INF203 : Compte-rendu TP5
C : types, tests, boucles, parcours de tableaux

Alexandre Dupré, Maxime Jaunatre, Clément Raspail | INF - 3
Mail | 2 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**.

1 Testons des entiers

[a]

```
[~/INF203/TP5]: clang -o exemple_generation exemple_generation.c
[~/INF203/TP5]: ./exemple_generation
J'ai genere l'entier 49
[~/INF203/TP5]: ./exemple_generation
J'ai genere l'entier 9
[~/INF203/TP5]: ./exemple_generation
J'ai genere l'entier 67
```

```
[~/INF203/TP5]: cat exemple_generation.c
#include <stdio.h>
#include "generer_entier.c"

int main() {
        long x = generer_entier(100);
        printf("J'ai genere l'entier %ld\n", x);
        if (x<42)
                printf("Trop petit\n");
        else if (x==42)
                printf("Youpie\n");
        else
                printf("Trop grand\n");
        return 0;
}
```

```
[~/INF203/TP5]: clang -o exemple_generation exemple_generation.c
[~/INF203/TP5]: ./exemple_generation
J'ai genere l'entier 48
Trop grand
[~/INF203/TP5]: ./exemple_generation
J'ai genere l'entier 6
Trop petit
[~/INF203/TP5]: ./exemple_generation
J'ai genere l'entier 92
Trop grand
```

```
0  [~/INF203/TP5]: cat exemple_generation.c
1  #include <stdio.h>
2  #include "generer_entier.c"
3
4  int main() {
5          int n = 5;
6          int i;
7          long x;
8
9          for (i = 0; i < n; i++) {
10         x = generer_entier(100);
11         printf("J'ai genere l'entier %ld\n", x);
12         if (x<42)
13                 printf("Trop petit\n");
14         else if (x==42)
15                 printf("Youpie\n");
16         else
17                 printf("Trop grand\n");
18         }
19         return 0;
20 }
```

```
[~/INF203/TP5]: clang -o exemple_generation exemple_generation.c
[~/INF203/TP5]: ./exemple_generation
J'ai genere l'entier 31
Trop petit
J'ai genere l'entier 33
Trop petit
J'ai genere l'entier 87
Trop grand
J'ai genere l'entier 57
Trop grand
J'ai genere l'entier 36
Trop petit
```

2 Provoquons un débordement

[b] La taille d'une variable **unsigned char** en bits est de [0 ;255] et fait 1 octet. On a une boucle while que l'on pensais infinie avant de tester, mais le débordement du type **unsigned char** permet de l'arreter.

```
[~/INF203/TP5]: clang −o deborde_char deborde_char.c
[~/INF203/TP5]: ./deborde_char
taille du type unsigned char : 1 octet(s)
255 + 1 = 0 donc ...
valeur maximum d'une variable de type unsigned char : 255
```

[c] La taille d'une variable **unsigned short** en bits est de [0 ;65535] et fait 2 octets.

```
[~/INF203/TP5]: clang −o deborde_short deborde_short.c
[~/INF203/TP5]: ./deborde_short
taille du type unsigned short : 2 octet(s)
65535 + 1 = 0 donc ...
valeur maximum d'une variable de type unsigned short : 65535
```

[d] La taille d'une variable **unsigned int** en bits est de [0 ;4294967295] et fait 4 octets.

```
[~/INF203/TP5]: clang −o deborde_int deborde_int.c
[~/INF203/TP5]: time ./deborde_int
taille du type unsigned int : 4 octet(s)
4294967295 + 1 = 0 donc ...
valeur maximum d'une variable de type unsigned int : 4294967295

real 0m11,066s
user 0m11,024s
sys 0m0,001s
```

[e] Le temps obtenu pour la commande **time *./deborde_int*** est de 11.024s alors que pour **time *./deborde_short*** le temps est de 0.003s

[f] On a essayé de calculer la relation entre les deux temps obtenus mais on obtient une erreur d'approximation d'environ 50%. Entre temps l'exécution d'un script avec long long tournais, et on l'a arrêté au bout de 20 minutes car on estimait que cela prendrait trop de temps. Nous n'avons pas réussis à estimer ce temps, mais il est vraisemblablement très long.

