TP5 : Makefile

Comment gérer les dépendances de tâches

La synthèse sous Alliance se décompose en plusieurs outils s'exécutant chronologiquement sur un flux de données. Chaque outil possède ses propres options donnant des résultats plus ou moins adaptés suivant l'utilisation que l'on veut faire du circuit.

Les dépendances de données dans le flux sont matérialisées dans la réalité par une dépendance de fichier. Le fichier Makefile exécuté à l'aide de la commande make permet gérer ces dépendances. Différents exemples de fichiers seront fournis durant le TP.

Ce TP n'étant pas un cours sur le Makefile, nous nous limiterons à expliquer l'usage qui est fait dans les exemples fournis.

1 Principe de base : Les Règles

Un Makefile est un fichier contenant une ou plusieurs règles traduisant les dépendances entre les actions et les fichiers.

Voici une règle type Makefile :

```
#Rq: chaque commande doit être précédée d?une tabulation
cible1 : dépendance1 dépendance2 ....
    commande_X
    commande_Y
```

Les dépendances et cibles représentent, en général, des fichiers. Seule la première régle du Makefile est examinée. Les règles suivantes sont ignorées si elles ne sont pas impliquées par la première. Si certaines dépendances d'une règle X sont elles-mêmes des règles dans le Makefile alors ces dernières seront examinées avant la règle X appelante. Pour chaque règle X examinée, si au moins une de ses dépendances est plus récente que sa cible alors les commandes de la règle X seront exécutées. Remarque : les commandes servent généralement à produire la cible (i.e un nouveau fichier). Une cible peut ne pas représenter un fichier. Dans ce cas, les commandes de cette règle seront toujours exécutées.

2 Règles de modèles

Ces règles sont plus polyvalentes car vous pouvez spécifier des règles de dépendance plus complexes. Une règle de modèle ressemble à une règle normale, sauf qu'un symbole (%) apparaît dans le nom de la cible. Les dépendances emploient également (%) pour indiquer la relation entre les noms de dépendance et le nom de la cible. La règle de modèle suivante spécifie comment tous les fichiers vst sont formés à partir des vbe.

```
#exemple de règle pour la synthèse
%.vst : %.vbe
    boog $*
```

3 Définitions de variables

On peut définir des variables en n'importe quel endroit du fichier Makefile, mais une écriture lisible nous amène à les définir en début de fichier.

```
#définitions de variables

MY_COPY = cp -r
MY_NUM = 42
MY_STRING = "hello"

Elles sont utilisables à n'importe quel endroit du Makefile. Elles doivent être précé-
dées du caractère '''$'''

#utilisation d?une variable dans une règle

copie:
    ${MY_COPY} digicode.vbe tmp/
```

4 Variables prédéfinies

- $@ Nom complet de la cible.
- $* Nom du fichier cible sans l'extension.
- $< Nom du premier fichier dépendant.
- $+ Noms de tous les fichiers dépendants avec des dépendances doubles répertoriées dans leur ordre d'apparition.
- $ Noms de tous les fichiers dépendants. Les doubles sont retirés.
- $? Noms de tous les fichiers dépendants plus récents que la cible.
- $% Nom de membre pour des cibles qui sont des archives (langage C). Si, par

exemple

La cible est libDisp.a(image.o), $% est image.o et $@ est libDisp.a.

5 travail à effectuer

Le but de ce TP est de reprendre le travail fait au TP4 de façon à l'automatiser par un Makefile.