Algorithmes et Pensée Computationnelle

Logiciels système

Le but de cette séance est de vous permettre de comprendre le rôle d'un système d'exploitation, de logiciels système, d'interpréteurs, de compilateurs et des librairies. Cette séance sera composée de deux parties. Une première partie qui sera consacrée à l'installation des outils de travail et une deuxième partie qui contient des questions à choix multiples en lien avec les notions abordées dans le cours.
Le code utilisé dans les questions est également disponible sur Moodle.

1 Installation des outils

Suivre le guide suivant pour installer les outils qui seront utilisés lors des prochaines séances de TP. Le guide d'installation des outils se trouve à cette adresse : `https://moodle.unil.ch/mod/resource/view.php?id=1357530`.

2 Systèmes d'exploitation

Le système d'exploitation (OS - Operating System) est le logiciel le plus important de l'ordinateur car il est responsable de tous les autres programmes et de leurs priorités d'exécution, il organise aussi les interactions et l'orchestration des différents composants physiques de la machine.

L'OS contrôle l'accès au matériel physique et gère le(s) processeur(s), la mémoire, le stockage, la sécurité et les périphériques externes. Par exemple, l'OS identifie quel utilisateur est connecté (au moment d'entrer le mot de passe à l'allumage), il reconnaît quelles touches du clavier et de la souris sont pressées, il affiche les images à l'écran et il sauvegarde les fichiers dans les disques durs internes ou externes.

La plupart du temps, de nombreuses applications sont exécutées au même moment sur le même ordinateur. Elles ont besoin d'utiliser les mêmes composants de l'ordinateur (CPU, RAM, Storage). Cependant, ces différents composants ne peuvent faire qu'une seule chose à la fois (même si ils les font très rapidement), c'est la raison pour laquelle l'OS orchestre chaque tâche de manière à ce que l'utilisateur ait l'impression que tout se déroule en même temps.

Question 1: (🕒 *2 minutes*) **QCM**
Lequel de ces logiciels **n**'est **pas** un système d'exploitation ?

1. Microsoft Office 2012
2. Microsoft Windows 98
3. Unix BSD
4. Gentoo
5. Aucune réponse correcte.

Question 2: (🕒 *2 minutes*) Quel est le rôle du noyau *Kernel* du système d'exploitation ? Trouvez la **mauvaise** réponse.

1. Servir d'interface de communication entre la partie logicielle et matérielle.
2. Gérer les ressources physiques de l'ordinateur.
3. Fournir les mécanismes d'abstraction du matériel.
4. Interpréter les instructions et les convertir en binaire.
5. Aucune réponse correcte.

Question 3: (🕒 *2 minutes*)
Avec lequel de ces éléments le système d'exploitation n'a aucune interaction ?

1. Le clavier et la souris.
2. Le processeur.

3. La carte réseau.
4. La mémoire vive (RAM).
5. Aucune réponse correcte.

Question 4: (🕒 *2 minutes*) Lequel de ces systèmes d'exploitation **ne** permet **pas** d'exécuter plus d'un programme en même temps ?

1. Ubuntu
2. MacOS
3. Windows XP
4. MS-DOS

> **Conseil**
>
> La plupart des systèmes d'exploitation actuels permettent d'exécuter plusieurs programmes en même temps. Pour répondre à cette question, pensez à vérifier la date de sortie des systèmes d'exploitation énumérés.

Question 5: (🕒 *3 minutes*) Que fait la commande suivante : **ls** *(Linux/MacOS)* / **dir** *(Windows)* ?

1. Affiche sous forme de liste uniquement l'ensemble des fichiers du disque.
2. Affiche sous forme de liste l'ensemble des processus en cours.
3. Retourne uniquement les dossiers du répertoire courant.
4. Affiche tous les dossiers et fichiers du répertoire courant.
5. Aucune réponse n'est correcte.

> **Conseil**
>
> Pour avoir une description détaillée d'une commande, vous pouvez ajouter **man** devant chaque commande sous Linux/MacOS ou ajouter **–h, ––help** ou **/?** après chaque commande sous Windows.

Question 6: (🕒 *2 minutes*)
Parmi les éléments suivants, lequel n'est pas géré par le système de fichiers (*filesystem*) ?

1. L'organisation des dossiers.
2. L'écriture et la lecture sur les disques.
3. Les différents chemins d'accès aux différents dossiers et fichiers.
4. L'organisation des fichiers.
5. Aucune réponse correcte.

3 Interpréteurs et compilateurs

Les ordinateurs ne "comprennent" pas les langages de programmation, il faut passer par un programme qui va convertir le code écrit par un humain en instructions que l'ordinateur comprend (à base de 0 et de 1).
Les interprètes et les compilateurs servent à faire ce travail de traduction, ils transforment donc les langages de programmation qui sont faits pour être compris et écrits par des humains, vers des instructions compréhensibles pour des ordinateurs.
La façon dont les interprètes et les compilateurs opèrent est différente et les deux approches apportent leur propre lot de bénéfices et d'inconvénients :

Compilateurs	Interpréteurs
Le programme est intégralement traduit à l'avance	Les instructions sont traduites à la volée une par une
Le compilateur trouve les erreurs à la compilation	L'interprète relève les erreurs pendant l'exécution du programme
Exécution plus rapide	Exécution plus lente
Code machine généré et optimisé à l'avance	Code machine généré à la volée et optimisé à chaque exécution
Ex : C, C++, Java, Scala, Rust, etc.	Ex : Bash, Python, Javascript, etc.

