INF203 : Compte-rendu TP6
C : Chaines de caracteres, structures

Alexandre Dupré, Maxime Jaunatre, Clément Raspail | INF - 3
Mail | 16 mars 2021

Syntaxe

Pour ce compte rendu la syntaxe des commandes sera la suivante :

```
[~chemin]: commande
retour de la commande
```

Example :

```
[~/INF203]: ls
sauve_TP1 TP1 TP2
```

Si la commande est interactive et demande d'appuyer sur entrée, une caractère '->' est indiqué.
Les fichiers sont en *italique* et les commandes (ou détails de retour de commande) en **gras**. Un script sera donc en gras quand il sera appelé comme une commande.
Les fichiers sources en C sont compilés avec clang, et un nom est donné avec l'option **clang -o**. Cela implique que les programmes seront appelés par un autre nom que **a.out**.

Pour s'échauffer : quelques gammes avec des chaînes de caractères

[a]

```
'p' 'l' 'o' 'm' 't' 'r' 'a' 'l' 'a' 'l' 'a' '\0'
```

[b]

```
[~/INF203/TP6]: clang −o chaines chaines.c ; ./chaines
un mot (pas trop long !) Ã  mesurer ?
cthulhu
longueur de la chaine cthulhu :7
On a bien copié ici le mot : cthulhu
Les deux chaines cthulhu et cthulhu sont identiques
On a bien concaténé ici le mot avec lui mÃªme : cthulhucthulhu
Les deux chaines cthulhucthulhu et cthulhu sont différentes
```

```
0  [~/INF203/TP6]: cat chaines.c
1  #include <stdio.h>
2  #include <string.h>
3  #include "generer_entier.c"
4  #include "bouts_de_phrases.c"
5
6  /* longueur de la chaine passee en parametre */
7  unsigned long mon_strlen(char *ch) {
8          int i;
9          for (i=0 ; ch[i] != '\0' ; i++)
10                 ;
11         return i ;
```

```
12 }
13
14 void mon_strcopy(char *destination, char *source){
15         int i;
16         for(i=0;source[i]!= '\0'; i++){
17                 destination[i] = source[i];
18         }
19         destination[i+1] = '\0';
20 }
21
22 /* ConcatÃ¨ne source Ã  la fin de destination */
23 void mon_strcat(char *destination, char *source){
24         int i;
25         int j = 0;
26         for(i = mon_strlen(destination) ;source[j]!= '\0'; i++){
27                 destination[i] = source[j];
28                 j++;
29         }
30         destination[i+1] = '\0';
31 }
32
33 int mon_strcmp(char *chaine1, char *chaine2){
34         int i;
35         int rep = 0;
36         for(i = 0; chaine1[i]!= '\0' || chaine2[i]!= '\0' ;i++){
37                 if (chaine1[i] != chaine2[i]){
38                         rep=1;
39                 }
40         }
41         return rep;
42 }
43
44 int main() {
45         char chaine[50] ;
46         unsigned long mon_resultat ;
47
48         printf("un mot (pas trop long !) Ã  mesurer ?\n") ;
49         scanf("%49s", chaine) ;
50         mon_resultat=mon_strlen(chaine) ;
51         if (mon_resultat == strlen(chaine) )
52                 printf("longueur de la chaine %s :%lu\n", chaine, mon_resultat) ;
53         else
54                 printf("non, la longueur de '%s' n'est pas %lu\n", chaine, mon_resultat) ;
55
56
57         char temp[mon_resultat *2 + 1];
58         mon_strcopy(temp, chaine);
59         printf("On a bien copié ici le mot : %s \n", temp);
60
61
62         int retour;
63         retour=mon_strcmp(temp,chaine);
64         if (retour == 0)
65                 printf("Les deux chaines %s et %s sont identiques\n", temp, chaine);
66         else
```

```
67                      printf("Les deux chaines %s et %s sont différentes\n", temp, chaine);
68
69
70              mon_strcat(temp, chaine);
71              printf("On a bien concaténé ici le mot avec lui mÃªme : %s \n", temp);
72
73
74              retour=mon_strcmp(temp,chaine);
75              if (retour == 0)
76                      printf("Les deux chaines %s et %s sont identiques\n", temp, chaine);
77              else
78                      printf("Les deux chaines %s et %s sont différentes\n", temp, chaine);
79              return 0;
```

