SÉANCE 1

Objectif

Le but de cette première séance est d'écrire des programmes simples et de vous familiariser avec les outils de travail : éditeur de texte, interface en ligne de commande, compilateur.

Outils

La suite suppose que vous êtes sous Fedora qui est une distribution du système d'exploitation Linux.
Vous aurez besoin de :

- un **terminal** depuis lequel vous taperez des lignes de commandes pour compiler vos programmes et lancer leur exécution.
- un **éditeur de texte** qui vous permettra de créer et modifier vos programmes sources (vous pouvez utiliser par exemple `Kate` ou `gedit`).

Supposons que votre numéro d'étudiant est `21601234`. Le répertoire (dossier) accessible en double-cliquant sur l'icône `Home` qui se trouve sur le bureau de l'interface graphique est `/home/21601234`. C'est dans le sous-répertoire `Documents` de ce répertoire que vous placerez les fichiers contenant vos programmes.

Dans le terminal, pour pouvoir compiler et lancer l'exécution de vos programmes, vous vous placerez dans le répertoire que vous aurez choisi pour placer les fichiers contenant vos programmes.

Désignation d'un fichier ou d'un répertoire

Le système de fichiers Unix suit un modèle hiérarchique : un répertoire « contient » des fichiers et des répertoires. Au sommet de la hiérarchie, le répertoire racine, désigné par `/`, contient d'autres répertoires : `home`, `dev`, `lib`, etc.

- Chaque répertoire contient deux répertoires particuliers :
  `.` : désigne le répertoire lui-même
  `..` : désigne le répertoire supérieur (le « père »).
- Chaque utilisateur dispose d'un **répertoire d'accueil** (*home directory*), appelé parfois « répertoire principal ».
  Exemple : `/home/21601234`
- À chaque instant, un utilisateur se trouve dans l'un des répertoires du système de fichiers : c'est le **répertoire courant** (*working directory*).
- **Désignation relative** : on désigne un fichier en donnant le chemin (suite des noms des répertoires séparés par le caractère `'/'`) qui mène à ce fichier en partant du répertoire **courant**.
  Exemple : si l'on suppose que le répertoire courant est `/home/21601234/Documents/REP1` et que le répertoire `/home/21601234/Documents/REP2` contient un fichier `f1`, alors la désignation relative du fichier `f1` est :
  `../REP2/f1`
- **Désignation absolue** : on désigne un fichier en donnant le chemin qui mène à ce fichier en partant du répertoire **racine** (elle commence toujours par `'/'`).
  Exemple : `/home/21601234/Documents/REP2/f1`

Format général d'une commande Unix

`nom_commande options arguments`

Les options se situent juste après le nom de la commande ; elles se caractérisent par le signe `-` suivi d'une lettre. Les arguments se situent après les options (s'il y en a). Les options et les arguments forment les paramètres de la commande.
Exemple de commande avec 2 options et 3 arguments :

```
ls -a -l fichier1 fichier2 fichier3
```

Quelques commandes utiles

Dans la suite, les arguments des commandes sont des désignations relatives ou absolues de fichiers ou de répertoires.

- `pwd`
  Affiche la désignation absolue du répertoire courant.
- `ls` [`-al`] [`FichierOuRépertoire` ...]
  Affiche le contenu d'un ou de plusieurs répertoires ou des renseignements concernant un ou plusieurs fichiers.
  - sans argument : affiche le contenu du répertoire courant ;
  - `-a` : affiche aussi les fichiers cachés (dont le nom commence par `.`)
  - `-l` : affiche des renseignements détaillés pour chaque fichier ou répertoire.
- `cd` [`Répertoire`]
  `Répertoire` devient le nouveau répertoire courant (« on se rend dans le répertoire de nom `Répertoire` »). Sans argument, on se rend dans le répertoire d'accueil.
- `mkdir Répertoire`
  Crée le répertoire `Répertoire`.
- `rmdir Répertoire`
  Supprime le répertoire `Répertoire`. Cette opération n'est possible que si `Répertoire` est vide.
- `cp FichierSource FichierDestination`
  Copie le fichier `FichierSource` dans `FichierDestination`.
- `mv Source Destination`
  Renomme (déplace) le fichier ou le répertoire `Source`.
- `rm Fichier` [...]
  Supprime le ou les fichiers dont les désignations sont passées en arguments. Attention, cette opération n'est pas réversible.

