

Algorithmes et Pensée Computationnelle

Logiciels système

Le but de cette séance est de comprendre le rôle d'un système d'exploitation, de logiciels système, d'interpréteurs, de compilateurs et des bibliothèques. Les exercices présentés au cours de cette séance se présentent sous deux sections. Une section composée de questions à choix multiples et une section permettant de découvrir et se familiariser avec votre futur environnement de travail.

Le code utilisé dans les questions est également disponible sur Moodle.

1 Systèmes d'exploitation

Le système d'exploitation (OS) 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 les processeurs, 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: (🕒 3 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.

>_ Solution

Microsoft Office 2012 est un ensemble d'applications bureautiques disponible sur les systèmes d'exploitation les plus populaires. Ce n'est donc pas un système d'exploitation, mais un logiciel d'application.

Question 2: (🕒 3 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.

>_ Solution

Le rôle du processeur est d'interpréter les instructions et de les convertir en binaire. Il gère les ressources de l'ordinateur et permet aux différents composants de communiquer entre eux.

Question 3: (🕒 3 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.

>_ Solution

Aucune réponse correcte.

En effet, le système d'exploitation a une interaction avec tous les éléments hardware (tous les composants physique de la structure de l'ordinateur) et avec tous les périphériques de l'ordinateur(tout composant qui n'est pas dans la structure de base de l'ordinateur comme une souris).

Question 4: (🕒 3 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

>_ Solution

MS-DOS est un système d'exploitation développé par Microsoft en 1981 d'abord pour l'IBM PC - modèle 5150 puis pour une plus large gamme d'ordinateurs personnels. Il est mono-utilisateur et ne supporte pas l'exécution de plusieurs programmes à la fois.

Si vous le souhaitez, vous pouvez explorer le code source de MS-DOS sur ce lien (<https://github.com/microsoft/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: (🕒 5 minutes) Que fait la commande suivante : **ls** (*Linux/MacOS*) / **dir** (*Windows*) ?

1. Liste 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.

>_ Solution

ls (*Linux/MacOS*) ou **dir** (*Windows*) permet d'afficher le contenu du répertoire courant.

💡 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: (🕒 5 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.

>_ Solution

L'écriture et la lecture sur les disques.

En effet, cet élément est géré par d'autres parties de l'OS, le filesystem ne gère que l'organisation des fichiers et dossiers.

2 Prise en main de l'environnement de travail

2.1 Installation des outils (🕒 30 minutes)

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=891556>.

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 installer 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`.
- Sous Windows, vous pouvez utiliser n’importe quel éditeur de texte. Notepad est installé par défaut sous Windows.

>_ Solution

Pour exécuter un programme **Python** en ligne de commande, rassurez-vous que votre environnement de travail est correctement installé. Puis, entrez la commande `python` suivi du nom de votre programme. Dans le cas de cette question, on fera : `python hello.py`.

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.

>_ Solution

Pour rajouter des arguments à un programme depuis la ligne de commande, il suffit d’exécuter le programme en suivant les étapes de la question 7 et rajouter les paramètres séparés par des espaces. Pour exécuter notre programme en passant des arguments, on fera : `python question8.py 4 5`. Le programme affichera **La somme de 4 et 5 est 9**.

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 passés.
- `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`.

>_ Solution

Avec le Terminal, rendez-vous dans le dossier où vous avez enregistré le fichier "Question9.java".

Compilez le code à l'aide de la commande `javac Question9.java` (en cas d'erreur voir ci-dessous.)

Pour exécuter notre programme en passant des arguments, on fera : `java Question9 Ceci est mon premier programme écrit en Java`. Le programme affichera :

Paramètre[0] : Ceci

Paramètre[1] : est

Paramètre[2] : mon

Paramètre[3] : premier

Paramètre[4] : programme

Paramètre[5] : écrit

Paramètre[6] : en

Paramètre[7] : Java

Sous Windows, depuis votre terminal, utilisez les commandes suivantes :

- `javac Question9.java` pour compiler votre programme.
- `java Question9 Ceci est mon premier programme écrit en Java` pour l'exécuter.

Erreur fréquente : En utilisant la commande `javac`, vous pouvez obtenir l'erreur suivante :

javac is not recognized as an internal or external command, operable program or batch file

Cette erreur se produit lorsque le chemin d'accès à java n'est pas défini dans votre système. Pour corriger cela, rendez-vous dans le dossier où vous avez installé java et copiez le chemin du fichier "bin".

Par exemple sous Windows, dans le terminal, entrez :

set path="C : \Program Files \Java \jdk1.8.0.121 \bin"

(Attention à bien entrer votre chemin comprenant la version que vous avez installée.)

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`.

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: (🕒 5 minutes) A quoi sert la fonction **gcd(a,b)** de la librairie **math** de Python ?

>_ Solution

Elle sert à calculer le plus grand diviseur commun entre a et b.

Question 11: (🕒 5 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.

>_ Solution

- Il faut utiliser la fonction **degrees(x)** en Python après avoir importé la librairie **math**.
- En Java, il faut utiliser la fonction **toDegrees(x)** de la librairie **java.lang.Math**.

Question 12: (🕒 5 minutes) En Python, à quoi sert la fonction **fsum()** et quels sont les paramètres à lui passer ?

>_ Solution

Elle sert à additionner les éléments d'une liste de façon plus précise que **sum()** sur des opérations à virgule flottante.

Question 13: (🕒 5 minutes) En Python, est-ce que la librairie **math** contient autre chose que des fonctions ? Si oui, quoi ?

>_ Solution

Oui, elle contient également des constantes, on peut citer **pi** par exemple.

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

>_ Solution

On peut citer **Numpy**, **Scikit**, **TensorFlow** qui sont utilisées pour faire du Machine Learning.

On peut aussi citer **Matplotlib** qui permet de créer des graphiques, **Pandas** qui permet de faire de l'analyse de données, **SQLAlchemy** qui permet de gérer des bases de données.