Premières fonctions, conditionnelles

Une feuille de TD sera proposée pour chaque semaine, un peu adaptée pour la séance de TP. Si vous n'avez pas atteint la section *renforcements* avant la fin des TD/TPs de la semaine, vous devez les terminer par un travail à la maison, et éventuellement poser vos questions à la séance suivante.

1 Prise en main de linux et du terminal

Si votre machine a démarré sous *windows*, redémarrez-là et sélectionnez *ubuntu*. L'interface utilisateur par défaut de ce système d'exploitation ressemble beaucoup aux interfaces que vous connaissez, et vous devriez vous y retrouver. La touche "windows" vous permet d'ouvrir le menu principal, vous reconnaitrez l'icone de *firefox* pour naviguer sur internet. L'icône représentant un porte document est l'explorateur de fichiers, et vous permet de naviguer dans votre arborescence.

Dans le menu à gauche, vous pouvez trouver un élément appelé terminal ou console (ou xterm). Vous pouvez aussi ouvrir le terminal avec la séquence CTRL+ALT+T. Le terminal est un outil de communication direct avec la machine. Bien que peu convivial, il peut s'avérer très efficace. Vous allez utiliser le terminal comme interface de communication lors de la compilation, autant le prendre en main.

Nous allons voir comment utiliser le terminal.

1. Observez d'abord le texte qui est affiché dans le terminal. On l'appelle *invite de commande*. Vous y voyez votre numéro d'étudiant (login), suivi d'un , suivi du nom de la machine, et de `:`. Le `~` qui suit vous indique le répertoire courant, `~` étant utilisé pour désigner votre répertoire personnel. Le $ final est juste là pour conclure l'invite.
2. Utilisez la commande `ls`. Elle vous permet de lister le contenu du répertoire courant. Si vous voulez plus de détails sur les fichiers listés, vous pouvez utiliser la commande `ls -l` (bien respecter les positions des espaces).
3. La commande `mkdir` vous permet de créer un répertoire. Créez un répertoire toto avec la commande `mkdir toto`. Avec `ls`, vérifiez que le répertoire a bien été créé. Créez de la même façon un repertoire *titi* et un répertoire *tata*.
4. La commande `cd` vous permet de changer de répertoire. Entrez dans le répertoire *toto* avec `cd toto` et vérifiez qu'il est vide. remarquez que votre invite de commande vous indique maintenant que vous vous trouvez dans le répertoire ~/toto.
5. le répertoire parent du répertoire courant est le répertoire `..`. Utilisez `cd ..` pour retourner dans votre répertoire principal. Entrez maintenant dans le répertoire *titi*. Constatez à nouveau qu'il est vide et le changement de l'invite. Sautez maintenant directement dans le répertoire *tata* avec la commande `cd ../tata`. Vous pouvez ainsi composer des adresses de répertoire pour vous déplacer plus rapidement dans les dossiers. Sautez maintenant dans le répertoire *titi*
6. La commande `mv` vous permet de déplacer des fichiers ou des répertoires. Pour cela, vous allez avoir besoin d'adresses composées. placez vous dans le répertoire *toto* et déplacer le répertoire titi dans celui-ci grace à la commande `mv ../titi ./` (dans laquelle `.` désigne le répertoire courant). Observez que le répertoire titi est bien dans votre emplacement courant, et allez vérifier qu'il ne se trouve plus dans votre répertoire personnel (~).
7. L'outil que vous allez utiliser initialement pour programmer est l'éditeur de texte de base de gnome, qui s'appelle `gedit`. Tapez la commande `gedit programme.c` pour créer un fichier *programme.c* et le modifier. Vous perdez la main du terminal (qui attend que vous ayez fini d'utiliser gedit pour reprendre) et une fenêtre d'édition s'ouvre. Lorsque vous fermez la fenêtre d'édition, vous reprenez la main sur le terminal. Vous pouvez aussi garder la main dans le terminal en tapant `gedit programme.c &` (avec l'ajout d'un & commercial).
8. Vous pouvez enfin tester la commande de suppression `rm`, qui vous permet de supprimer un fichier. Pour supprimer un répertoire et tout ce qu'il contient, vous pouvez utiliser `rm -r` (pour récursivement).

2 Compilation du *C*

9. Dans un fichier bonjour.c, avec l'extension `.c`, écrivez le programme *Hello world* suivant :

```
#include <stdio.h>
int main(void){
  printf("Hello world\n");
  return 0;
}
```

On notera le caractère `\n` qui sert de retour à la ligne.

10. Sauvegarder le fichier, et compilez le avec `gcc -Wall bonjour.c -o bonjour`. Ceci crée un fichier `bonjour` qui contient votre programme.
11. Lancez votre programme en tapant `./bonjour` dans le terminal. Vous devriez voir apparaître "Bonjour" dans la console.

La fonction `printf` de la bibliothèque standard `<stdio.h>` vous permet de faire des affichages dans la console. On peut faire des affichages formatés avec des balises dans le texte. En particulier, la balise `%d` permet d'afficher un entier, `%f` un nombre à virgule.

