



Université Sidi Mohamed Ben Abdellah
Faculté des Sciences Dhar Mahraz Fes
Département d'Informatiques



Compte rendue :

**Travaux dirigés de système d'exploitation N°3 :
la lecture et l'écriture des périphériques.**

Réalisé par :

EZZOUHRI Amal

JANATI IDRISSE Meryem

Professeur : M.Meknassi



2014/2015

Introduction : On peut classer généralement les périphériques en deux types : les périphériques d'entrée et les périphériques de sortie. Les périphériques d'entrée servent à fournir des informations (ou données) au système informatique : clavier (frappe de texte), souris (pointage), scanneur (numérisation de documents papier), micro, webcam, etc. Les périphériques de sortie servent à faire sortir des informations du système informatique : écran, imprimante, haut-parleur, etc. On peut également rencontrer des périphériques d'entrée-sortie qui opèrent dans les deux sens : un lecteur de CD-ROM ou une clé USB, par exemple, permettent de stocker des données (sortie). Une autre catégorie peut être ajoutée à ce dernier type, il s'agit des périphériques multifonctions (MFD pour Multi-Functional Device en anglais) comme un caméscope qui fait office d'appareil photo, de webcam, et de disque externe ou encore d'une imprimante qui fait aussi office de scanneur. Alors comment se passe la lecture et l'écriture d'un périphérique sous linux ?

Enoncé : dans ce rapport on va détailler comment faire la lecture et l'écriture d'un périphérique sous unix. Dans un premier temps nous allons parler de façons générale, puis prendre quelques exemples des périphériques.

Généralités :

Pour mieux comprendre comment se fait la lecture et l'écriture d'un périphérique, prenons un média quelconque, une disque, un CD ou encore une clef USB. Deux aspects le concernent peuvent être considérer : le format des données que contient le support, leur organisation, et le lecteur, le matériel, capable d'extraire du monde physique (surface magnétique, mémoire flash, etc.) les dites données.

L'organisation interne des données est appelé un système de fichier (ou FS pour File System). Cela correspond à la manière que l'on a d'organiser les 0 et les 1 de sorte à former des noms dossiers, des noms de fichier, des contenus de fichier.

Les exemples :

1-la carte graphique

Les cartes graphiques sont (quasiment) toujours fonctionnelles sous Ubuntu grâce à des pilotes libres installés et utilisés automatiquement.

Lorsque d'autres choix de pilotes sont disponibles il se fait depuis le gestionnaire de pilotes propriétaires. Ubuntu tend à intégrer de plus en plus de choix de pilote en utilisant ce système qui est chaudement recommandé.

Toutefois les autres méthodes d'installation de pilotes sont toujours possibles, et parfois encore nécessaires. Elles sont détaillées dans les documentations ci-dessous qui sont classées par marque (après les documentations pouvant s'appliquer toutes marques confondues).

Identifier sa carte graphique

Pour connaître les caractéristiques de la carte graphique, on peut utiliser depuis un terminal l'une des commandes suivantes :

- `lspci | grep "VGA compatible controller"`

- `lshw -enable pci -class display`

On reçoit par exemple :

```
00:02.0 VGA compatible controller [0300]: Intel Corporation 2nd Generation Core
Processor Family Integrated Graphics Controller [8086:0116] (rev 09) (prog-if 00
[VGA controller])
```

On peut pour connaître le pilote utilisé, en consultant le log le plus récent de Xorg:

