R·I·T

Files and Streams

File Directories

File Directory

A set of files and other (sub)directories

Principle function: help people find their way around data on a system

Implementation

Directories are stored as additional files

OS maintains a *file descriptor* for each file and directory in the file system (e.g. on disc)

Absolute and Relative Paths

Absolute Path

Path from the root (top) directory in a directory tree to the desired directory/file

- e.g. “/home/zanibbi/comp232/slides.ppt”
- e.g. “D:\myfiles\zanibbi\comp232\slides.ppt”

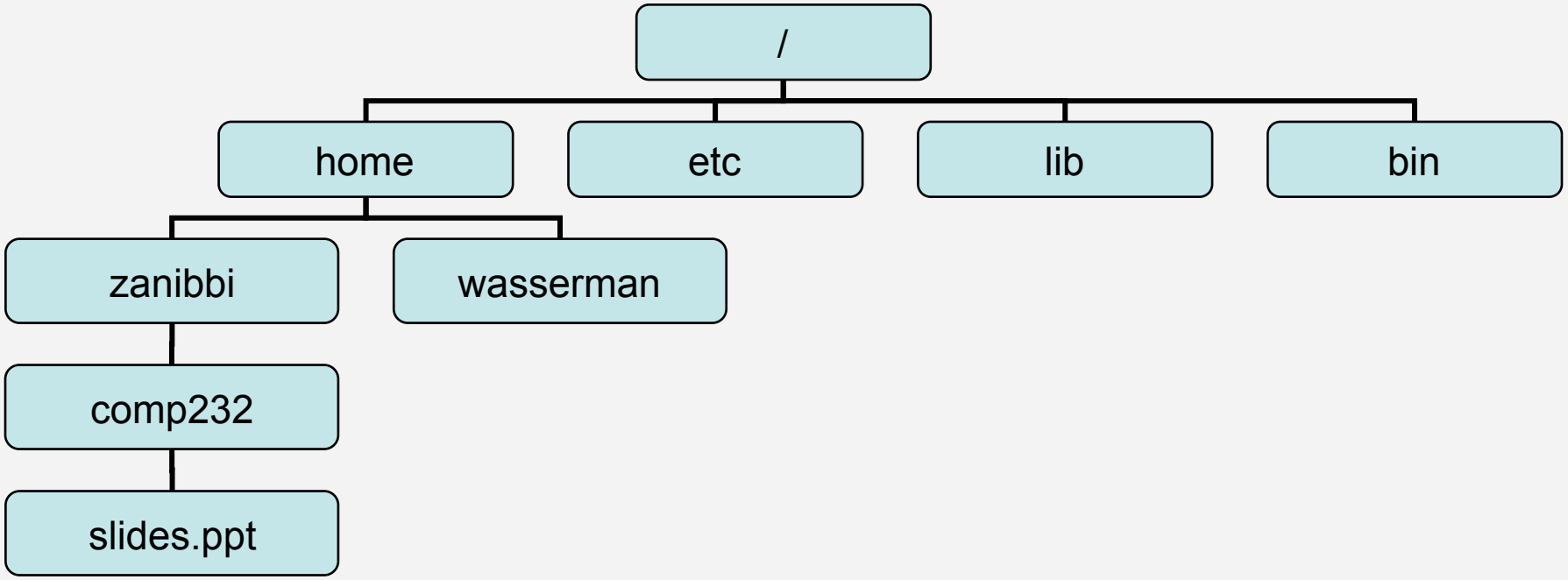
Relative Path

Path from (“relative to”) a given directory

- (usually current)
- e.g. : “comp232/slides.ppt” (from /home/zanibbi)
- e.g. : “comp232\slides.ppt” (from D:\myfiles\zanibbi)

Example: Unix File Directory

Root Directory: “/” (e.g. “cd /”)



File Class in Java

Purpose

- Represents file attributes, as maintained by the operating system in *file descriptors*
- May be used to rename, delete files
- Directories also can be represented by File objects
- The File class is not used to read and write file data

Path Separators ("\\" and "/")

- Because "\" is an escape character in Java Strings, directory separators for Windows must be indicated using "\\"
- e.g. "C:\\book\\Welcome.java"
- When giving relative paths in Java, you can use "/", as on Unix systems (works for all platforms). This is recommended.

Example

TestFileClass.java

Writing to Text Files in Java Using PrintWriter

PrintWriter

Allows programmer to use the same print, println, and printf commands used with System.out, but sends data to a file rather than the standard output.

Opening a File for Writing

PrintWriter outputFile = new PrintWriter(new File("FileInCurrentDir.txt"));

PrintWriter outputFile = new PrintWriter("FileInCurrentDir.txt");

Important

- It is important to explicitly close() a file, to make sure that the data written is properly saved, and to release resources needed for the file.
- e.g. outputFile.close();

Example

WriteData.java

Reading Text Files in Java Using Scanner

Scanner

Reads input from a text file one *token* at a time. A *token* is a series of adjacent non-whitespace characters (newlines, spaces, tabs separate tokens)

Opening a File for Reading

Scanner stdIn = new Scanner(Standard.in);

- Standard.in is the standard input, a file defined for all programs running on a machine. Usually the standard input contains captured keyboard strokes.

Scanner input = new Scanner(new File("FileName.txt"));

Example

ReadData.java

Input and Output Streams

(Byte) Streams

– Store data read from or to be written to system resources (e.g. physical file, monitor, keyboard)

– Represented in memory as a sequence of “raw” bytes

– No encoding/organization assumed

– In Java: represented and utilized through objects (most found in java.io package)

Default Streams for Programs in Most OS’s

Standard input (System.in) – usually from keyboard.

Standard output (System.out) – usually sent to terminal.

Standard error (System.err) – usually sent to terminal also

Redirecting Standard Output, Error, and Input (bash shell)

Redirecting Standard Output

java Streams > output

Redirecting Standard Input

java Streams 2> error

Redirecting Both Together or Separately

java Streams &> output (one file)

java Streams > output 2> error

Redirecting Standard Input

java Streams < textfile