Exercice complémentaire

```
33 int mon_strcmp(char *chaine1, char *chaine2){
34              int i;
35              int rep = 0;
36              for(i = 0; chaine1[i]!= '\0' || chaine2[i]!= '\0' ;i++){
37                      if (chaine1[i] != chaine2[i]){
38                              if (chaine1[i] < chaine2[i])
39                                      rep = -1;
40                              else
41                                      rep = 1;
42                              return rep;
43                      }
44              }
45              return rep;
```

Fabriquer des phrases

[c]

```
0  [~/INF203/TP6]: cat phrases.c
1  #include <stdio.h>
2  #include <string.h>
3
4  int main() {
5               char Sujet[50]="la petite souris" ;
6               char Verbe[50]="mange" ;
7               char Compl[50]="le gros chat" ;
8               char Phrase[150] ;
9
10              strcat(Phrase,Sujet);
11              strcat(Phrase, " ");
12              strcat(Phrase,Verbe);
13              strcat(Phrase, " ");
14              strcat(Phrase,Compl);
15              printf("%s\n", Phrase);
16              return 0;
17 }
```

```
[~/INF203/TP6]: clang -o phrases phrases.c ; ./phrases
la petite souris mange le gros chat
```

[d] On compte les espaces pour compter les mots. Il ne faut pas oublier d'initier à 1 pour le premier mot de la phrase.

```
[~/INF203/TP6]: cat phrases.c
#include <stdio.h>
#include <string.h>
#include "bouts_de_phrases.c"
#include "generer_entier.c"


int nb_mots(char *Phrase){
        int i,j = 1;
        for (i=0; Phrase[i] != '\0'; i++){
                if(Phrase[i] == ' ')
                        j++;
        }
        return j;
}


int main() {
        char Sujet[50]="la petite souris" ;
        char Verbe[50]="mange" ;
        char Compl[50]="le gros chat" ;
        char Phrase[150] ;
        int res,i;

        strcat(Phrase,Sujet);
        strcat(Phrase, " ");
        strcat(Phrase,Verbe);
        strcat(Phrase, " ");
        strcat(Phrase,Compl);
        // printf("%s\n", Phrase);

        for(i = 0; i < 3; i++){
                Phrase[0] = '\0'; // reinitialise phrase
                strcat(Phrase,sujet[generer_entier(11)]);
                strcat(Phrase, " ");
                strcat(Phrase,verbe[generer_entier(11)]);
                strcat(Phrase, " ");
                strcat(Phrase,complement[generer_entier(11)]);
                printf("%s\n", Phrase);
                res = nb_mots(Phrase);
                printf("La phrase : %s contient %d mots\n", Phrase, res);
        }
        return 0;
}
```

```
[~/INF203/TP6]: clang -o phrases phrases.c ; ./phrases
Guillaume H. détruit son steack-frites
La phrase : Guillaume H. détruit son steack-frites contient 5 mots
un inconnu avale bruyamment la fumee qui sort du clavier
La phrase : un inconnu avale bruyamment la fumee qui sort du clavier contient 10 mots
Oscar trouve enfin la fumee qui sort du clavier
La phrase : Oscar trouve enfin la fumee qui sort du clavier contient 9 mots
```

