiquée lors de l'invocation du makefile).