Makefile

Sylvain Jubertie

Département Informatique
IUT Orléans

Plan

Making Makefiles...

Objectifs

Faciliter le processus de compilation :

- appels de gcc
- création de bibliothèques
- options de compilation

Centraliser la gestion du projet :

- compilation
- documentation
- nettoyage

Making Makefiles...

Principe, avantages

- Simplifier le processus de compilation.
- Ne recompiler que ce qui est nécessaire : `make` se base sur les dates de création/modification des fichiers pour déterminer ce qui doit être recompilé.
  Si un fichier `.c` est plus récent que le fichier `.o` correspondant c'est que le fichier `.c` a été modifié depuis la génération du `.o` donc il faut le recompiler et refaire l'édition de lien pour regénérer l'exécutable/la bibliothèque.

Making Makefiles...

Exemple simple

```
main : main.c
    gcc -o main main.c
```

Principe

```
cible : source
    cmd pour passer de la source à la cible
```

Making Makefiles...

Exemple plus compliqué

```
main: main.o
    gcc -o main main.o
main.o: main.c
    gcc -c main.c
```

Making Makefiles...

Syntaxe

```
main: main.o
<tab>gcc -o main main.o
main.o: main.c
<tab>gcc -c main.c
```

Making Makefiles...

Autres possibilités

- définition de variables

```
SRC=file1.c
```

- nettoyage

```
clean:
    rm *.o
```

- documentation

```
doc:
    doxygen ...
```

Making Makefiles...

Putting it all together

```
BIN=main

all: ${BIN}
${BIN}: main.o autrefichier.o
    ${CC} -o main main.o autrefichier.o
main.o: main.c
    ${CC} -c main.c
autrefichier.o: autrefichier.c autrefichier.h
    ${CC} -c autrefichier.c
...
```

Making Makefiles...

Attention aux headers

Les fichiers objets `.o` doivent être regénérés si les dépendances : `.c` et `.h` sont modifiés MAIS ON NE COMPILE JAMAIS UN .H CAR LES .H SONT INCLUS PAR LE PREPROCESSEUR DANS LE .C. DONC RECOMPILER LE .C SUFFIT ! ON DONNE JUSTE L'INFORMATION DE DEPENDANCE AU MAKEFILE POUR QU'IL VERIFIE LES DATES DE MODIFICATIONS.

Making Makefiles...

Réalisation du Makefile

Le fichier Makefile peut être réalisé à partir de l'analyse des dépendances entre les sources et les cibles.
Ces dépendances peuvent être représentées sous forme d'un graphe orienté acyclique (graphe sans cycles) mais on rencontre souvent la forme d'un arbre qui est un cas particulier de graphe orienté acyclique.

Making Makefiles...

Exemple simple

Pour générer le programme main, il faut un fichier objet main.o qui lui même nécessite un fichier main.c pour être généré.

Making Makefiles...

Exemple arbre

Le programme main nécessite deux fichiers objets main.o et file1.o qui eux-mêmes nécessitent respectivement les fichiers main.c et file1.c pour être généré.

Making Makefiles...

Comment écrire un Makefile à partir du graphe des dépendances ?

1. Pour chaque noeud du graphe, mettre le noeud comme cible et ses fils comme sources.
2. Compléter ensuite en ajoutant les commandes nécessaires pour passer des sources à la cible.

Making Makefiles...

Variables internes

- nom de la cible : $@
- nom de la 1ère dépendance $ <
- liste des dépendances $^
- liste des dépendances + récentes que la cible $?
- nom de la cible sans suffixe $*

Making Makefiles...

Simplification

```
CC=gcc
CFLAGS=-O3 -march=native
EXEC=main

all: ${BIN}
${BIN}: main.o autrefichier.o
    ${CC} -o $@ $^
main.o: main.c
    ${CC} ${CFLAGS} -c $<
autrefichier.o: autrefichier.c
    ${CC} ${CFLAGS} -c $<
...
```

Making Makefiles...

Encore trop long...

Les commandes dans les cas suivants sont identiques :

```
main.o: main.c
    ${CC} -c $<
autrefichier.o: autrefichier.c
    ${CC} -c $<
```

Making Makefiles...

Factorisation

```
CC=gcc
CFLAGS=-O3 -march=native
BIN=main

all: ${BIN}
${BIN}: main.o autrefichier.o
    ${CC} -o $@ $^
%.o: %.c
    ${CC} ${CFLAGS} -c $<
```

Making Makefiles

Règles implicites

Certaines règles sont déjà incluses par défaut par la commande `make`, par exemple comment transformer un `.c` en `.o`.

Making Makefiles...

Règles implicites

```
CFLAGS=-O3 -march=core2

all: main
main: main.o autrefichier.o
```

Making Makefiles...

Autres règles

Nettoyage, documentation, etc :

```
all: main ...

clean:
    rm -f *.o *.so *.a

doc:
    doxygen ...
```

Dans ce cas, les cibles ne sont pas des fichiers et il faut désactiver la vérification des dates entre cibles et dépendances en ajoutant les cibles à la cible `.PHONY`

```
.PHONY: all clean doc
```