2 – Files and their Use

Week 8 Presentation 2

Memory

- Working in main memory is nice but everything goes when the computer is switched off (or crashes)

- Need to persist program results somewhere

"save" / "keep"

- That is the idea of persistence: "somewhere" can be many different places (e.g. file, remote computer, etc, etc)

So we have to work as much as we can in memory, and only in memory, for speed ...

Memory

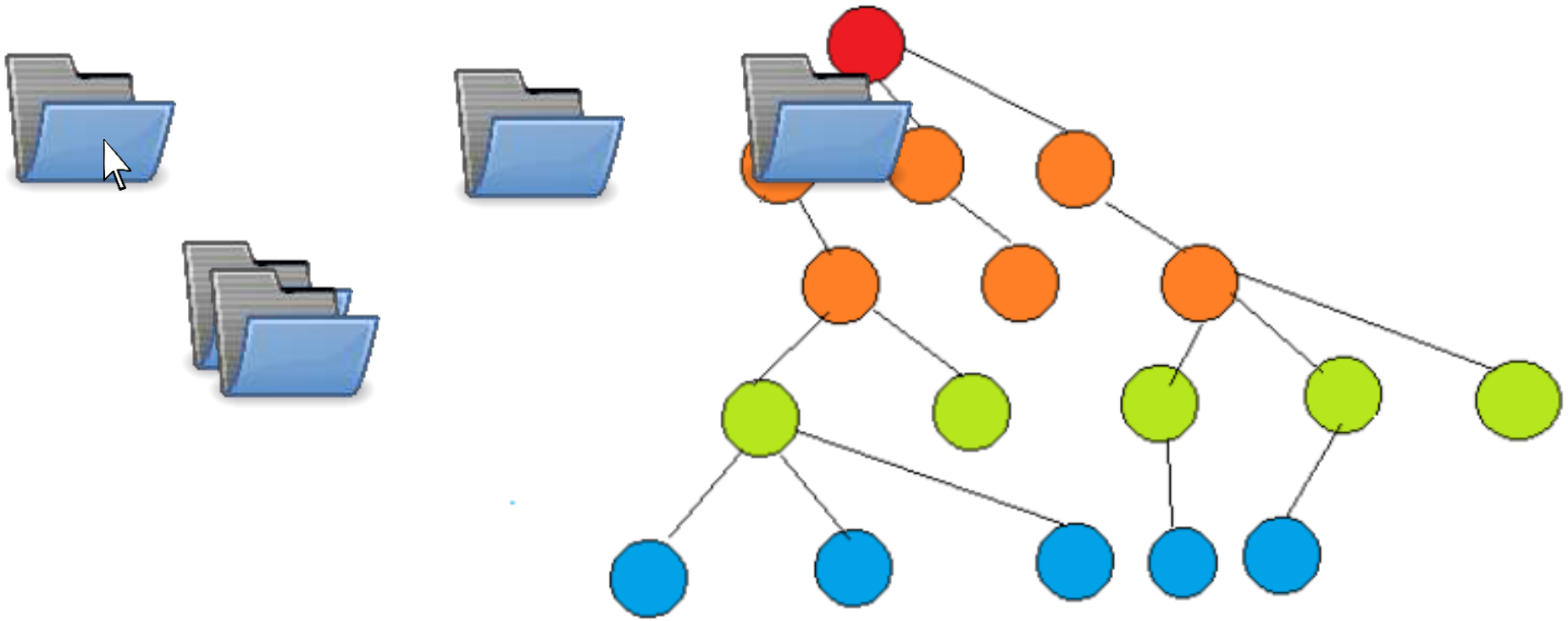
Mostly work in memory for speed.

... while we still need a safety net.

But write to files for safety.

File System

Directories/folders



Stream Redirection

Instead of the keyboard

$ java MyProgram < input_file

$ java MyProgram > output_file

Instead of the screen

One very easy way to save data and to restore it is, instead of doing it in Java, to let the system do it through what is known as "Stream redirection". Your program may read from what it thinks is the keyboard when in reality input comes from a file, and what is written to the screen can also be redirected to a file. System.out and System.err can be separated.

