Makefiles

Démonstration OS n°7

06/04/2015

Romain Chanoir

Un Makefile, c’est quoi ?

- Composé de plusieurs règles
- Puissant
- Permet d’automatiser la compilation
- Standard

Cibles, dépendances et commandes

- Structure générale d'un Makefile :

```
Cible : dependance
Commandes
```

= une règle

- Lorsque l'on utilise make, la première règle ou la règle spécifiée est évaluée
- Si une dépendance est la cible d'une autre règle, cette règle est évaluée à son tour
- Si la cible ne correspond pas à un fichier existant ou si un fichier dépendance est plus récent que la règle, les commandes sont exécutées

Exemple

- On a trois fichiers : hello.c, hello.h et main.c

```
hello: hello.o main.o
	gcc -o hello hello.o main.o

hello.o: hello.c
	gcc -o hello.o -c hello.c -W -Wall

main.o: main.c hello.h
	gcc -o main.o -c main.c -W -Wall
```

Cibles standards (conventions)

- all : première cible du fichier, permet de construire complètement le projet
- clean : permet de supprimer tout les fichiers intermédiaires
- mrproper : supprime tout ce qui peut être régénéré

Exemple

```
all: hello

hello: hello.o main.o
	gcc -o hello hello.o main.o

hello.o: hello.c
	gcc -o hello.o -c hello.c -W -Wall

main.o: main.c hello.h
	gcc -o main.o -c main.c -W -Wall

clean:
	rm -rf *.o

mrproper: clean
	rm -rf hello
```

Variables

- On peut définir des variables : **NOM=valeur**
- Pour les appeler : **$(NOM)**

- Variables courantes :
  - **CC** : compilateur utilisé
  - **CFLAGS** : options de compilation
  - **LDFLAGS** : options d'édition de liens
  - **EXEC** ou **TARGET** : regroupe les exécutables

- Liste des variables conventionnelles : http://www.tutorialspoint.com/makefile/makefile_quick_guide.htm

Variables internes

- $@ : nom de la cible
- $< : nom de la première dépendance
- $^ : liste des dépendances
- $? : liste des dépendances plus récentes que la cible
- $* : nom d’un fichier sans son suffixe

Exemple

```
CC=gcc
CFLAGS=-W -Wall
LDFLAGS=
EXEC=hello

All: $(EXEC)

Hello: hello.o main.o
	$(CC) -o $@ $^ $(LDFLAGS)

hello.o: hello.c
	$(CC) -o $@ -c $< $(CFLAGS)

main.o: main.c hello.h
	$(CC) -o $@ -c $< $(CFLAGS)

Clean:
	Rm -rf *.o

Mrproper: clean
	Rm -rf $(EXEC)
```

Règles d'inférences

- Règles génériques :
  - **%.o : %.c** ou **.o.c:**

```
CC=gcc
CFLAGS=-W -Wall
LDFLAGS=
EXEC=hello

all: $(EXEC)

hello: hello.o main.o
	$(CC) -o $@ $^ $(LDFLAGS)

main.o: hello.h

%.o: %.c
	$(CC) -o $@ -c $< $(CFLAGS)

clean:
	rm -rf *.o

mrproper: clean
	rm -rf $(EXEC)
```

Cible .PHONY

- Permet d'éviter un conflit avec un fichier du même nom.
  - Exemple : Si un fichier clean se trouve dans le répertoire du Makefile, la règle **clean** ne sera jamais exécutée. En effet, comme la cible n'a pas de dépendances, le fichier clean sera toujours considéré à jour et la règle ne sera pas exécutée
- On doit rajouter cette ligne pour éviter tout conflit :

```
.PHONY: clean mrproper
```

Conditions

```
DEBUG=yes

all: $(EXEC)
ifeq ($(DEBUG),yes)
        @echo "Génération en mode debug"
else
        @echo "Génération en mode release"
endif
```

```
DEBUG=yes

all: $(EXEC)
ifneq ($(DEBUG),yes)
        @echo "Génération en mode release"
else
        @echo "Génération en mode debug"
endif
```

Conditions

```
ifdef variable
      block if variable is non-empty
else
      block if variable is empty
endif

ifndef variable
      block if variable is empty
else
      block if variable is non-empty
endif
```

Plusieurs Makefiles

- On peut appeler un autre Makefile grâce à **$(MAKE)**
  - Si on veut lui transférer des variables, on utilise **export**

```
export CC=gcc
export CFLAGS=-W -Wall -ansi -pedantic
export LDFLAGS=
HELLO_DIR=hello
EXEC=$(HELLO_DIR)/hello

all: $(EXEC)

$(EXEC):
	@(cd $(HELLO_DIR) && $(MAKE))

.PHONY: clean mrproper $(EXEC)

clean:
	@(cd $(HELLO_DIR) && $(MAKE) $@)

mrproper: clean
	@(cd $(HELLO_DIR) && $(MAKE) $@)
```

Bonus

- Stopper l'exécution pour appeler d'autres Makefiles : **include** *filenames*
- Écrire une variable sur plusieurs lignes : \
- Rendre les commandes silencieuses : @

Sources

- http://gl.developpez.com/tutoriel/outil/makefile/
- http://www.gnu.org/software/make/manual/make.html
- http://www.tutorialspoint.com/makefile/makefile_quick_guide.htm
- https://ensiwiki.ensimag.fr/images/e/eb/Makefile.pdf