[e] Cela n'aurait pas marché car on ne pointe pas joueur directement, il faut donc préciser (joueur* j) pour que cela fonctionne correctement.
[f]

```
0  [~/INF203/TP6]: cat billes.c
1  #include <stdio.h>
2  #include <string.h>
3
4  typedef struct {
5          char pseudo[20];
6          int nb_billes;
7  } joueur;
8
9  joueur atchoum = { "Atchoum", 42 };
10 joueur dormeur = { "Dormeur", 25 };
11 joueur grincheux = { "Grincheux", 3 };
12 joueur joyeux = { "Joyeux", 100 };
13 joueur prof = { "Prof", 2 };
14 joueur simplet = { "Simplet", 0 };
15 joueur timide = { "Timide", 12 };
16
17 void afficher_joueur_V1(joueur j) {
18         printf("%s a %d billes\n", j.pseudo, j.nb_billes);
19 }
20
21 void afficher_joueur_V2(joueur* pj) {
22         printf("%s a %d billes\n", pj->pseudo, pj->nb_billes);
23 }
24
25 void lire_joueur(joueur* pj){
26         char t_pseudo[20] ;
27         int nbille ;
28         scanf("%19s", t_pseudo) ;
29         scanf("%d", &nbille) ;
30         strcpy(pj->pseudo,t_pseudo);
31         pj->nb_billes = nbille;
32 }
33
34 void transferer_V1(joueur j1, joueur j2, int nb){
35         j1.nb_billes = j1.nb_billes - nb;
36         j2.nb_billes = j2.nb_billes + nb;
37 }
38
39 void transferer_V2(joueur* j1, joueur* j2, int nb){
40         j1->nb_billes = j1->nb_billes - nb;
41         j2->nb_billes = j2->nb_billes + nb;
42 }
43
44
45 int main() {
46         afficher_joueur_V1(atchoum);
47         afficher_joueur_V2(&simplet);
48         lire_joueur(&simplet);
49         afficher_joueur_V2(&simplet);
50         printf("\nOn test transferer_V1 avec 10 billes\n");
```

```
51          transferer_V1(atchoum,simplet,10);
52          afficher_joueur_V2(&atchoum);
53          afficher_joueur_V2(&simplet);
54          printf("\nOn test transferer_V2 avec 10 billes\n");
55          transferer_V2(&atchoum,&simplet,10);
56          afficher_joueur_V2(&atchoum);
57          afficher_joueur_V2(&simplet);
58          return 0;
59  }
```

```
[~/INF203/TP6]: clang -o billes billes.c ; ./billes
Atchoum a 42 billes
Simplet a 0 billes
Roger
1
Roger a 1 billes

On test transferer_V1 avec 10 billes
Atchoum a 42 billes
Roger a 1 billes

On test transferer_V2 avec 10 billes
Atchoum a 32 billes
Roger a 11 billes
```

Commande de la semaine : tr

[g] *On a pris la liberté de modifier légèrement exemple_tr...*

```
[~/INF203/TP6]: tr aeiou vwxyz < exemples_tr
Ww'rw ny strvngwrs ty lyvw
Yyz knyw thw rzlws vnd sy dy I
A fzll cymmxtmwnt's whvt I'm thxnkxng yf
Yyz wyzldn't gwt thxs frym vny ythwr gzy

I jzst wvnnv twll yyz hyw I'm fwwlxng
Gyttv mvkw yyz zndwrstvnd

Nwvwr gynnv gxvw yyz zp
Nwvwr gynnv lwt yyz dywn
Nwvwr gynnv rzn vryznd vnd dwswrt yyz
Nwvwr gynnv mvkw yyz cry
Nwvwr gynnv svy gyydbyw
Nwvwr gynnv twll v lxw vnd hzrt yyz
```

La commande utilisée est :

```
[~/INF203/TP6]: cat exemples_tr | tr −s '\n' > fich2 ; cat fich2
We're no strangers to love
You know the rules and so do I
A full commitment's what I'm thinking of
You wouldn't get this from any other guy
I just wanna tell you how I'm feeling
Gotta make you understand
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
Never gonna say goodbye
Never gonna tell a lie and hurt you
```