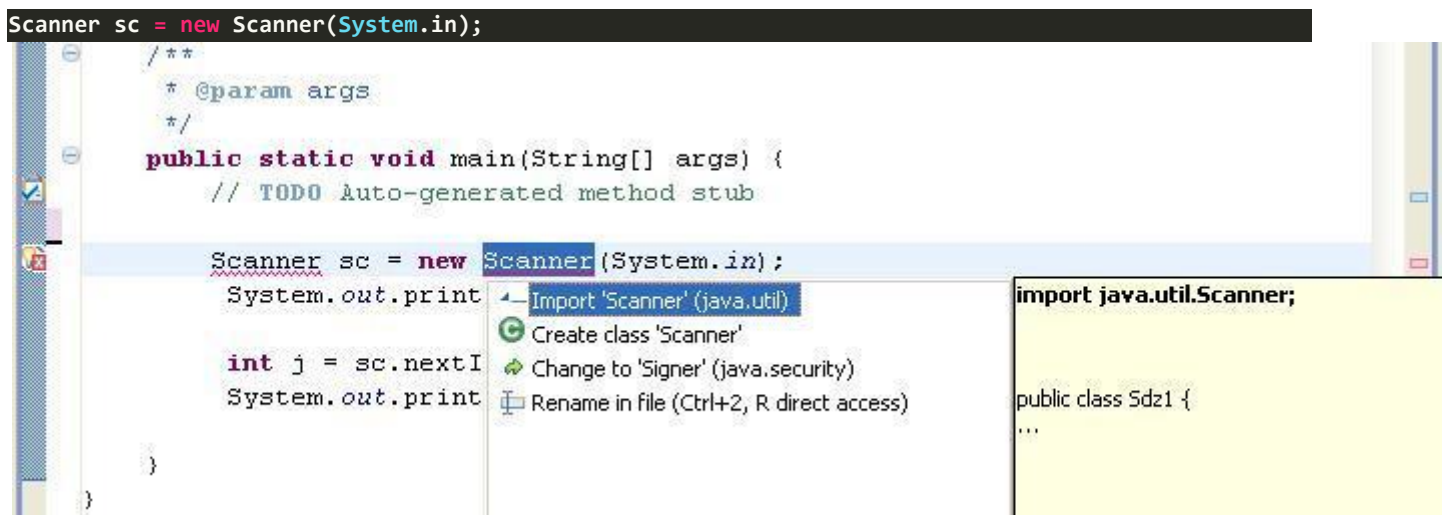
```
grep /drivers/ /var/log/Xorg.0.log
```

qui renvoie par exemple:

```
(II) Loading /usr/lib/xorg/modules/drivers//radeonhd_drv.so
```

2-les entrées clavier:

Les variables de type **String** sont en réalité des objets de type **String**. Pour que Java puisse lire ce qu'on tape au clavier, nous allons devoir utiliser un objet de type **Scanner**. Cet objet peut prendre différents paramètres, mais ici nous n'en utiliserons qu'un : celui qui correspond à l'entrée standard en Java. Lorsqu'on fait `System.out.println()`; nous appliquons la méthode `println()` sur la sortie standard ; ici, nous allons utiliser l'entrée standard `System.in`. Donc, avant d'indiquer à Java qu'il faut lire ce que nous allons taper au clavier, nous devons instancier un objet **Scanner**.



Importer la classe Scanner :

`import java.util.Scanner;`

Voilà ce que nous avons fait. Il fallait indiquer à Java où se trouve la classe **Scanner**. Pour faire ceci, nous devons importer la classe **Scanner** grâce à l'instruction **import**. La classe que nous voulons se trouve dans le package **java.util**.

Un package est un ensemble de classes. En fait, c'est un ensemble de dossiers et de sous-dossiers contenant une ou plusieurs classes.

Les classes qui se trouvent dans les packages autres que **java.lang** (package automatiquement importé par Java, on y trouve entre autres la classe **System**) sont à importer à la main dans vos

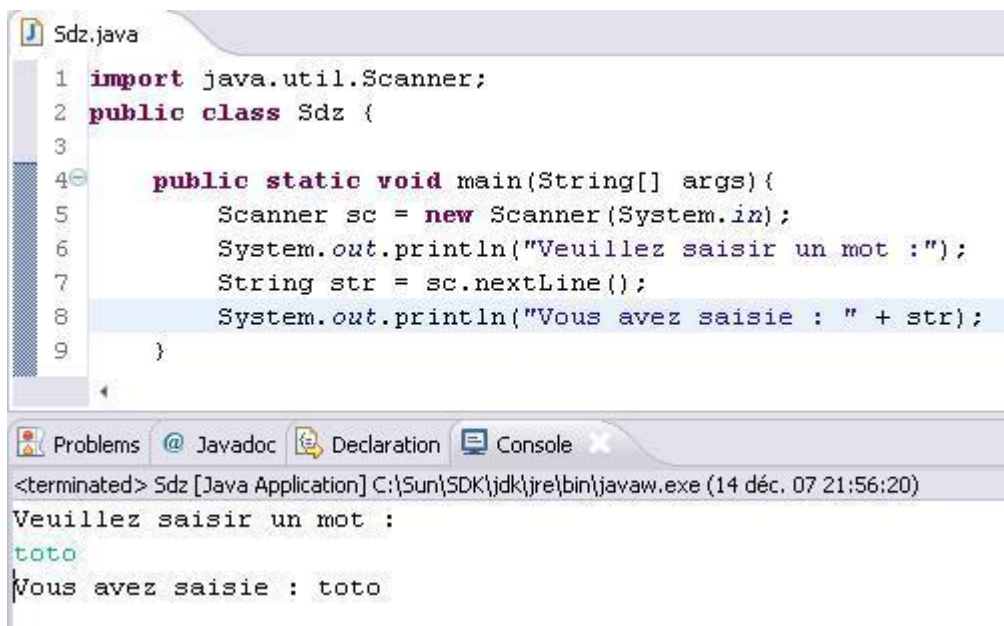
classes Java pour pouvoir vous en servir. La façon dont nous avons importé la classe `java.util.Scanner` dans Eclipse est très commode. On peut aussi le faire manuellement :

```
//Ceci importe la classe Scanner du package java.util
import java.util.Scanner;
//Ceci importe toutes les classes du package java.util
import java.util.*;
```

