

Lectura de fitxers binaris d'accés directe en java

Recordem que havíem classificat els fitxers en funció dels seu mètode d'accés com:

Fitxers seqüencials

Son aquells que per accedir a una determinada posició del fitxer, hem de travessar totes les anteriors. Podem assimilar aquest tipus de fitxers a les antigues cintes de cassette, on per escoltar una determinada cançó, hem de travessar totes les que trobem abans d'aquesta. Aquest mode d'accés, que podem utilitzar tant en fitxers de **text** com **binaris**, és útil quan volem per exemple traslladar tot el contingut del fitxer des del disc a la memòria.

Fitxers d'accés directe

Son els que podem accedir a qualsevol contingut del fitxer a partir de la seva posició, de forma semblant a com faríem amb un vector. Aquest mode d'accés és útil sobre tot en fitxers binaris, ja que en fitxers de text, degut a la codificació diferent dels caràcters no anglesos, pot resultar complicat accedir a un caràcter determinat dintre del fitxer.

Els fitxers amb els que hem treballat fins ara, fitxers de text, son d'accés seqüencial, es a dir, que llegim el fitxer des del començament i l'escrivim de la mateixa forma. Un cas particular és quan escrivim al final d'un fitxer obert en mode **append**, on l'accés és directe, però només al final del fitxer, i a partir d'allà ja és seqüencial.

L'accés directe o aleatori a un fitxer de text no té gaire sentit, ja que les línies del fitxer, tindran una o altra llargària depenent de la codificació d'aquest i també del sistema operatiu utilitzat (en particular, el final de línia pot ser un o dos caràcters com ja s'ha comentat). Per aquest motiu, és de poca utilitat un accés directe en aquests tipus de fitxers.

En els fitxers binaris tanmateix, un accés directe o aleatori a una determinada posició, ens pot ser útil, ja que aquests fitxers no canvien amb el sistema operatiu utilitzat ni tenen codificacions diferents. Tot i el que s'ha comentat referent als fitxers de text, un fitxer de text és un cas particular dels fitxers binaris, i pot ser obert com si d'un fitxer binari es tractés, fent servir amb ell qualsevol de les instruccions que es comentaran a continuació.

Els fitxers d'accés directe o aleatori

Els fitxers d'accés directe es representen en **java** per la classe **RandomAccessFile** del paquet **java.io**. Els mètodes d'aquesta classe ens permetran obrir un arxiu només per llegir (mode "**r**") o be de forma que puguem tant llegir com escriure en ell (mode "**rw**" d'altres llenguatges com **C** o **Python**).

En resum, la forma de procedir per l'accés a fitxers amb aquesta classe podria ser:

1. Ens creem un objecte **File** per saber si el fitxer existeix i altres gestions, com per exemple determinar la grandària. Als fitxers binaris, això esdevé fonamental, ja que com vam comentar, no disposen d'un caràcter de finalització específic de l'estil **<EOF>** (**end of file**). Per exemple: **File fitxer = new File("C:/Java/Exemple.java");**. Això ho farem també en un context de tractament dels errors que es puguin produir (dintre d'un **try**).
2. Podem a partir d'aquest objecte de la classe **File** (**fitxer** a l'exemple) determinar si existeix **fitxer.exists()**, es tracta d'un arxiu **fitxer.isFile()** (o be d'un directori **fitxer.isDirectory()**), si es pot llegir (**fitxer.canRead()**), si es pot escriure (**fitxer.canWrite()**), etc.
3. Ens podem apuntar la longitud del fitxer, ja que no podem llegir o posicionar l'apuntador al fitxer més enllà del final del mateix: **long bytesFitxer = fitxer.length();**
4. Obrim el fitxer, ja sigui per lectura o per lectura-escriptura:
 - a. Per lectura: **RandomAccessFile file = new RandomAccessFile(fitxer,"r");**
 - b. Per escriptura: **RandomAccessFile file = new RandomAccessFile(fitxer,"rw");**
5. Fem servir els mètodes de la classe per llegir o escriure en el fitxer
6. Saber on es troba l'apuntador dintre del fitxer (posició on escriurem o llegirem del fitxer): **posició = getFilePointer()** o posar-lo al lloc que ens convingui: **file.seek(posicio);**

