SDA1 Travaux Pratiques n°5 Année scolaire 2017-2018

Traitement de chaînes de caractères

Remarque : Ce TP se déroulera en 4 parties. Les 4 parties sont notées

Objectifs

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

Compétences attendues

- Compilation séparée et modularité
- Faire des tests
- Faire un compte-rendu

Rappels sur la compilation séparée

Les grands projets informatique 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 oeuvre 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 contruire 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.

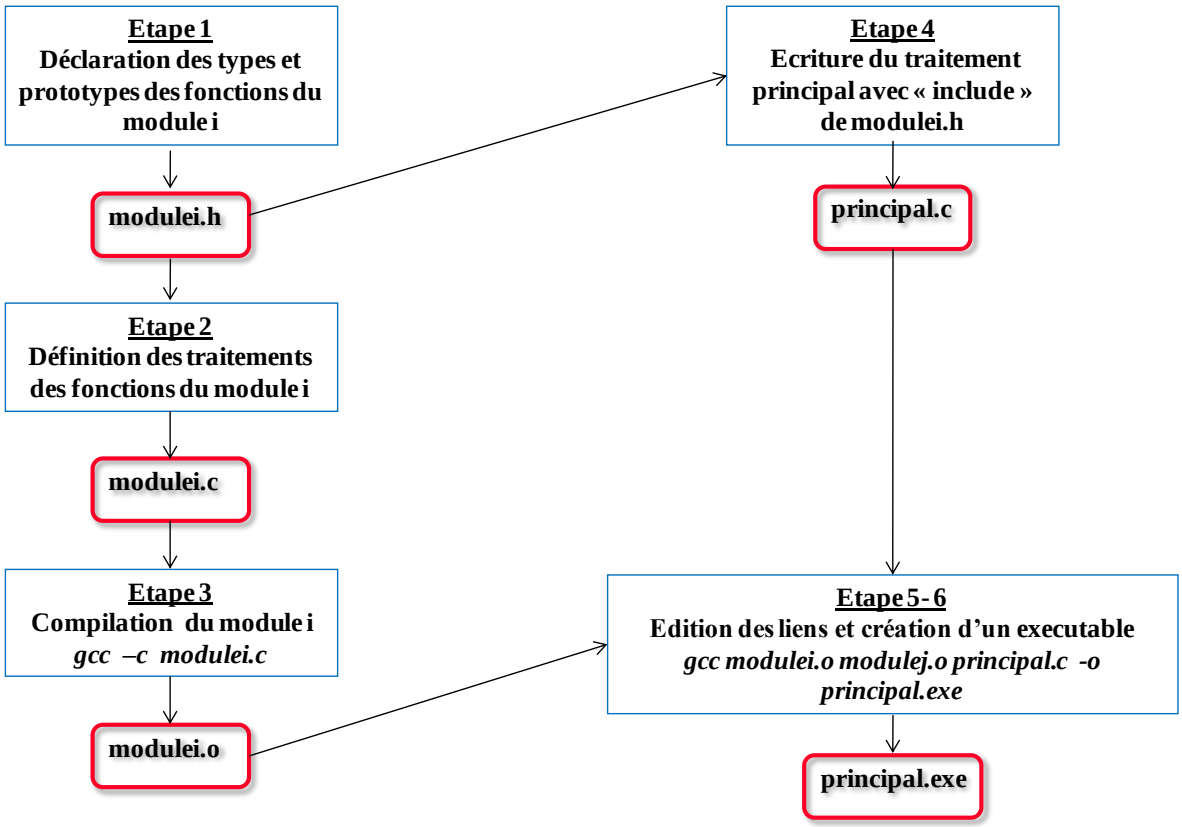

Figure 1. Principe de la compilation séparée

Enonce du sujet du TP 5

On considère des chaines d'au plus 80 caractères. Ces chaines peuvent être :

- vides,
- composées d'un seul mot,
- composées de plusieurs mots, séparés chacun par un seul espace.

De plus ces chaines auront la particularité de ne jamais commencer ou finir par un espace.

Exemple :

- L1= "" chaine vide
- L2= "exemple" 1 seul mot
- L3="titi et gros minet" plusieurs mots

Partie I : Bibliothèque de gestion de chaines de caractères

Ecrire une bibliothèque contenant les fonctions suivantes :

char *lire (char * ch) ;
La fonction lire réalise la saisie d'une chaine et la formate de manière à ce qu'elle respecte les spécifications imposées au début de cet énoncé.