12. Testez les affichages formatés dans la console en modifiant votre programme main pour ajouter les lignes :

```
int entier = 42;
double decimal = 5.1;
printf("Par exemple, %d est entier et %f est décimal.\n", entier, decimal);
```

Dans la suite, pour chacune de vos fonctions, ajoutez une fonction *main* qui vous permettra de tester votre fonction. Avec l'inclusion de la bibliothèque `<stdio.h>` et l'utilisation de la fonction `printf`, vous pourrez afficher vos résultats. Pour plus d'exemples de printf, `https://fr.wikipedia.org/wiki/Printf`

3 Fonctions simples

1. Écrivez une fonction `poly` qui prend en argument une valeur (*double*) x et retourne la valeur (*double*) de x^2+x+1.
2. Écrivez une fonction `main` qui appelle `poly` sur une valeur de votre choix et affiche le résultat.
3. Écrivez une fonction `moyenne` qui prend en argument deux valeurs (*double*) a et b et retourne la moyenne des deux.

4 Conditionnelles

Une année est bissextile si son numéro est divisible par 4 mais pas par 100, ou est divisible par 400.

4. Écrivez de deux façons une fonction qui prend en entrée une année (sous forme d'un entier) et renvoi le nombre de jours de l'année (sous forme d'un entier).
   — Avec un seul *if-else* mais une expression logique qui traduit l'ensemble de la contrainte ci-dessus.
   — Avec plusieurs *if-else* imbriqués mais que des tests simples.

4.1 Maximum

5. Écrivez une fonction `max2` qui retourne le maximum de deux entiers.
6. Écrivez une fonction `max3` qui retourne le maximum de trois entiers
   — sans utiliser la fonction précédente.
   — en l'utilisant, avec une variable intermédiaire.
   — en l'utilisant, en une seule instruction.

On notera que l'implémentation de `max3` la plus qualitative, pour sa facilité de lecture, est sans doute la seconde.

4.2 Le parc d'attraction

Un parc d'attraction calcule le tarif d'entrée en fonction de l'heure d'arrivée des clients. Le parc est ouvert de 10h à 18h. Le tarif est calculé suivant le principe :

— Les enfants de moins de 5 ans et les personnes de plus de 70 ans ne payent pas.
— Pour toute personne agée de moins de 12 ans ou de plus de 60 ans, un tarif de base de 6 euros est appliqué auquel s'ajoutent 2 euros par heure complète d'ouverture restante. Le tarif par personne ne peut pas dépasser 18 euros.
— Pour les autres, un tarif de base de 7 euros est appliqué auquel s'ajoutent 3 euros par heure complète d'ouverture restante. Le tarif par personne ne peut pas dépasser 22 euros.

Ainsi, un enfant de 11 ans arrivant à 14h17 devra payer 12 euros, une femme de 38 ans arrivant à 11h42 payera le tarif complet de 22 euros.

7. Écrivez l'algorithme qui étant donnés l'âge d'une personne (*int*) et son heure d'arrivée, calcule et affiche le montant de son entrée. L'heure est représentée par un entier correspondant au numéro de l'heure en cours, on supposera que celle-ci est toujours entamée, et donc non comptée dans le coût de l'entrée.

5 Renforcements

Cette section permettra chaque semaine aux plus rapides de continuer de progresser.

5.1 Résolution d'une équation du second degré.

8. Écrivez une fonction `solve` qui prend en entrée les trois coefficients a, b, et c de l'équation $ax^2 + bx + c = 0$ et retourne le nombre de solutions de l'équation.
9. Modifiez votre fonction pour qu'elle affiche aussi les éventuelles solutions trouvées.
10. Pourquoi dans un cas, on vous demande de retourner le nombre de solutions, et dans l'autre d'afficher les solutions ?

5.2 Reprographie

Le service de reprographie propose les photocopies avec le tarif suivant : les 10 premières coûtent 20 centimes l'unité, les 20 suivantes coûtent 15 centimes l'unité et au-delà de 30 le coût est de 10 centimes.

11. Écrivez une fonction `coutPhotocopies` qui retourne le prix à payer en centimes pour un nombre de photocopies donné en paramètre.

5.3 Horloges

12. Écrivez une fonction `seconde_vers_horaire` qui prend en entrée un nombre de secondes depuis le début de la journée et affiche l'heure au format habituel. 53537 deviendra par exemple 14 :52 :17.
13. Écrivez une fonction `une_seconde_plus_tard` qui prend en entrée un horaire et affiche l'heure qu'il sera une seconde après. L'entrée se fera sous la forme de trois entiers pour les heures, minutes et secondes.
14. Écrivez une fonction qui fait le processus inverse de la question 12, à savoir prend les trois paramètres heure, minute et seconde et calcule le nombre de secondes que cela représente.
15. Écrivez une fonction `duree` qui prend en paramètres deux horaires hh :mm :ss et qui calculent le temps qui les séparent (en prenant éventuellement en compte un jour d'écart si le deuxième horaire est plus tôt que le premier).