Types of Files

- Traditionally, there are mainly two types of files

- Binary files

- Text files

Text files only contain printable characters

But if there are many different examples of file extensions and many types of files, **and** they all fall in one of two categories: text or binary - then the only problem is, whatever they are, they are all made of 0s and 1s. So really it's all a matter of interpretation.

HOW TO INTERPRET THE 0S AND 1S

Interpretation is the big, hard question. You cannot guess the meaning of 1s and 0s just by looking at them. You must have an idea already. And even with text, there are many different ways to encode one single character (and don't believe that the problem doesn't exist even with basic Latin letters – there is another encoding system than ASCII called EBCDIC and the bits meaning 'a' in ASCII mean '/' in EBCDIC). If you haven't the key allowing you to decrypt the bits, you are lost.

Types of Files

- Traditionally, there are mainly two types of files

- Binary files

- Text files

So the true definition of a "text file" is that it only contains characters that you can print (including spaces and carriage returns) when you decrypt the file as a bunch of characters.

only printable characters WHEN DECODING AS CHARACTERS

Text Files

There are many types of text files: not only files with the extension .txt!

They can be opened with a ”text editor”
e.g. Notepad in windows, more in linux

Code - .c, .h, .py, .php, .java, .bat, .sh, …
Plain text - .txt, .ini
Text with readable tags - .html, .rtf, .xml
Data as text - .csv

All these only contain printable characters

Binary Files

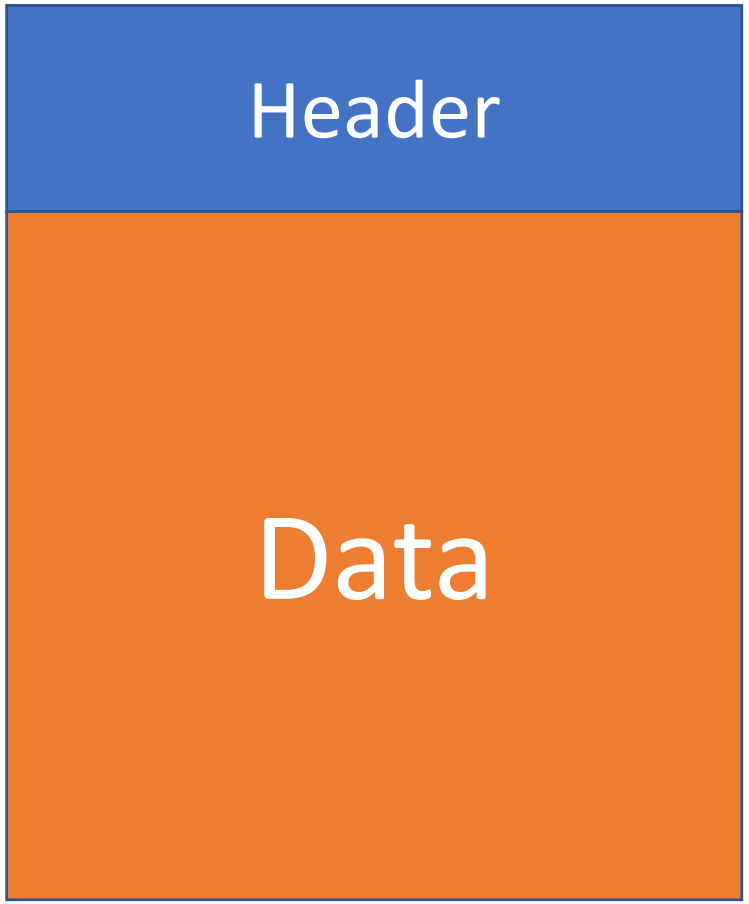
There are also many types of binary files… Including used by text applications…

They can only be opened with a **special** program!

Compiled program - .o, .exe. class
Archives, compressed files - .tar, .tgz, .zip
Encrypted files
Text with non readable formatting - .docx, .pdf
Multimedia - .gif, .mp4, .png, .flv, …

Binary Files

Most often a binary file contains a header part that describes the structure, followed by data proper.