Question 7: (*10 minutes*) En utilisant l'invite de commande (Terminal), créer un fichier contenant les instructions suivantes :

```
1        if __name__ == "__main__":
2            print("Hello world")
```

Enregistrer le fichier sous **hello.py**. Utiliser l'interpréteur de Python pour exécuter le programme que vous venez de créer.

Conseil

- Assurez-vous d'avoir correctement installé les outils de développement avant d'exécuter le programme.
- Utiliser un éditeur de texte présent sur votre ordinateur pour écrire directement votre programme depuis le terminal sous MacOS ou Linux. Sous MacOS, nous vous conseillons d'utiliser Nano (installé par défaut)en tapant **nano hello.py**. Sous Linux (Debian/Ubuntu), vous pouvez installer **Nano** en utilisant la commande suivante : **sudo apt-get install nano**. Alternativement, il est possible de créer un fichier vide directement depuis le terminal au moyen de la command **touch > hello.py**.
- Sous Windows, vous pouvez utiliser n'importe quel éditeur de texte. Notepad est installé par défaut sous Windows.

Question 8: (*10 minutes*) En utilisant l'invite de commande (Terminal), exécutez le programme Python suivant en lui passant des paramètres.

```
1  import sys
2
3  if __name__ == '__main__':
4   arguments = sys.argv
5   if len(arguments) == 3:
6     print("La somme de {} et {} est {}".format(sys.argv[1], sys.argv[2], int(sys.argv[1])+int(sys.argv[2])))
7   else:
8     print("Assurez-vous de passer deux arguments à votre programme")
```

Conseil

- Le code ci-dessus est disponible sur Moodle.
- Le programme affiche la somme de deux nombres passés en paramètres. Pour que le programme fonctionne, assurez-vous de passer des valeurs numériques.
- **sys.argv** est une liste contenant les paramètres passés au programme. Le premier paramètre contient le chemin d'accès au fichier exécuté.

Question 9: (*10 minutes*) Soit le programme suivant écrit en Java.

```
1  public class Question9 {
2    public static void main(String args[]) {
3      for(int i = 0; i<args.length; i++) {
4        System.out.println("Paramètre[" + i + "]: " + args[i]);
5      }
6    }
7  }
```

Exécutez-le en utilisant l'invite de commande (Terminal) en vous assurant d'y ajouter des paramètres.

Conseil

- Le code ci-dessus est disponible sur Moodle.
- Le programme affiche l'ensemble des paramètres qui lui auront été passé.
- **String args[]** est une liste (de chaînes de caractères) contenant les paramètres passés au programme. Pour accéder à un élément de cette liste, vous pouvez faire **args[index]** où **index** est un entier représentant la position dans la liste **args**.
- Pour naviguer dans votre système de fichiers, pensez à utiliser les commandes présentées à la diapositive 20 du cours.

4 Librairies

Une librairie est un ensemble de fonctions qui ont pour but d'être utilisées par d'autres programmes, mais une librairie seule ne suffit pas pour faire un programme. Par exemple, que ce soit Google Chrome, Word, ou Instagram, presque tous les programmes ont besoin de manipuler des listes d'objets. Au lieu de réécrire l'ensemble des fonctions qui permettent de créer et de gérer des listes, ces différents programmes utilisent tous une librairie écrite par un tiers qui offre toutes les fonctionnalités attendues d'une liste.

En Python, de nombreuses librairies sont disponibles de base (**list**, **set**, **dict**, **str**, etc.) et de nombreuses autres peuvent être utilisées en utilisant le mot-clé **import**.
Le but des exercices ci-dessous est de vous apprendre à lire la documentation d'un langage de programmation. Cela vous permettra d'être plus autonome pour la suite du cours.

Dans l'exemple suivant, nous importons la librairie **math** qui propose de nombreuses fonctions mathématiques et nous essayons la fonction factorielle pour calculer **5!**.

>_Exemple

```
1  import math
2  fact = math.factorial(5)
3  print(fact)
```

Conseil

La documentation de la librairie **math** sous Python est disponible sur ce lien : `https://docs.python.org/3/library/math.html`
Quand vous ne comprenez pas comment fonctionnent certaines fonctions issues d'une librairie, pensez à lire la documentation. Toutes les fonctions disponibles dans la librairie en question y sont présentées.

Question 10: (*3 minutes*) À quoi sert la fonction **gcd(a,b)** de la librairie **math** de Python ?

Question 11: (*3 minutes*) En Python et en Java, quelles fonctions utiliseriez-vous pour convertir un angle, dont la valeur est en Radians, en Degrés ?

Conseil

Pensez à lire la documentation des librairies **Math** sous Java et Python.

Question 12: (*3 minutes*) En Python, à quoi sert la fonction **fsum()** et quels sont les paramètres à lui passer ?

Question 13: (*3 minutes*) En Python, est-ce que la librairie **math** contient autre chose que des fonctions ? Si oui, quoi ?

Question 14: (🕒 *10 minutes*) Cherchez sur le net d'autres librairies disponibles pour Python. Citez-en 3 et ajouter le lien vers leur documentation officielle.