SDA1 Travaux Pratiques n°4 Année scolaire 2014-2015

Remarque : Ce TP se déroulera en 5 parties. La séance est non notée. Par contre vous ferez au fur et à mesure un compte-rendu de votre travaille en répondant aux questions qui vous sont posés. Vous déposerez ce compte-rendu dans la zone de dépôt du serveur pédagogique nti.ec-lille.fr.

Objectifs

- Manipulation des chaines de caractères
- Manipulation des fonctions
- Utilisation de pointeurs

Compétences attendues

- Utilisation de l'aide du système d'exploitation pour savoir comment utiliser une fonction
- Compilation séparée et modularité
- Faire un compte-rendu

Rappels sur la compilation séparée

Les grands projets informatiques font appel à des équipes de développement. Cela amène naturellement à découper le développement en différents modules qui sont confiés à chaque membre de l'équipe. Cela amène à considérer 3 catégories de fichiers dans une application :

- le fichier de déclaration de la structure de données manipulées par le module que nous nommerons de manière générique <modulei>.h,
- le fichier de définition des traitements du module que nous nommerons <modulei>.c
- le fichier relatif au programme prinicpal que nous désignerons <principal>.c.

Après l'édition de ces fichiers nous devons compiler <modulei>.c par la commande

gcc -c <modulei>.c

Le résultat est le fichier <modulei>.o

Remarque : Bien faire attention à l'option de compilation -c qui limite l'exécution de gcc à mettre en œuvre les étapes de pré-processing et de compilation. Le fichier « .o » n'est généré que s'il n'y a pas d'erreurs

On peut ensuit construire l'application exécutable en utilisant les fichiers objets obtenus à partir de chaque module. On execute alors la commande :

gcc <modulei>.o … <modulek>.o <principal>.c -o <principal>.exe

On notera l'utilisation de l'option « -o » qui demande à gcc d'éxécuter les 3 étapes de son fonctionnement : pré-processing, compilation et édition de liens.

Remarque : On peut compiler le fichier <principal>.c avec la commande « gcc -c <principal>.c » avant l'option de l'ensemble des modules. Cela permet de corriger les erreurs de compilation. Mais ensuite, pour l'obtention du programme principal, il faut repartir du fichier source <principal>.c et non pas du fichier objet.