Attributes

Bunch of 0s and 1s

Binary Files

Memory structures written ”as is”
More compact (no encodings etc)
No conversion during I/O
May be portability problems between computers (windows - mac etc)

One issue is the “small endian”/ ”big endian” issue which is a hardware spec.
The 4 bits that make up ½ a byte might be swapped

Stream Redirection

Text Files

$ java MyProgram < input_file

$ java MyProgram > output_file

Although nothing forbids writing to and from binary files stream redirection usually uses text files

ToString()

```
public static void main(String args[]){
        int i=200;
        String s=String.valueOf(i);
        System.out.println(i+100);
        System.out.println(s+100);
}
```

Output:

300

200100

Key Point: whether you are calling println with an integer, whether you are explicitly calling toString() or not, a conversion occurs.

```
Integer.toString(int_val);
```

binary internal computer representation

To digits (characters)

ToString()

A number has to be turned into a string of digits for output.

8 bit word

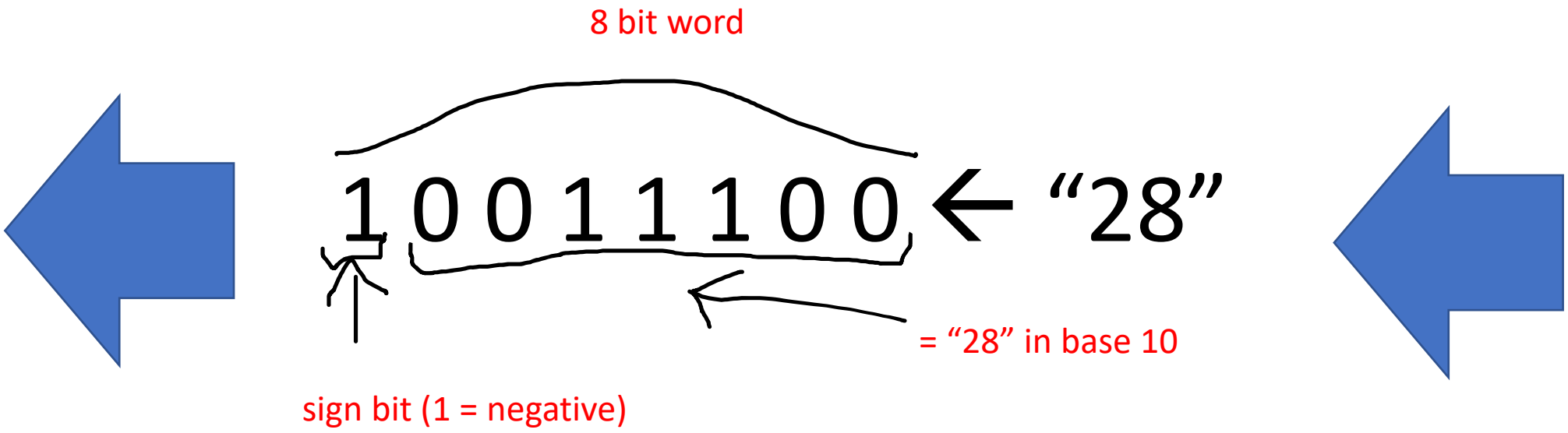
1 0 0 1 1 1 0 0 → “28”

= “28” in base 10

sign bit (1 = negative)

```
if number is negative display '-'
loop on decreasing powers of 10
        get the result r of the integer division of the number by the power of 10
        if we have already displayed a non zero digit
                display the digit corresponding to the code of '0' plus r
         else the digit is not zero
                record that we have found a non zero digit
                display the digit corresponding to the code of '0' plus r decrease the number by r times the power of 10
                processed
end loop
```

Input requires the opposite



Integer.parseInt() or the method nextInt() of a Scanner object perform the reverse operation

HOW TO INTERPRET THE 0S AND 1S

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello!");
    }
}
```

A program has, to "understand a file", a number of options.

How to interpret the 0s and 1s

A program has, to "understand a file", a number of options.

1. Assume that it is what we expect…

For instance text.

This is the simplest option: ”when the only tool is a hammer everything looks like a nail!”

