Operating Systems and File Output

Colorado State University

Department of Computer Science

Slides Originally Created by Albert Lionelle (Albert.Lionelle@colostate.edu)
updated by Marcia Moraes (marcia.moraes@colostate.edu)

Announcements

https://twitter.com/BradSugars/status/897850608312561664

TODO Reminders:

Readings are due **before** lecture

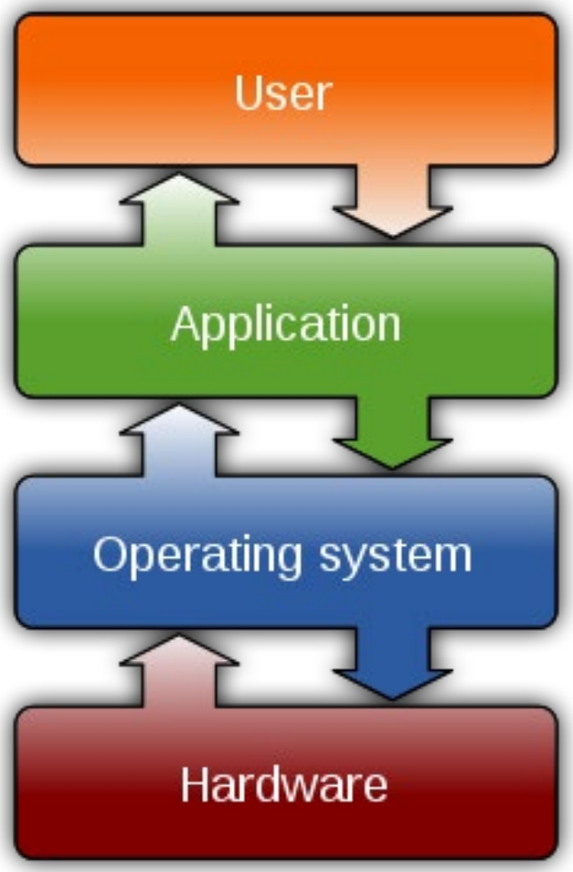
- Reading 17 (zybooks) – you should have already done that ☺
- Lab 11
- Reading 18 (zyBooks)
- Lab 12
- Reading 19 (zybooks)