Remarque : les chaînes brutes saisies peuvent être de la forme ‘’çè1234 323**Ceci** 4512 **est** à’-è21**un exemple** &é »’(-è34’’. Doivent être transformée en ‘’Ceci est un exemple’’.

int strpos(const char * m, const char *ch) ;
La fonction strpos retourne la position du mot pointé par m dans la chaîne pointée par ch sinon -1.

Remarque : ch peut valoir ‘’Ceci est un mauvais **etes** et ci un bon **test** choisi’’ et mot vaut ‘’**test’**. Le mot ‘’etes’’ contient les premières lettres de ‘’test’’. Mais je on va tester également avec ch ‘’Voici un des tes tests !!!’’. Le mot ‘’tes’’ contient les premières lettre de test.

char * strmin(char * ch) ;
Elle convertit en minuscule les caractères de la chaine pointée par ch. Retourne un pointeur sur le début de cette chaine.

Partie II

En utilisant la bibliothèque « string.h », réaliser les fonctions ou macro-fonctions suivantes :

1. Macro-fonctions

VIDE(ch)
Elle permet de tester si une chaine ch est vide.

EGAL(ch1, ch2)
Elle permet de comparer si les chaines ch1 et ch2 sont identiques.

2. Fonctions

char *premier(const char * ch, char * m) ;
Elle retourne un pointeur sur la chaîne **m** contenant le premier mot de la chaîne pointée par ch.

char *saufpremier(const char * ch, char * restch) ;
Elle retourne un pointeur sur une chaîne restch contenant tout sauf le premier mot de la chaîne pointée par ch.

char *phrase (char * ch1, char * ch2) ;
Elle retourne un pointeur sur une nouvelle chaîne construite en prenant tous les mots de la chaine pointée par ch1, suivis d'un espace et de tous les mots de la chaîne pointée par ch2 dans la limite des 80 caractères.

Exemple :
phrase("titi et gros ", "minet") → "titi et gros minet"
phrase("titi et gros minet", " ") → "titi et gros minet"

Partie III

En utilisant désormais que les fonctions définies en première partie, définir les fonctions suivantes :

char * dernier(const char * ch, char * m) ;
Elle retourne un pointeur sur une chaîne m contenant le dernier mot de la chaîne pointée par ch.

char * saufdernier(const char * ch, char * debch) ;
Elle retourne un pointeur sur une chaîne debch contenant tout sauf le dernier mot de la chaîne pointée par ch.

char * miroir(const char * ch, char * mirch) ;
Elle retourne un pointeur sur une chaîne mirch contenant tous les mots de la chaîne pointée par ch, pris dans l'ordre inverse.
Exemple : miroir("ceci est un exemple") → "exemple un est ceci"

int member(const char *m, const char *ch) ;
Elle retourne VRAI si le mot pointé par m est contenu dans la chaîne pointée par ch.

char * efface(const char * m, char * ch) ;
Elle retourne un pointeur sur une chaîne contenant tous les mots de la chaîne pointée par ch, sauf ceux égaux au mot pointé par m.

Partie IV : Implémentation d'un « interpréteur » de commande

Afin de tester les différences fonctions demandées dans ce TP, vous réaliserez un mini-interpréteur de commandes de la manière suivante :
Soit la liste de commandes : (PRE, SFP, PHR, DER, MIR, MIR, MMB, EFF, HLP, FIN).

La commande PRE affichera le 1er mot d'une chaine, etc..., HLP permettra d'afficher la liste des commandes disponibles, et la commande FIN permettra de quitter le programme.

```
/* Interpreteur de commande */
#define     MAX     81
#define     TRUE    1
#define     FALSE   0

main()
{
```

```
char cde[5], s1[MAX], s2[MAX] ;
                    /*0     4     8     12  16  20    24   28   32   36  */
char *liste_cdes= "pre-sfp-phr-der-sfd-mir-mmb-eff-hlp-fin" ;
int pos, termine=FALSE ;

imprimer_le_menu() ; / / Affiche la liste des commandes (a realiser)

printf("Tapez votre commande : ") ;
lire(cde) ;
pos=strpos(strmin(cde),liste_cdes) ;
switch (pos)
{
case 0 :
        printf("Tapez une premiere chaine de caracteres (s1) : ") ;
        lire(s1) ;
        printf("1er mot= %s \n », premier(s1) ;
        break ;
case 4 :
        …
        break ;
case 8 :
        …
        break ;
…
case 32 : imprimer_le_menu() ;
        break ;
case 36 : termine=TRUE ;
        break ;
default : printf("\007\007 Cde inconnue. Recommencez !!!\n") ;
        }
  }
while (!termine) ;
}
```

Compléter le programme principal suivant pour mettre en œuvre les différentes commandes du mini-interpréteur.