3 Rangeons !

3.1 Affichages successifs

```
0  [~/INF203/TP5]: cat exemple_generation_tableau.c
1  #include <stdio.h>
2  #include "generer_entier.c"
3
4  // affiche a l'écran T[0..nb-1]
5  void afficher(long T[], int nb) {
6          int i;
7          printf("[ ");
8          for (i = 0; i < nb; i++)
9                  printf("%ld ", T[i]);
10         printf("]\n");
11 }
12
13 int main() {
14         int Taille = 20;
15         long T[Taille];
16         int i;
17         long valeur ;
18         // on initialise le tableau pour pas récuperer la mémoire précédente
19         for (i = 0; i < Taille; i++)
20                 T[i] = 0;
21         for (i = 0; i < Taille; i++) {
22                 valeur = generer_entier(100) ;
23                 T[i] = valeur ;
24                 afficher(T, Taille);
25         }
26         return 0;
27 }
```

```
[~/INF203/TP5]: clang -o exemple_generation_tableau exemple_generation_tableau.c
[~/INF203/TP5]: ./exemple_generation_tableau
[ 55 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 55 51 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 55 51 49 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 55 51 49 32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 55 51 49 32 67 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 55 51 49 32 67 52 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 55 51 49 32 67 52 60 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 55 51 49 32 67 52 60 85 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 55 51 49 32 67 52 60 85 99 0 0 0 0 0 0 0 0 0 0 0 ]
[ 55 51 49 32 67 52 60 85 99 90 0 0 0 0 0 0 0 0 0 0 ]
[ 55 51 49 32 67 52 60 85 99 90 69 0 0 0 0 0 0 0 0 0 ]
[ 55 51 49 32 67 52 60 85 99 90 69 37 0 0 0 0 0 0 0 0 ]
[ 55 51 49 32 67 52 60 85 99 90 69 37 29 0 0 0 0 0 0 0 ]
[ 55 51 49 32 67 52 60 85 99 90 69 37 29 49 0 0 0 0 0 0 ]
[ 55 51 49 32 67 52 60 85 99 90 69 37 29 49 57 0 0 0 0 0 ]
[ 55 51 49 32 67 52 60 85 99 90 69 37 29 49 57 23 0 0 0 0 ]
[ 55 51 49 32 67 52 60 85 99 90 69 37 29 49 57 23 93 0 0 0 ]
[ 55 51 49 32 67 52 60 85 99 90 69 37 29 49 57 23 93 40 0 0 ]
[ 55 51 49 32 67 52 60 85 99 90 69 37 29 49 57 23 93 40 52 0 ]
[ 55 51 49 32 67 52 60 85 99 90 69 37 29 49 57 23 93 40 52 15 ]
```