- RPA 9

Keep practicing your RPAs in a spaced and mixed manner ☺

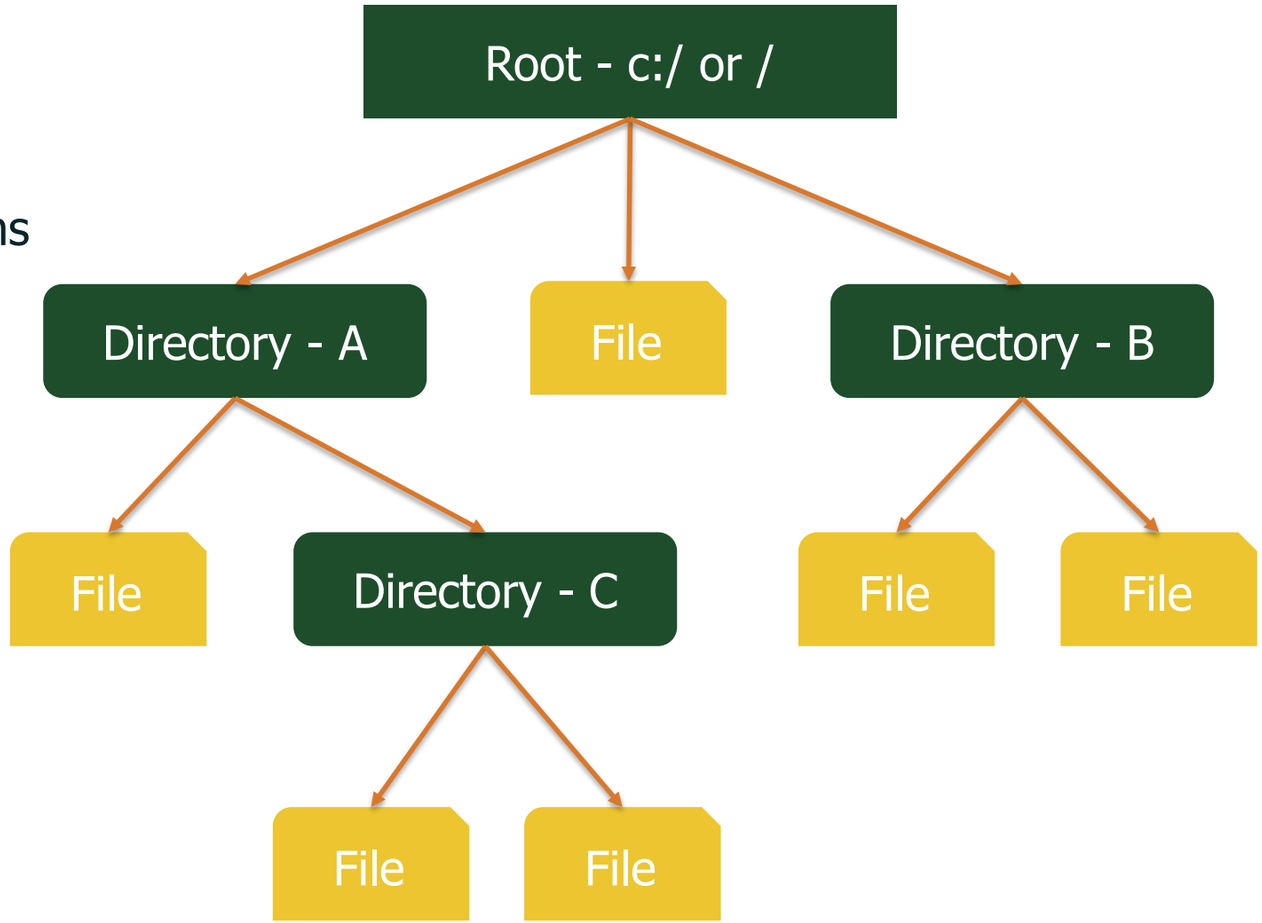
Operating Systems

- You use them daily
  - Most common OS in the world?
  - Android
    - Written in Java w/ Kotlin
- The control
  - Resources
  - Hardware Interaction
  - Devices
  - Running applications, memory, etc
  - Files!