Més informació als enllaços:

<https://docs.oracle.com/javase/8/docs/api/java/io/RandomAccessFile.html>

<https://docs.oracle.com/javase/8/docs/api/java/io/RandomAccessFile.html#mode>

Lectura de fitxers binaris en mode cru

Els fitxers binaris d'accés directe, es poden llegir i escriure en mode **cru** (**raw** en anglès), es a dir, podem llegir o escriure **bytes** individuals al fitxer (i també vectors de bytes). Això fa que tinguem un control absolut sobre el contingut del fitxer.

Els mètodes de la classe també ens permeten llegir i escriure qualsevol dels tipus primitius del llenguatge **java** al fitxer, ja que la classe sap com convertir aquests tipus a bytes, seguint el mecanisme de representació interna. Fins i tot podem escriure cadenes al fitxer.

Els fitxers binaris creats amb la classe **RandomAccessFile** tindran un sistema de representació que moltes vegades serà difícil de conèixer, i si be tot funcionarà si fem servir les instruccions adequades per llegir i escriure informació al fitxer (les complementàries), aquesta informació no podrà ser consultada des de fora del nostre programa (en un fitxer de text, podem saber si s'ha escrit be simplement obrint el fitxer amb una altra aplicació).

Exemple de programa de lectura seqüencial d'un fitxer binari

Un programa força útil que podem programar en **java** per llegir fitxers binaris és **Dump.java**, que podem utilitzar per representar el contingut d'un fitxer a la pantalla, tant en format **hexadecimal** com en mode **text** (molt útil pels fitxers de text, es a dir, aquells que tinguin un contingut que nosaltres puguem interpretar).

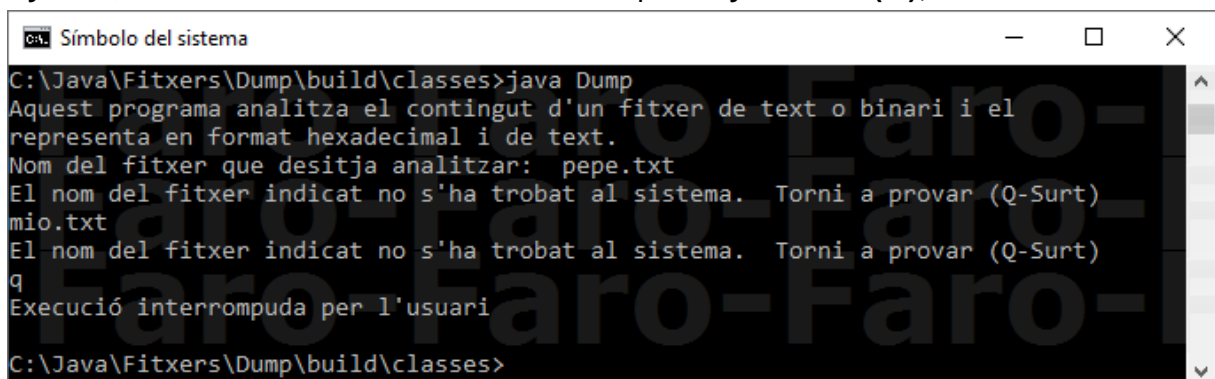
Aquests programes, visualitzen la informació dels fitxers en pàgines. Cada pàgina conté **256 bytes** o caràcters (llegirem el fitxer en blocs de **256 bytes**) que es representen normalment amb **16** fileres de **16** caràcters cadascuna (**16 x 16 = 256**).

Farem un programa que admetrà com a únic paràmetre el nom d'un fitxer (es pot incloure la trajectòria completa fins arribar a l'arxiu), i la sortida del qual serà el contingut del fitxer, representat en hexadecimal i text. La crida al programa podria ser:

java Dump C:\Java\Fitxers\Dump\build\classes\Dump.py

On a més del nom del programa (**Dump.class**), hem indicat quin és el fitxer (és codi font en **python**) que volem analitzar. Si us fixeu, no és necessari duplicar les barres '****' quan aquestes son llegides des del teclat o be des d'un fitxer, però sí quan les escrivim en una cadena al nostre programa.