```
As-MacBook-Pro:code ag$ javac Hello.java
As-MacBook-Pro:code ag$ ls
Hello.class     Hello.java
As-MacBook-Pro:code ag$ cat Hello.class
????:

java/lang/Object<init>()V



java/lang/SystemoutLjava/io/PrintStreamHello!

java/io/PrintStreamprintln(Ljava/lang/String;)VHelloCodeLineNumberTablemain([Lja
va/lang/String;)V
SourceFile
??llo.java!*??  %       ?
As-MacBook-Pro:code ag$
```

cat writes everything to the screen – here the result looks like garbage!

HOW TO INTERPRET THE 0S AND 1S

A program has, to "understand a file", a number of options.

2. Assume that the extension is correct

Hello.class is a java bytecode file

I renamed Hello.class to Hello.c and tried to compile it. I got 105 warnings and 11 errors but the compiler tried

HOW TO INTERPRET THE 0S AND 1S

A program has, to "understand a file", a number of options.

3. Check the header for a “magic number”

Binary files usually contain a "signature" in their header, a small number of bytes that are very specific to one type of files. You don't need to trust the extension.

- hexdump just dumps the file contents
- All class files start with the same bytes.

cafe babe

```
$ hexdump -n 1024 Hello.class
0000000 ca fe ba be 00 00 00 3a 00 1d 0a 00 02 00 03 07
0000010 00 04 0c 00 05 00 06 01 00 10 6a 61 76 61 2f 6c
0000020 61 6e 67 2f 4f 62 6a 65 63 74 01 00 06 3c 69 6e
0000030 69 74 3e 01 00 03 28 29 56 09 00 08 00 09 07 00
0000040 0a 0c 00 0b 00 0c 01 00 10 6a 61 76 61 2f 6c 61
0000050 6e 67 2f 53 79 73 74 65 6d 01 00 03 6f 75 74 01
0000060 00 15 4c 6a 61 76 61 2f 69 6f 2f 50 72 69 6e 74
0000070 53 74 72 65 61 6d 3b 08 00 0e 01 00 06 48 65 6c
0000080 6c 6f 21 0a 00 10 00 11 07 00 12 0c 00 13 00 14
0000090 01 00 13 6a 61 76 61 2f 69 6f 2f 50 72 69 6e 74
00000a0 53 74 72 65 61 6d 01 00 07 70 72 69 6e 74 6c 6e
00000b0 01 00 15 28 4c 6a 61 76 61 2f 6c 61 6e 67 2f 53
00000c0 74 72 69 6e 67 3b 29 56 07 00 16 01 00 05 48 65
00000d0 6c 6c 6f 01 00 04 43 6f 64 65 01 00 0f 4c 69 6e
00000e0 65 4e 75 6d 62 65 72 54 61 62 6c 65 01 00 04 6d
00000f0 61 69 6e 01 00 16 28 5b 4c 6a 61 76 61 2f 6c 61
0000100 6e 67 2f 53 74 72 69 6e 67 3b 29 56 01 00 0a 53
0000110 6f 75 72 63 65 46 69 6c 65 01 00 0a 48 65 6c 6c
0000120 6f 2e 6a 61 76 61 00 21 00 15 00 02 00 00 00 00
0000130 00 02 00 01 00 05 00 06 00 01 00 17 00 00 00 1d
0000140 00 01 00 01 00 00 00 05 2a b7 00 01 b1 00 00 00
0000150 01 00 18 00 00 00 06 00 01 00 00 00 02 00 09 00
0000160 19 00 1a 00 01 00 17 00 00 00 25 00 02 00 01 00
0000170 00 00 09 b2 00 07 12 0d b6 00 0f b1 00 00 00 01
0000180 00 18 00 00 00 0a 00 02 00 00 00 04 00 08 00 05
0000190 00 01 00 1b 00 00 00 02 00 1c
```

Next: Streams