Licence 1 - section B
TP 1 d'éléments d'informatique

Catherine RECANATI – Département d'Informatique – Institut Galilée

Semaine du 7 au 11 novembre 2016

C'est leur première manip. de shell pour utiliser gcc, lancer le programme, etc. On donne les points à suivre mais il vaut mieux les leur expliquer, et vérifier qu'ils indentent bien leur programme (en les renvoyant au fichier sur la présentation des programmes qui figure sur le site de l'ENT si nécessaire).

1 Premiers pas sous Shell Linux et langage C

1. Depuis votre répertoire principal (base de votre arborescence de fichiers), créez le répertoire TP1 en tapant la commande `mkdir TP1` dans une fenêtre de terminal.
2. Allez dans le répertoire TP1 grâce à la commande `cd` (= *change directory*) en tapant : `cd TP1`.
3. Lancez l'éditeur de texte `gedit` pour créer un nouveau fichier source appelé `bonjour.c`, en tapant la commande `gedit bonjour.c &`. Le caractère & en fin de commande permet de lancer l'exécution de la commande en arrière plan (et ainsi garder la main dans la fenêtre du terminal, sans avoir à attendre la fin de la commande, ici donc sans attendre la fin de l'éditeur `gedit`). L'intérêt ici, sera de pouvoir lancer la compilation du programme et son exécution sans avoir à quitter l'éditeur `gedit`.

Exercice 1.1 Programme Bonjour !

Ce premier programme devra afficher Bonjour ! Vous le composerez en recopiant le programme du cours ou un programme donné en exemple en TD. Votre éditeur `gedit` vous assistera en coloriant automatiquement les mots du code saisi.

Pour réaliser l'affichage de Bonjour !, vous utiliserez l'instruction `printf("Bonjour !\n");` Le `\n` représente l'impression d'un saut à la ligne après l'impression de Bonjour ! `printf` est une fonction définie dans la bibliothèque standard d'entrée/sortie du C, et pour que le compilateur trouve cette définition, il faut insérer la ligne suivante au début de votre programme :

```
#include <stdio.h>          /* pour pouvoir utiliser printf */
```

1. Après avoir fini d'écrire votre programme, enregistrez-le.
2. Pour créer un programme exécutable à partir de votre fichier source, vous devez *compiler* votre fichier avec la commande `gcc`, qu'on détaillera au prochain cours :
   `gcc -Wall bonjour.c -o bonjour.exe`
   Si le compilateur vous signale des erreurs, corrigez-les dans l'éditeur, enregistrez et recompilez.
3. Quand l'étape précédente ne vous signale plus d'erreurs, lancez l'exécution du programme en tapant simplement le nom de son fichier code exécutable :
   `bonjour.exe` (ou `./bonjour.exe`).

Modifiez le programme de manière à ce qu'il affiche votre prénom après «Bonjour !». Vous répéterez ces trois étapes (écrire/sauvegarder, compiler puis exécuter), dans tous les TP.

Exercez-vous à apprendre les raccourcis clavier des diffrentes commandes de l'éditeur pour éviter d'utiliser la souris qui est moins rapide.

Exercice 1.2 Affichages

1. Écrire un programme `coucou.c` qui affiche à l'écran « Coucou ».
2. Modifier ce programme pour qu'il affiche à l'écran « Coucou » sur cinq lignes de deux façons :
   — avec cinq printf ;
   — avec un seul printf.
3. Modifier ce programme pour qu'il affiche à l'écran l'évaluation de l'expression `7 * 3 + 2`.

4. Modifier à nouveau ce programme pour qu'il affiche à l'écran l'évaluation de l'expression `3 * x + 2`, avec la variable entière `x` initialisée à une valeur quelconque dans le programme.