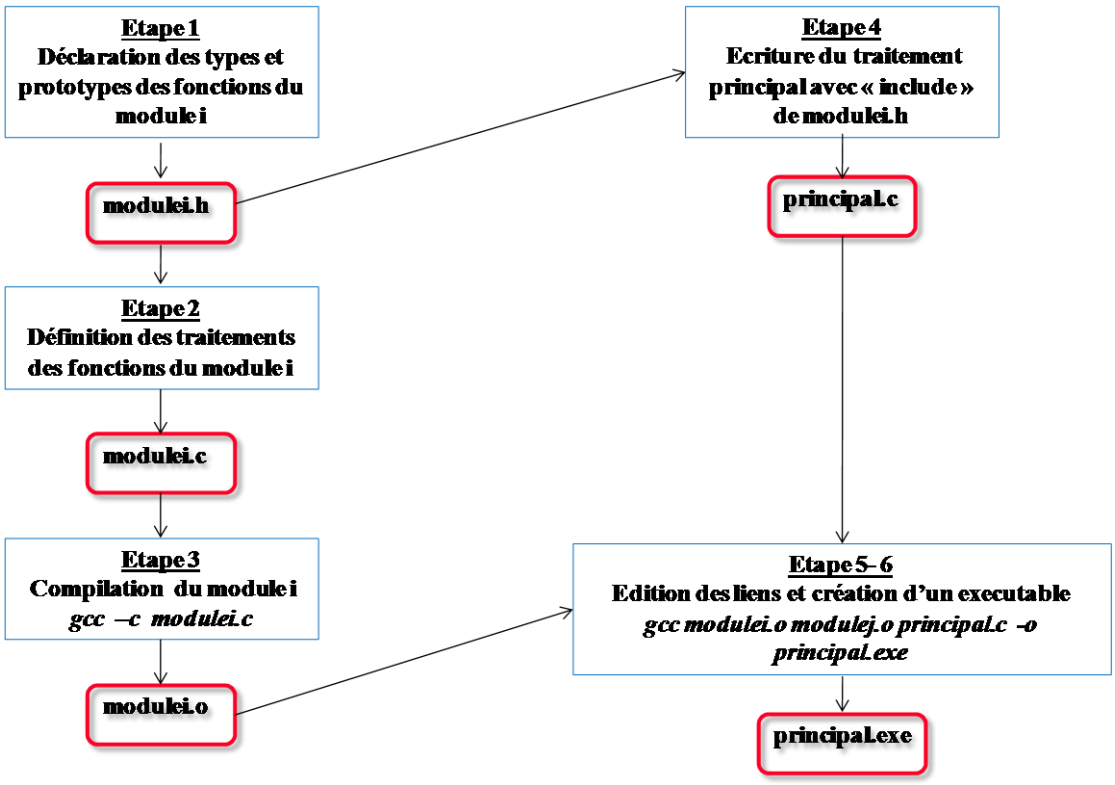

Figure 1. Principe de la compilation séparée

```
#ifndef chaine_h
#define chaine_h
#define  MAX      80
typedef char TCHAINE[MAX];

//Prototypes des fonctions
int StringLenght(const char *);
char *StringCopy(char*, const char*);
#endif
```

Figure 2. Mise en oeuvre de la compilation conditionnelle utilisée en compilation séparée

Partie I – Utilisation du man pour spécifier les paramètres des fonctions et leur description (1h)

1. Pour comprendre le fonctionnement de strcpy, faire « man strcpy ».

Déterminer le nom de la bibliothèque à inclure dans une application pour utiliser cette fonction.

Vérifier les paramètres définis et comparer ces paramètres avec ceux de la fonction StringCopy définie dans le squelette de l'application donnée par votre enseignant.

2. Qu'elle est la fonction de bcopy ? Comparer son fonctionnement à celui de strcpy

3. Utiliser l'aide en ligne (man) pour comprendre le fonctionnelent de gcc. Quels est le rôle du paramêtre de compilation « -E » ? Quels sont les étapes de gcc exécutées avec un fichier ayant l'extension « .i » ?

4. Sur l'exemple de « strcpy » et « Strincopy », compléter la déclaration des spécifications des traintements de chaine de caractères suivants (On précisera les signatures de chaque traitement et le cartouche associé sera complété)

StringLenght
StringNCopy
StringConcat
StringNConcat
StringCompare

Partie II – Développement de fonctions de traitement de chaines de caractères : approche tableau (1h)

Dans cette partie, vous utiliserez le squelette de l'application donnée par l'enseignant.

1. Vous ferez une copie de ce programme en un fichier appelé tp4_chaines_partie2.c.

2. Vous complèterez la définition des traitements, selon l'exemple de la fonction StringCopy.

3. Vous complétez le programme principal selon l'exemple de l'appel de ≪ StringCopy ≫. Compilez le programme au fur et à mesure (après chaque traitement) pour corriger les erreurs de syntaxe puis tester le traitement.

Parti III - Compilation séparée (45 mn)

1. Utiliser tp4_chaines_partie2.c ou le nouveau fichier donné par votre enseignant

(tp4_chaines_partie3.c). Séparer ce fichier en 3 fichiers : chaines.h, chaines.c et mainchaine.c.

2. Utiliser la compilation conditionnelle pour inclure chaines.h dans chaines.c et mainchaine.c.

3. Quelle est l'intéret de cette compilation conditionnelle ?

4. Utiliser l'option de compilation -E pour compiler chaines.c . Editer le fichier résultat que vous aurez appelé chaines.i. Que constatez-vous dans ce fichier ?

5. Faites la compilation séparée ? Pouvez-vous pour cela utiliser chaines.i ?

Partie IV - Conception d'une nouvelle version de chaines.h basées sur les pointeurs (1h)

1. Copier chaines.c en un fichier appelé chaines_v2.c

2. Editer ce fichier et modifier la définition de StringCopy en utilisant que la notation pointeur, selon l'exemple :

3. Sauvegarder les modifications, et refaire la compilation séparée.

4. Itérer le processus précédent (question 3) pour chaque fonction de traitement des chaines de caractères ?

Partie V - Rédaction d'un compte-rendu (15mn)

Finaliser votre compte-rendu.

Que pensez-vous de la compilation séparée ?