Voici l'instruction pour permettre à Java de récupérer ce que vous avez saisi pour ensuite l'afficher :

```
Scanner sc = new Scanner(System.in);
System.out.println("Veuillez saisir un mot :");
String str = sc.nextLine();
System.out.println("Vous avez saisi : " + str);
```

Une fois l'application lancée, le message qu'on a écrit auparavant s'affiche dans la console, en bas d'Eclipse.



The screenshot shows the Eclipse IDE with a file named `Sdz.java` open. The code in the editor is as follows:

```
1 import java.util.Scanner;
2 public class Sdz {
3
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         System.out.println("Veuillez saisir un mot :");
7         String str = sc.nextLine();
8         System.out.println("Vous avez saisi : " + str);
9     }
}
```

Below the editor, the `Console` tab is active, showing the following output:

```
<terminated> Sdz [Java Application] C:\Sun\SDK\jdk\jre\bin\javaw.exe (14 déc. 07 21:56:20)
Veuillez saisir un mot :
toto
Vous avez saisi : toto
```

Saisie

Si on remplace la ligne de code qui récupère une chaîne de caractères comme suit :

```
Scanner sc = new Scanner(System.in);
System.out.println("Veuillez saisir un nombre :");
int str = sc.nextInt();
System.out.println("Vous avez saisi le nombre : " + str);
```

On constate que lorsqu'on introduit notre variable de type `Scanner` et qu'on introduit le point permettant d'appeler des méthodes de l'objet, Eclipse vous propose une liste de méthodes associées à cet objet (ceci s'appelle l'auto complétion) ; de plus, lorsqu'on commence à taper le début de la méthode `nextInt()`, le choix se restreint jusqu'à ne laisser que cette seule méthode.

Pour lire un `int`, on doit utiliser `nextInt()`. De façon générale, pour récupérer un type de variable, il suffit d'appeler `next<Type de variable commençant par une majuscule>`.

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
double d = sc.nextDouble();
long l = sc.nextLong();
byte b = sc.nextByte();
//Etc.
```

Il y a un type de variables primitives qui n'est pas pris en compte par la classe **Scanner** : il s'agit du type **char**.

Voici comment on pourrait récupérer un caractère :

```
System.out.println("Saisissez une lettre :");
Scanner sc = new Scanner(System.in);
String str = sc.nextLine();
char caract = str.charAt(0);
System.out.println("Vous avez saisi le caractère : " + caract);
```

Nous avons récupéré une chaîne de caractères, puis utilisé une méthode de l'objet **String** (ici, **charAt(0)**) afin de récupérer le premier caractère saisi. Même si on tape une longue chaîne de caractères, l'instruction **charAt(0)** ne renverra que le premier caractère.

Une précision s'impose, toutefois : la méthode **nextLine()** récupère le contenu de toute la ligne saisie et replace la « tête de lecture » au début d'une autre ligne. Par contre, si on invoque une méthode comme **nextInt()**, **nextDouble()** et qu'on invoque directement après la méthode **nextLine()**, celle-ci ne nous invitera pas à saisir une chaîne de caractères : elle videra la ligne commencée par les autres instructions. En effet, celles-ci ne repositionnent pas la tête de lecture, l'instruction **nextLine()** le fait à leur place. Pour faire simple, ceci :

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Saisissez un entier : ");
        int i = sc.nextInt();
        System.out.println("Saisissez une chaîne : ");
        String str = sc.nextLine();
        System.out.println("FIN ! ");
    }
}
```

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Saisissez un entier : ");
        int i = sc.nextInt();
        System.out.println("Saisissez une chaîne : ");
        //On vide la ligne avant d'en lire une autre
        sc.nextLine();
        String str = sc.nextLine();
        System.out.println("FIN ! ");
    }
}
```

En résumé

- La lecture des entrées clavier se fait via l'objet **Scanner**.
- Ce dernier se trouve dans le package **java.util** que vous devrez importer.
- Pour pouvoir récupérer ce nous allons taper dans la console, nous devons initialiser l'objet **Scanner** avec l'entrée standard, **System.in**.

- *Il y a une méthode de récupération de données pour chaque type (sauf les char)
: `nextLine()` pour les `String`, `nextInt()` pour les `int` ...*