IN405 – Feuille de TD #2
Tout ce qu'il faut savoir sur le terminal (en bref)

Objectif : initiation/rappels à l'usage du terminal, la compilation et le débogage.
Instructions :

- *vous aurez besoin de l'archive* `td2-contents.tar.gz`.
- *l'exercice 2.5 et la question 13 requiert le fichier* `mystery-code.c`

Exercice 2.1 - Compréhension des commandes de base

Soit la liste de commandes/exécutables suivante :

```
 cat   cd  cp  diff  echo  gcc  gdb   ls   make  man
mkdir  mv  rm  rmdir  sudo  tar  time  touch  vi
```

1- Donnez une brève description pour chacune des commandes.

2- Quelles commandes consistent en l'exécution d'un binaire ?

3- Quels chemins sont représentés par les symboles suivants : `.`, `..`, `~`.

Exercice 2.2 - Première utilisation du terminal

Pour chacune des questions suivantes, exécutez la commande correspondante.

4- Déplacez vous dans le répertoire temporaire de votre système de fichiers.

5- Créez le répertoire `project` ainsi que les sous-répertoires `doc`, `include` et `src`.

6- Au sein du dossier `project`, créez un fichier `README` contenant votre nom et prénom. Créez le fichier `func.h` dans `include`, les fichiers `main.c` et `func.c` dans `src`.

7- Affichez la hiérarchie complète du répertoire `project` et des ses sous-répertoires, puis écrivez ce résultat dans `contents.txt`.

8- Créez une copie du répertoire `project` que vous nommerez `projectV2`. Supprimez le répertoire `project`.

9- Créez l'archive `pv2.tar` contenant l'ensemble du répertoire `projectV2`.

Exercice 2.3 - Premier script Shell

Afin d'automatiser l'exécution de commandes (comme par exemple la compilation d'un projet ou l'exécution d'un jeu de tests), il est possible de les rassembler dans un fichier. Ce type de fichier est appelé script.

10- Placez l'ensemble des commandes écrites dans l'exercice 2.2 dans un script Shell, et exécutez-le. Le résultat est-il le même que dans l'exercice 2.2 ?

Exercice 2.4 - Shell en C

Il vous est possible d'exécuter des commandes Shell en C, soit en exécutant un script grâce à la suite d'appels `exec()`, soit en exécutant directement une commande avec l'appel `system`.

11- A l'aide de la fonction `system()`, faites un programme C affichant le contenu de votre répertoire personnel.

Exercice 2.5 - Débogage

Le débogage est une technique qu'il vous faut pratiquer pour gagner du temps sur le développement d'un programme. Par exemple, en cas d'erreur de segmentation, l'utilisation de gdb sur un code C vous indiquera directement quelle instruction provoque l'erreur. Si vous utilisiez des affichages `printf()`, vous devrez en mettre entre chaque instruction pour obtenir le même résultat. Le programme `mystery-code.c` qui vous est fourni présente un(?) bogue, à vous de le débusquer.

12- Compilez le programme `mystery-code.c` en utilisant l'option `-g` de `gcc`, puis déboguez-le à l'aide de `gdb` jusqu'à atteindre l'exécution normale du programme.

Rappel des commandes gdb :
`break fichier:ligne` – *ajout d'un point d'arrêt dans le code*
`run arg1 arg2 ...` – *exécution du programme*
`CTRL + c` – *envoi d'un signal d'interruption au programme*
`next` – *exécution de l'instruction suivante*
`continue` – *reprise de l'exécution du programme*
`print var` – *affichage du contenu d'une variable*
`backtrace` – *affichage de la pile d'appels des fonctions*
`up/down i` – *remontée/descente de i dans la pile d'appels*
`quit` – *arrêt du débogueur*

Exercice 2.6 - *Est-ce que scripter c'est développer ?*

Un script Shell peut, tout comme un programme C, prendre des arguments lors de son exécution. Supposons la commande `./script.sh hello world` : les arguments `$0`, `$1` et `$2` contiennent respectivement les chaînes de caractères `./script.sh`, `hello` et `world`.

Un script Shell peut également utiliser des blocs de contrôle (if, for, etc.). Le script `shell-example.sh` vous montre quelques exemples d'utilisation de ces blocs.

13- Écrivez un script Shell qui compile le fichier `mystery-code.c` de l'exercice 2.5 et qui exécute le programme, seul si vous donnez l'argument `release` au script, avec `gdb` si vous donnez l'argument `debug`.

14- Écrivez un script Shell qui prend un entier positif n en paramètre et affiche la somme des entiers de 0 à n.

15- Écrivez un script Shell qui prend trois arguments *op*, a, b avec *op* une opération parmi {`+`, `-`, `*`, `/`}, a et b des entiers, et affiche le résultat de l'opération a *op* b.