

Makefile, modules, préprocesseur

Ensimag 1A Apprentissage, Logiciel de Base

mai 2012

1 Makefile

Documentation : <http://www.gnu.org/software/make/manual/>

1.1 Version de base

```
CC = gcc
```

```
CFLAGS = -Werror -Wextra -Wall -pedantic -ansi -g
```

```
LD = gcc
```

```
LDFLAGS =
```

```
monprog: module1.o module2.o monprog.o
```

```
$(LD) $(LDFLAGS) -o monprog module1.o module2.o monprog.o
```

```
# ^^-- ceci est une tabulation et SURTOUT PAS DES ESPACES
```

```
module1.o: module1.c
```

```
$(CC) $(CFLAGS) -c module1.c
```

```
module2.o: module2.c
```

```
$(CC) $(CFLAGS) -c module2.c
```

```
monprog.o: monprog.c
```

```
$(CC) $(CFLAGS) -c monprog.c
```

1.2 Version moins de base

```
.PHONY: clean real-clean
```

```
CC = gcc
```

```
CFLAGS = -Werror -Wextra -Wall -pedantic -ansi -g
```

```
LD = gcc
```

```
LDFLAGS =
```

```

BIN = monprog

OBJS = module1.o module2.o monprog.o

$(BIN): $(OBJS)
    $(LD) $(LDFLAGS) -o $(BIN) $(OBJS)

module1.o: module1.c
    $(CC) $(CFLAGS) -c module1.c

module2.o: module2.c
    $(CC) $(CFLAGS) -c module2.c

monprog.o: monprog.c
    $(CC) $(CFLAGS) -c monprog.c

clean:
    $(RM) $(OBJS)

real-clean:
    $(MAKE) clean
    $(RM) $(BIN)

```

1.3 Version avec règles génériques

```

.PHONY: clean real-clean

CC = gcc
CFLAGS = -Werror -Wextra -Wall -pedantic -ansi -g

LD = gcc
LDFLAGS =

BIN = monprog

OBJS = module1.o \
      module2.o \
      monprog.o

$(BIN): $(OBJS)
    $(LD) $(LDFLAGS) -o $@ $^

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

```

```
clean:
    $(RM) $(OBJS)
```

```
real-clean:
    $(MAKE) clean
    $(RM) $(BIN)
```

1.4 Règles par défaut

```
.PHONY: clean real-clean
```

```
CC = gcc
CFLAGS = -Werror -Wextra -Wall -pedantic -ansi -g
```

```
LD = gcc
LDFLAGS =
```

```
BIN = monprog
OBJS = module1.o \
      module2.o \
      monprog.o
```

```
$(BIN): $(OBJS)
```

```
clean:
    $(RM) $(OBJS)
```

```
real-clean:
    $(MAKE) clean
    $(RM) $(BIN)
```

1.5 Manipulation de chaînes

```
.PHONY: clean real-clean
```

```
CC = gcc
CFLAGS = -Werror -Wextra -Wall -pedantic -ansi -g
```

```
LD = gcc
LDFLAGS =
```

```
BIN = monprog
BASE = $(basename $(shell ls -1 *.c))
OBJS = $(foreach f,$(BASE),$(addsuffix .o,$(f)))
```

```
$(BIN): $(OBJS)
```

```
clean:
    $(RM) $(OBJS)

real-clean:
    $(MAKE) clean
    $(RM) $(BIN)
```

2 Exemple

Pour cette section, un squelette de code se trouve dans le répertoire `c/2-c-et-makefiles/` de votre archive Git.

2.1 Mini module de statistiques

Question 1 *Implémenter un mini module de statistiques avec les spécifications suivantes :*

- Un utilisateur du module doit l’initialiser en appelant la fonction `stat_init`,
- Un utilisateur du module doit le terminer en appelant la fonction `stat_end`,
- L’utilisateur donne plusieurs valeurs, avec la fonction `stat_entrer_valeur`.
- L’utilisateur a accès à la valeur minimum entrée, la valeur maximum, et la moyenne avec les fonctions `stat_min()`, `stat_max()`, `stat_moyenne()`.

Pour vous gagner du temps, le fichier `stats.h` est fait pour vous dans le squelette. Il reste à écrire un fichier `stats.c`. On respectera les règles de codage suivantes :

- Toutes les fonctions publiques (qui apparaissent dans le fichier `.h`) sont préfixées par `stat_`;
- Le fichier `.h` ne contient que ce qui est nécessaire au monde extérieur (le reste est plus ou moins l’équivalent du « `private` » en Ada ou Java);
- Déclarer les fonctions et variables globales du `.c` en « `static` » (symbole local à une unité de compilation), sauf si on veut les exporter dans le `.h`, pour éviter les clashes de noms avec d’autres modules.
- Exporter les fonctions et variables globales publiques dans le `.h` avec le mot clé « `extern` ».
- Les commentaires du fichier `.h` doivent décrire l’interface du module, ceux du `.c` doivent décrire l’implémentation.

2.2 Utilisation des modules statistiques et listes

Question 2 *En utilisant le module de statistiques ci-dessus, et le modules de listes fourni, implémenter un module `stat_listes` qui propose la fonction*

```
stat_listes_moyenne(const liste* const l)
```

Question 3 *Implémenter un programme principal, `moyenne_liste.c` qui demande une liste d’entiers positifs au clavier, utilisant `-1` comme terminateur, qui entre les entiers dans une liste, et qui affiche la moyenne des éléments à l’écran.*

Question 4 *Représenter de manière graphique les dépendances entre les fichiers des modules `liste`, `stat`, `stat_listes` et du programme principal.*

Question 5 *Proposez un Makefile pour compiler ce petit projet.*