Correction 1.1 *Affichages*

```
1    /* declaration de fonctionnalites supplementaires */
2    #include <stdlib.h> /* pour EXIT_SUCCESS */
3    #include <stdio.h> /* pour printf */
4
5    /* declaration constantes et types utilisateurs */
6
7    /* declaration de fonctions utilisateurs */
8
9    /* fonction principale */
10   int main()
11   {
12      /* affiche coucou sur une ligne 5 fois de suite */
13      printf("Coucou\nCoucou\nCoucou\nCoucou\nCoucou\n");
14
15      /* valeur fonction */
16      return EXIT_SUCCESS;
17   }
------------
```

```
 1   /* declaration de fonctionnalites supplementaires */
 2   #include <stdlib.h> /* EXIT_SUCCESS */
 3   #include <stdio.h> /* printf */
 4
 5   /* declaration constantes et types utilisateurs */
 6
 7   /* declaration de fonctions utilisateurs */
 8
 9   /* fonction principale */
10  int main()
11  {
12       /* declaration et initialisation des variables */
13      int x = 3;
14
15       printf("x * 7 + 2 = %d\n", x * 7 + 2);
16
17     /* valeur fonction */
18     return EXIT_SUCCESS;
19  }
```

2 Lire et imprimer des variables (vu en TD)

On a vu en cours la fonction printf qui utilise des indications de formats avec le symbole %, comme par exemple %d indiquant un format d'impression décimal. Il existe d'autres formats reconnus par printf comme le format %x des nombres hexadécimaux. Le tableau suivant indiquent divers formats que vous pourrez tester dans un programme :

Formats	Sorties	Exemples
%d ou %i	Entier decimal signe	392
%u	Entier decimal non signe	7235
%o	Octal non signe	610
%x	Entier hexadecimal non signe	7fa
%X	Entier hexadecimal non signe	7FA
%f	Decimal virgule flottante, minuscules	392.65
%F	Decimal virgule flottante, majuscules	392.65
%e	Notation scientifique (mantisse/exposant min)	3.9265e+2
%E	Notation scientifique (mantisse/exposant maj)	3.9265E+2

```
%g          Utilise la representation la + courte: %e ou %f        392.65
%G          Utilise la representation la + courte: %E ou %F        392.65
%c          Caractere (char)                                         a
%s          Chaine de caracteres                                  bonjour
%p          Adresse  (pointeur)                                   b8000000
```

A noter que pour les flottants, on peut indiquer le nombre de caractères figurant avant et après la virgule, comme avec `%3.1f` pour obtenir l'affichage 392.6 au lieu de 392.65.

La fonction qui permet de lire des caractères du clavier pour initialiser une variable s'appelle `scanf`. Elle prend (comme `printf`) en premier argument une chaine de caratères contenant des indications de format, et en second argument l'adresse en mémoire d'une variable. Pour connaitre l'adresse d'une variable, on lui appliquera l'opérateur & (prononcer "et commercial" ou "esperluette").

Ainsi, les lignes de programme suivantes permettent de lire une variable initialisée d'abord à zéro à partir des caractères entrés au clavier par l'utilisateur du programme :

```
1    int variable = 0;
2    scanf ("%d", &variable);   /* & est l'operateur d'adresse */
3    printf ("%d\n", variable);
```

La plupart des formats d'impression de printf du tableau précédent restent valables comme format de lecture pour `scanf`. Ainsi on pourra utiliser `%d %i` et `%u` pour lire des décimaux,`%x` et `%X` pour des hexadécimaux, `%o` pour des nombres en notation octale, `%e` , `%f` , et `%g` pour lire des float, et `%lf` pour des double (le l devant le f signifie long). Enfin `%c` et `%s` permettront de lire respectivement un caractère et une chaine de caractères.

A noter que si l'on préfixe l'indicateur de format d'un nombre, cela précisera le nombre de caractères qui seront lus par `scanf`.

```
1   int nb_de_5_chiffres = 0;
2   scanf ("%5d", &nb_de_5_chiffres);
3   printf ("---> %d\n", nb_de_5_chiffres);
```

Exécutions :

```
18
---> 18
1000
---> 1000
1584669842
---> 15846
```

Mais attention, si l'utilisateur tape plus de caracères que nécessaire, les caracères supplémentaires ne seront pas lus mais resteront dans le tampon qui lit les entrées. C'est eux qui se présenteront ensuite pour être lus si on fait un nouvel appel à la fonction scanf.

Exercice 2.1 Lire et imprimer une variable.

1. Ecrire un programme dont la fonction principale lit une variable x de type unsigned long int avec scanf et l'imprime ensuite dans les formats %lu, %lo et %lX.
2. Modifier le programme précdent pour que la fonction main appelle une fonction imprime(x) qui imprime son argument dans les formats %lu, %lo et %lX.

