Lang Ludovic
Moissonnier Shane
Noel-lardin Thomas
Martinez Anthony

Compte-rendu séance 3 :

Rappel séance 2 :

Depuis le début du projet nous avons commencé à enlever des bugs au programme et nous avons rendu le code modulaire.

Voici une liste des bugs que nous avons corrigé :

- warning : implicit declaration of function ‘setCharnum’
- warning : conflicting types for ‘setCharnum ‘
- In function ‘compareWord’:
  warning : unused variable ‘j’
  int i, j = 0;
- In function ‘getSizeMaillon’:
  dico.c:175:1: warning: control reaches end of non-void function
- On a aussi obtenu une erreur de segmentation lors de la première exécution que nous avons réglé en contrôlant l’ouverture des fichiers.

Par la suite nous avons changé la fonction main pour qu’elle puisse prendre en argument le fichier que l’on souhaite trier dans le dictionnaire :

```
int main(int argc, char* argv[]) {
    if (argc < 2){ //On contrôle qu’on ai le bon nombre d’argument
        printf("Vous avez oublié le nom du fichier d'entrée : \n\t./%s <NOM_FICHIER>\n", argv[0]);
        exit(0);
    }
    ...

}
```

Et pour finir la séance 2 nous avons organisé notre projet. Voici la modularité que nous avons choisit :

main.c : contenant juste la fonction main

Et voici les différents fichier avec ce qu’ils contiennent :

- charFunc.c : avec toutes les fonctions sur les caractères
- conversion.c : avec toutes les fonctions de conversion
- dataStruct.h : un fichier contenant les différentes structures
- dico.c : avec les fonctions sur le dictionnaire
- globals.h : contenant toutes les variables globales
- macro.h : contenant tous les macros
- maillon.c : contient les fonctions sur les maillons
- wordFunc.c : contient les fonctions sur les mots

Le but de la séance 3 était de créer des dossiers pour avoir une meilleure organisation dans notre projet, et aussi de créer une bibliothèque contenant les fonctions que nous allons réutiliser.

Voici l'arbre contenant la structure du projet :

```
.
├── compte-rendu.odt
├── dictionnaires.txt        //Fichier de sortie contenant le dictionnaire créé
├── donnees                  //Contient tous les fichiers tests
│   ├── diction.txt
│   ├── hugo1.txt
│   ├── hugo.txt
│   ├── orwell.txt
│   └── wells.txt
├── main                     //Exécutable créé par le Makefile
├── Makefile                 //Makefile (code donné plus bas)
├── my_header                //Dossier contenant tous les fichiers .h
│   ├── charFunc.h
│   ├── conversion.h
│   ├── dataStruct.h
│   ├── dico.h
│   ├── globals.h
│   ├── macro.h
│   ├── maillon.h
│   └── wordFunc.h
├── my_lib                   //Dossier contenant les bibliothèques statique et dynamique
│   ├── libDynamique.so
│   ├── lib_obj              //Contient tous les fichiers objet compilé avec l'option -fPIC
│   │   ├── charFunc.o
│   │   ├── conversion.o
│   │   ├── dico.o
│   │   ├── maillon.o
│   │   └── wordFunc.o
│   └── libStatique.a
├── my_obj                   //Contient tous les fichiers objets
│   ├── charFunc.o
│   ├── conversion.o
│   ├── dico.o
│   ├── maillon.o
│   ├── main.o
│   └── wordFunc.o
├── my_src                   //Contient tous les codes sources
│   ├── charFunc.c
│   ├── conversion.c
│   ├── dico.c
│   ├── maillon.c
│   ├── main.c
│   └── wordFunc.c
├── README
├── test.txt
└── toto.txt
```

Voici le code de notre Makefile :

```
#####################################################
#	Compiler bibliothèque statique -> make LIB=s
#	Compiler bibliothèque dynamique -> make LIB=d
#	Par défaut, compile avec fichier locaux
#####################################################

INCLUDEDIR_1=my_header
SRCDIR=my_src
OBJDIR=my_obj
LIBDIR=my_lib

CC=gcc
SRCS=$(wildcard my_src/*.c)
OBJS=$(SRCS:$(SRCDIR)/%.c=$(OBJDIR)/%.o)
CFLAGS=-Wall -I $(INCLUDEDIR_1) -g

# Objets pour la bibliotèque statique
OBJ_LIB=$(patsubst my_obj/main.o,,$(OBJS))

# Objets pour la bibliotèque dynamique
SRCS2=$(patsubst my_src/main.c,,$(SRCS))
OBJDIR2=$(LIBDIR)/lib_obj
OBJ_LIB2=$(SRCS2:$(SRCDIR)/%.c=$(OBJDIR2)/%.o)

# Choisi l'option de linkage pour les bibliothèques
ifeq ($(LIB),s)
LDFLAGS= -L $(LIBDIR)/ -static -lStatique
endif
ifeq ($(LIB),d)
LDFLAGS= -L $(LIBDIR)/ -Wl,-Bdynamic -lDynamique
endif

all: bibli2 bibli main

# Règle générique
# Créer les fichiers objs à partir des sources (my_src) dans le répertoire my_obj
$(OBJDIR)/%.o:$(SRCDIR)/%.c
	$(CC) $(CFLAGS) -c $< -o $@

# Compile le programme main avec les fichiers objs (my_obj)
main:$(OBJS)
	$(CC) $(OBJDIR)/* -o $@ $(LDFLAGS)

# Créer la bibliothèque statique libStatique.a dans le répertoire my_lib
bibli:$(OBJ_LIB)
	ar r $(LIBDIR)/libStatique.a $(OBJ_LIB)

# Règle générique
# Créer les fichiers objs à partir des sources (my_src) dans le répertoire my_lib/lib_obj
$(OBJDIR2)/%.o: $(SRCDIR)/%.c
```

```
	$(CC) $(CFLAGS) -fPIC -c $< -o $@

# Crée la bibliothèque dynamique libDynamique.so dans le répertoire my_lib
# récupère les fichiers objs dans le répertoire my_lib/lib_obj (OBJ_LIB2)
bibli2:$(OBJ_LIB2)
	$(CC) -shared -o $(LIBDIR)/libDynamique.so $(OBJ_LIB2)

clean:
	-rm $(OBJDIR)/*.o $(LIBDIR)/*.* $(OBJDIR2)/*
```

Actuellement notre projet compile, nous obtenons juste une erreur de segmentation lors de l'exécution du programme, lorsque le fichier d'entrée ne contient que 1 ligne.