Si no passem cap paràmetre al programa (o passem 2 o més paràmetres, o el nom del fitxer introduït no es correspon amb un fitxer vàlid al sistema), aquest demanarà el nom de l'arxiu que es vol analitzar, fins que la cadena introduïda, correspongui amb un nom de fitxer que existeixi al sistema o be l'usuari indiqui la lletra **q** (**quit**), que interromprà el procés. Recordeu que per sortir d'un programa en **java**, podem fer servir el mètode **exit** de la classe **System**, i hem de retornar el codi d'error. Exemple: **System.exit(-1);**



L'aspecte de la nostra aplicació podria ser el que s'indica:

```

Símbolo del sistema - java Dump Dump.py
Pàgina 1
Direcció          Contingut en hexadecimal          Text ASCII
00000000000000  23 20 2D 2A 2D 20 63 6F 64 69 6E 67 3A 20 75 74  # -*- coding: ut
00000000000016  66 2D 38 20 2D 2A 2D 20 0A 69 6D 70 6F 72 74 20  f-8 -*- .import
00000000000032  72 61 6E 64 6F 6D 0A 69 6D 70 6F 72 74 20 6F 73  random.import os
00000000000048  2E 70 61 74 68 20 0A 69 6D 70 6F 72 74 20 73 79  .path .import sy
00000000000064  73 0A 66 72 6F 6D 20 73 75 62 70 72 6F 63 65 73  s.from subprocess
00000000000080  73 20 69 6D 70 6F 72 74 20 63 61 6C 6C 0A 0A 23  s import call..#
00000000000096  20 43 61 6E 76 69 61 72 20 65 6C 20 63 6F 6C 6F  Canviar el colo
00000000000112  72 20 64 65 20 74 65 78 74 20 69 20 66 6F 6E 73  r de text i fons
00000000000128  20 61 20 6C 61 20 63 6F 6E 73 6F 6C 61 0A 23 20  a la consola.#
00000000000144  68 74 74 70 3A 2F 2F 63 72 79 73 6F 6C 2E 6F 72  http://crysol.or
00000000000160  67 2F 65 73 2F 6E 6F 64 65 2F 31 30 31 30 0A 0A  g/es/node/1010..
00000000000176  23 20 49 6D 70 6F 72 74 65 6D 20 65 6C 20 6E 6F  # Importem el no
00000000000192  73 74 72 65 20 6D C3 B2 64 75 6C 20 64 65 20 6C  stre m..dul de l
00000000000208  65 63 74 75 72 61 20 64 65 20 74 65 63 6C 61 74  ectura de teclat
00000000000224  2E 20 20 54 72 6F 62 61 74 20 61 3A 20 20 68 74  . Trobat a: ht
00000000000240  74 70 3A 2F 2F 68 6F 6D 65 2E 77 6C 75 2E 65 64  tp://home.wlu.ed

S-Següent  Q-Sortir  nnnn-Pàgina inicial a visualitzar : 19

```

A la part inferior, mostrarem un menú, que ens permet navegar per les pàgines del fitxer. Amb **A-Anterior** anem a la pàgina anterior del fitxer (prèvia a la que s'està visualitzant), no apareix ara perquè estem a la primera pàgina. La idea és que al menú només apareguin les opcions possibles en cada moment. En aquest cas **S-Següent**, que visualitzarà la següent pàgina (a l'exemple la **2**), **Q-Sortir**, termina l'execució del programa. També podem indicar al programa la pàgina que volem visualitzar (com s'indica a l'exemple, on hem demanat de veure la pàgina **19**, que és l'última del fitxer).

```

Símbolo del sistema - java Dump Dump.py
Pàgina 19
Direcció          Contingut en hexadecimal          Text ASCII
00000000004608  09 09 09 09 70 61 73 20 2B 3D 20 31 0A 09 09 09  ....pas += 1....
00000000004624  09 0A 09 23 20 53 75 70 70 6F 72 74 20 6E 6F 72  ...# Support nor
00000000004640  6D 61 6C 2D 74 65 72 6D 69 6E 61 6C 20 72 65 73  mal-terminal res
00000000004656  65 74 20 61 74 20 65 78 69 74 0A 09 61 74 65 78  et at exit..atex
00000000004672  69 74 2E 72 65 67 69 73 74 65 72 28 73 65 6C 66  it.register(self
00000000004688  2E 73 65 74 5F 6E 6F 72 6D 61 6C 5F 74 65 72 6D  .set_normal_term
00000000004704  29 20 0A 09 09 0A 0A 0A 0A  ....

A-Anterior  Q-Sortir  nnnn-Pàgina inicial a visualitzar :