Correction 2.1 *Lire et imprimer une variable.*

1. *Le premier programme est :*

```
1  /* declaration de fonctions ou constantes externes */
2  #include <stdlib.h>    /* pour EXIT_SUCCESS */
3  #include <stdio.h>    /* pour la fonction printf */
4
5   /* fonction principale */
6   int main()
```

```
 7  {
 8     /* declaration de variables locales */
 9     unsigned long int x;
10
11     /* initialisation de x */
12     printf("Entrez un nombre entier : ");
13     scanf("%lu", &x);
14
15     /*  impression de x dans diffrents formats */
16     printf("x = %lu en format \%lu\n", x);
17     printf("x = %lo en format octal \%lo\n", x);
18     printf("x = %lX en format hexadecimal \%lX\n", x);
19
20     /* valeur de retour de la fonction main */
21     return EXIT_SUCCESS;
22  }
```

2. *Le second programme pourrait être :*

```
 1  /* declaration de fonctions ou constantes externes */
 2  #include <stdlib.h>    /* pour EXIT_SUCCESS */
 3  #include <stdio.h>    /* pour la fonction printf */
 4
 5   /* declaration de la fonction imprime */
 6   void imprime (unsigned long int y) ;
 7
 8   /* fonction principale */
 9   int main()
10   {
11     /* declaration de variables locales */
12      unsigned long int x;
13
14      /* initialisation de x */
15      printf("Entrez un nombre entier : ");
16      scanf("%lu", &x);
17
18      printf("La valeur de x \n");
19      imprime (x);
20
21      /* valeur de retour de la fonction main */
22      return EXIT_SUCCESS;
23   }
24
25   /* definition de la fonction imprime */
26   void imprime ( unsigned long int y) {
27      /* imprime y dans divers formats */
28      printf("     = %lu en format \%lu\n", y);
29      printf("     = %lo en format octal \%lo\n", y);
30      printf("     = %lX en format hexadecimal \%lX\n", y);
31    }
```

3 Quelques petits programmes

Pour tous ces programmes, il faudrait que vous insistiez, lorsque vous passerez entre les machines en TP, sur l'indentation, et sur le fait qu'ils ne doivent pas utiliser de caractères blancs (espace), mais uniquement les tabulations pour réaliser cette indentation. J'ai mis en ligne sur l'ENT un fichier contenant quelques instructions. Revoyez-les ce document.

Exercice 3.1 Conversion Fahrenheit-Celsius.

Écrire un programme qui demande à l'utilisateur d'entrer une température en degrés Fahrenheit et affiche sa conversion en degrés Celsius. On s'appuiera sur la formule `[C] = ([F] - 32) x 5/9`.

Correction 3.1 *Conversion Fahrenheit-Celsius.*

```
 1  /* declaration de fonctions ou constantes externes */
 2  #include <stdlib.h>    /* pour EXIT_SUCCESS */
 3  #include <stdio.h>    /* pour printf et scanf */
 4
 5   /* fonction principale */
 6   int main()
 7   {
 8     /* declaration de variables locales */
 9      long  x;          /* pour la temperature. On pourrait aussi la declarer float
10
11      /* lecture au clavier de x */
12      printf("Entrez une temperature entiere en degres Fahrenheit : ");
13      scanf("%ld", &x);
14
15      x = (x - 32)*5/9 ;
16      printf("        =   %ld en degres Celsius\n", x);
17
18      return EXIT_SUCCESS;
19   }
```

Exercice 3.2 Majeur ou mineur ?

Écrire un programme qui demande à l'utilisateur d'entrer son âge et affiche ensuite si la personne est majeure ou mineure. L'algorithme devra d'abord être écrit en français en mettant l'indentation.

Correction 3.2 *Majeur ou mineur ?*

```
 1  /* declaration de fonctions ou constantes externes */
 2  #include <stdlib.h>    /* pour EXIT_SUCCESS */
 3  #include <stdio.h>    /* pour printf et/ou scanf */
 4
 5   /* fonction principale */
 6   int main()
 7   {
 8     /* declaration de variables locales */
 9      unsigned int x;          /* pour l'age de l'utilisateur */
10
11      /* initialisation de x */
12      printf("Entrez votre age : ");
13      scanf("%u", &x);
14
15      if  ( x >=  18)
16             printf("Vous etes majeur \n");
17      else
18             printf("Vous etes mineur \n");
19
20      return EXIT_SUCCESS;
21   }
```

Exercice 3.3 Année bissextile ?

