Guide make Pour l'Impatient(e)

"make - GNU make utility to maintain groups of programs", /usr/bin/man 2004.
"Le Guide *make* Pour l'Impatient(e) est un tas de mots gérés par un tas de commandes elles-même gérées par un fichier *Makefile* écrit pour la commande *make*.", nj 2004.

Point de départ : un ensemble de fichiers, un ensemble de dépendances relatives à ces fichiers.
Objectif : utiliser un fichier de configuration appelé *Makefile* pour automatiser la gestion des dépendances et éviter de retenir des commandes à rallonge. Il est courant pour cela d'utiliser la commande *make*.

1 fichier *Makefile*

Ce fichier contient une liste de dépendances et d'actions. Les dépendances maintiennent la bonne constitution des programmes. Les actions décrivent les commandes à exécuter. Il est spécifié à la commande *make* par l'option `-f nomDuFichierMakefile`. Par défaut, make recherche un fichier nommé *Makefile*.
Dans ce fichier :

`#`	commente tout ce qui suit sur cette ligne,
`CC = gcc`	indique à *make* de remplacer toutes les expressions `$(CC)` par `gcc`,
`A.o :`	définit une référence-cible,
`A.o : A.c A.h`	définit une référence-cible avec deux dépendances.

Le fichier cible `A.o` est valide si les dates de modification des fichiers `A.c` et **`A.h`** ne sont pas postérieures à celle du fichier **`A.o`** (`all` est la cible par défaut de la commande `make`),

`<tab>$(CC) -o A.c`	est un exemple de commande.

Les commandes sont toujours précédées d'une tabulation. Pour associer des commandes à une référence, il suffit de les placer sur les lignes suivant cette référence,

`AB.o : A.o B.o`	impose aux références `A.o` et `B.o` invalides d'être reconstruites

avant l'exécution des commandes associées à `AB.o`. Une (ou plusieurs) référence(s) est(sont) évaluée(s) par la commande *make ref (ref2 ...)* (ici *make AB.o* et *make appli propre* ci-dessous).

exemple :

```
CC = gcc
CFLAGS = -g -Wall
obj = A.o B.o
all : appli

appli : $(obj)
	$(CC) $(CFLAGS) -o appli $(obj)
A.o : A.c A.h
	$(CC) $(CFLAGS) -c A.c
B.o : B.c B.h
	$(CC) $(CFLAGS) -c B.c
propre :
	rm -f *~  $(obj)
```

lien utile :

- http ://www.gnu.org/software/make/make.html

Autres options de la commande **make** :

`-p`	affiche la liste des règles par défaut,
`-n`	affiche les commandes à éxecuter sans les éxecuter,
`-j 3`	tente d'éxecuter simultanéement au plus 3 commandes indépendantes,
`-i`	ignore les codes d'erreur des commandes dans l'éxecution des commandes,
`-r`	ignore toutes les règles de dépendances par défaut,
`-d`	affiche la séquence des traitements testés par la commande **make**.

2 macro-définitions standard

`$@`	est le nom de la référence-cible en cours d'examen,
`$<`	est le nom de la première dépendance de la liste des dépendances.
`$?`	est la liste des dépendances plus récentes que la référence-cible,
`$^`	est la liste de toutes les dépendances,
`$*`	est le nom de la référence-cible en cours de traitement privée de son suffixe.

3 règles génériques et fonctions

Une règle générique est définie par :

`%.b : %.a`	tout fichier de suffixe `.b` dépend du fichier de même nom de suffixe `.a`,

La syntaxe d'une fonction est : `$(fonction paramètres)`.
Pour toutes les fonctions suivantes, la valeur de retour est affectée à la variable `RES` :

`RES=$(wildcard *.a)`	retourne la liste des fichiers de suffixe `.a`,
`RES=$(subst from,to,text)`	retourne le texte `text` en substituant toutes les occurences de `from` par `to` dans `text`,
`RES=$(findstring text,find)`	retourne le texte `find` si `find` est dans `text`,
`RES=$(strip text)`	retourne le text `text` en supprimant les espaces au début et à la fin de `text`,
`RES=$(shell ls -l)`	retourne le résultat de l'éxecution de la commande `ls -l`,
`RES=$(VAR:.a=.b)`	comme `RES=$(subst .a,.b,$(VAR))`,
`RES=$(dir src/file.c)`	retourne `src`,
`RES=$(notdir src/file.c)`	retourne `file.c`,

Il est possible de vérifier des conditions avec la syntaxe
`ifeq CONDITIONAL-DIRECTIVE <tab>TEXT-IF-TRUE else <tab>TEXT-IF-FALSE endif.`

exemples :

```
TEX = latex
TEXTOPS = dvips
FICHIERSTEX = $(wildcard *.tex)

all : $(FICHIERSTEX:.tex=.ps)

%.dvi : %.tex
	$(TEX) $<
	rm $(@:.dvi=.aux)
	rm $(@:.dvi=.log)
%.ps : %.dvi
	$(TEXTOPS) $< -o $@
```

```
OS=$(shell uname)
all : deuxtests
deuxtests :
ifeq ($(OS),Linux)
	echo OS est Linux
else
	echo OS pas Linux mais $(OS)
endif
ifeq (1.tex,$(wildcard 1.tex))
	echo 1.tex dans rep courant
else
	echo 1.tex pas dans rep courant
endif
```