3.2 Tri du tableau à la volée

[g] On appel la fonction *inserer* avec en parametre T, i et valeur

```
0  [~/INF203/TP5]: cat #include <stdio.h>
1  #include "generer_entier.c"
2
3  // affiche a l'écran T[0..nb−1]
4  void afficher(long T[], int nb) {
5          int i;
6          printf("[ ");
7          for (i = 0; i < nb; i++) {
8                  printf("%ld ", T[i]);
9          }
10         printf("]\n");
11 }
12
13 void echanger(long Tab[], int i, int j) {
14         long tmp;
15         tmp = Tab[i];
16         Tab[i] = Tab[j];
17         Tab[j] = tmp;
18 }
19
20 /* inserer a sa place l'entier val dans la sequence triee Tab[0..nb−1] */
21 void inserer(long Tab[], int nb, int val){
22         Tab[nb] = val;
23         int i = 0;
24         while (Tab[nb − 1 − i ] > Tab[nb − i] && nb − i > 0)
25         {
26                 echanger(Tab, nb − i, nb − 1 − i);
27                 i++;
28         }
29
30 }
31
32 int main() {
33         int Taille = 20;
34         long T[Taille];
35         int i;
36         long valeur ;
37         // on initialise le tableau pour pas récuperer la mémoire précédente
38         for (i = 0; i < Taille; i++)
39                 T[i] = 0;
40
41         for (i = 0; i < Taille; i++) {
42                 valeur = generer_entier(100) ;
43                 inserer(T,i,valeur) ; // T[i] = valeur;
44                 afficher(T, Taille);
45         }
46         return 0;
47 }
```

```
[~/INF203/TP5]: clang -o exemple_generation_tableau exemple_generation_tableau.c
[~/INF203/TP5]: ./exemple_generation_tableau
[ 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 19 50 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 19 50 56 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 19 50 56 72 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 19 50 55 56 72 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 19 50 55 56 72 80 0 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 19 50 55 56 72 80 87 0 0 0 0 0 0 0 0 0 0 0 0 ]
[ 19 24 50 55 56 72 80 87 0 0 0 0 0 0 0 0 0 0 0 ]
[ 19 24 50 55 56 58 72 80 87 0 0 0 0 0 0 0 0 0 0 ]
[ 19 24 50 55 56 58 72 80 81 87 0 0 0 0 0 0 0 0 0 ]
[ 19 24 50 55 56 58 72 80 81 87 97 0 0 0 0 0 0 0 0 ]
[ 19 24 39 50 55 56 58 72 80 81 87 97 0 0 0 0 0 0 0 ]
[ 19 24 39 46 50 55 56 58 72 80 81 87 97 0 0 0 0 0 0 ]
[ 19 24 39 46 50 55 56 58 72 80 81 87 97 99 0 0 0 0 0 ]
[ 19 24 39 46 50 55 56 58 64 72 80 81 87 97 99 0 0 0 0 ]
[ 19 24 39 46 50 55 56 58 64 72 80 81 86 87 97 99 0 0 0 ]
[ 19 24 38 39 46 50 55 56 58 64 72 80 81 86 87 97 99 0 0 ]
[ 19 24 38 39 46 50 53 55 56 58 64 72 80 81 86 87 97 99 0 ]
[ 19 24 38 39 46 47 50 53 55 56 58 64 72 80 81 86 87 97 97 99 ]
```

3.3 Les commandes de la semaine : head et tail

[h] **tail -n 4** renverra les 4 derniere ligne au lieu des 10 alors que **tail -n +4** renverra toute les lignes a partir de la ligne 4.
[i]

```
0  [~/INF203/TP5]: cat ligne_par_4.sh
1  #!/bin/bash
2
3  ligne=$(wc -l $1 | cut -d' ' -f1)
4  nb=$(expr $ligne / 4 )
5
6  i=0
7  while [ $i -lt $nb ]
8  do
9      numligne=$( expr $i \* 4 + 1 )
10     tail -n +$numligne $1 | head -n 4
11     echo ",,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,"
12     i=$(expr $i + 1 )
13 done
14
15 numligne=$( expr $i \* 4 + 1 )
16 tail -n +$numligne $1 | head -n 4
17 echo ''
```

```
[~/INF203/TP5]: chmod u+x ligne_par_4.sh
[~/INF203/TP5]: ./ligne_par_4.sh deborde_char.c
#include <stdio.h>
#include <stdlib.h>



,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
int main() {
	unsigned char courant = 0, prochain = 1;

	printf("taille du type unsigned char : %ld octet(s)\n", sizeof(unsigned char));
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

	while (prochain > courant) {
		courant = prochain;
		prochain = prochain + 1;
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
	}

	printf("%u + 1 = %u donc ...\n", courant, prochain);
	printf("valeur maximum d'une variable de type unsigned char : %u\n", courant);
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

	return 0;
}
```