Commande de compilation

Utilisez la commande de compilation suivante :

```
gcc -Wall fichier.c -o fichier
```

L'option `-Wall` demande au compilateur de vérifier de manière plus stricte la syntaxe du programme afin de faciliter la détection des erreurs.

Si votre programme fait appel à des fonctions de la bibliothèque mathématique standard (`sqrt`, `fabs`, `cos`, etc.), ajoutez au début de votre programme source la ligne :

```
#include <math.h>
```

et n'oubliez pas d'ajouter à la fin de la commande de compilation l'option `-lm`, c'est-à-dire :

```
gcc -Wall fichier.c -o fichier -lm
```

Exercices

☞ Exercice 1 (Utilisation des commandes Unix)

Dans le terminal, testez les commandes `pwd`, `ls`, `mkdir` et `cd`. Placez-vous dans votre répertoire `Documents` (par exemple `/home/21601234/Documents/`) et créez un sous-répertoire `PROGC` dans lequel vous placerez vos programmes. Placez-vous dans `PROGC`. Depuis l'éditeur de texte, pensez à enregistrer vos programmes sources dans ce répertoire.

☞ Exercice 2 (Premier programme)

Écrivez un programme qui affiche un message simple à l'écran. Pour cela, appliquez les étapes de la chaîne de production vues en cours.

☞ Exercice 3 (Carré d'un entier)

Écrivez un programme qui lit un nombre entier tapé au clavier et qui affiche son carré. Par exemple, si l'entier tapé est `12`, le programme doit afficher `144`.

☞ Exercice 4 (Racine carrée d'un réel)

Copiez le programme source de l'exercice 3 dans un autre fichier et modifiez-le pour lire un nombre réel et afficher sa racine carrée. Pour cela, vous utiliserez une variable de type `double` et la fonction `sqrt` de la bibliothèque mathématique standard.

☞ Exercice 5 (Permutation des valeurs de deux variables)

Écrivez un programme qui lit deux entiers, les range dans deux variables `a` et `b`, permute les valeurs des deux variables et les affiche à l'écran.

☞ Exercice 6 (Conversion de température)

Écrire un programme qui lit une température exprimée en degrés Farenheit et l'affiche après l'avoir convertie en degrés Celsius. La température lue sera un entier. La température affichée sera, elle aussi, un entier. Vous n'utiliserez que des variables entières de type `int` (pas de flottants) dans votre programme.

La conversion Farenheit (F) $\rightarrow$ Celsius (C) est assurée par la formule suivante :

$$C = \frac{5}{9}(F - 32)$$

☞ Exercice 7 (Carré d'un petit entier)

Copiez le programme source de l'exercice 3 dans un autre fichier et modifiez-le de telle sorte que :

- l'entier lu soit stocké dans une variable de type `unsigned char` ;
- le carré soit stocké dans une autre variable de type `unsigned char`.

Pour lire une valeur et la stocker dans une variable de type `unsigned char`, utilisez la fonction `scanf` avec le spécificateur de format `%hhu`.

Testez votre programme en tapant l'entier `8`. Refaites un test avec l'entier `16`. Expliquez le résultat obtenu.

Pour aller plus loin

☞ Exercice 8 (Min et max d'une série de cinq réels)

Ecrire un programme qui demande à l'utilisateur de saisir cinq nombres réels et qui affiche ensuite le plus petit et le plus grand nombre de cette série.