```

En aquesta pàgina apreciem:

- A la part superior indiquem el número de pàgina que estem visualitzant. La primera pàgina la numerarem com a **1**, però internament, és convenient que sigui la **0**.
- Com que el fitxer té una longitud determinada, que molt probablement no serà múltiple de **256**, l'última pàgina del fitxer apareixerà normalment incompleta (com passa a l'exemple amb el fitxer visualitzat).
- No apareix l'opció **S-Següent**, ja que estem a l'última pàgina del fitxer.
- És un fitxer **Linux**, ja que els retorns de carro es codifiquen com **0A = 10 = NL** (New Line).

Com a complement, pels que vulgueu aventurar-vos a programar aquesta aplicació, us poso el codi de la funció més interessant, la que escriu la pàgina del fitxer a la consola. També s'ha indicat la funció auxiliar que és cridada des d'aquesta :

Funció que escriu la cadena que li passem centrada en un camp de la longitud que l'indiquem com a segon paràmetre, i precedida d'espais. La funció que escriu el text de la pàgina a la pantalla la fa servir:

```
private static String centraCadena(String s, int numEspais){
    String cad = "";
    for (int n=0; n<(numEspais-s.length())/2; n++){
        cad += ' ';
    }
    return cad + s;
}
```

La funció que escriu el text de la pàgina podria ser programada com:

```
private static void escriuPagina(byte [] p, long pag, int numCar){
    // Li passem una pàgina (256 bytes) i la escriu en hexadecimal i text
    // pag és el número de la pàgina, i numCar el nombre de bytes a la pàgina
    System.out.println(centraCadena("Pàgina " + (pag+1),80));
    System.out.println(" Direcció Contingut en hexadecimal"
        + " Text ASCII");
    for (int n=0; n<16; n++){
        if (n*16 < numCar)
            System.out.print(String.format("%013d ", pag*256+n*16));
        else
            System.out.print(" ");
        for (int m=0; m<16; m++){
            if (n*16+m < numCar)
                System.out.print(String.format("%02X ", p[n*16+m]));
            else
                System.out.print(" ");
            System.out.print(" ");
            for (int m=0; m<16; m++){
                if (n*16+m < numCar)
                    if (p[n*16+m] >= 32)
                        System.out.print((char) p[n*16+m]);
                    else
                        System.out.print(".");
                else
                    System.out.print(" ");
            }
            System.out.println(); // Alimentem línia
        }
    }
}
```

Fixeu-vos que passem a la funció el número de caràcters de la pàgina perquè pugui visualitzar-la incompleta si és el cas. En aquest sentit, si **p** és un vector de **256** bytes que al nostre programa emmagatzema la pàgina llegida del fitxer, el següent tros de codi representa com llegir i visualitzar aquesta pàgina. La crida a la funció de lectura retorna el nombre de bytes llegits realment del fitxer, que seran els que haurem de passar a la funció **escriuPagina** i que seran els que la funció visualitzarà a l'última pàgina de l'arxiu.

```
llegits = file.read(p); // Llegim els següents 256 bytes
escriuPagina(p,pagina,llegits);
```

Funció que no es fa servir al programa, només s'utilitza per provar l'anterior. Poseu al mètode **main()**:

```
byte [] p = creaVector();
escriuPagina(p,12,256);
```

```
private static byte[] creaVector(){
    // Crea un vector de 256 caràcters aleatoris (0-255)
    byte [] v = new byte[256];
    for (int n=0; n<256; n++){
        v[n] = (byte) (Math.random()*256);
    }
    return v;
}
```