Java System/Tube

TP1

06/03/2021

Exercice 1 : Ecrire un ou des programme(s) pour obtenir les éléments suivants sur l'application et Java

- Le nombre de processeur disponible
- La taille max de la mémoire
- La taille totale de la mémoire
- La taille disponible
- Les variables d'environnement provenant du système
- Les « Properties » du système Java

```java
package tp_reseau;

import java.util.Enumeration;
import java.util.Hashtable;

public class Exercice1 {
    public static void main (String[] args) {
        System.out.println("Properties :");
        Hashtable<Object, Object>props = System.getProperties();
        Enumeration<Object>keys = props.keys();
        while(keys.hasMoreElements()) {
            Object key = keys.nextElement();
            System.out.println(key + ":" + props.get(key));
        }
        System.out.println("Available processors :" + Runtime.getRuntime().availableProcessors());
        System.out.println("Number_of_processors :" + System.getenv("Number_of_processors"));
    }
}
```

Affichage de la console

```
Properties :
sun.desktop:gnome
awt.toolkit:sun.awt.X11.XToolkit
java.specification.version:11
sun.cpu.isalist:
sun.jnu.encoding:UTF-8
java.class.path:/home/tpuser/Bureau/tp_reseau/bin
java.vm.vendor:Ubuntu
sun.arch.data.model:64
java.vendor.url:https://ubuntu.com/
user.timezone:
os.name:Linux
java.vm.specification.version:11
sun.java.launcher:SUN_STANDARD
user.country:FR
sun.boot.library.path:/usr/lib/jvm/java-11-openjdk-amd64/lib
sun.java.command:tp_reseau.Exercice1
jdk.debug:release
sun.cpu.endian:little
user.home:/home/tpuser
user.language:fr
java.specification.vendor:Oracle Corporation
java.version.date:2021-04-20
java.home:/usr/lib/jvm/java-11-openjdk-amd64
file.separator:/
java.vm.compressedOopsMode:32-bit
line.separator:

java.specification.name:Java Platform API Specification
java.vm.specification.vendor:Oracle Corporation
java.awt.graphicsenv:sun.awt.X11GraphicsEnvironment
sun.management.compiler:HotSpot 64-Bit Tiered Compilers
java.runtime.version:11.0.11+9-Ubuntu-0ubuntu2.18.04
user.name:tpuser
path.separator::
os.version:4.15.0-112-generic
java.runtime.name:OpenJDK Runtime Environment
file.encoding:UTF-8
java.vm.name:OpenJDK 64-Bit Server VM
java.vendor.url.bug:https://bugs.launchpad.net/ubuntu/+source/openjdk-lts
java.io.tmpdir:/tmp
```

```
java.version:11.0.11
user.dir:/home/tpuser/Bureau/tp_reseau
os.arch:amd64
java.vm.specification.name:Java Virtual Machine Specification
java.awt.printerjob:sun.print.PSPrinterJob
sun.os.patch.level:unknown
java.library.path:/usr/java/packages/lib:/usr/lib/x86_64-linux-gnu/jni:/lib/x86_64-linux-gnu:/usr/lib/x86_64-linux-gnu:/usr/lib/jni:/lib:/usr/lib
java.vm.info:mixed mode, sharing
java.vendor:Ubuntu
java.vm.version:11.0.11+9-Ubuntu-0ubuntu2.18.04
sun.io.unicode.encoding:UnicodeLittle
java.class.version:55.0
Available processors :2
Number_of_processors :null
```

System.getProperties renvoie toutes les « propriétés d'environnement Java » connues. C'est assez similaire à la variable d'environnement « PATH ».

System.getProperties renvoie toutes les « propriétés système » connues dans un objet similaire à un tableau. Chaque élément de ce tableau est composé d'une clé, qui est le nom de la propriété et l'information associée.

Exercice 2 : Ecrire trois classes pour tester le principe du Tube. Par exemple en écrivant une suite de caractère (ex chaque 1/2 seconde).

Une classe pour écrire

```java
package tp_reseau;

import java.io.IOException;
import java.io.PipedOutputStream;

public class ThreadEcriture extends Thread {

    private PipedOutputStream output;

    // Constructeur
    public ThreadEcriture(PipedOutputStream output) {
        super();
        this.output = output;
    }


    @Override
    public void run() {
        try {
            java.util.Scanner entree =   new java.util.Scanner(System.in);

            System.out.println("Entrez trois mots de votre choix, séparés par un saut de ligne.");
            for (int i = 0; i < 3; i++) {
                String mot = entree.next();
                output.write(mot.getBytes());
                // Sleep de 0.5 secondes
                this.sleep(500);
            }
        } catch (IOException | InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Une classe pour lire

```java
package tp_reseau;

import java.io.IOException;
import java.io.PipedInputStream;

public class ThreadLecture extends Thread{
    private PipedInputStream input;

    // Constructeur
    public ThreadLecture(PipedInputStream input) {
        super();
        this.input = input;
    }

    // Méthode exécutée au start
    @Override
    public void run() {
        try {
            int data = input.read();
            while(data != -1){
                System.out.print((char) data);
                data = input.read();
            }
        } catch (IOException e) {
        }
    }

}
```

Une classe main

```java
package tp_reseau;

import java.io.IOException;
import java.io.PipedInputStream;
import java.io.PipedOutputStream;;

public class ThreadMain {
    public static void main (String[] args) throws IOException {
        final PipedOutputStream output = new PipedOutputStream();
        final PipedInputStream  input  = new PipedInputStream(output);

        // Création des threads
        ThreadEcriture ecriture = new ThreadEcriture(output);
        ThreadLecture lecture = new ThreadLecture(input);

        // Start des threads
        ecriture.start();
        lecture.start();
    }
}
```

Exercice 3 : Ecrire un programme java permettant de lancer une commande Unix comme 'ls -l', d'afficher le résultat de la commande, d'éventuels erreurs, attendre la fin de la commande et récupérer le code de retour. On pourra le faire évoluer en mini-shell.

```java
package tp_reseau;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class CommandeUnix {
    public static void main (String [] args) {

        // Commande d'un objet String contenant la commande
        String commande = "ls -l";

        // Création d'un objet runtime
        Runtime runtime = Runtime.getRuntime();

        // Création d'un nouveau proccessus qui gère la commande
        Process p = null;

        // Gestion des exceptions
        try {
            p = runtime.exec(commande); // Lancement de la commande (exec) et stockage du process lié à cette commande dans p

            // La sortie écrite du process p s'obtient avec p.getInputStream()
            // Pour transformer ça en String on passe par un InputStreamReader et un BufferedReader pour alimenter un StringBuilder
            BufferedReader reader = new BufferedReader(new InputStreamReader(p.getInputStream()));
            StringBuilder builder = new StringBuilder();
            String line = null;
            while ( (line = reader.readLine()) != null) {
               builder.append(line);
               builder.append(System.getProperty("line.separator"));
            }

            String result = builder.toString(); // Récupération de la valeur construite
            System.out.println(result);
        } catch (IOException e) {
            // Gestion du catch
            e.printStackTrace();
            System.out.println("Une erreur est survenue lors de l'éxécution " + commande);
        }
    }
}
```