TP3 Ingeniérie logicielle en LaTeX

AUDIC Bryan & NAPOLEON Sebastien

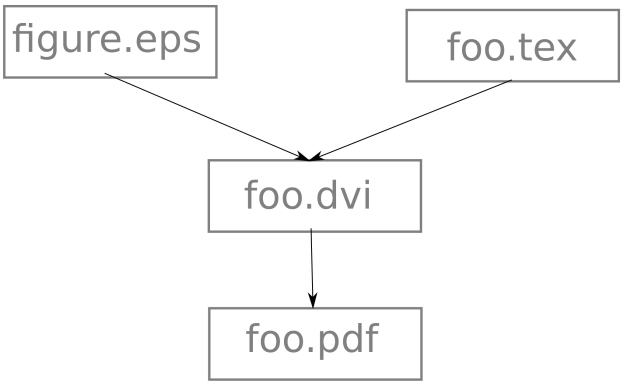
September 18, 2018

1 Insertion d'une image

Voici l'image créée sous Inkscape :

2 Partie 1.2

Question 3 : graphe des dépendances



Question 4 : Premier Makefile

```
all : foo.pdf clean

foo.pdf : foo.dvi
dvipdf foo.dvi
foo.dvi : foo.tex figure.eps dependance.eps
latex foo.tex
clean :
rm -rf foo.aux foo.log
```

Question 5 : Comportement Makefile 1 - Pas de nouvelle compilation. 2 - Après utilisation du touch, la compilation s'effectue à nouveau car l'horodatage a changé. 3 - La compilation s'effectue car le .tex est plus récent que le .dvi alors que le .pdf est plus récent que le .tex. Cela s'explique par le fait que le systeme de makefile remonte la chaine de dépendance et ne s'arrête pas seulement au début. Ainsi, le .tex est recompilé et le .dvi est devenu obsolète. Même chose pour le .dvi et le .pdf.

3 Partie 1.3

Question 6 : Règle Makefile

```
all : pdf clean

pdf : foo.dvi
dvipdf foo.dvi
foo.dvi : foo.tex figure.eps dependance.eps
latex foo.tex
dvi : foo.dvi
xdvi foo.dvi
clean :
rm -rf foo.aux foo.log
```

A partir d'ici, nous n'avons pas eu le temps de modifier le reste du code pour y ajouter le foo.dvi partout. Normalement il faudrait cette étape intermédiaire pour diffencier la compilation du .tex et l'affichage du .dvi. Le code dans le reste du fichier est fonctionnel.

Question 7 : Macro

```
#MACROS
LATEX=latex
BASENAME=foo
DVItoPDF=dvipdf
DPFLAGS=
VIEWER=xdvi
PDFVIEWER=evince

#REGLES
all : pdf clean

pdf : dvi
$(DVItoPDF) $(BASENAME).dvi
dvi : $(BASENAME).tex figure.eps dependance.eps
$(LATEX) $(BASENAME).tex
$(VIEWER) $(BASENAME).dvi
```

```
clean :
rm -rf $(BASENAME).aux $(BASENAME).log
```

Question 8 : Macros spéciales

```
#MACROS
LATEX=latex
BASENAME=foo
DVItoPDF=dvipdf
DPFLAGS=
VIEWER=xdvi
PDFVIEWER=evince

#REGLES
all : pdf clean

pdf : dvi
$(DVItoPDF) $(BASENAME).$<
dvi : $(BASENAME).tex figure.eps dependance.eps
$(LATEX) $<
$(VIEWER) $(BASENAME).$@
clean :
rm -rf $(BASENAME).aux $(BASENAME).log
```

Question 9-10 : Makefile générique

```
#MACROS
LATEX=latex
BASENAME=foo
DVItoPDF=dvipdf
DPFLAGS=
VIEWER=xdvi
PDFVIEWER=evince

#REGLES
#all : pdf clean
%.pdf : %.dvi
$(DVItoPDF) $<
%.dvi : %.tex figure.eps dependance.eps
$(LATEX) $<
$(VIEWER) $@
clean :
rm -rf $(BASENAME).aux $(BASENAME).log

help :
@echo " nomDuFichier.dvi : Genere le dvi associer a un fichier tex et l'affiche."
```

```
@echo " nomDuFichier.pdf : Genere le pdf associer a un fichier dvi."
@echo " clean : Supprime les fichiers .aux .log."
```

4 Partie 1.4

Question 11 : Compilation en C La direcive défault sert à indiquer un message en cas de cible incorrect pour la commande make. La valeur de OBJS est : hello-world.o car on substitue le .c par .o. La règle de construction de l'application, après substitution, est : gcc hello-world.o -o hello-world

Question 12 : Suffixes en C Il y a une erreur car que le .o n'existe pas. Si on met .SUFFIXES en commentaire, les suffixes ne sont pas supprimés par défaut et donc conservés. Donc la compilation fonctionne.

Question 13 : Création d'un fichier compressé

```
APPL = hello-world

SHELL = /bin/sh

# paramètres de compilation et d'édition de liens
CC = gcc
LD = $(CC)
CFLAGS =
LDFLAGS =
CPPFLAGS =
LOADLIBS =
LDLIBS =
TAROPT=zcvf

# objets de l'application
SRCS = hello-world.c
OBJS = $(subst .c,.o,$(SRCS))
EXEC = $(APPL)

# directive d'inhibition des suppressions
.PRECIOUS: $(EXEC)

# directive d'action par défaut, en cas d'erreur
.DEFAULT: ; @echo "$< n'est pas une cible valide"

# directive d'application des règles implicites
.SUFFIXES:             # supprime les suffixes par défaut
#.SUFFIXES: .c .o      # définit une liste de suffixes
```

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



# règle de construction de l'application
$(EXEC): $(OBJS)
$(LD) $(LDFLAGS) $^ -o $@ $(LOADLIBS) $(LDLIBS)

$(APPL).tar.gz: $(OBJS)
tar $(TAROPT) $@ $(APPL)

clean:
-@rm -f $(OBJS) $(EXEC) core a.out 2>/dev/null
```