Écrire un programme qui demande à l'utilisateur d'entrer une année et qui affiche ensuite si l'année est bissextile ou non. On rappelle qu'une année est bissextile dans les deux cas suivants :

1. si l'année est divisible par 4 et non divisible par 100, ou
2. si l'année est divisible par 400.

Donc en particulier l'année 1900 n'est pas bissextile. Par contre l'an 2000 l'est.

Correction 3.3 *Année bissextile ?*

Pour déterminer si l'année entrée par l'utilisateur est bissextile, on va utiliser une variable booléenne qui sera initialisée à zéro pour dire que l'année n'est pas bissextile, et qui vaudra 1 à la fin du programme si l'année s'avère

avoir franchi un test montrant qu'elle est bissextile. Ainsi l'algorithme est :

declarer une variable (booléenne) bissextile
et initialiser cette variable à 0 (FAUX)
demander a l'utilisateur d'entrer une annee
et lire ce nombre dans la variable an
si an est divisible par 4 ET an n'est pas divisible par 100
alors affecter la valeur 1 (VRAI) à bissextile
si an est divisible par 400
alors affecter la valeur 1 (VRAI) à bissextile
si bissextile vaut 1 (est VRAI)
alors imprimer que l'année est bissextile
sinon
imprimer que l'année n'est pas bissextile

D'où le programme :

```
  ...
3   /* fonction principale */
4   int main()
5   {
6     /* declaration de variables locales */
7      unsigned int an ;          /* pour l'annee entree par l'utilisateur */
8     int  bissextile ;        /* variable booleenne */
9
10     /* initialisation des variables bissextile et an */
11     bissextile = 0;
12     printf("Entrez une annee : ");
13     scanf("%u", &an);
14
15     if  (  (an%4 == 0)  &&  (an%100 != 0)  )
16            bissextile = 1 ;
17     if    (an%400 == 0)
18            bissextile = 1 ;
19
20     if (bissextile == 1)
21            printf("L'annee %d est bissextile\n",  an);
22     else
23            printf("L'annee %d n'est pas bissextile\n",  an);
24     return EXIT_SUCCESS;
25  }
```

Exercice 3.4 Minimum de 3 valeurs.

Soient 3 variables `a`, `b`, `c`, initialisées au clavier par l'utilisateur. Écrire une fonction minimum (a, b, c) qui calcule et retourne le minimum de ces 3 valeurs. Appeler la fonction dans le programme principal et imprimer le résultat.

Correction 3.4 *Minimum de 3 valeurs.*

L'algorithme de la fonction principale (main) :
Demander a l'utilisateur d'entrer 3 valeurs
appeler la fonction minimum et mémoriser le résultat dans res
imprimer le résultat res

La déclamation de la fonction minimum est
`int  minimum (int a, int b, int c)` *et son algorithme :*
Déclarer une variable resultat qui contiendra à la fin le minimum des 3 valeurs
si a > b
alors resultat = b
sinon
alors resultat = a
(On a maintenant le plus petit de a ou b dans la variable resultat)

On compare maintenant c avec ce premier minimum des 2 valeurs :
si c < resultat
alors resultat = c
(sinon resultat contient deja le minimum)
retourner resultat

```
 1  /* declaration de fonctions ou constantes externes */
 2  #include <stdlib.h>    /* pour EXIT_SUCCESS */
 3  #include <stdio.h>    /* pour printf et scanf */
 4
 5   /* declaration de la fonction minimum */
 6   int minimum (int a, int b, int c) ;
 7
 8   /* fonction principale */
 9   int main()
10   {
11     /* declaration de variables locales */
12      int x, y, z ;          /* pour les trois valeurs a comparer */
13      int res ;             /* pour memoriser le resultat */
14
15      /* initialisations des trois valeurs a comparer */
16      bissextile = 0;
17      printf("Entrez 3 entiers spares par des blancs: ");
18      scanf("%d%d%d\n", &x, &y, &z);
19
20      res = minimum (x, y, z);
21
22      printf("Le minimum de %d, %d, %d est %d\n", x, y, z, res);
23      return EXIT_SUCCESS;
24   }
25
26   /* definition de la fonction minimum  */
27   int minimum(int a, int b, int c)
28   {
29       int resultat;
30
31       if ( a > b)
32              resultat = b;
33       else
34              resultat = a;
35       if (c < resultat)
36              resultat = c;
37       return resultat;
38    }
```