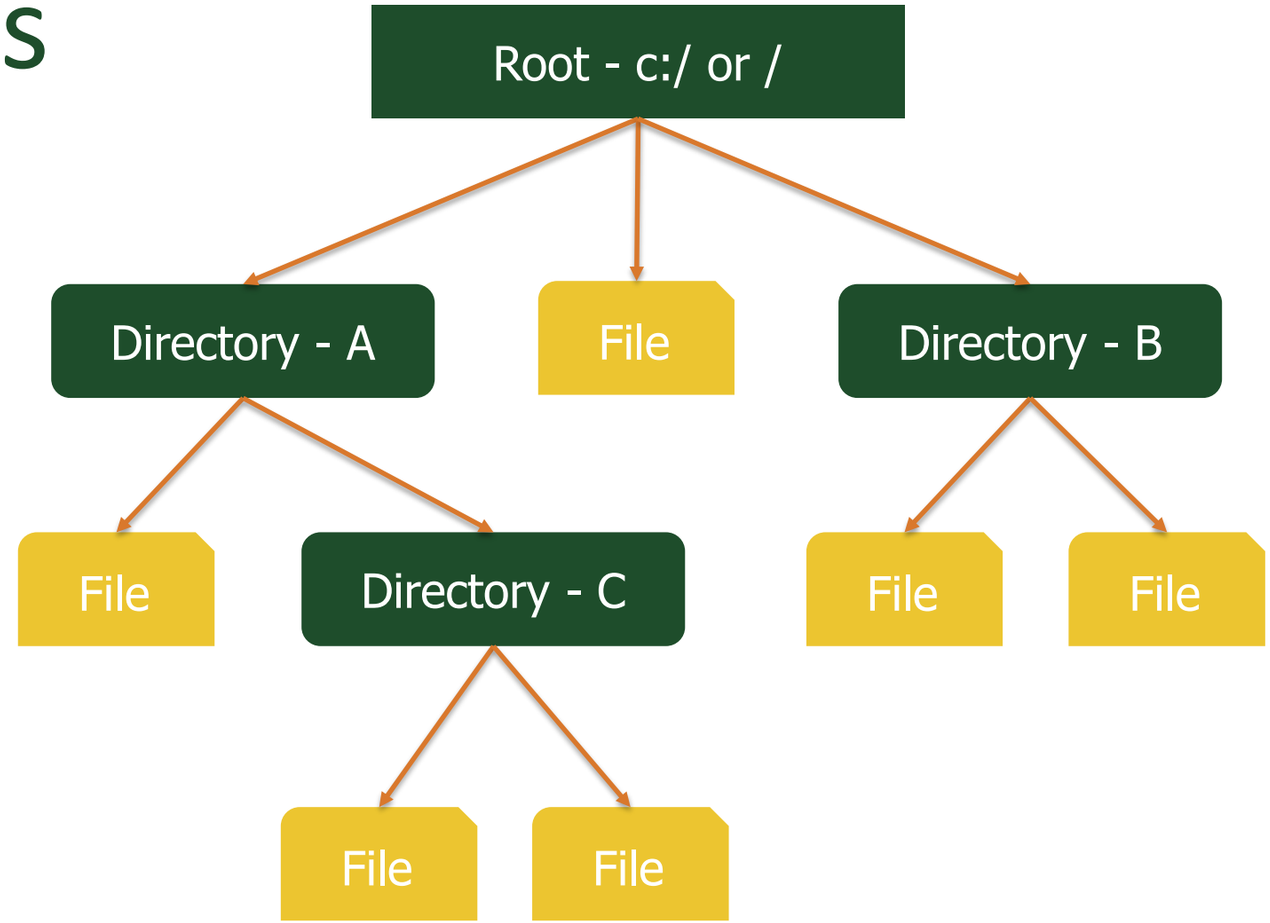
File Systems

- Program that helps manage files and other programs
- Directory Structure
  - Relative
    - Based on current location
  - Absolute
    - Based on Root, the top of the hierarchy
- Key “shortcuts”
  - . (yes dot) - current directory
  - .. (directory above this directory)

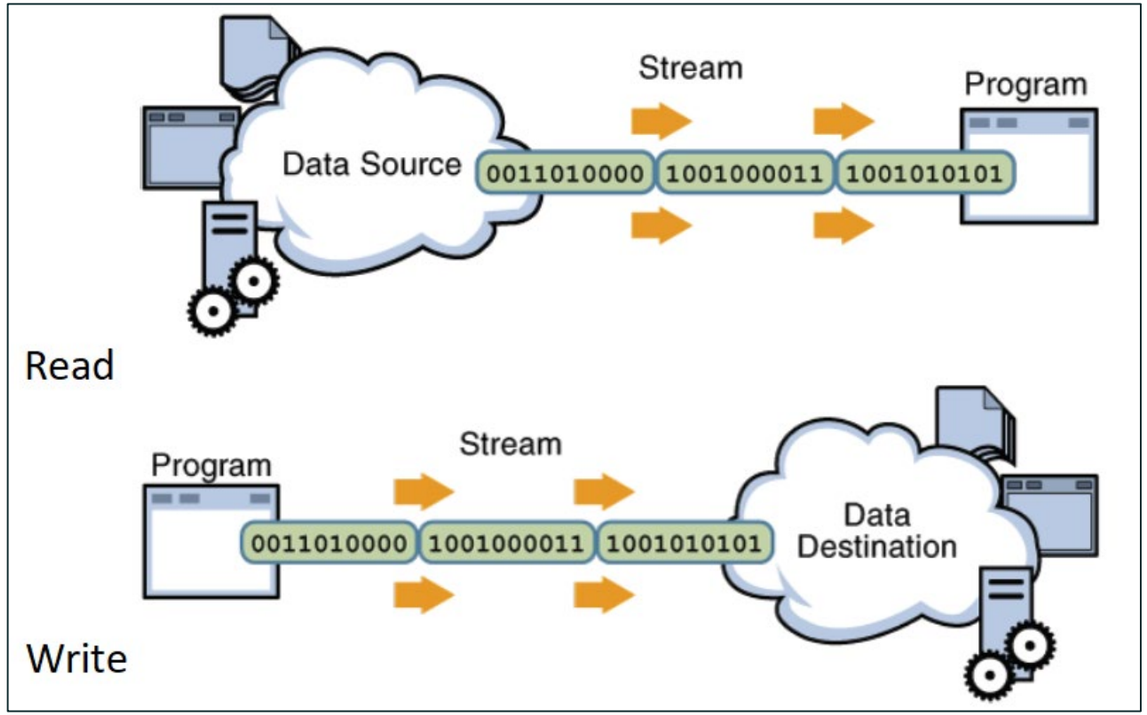


File Systems - Examples

- Windows: drive letter with C:/, D:/ etc
- Linux/MacOS/Unix: just a "/"
- "/Directory - A/file"
- "/Directory - A/Directory - C/file"
- "C:\Directory - B\file"
- "/file"



Files and Streams



To read we used:

```
Scanner in = new Scanner(new File("input.txt"));
```

To write we will use:

```
PrintWriter writer =
        new PrintWriter(new File(“notes.txt”);
```

Or

```
PrintWriter writer =
        new PrintWriter(new FileOutputStream(“notes.txt”);
```

File Object in Java

- Has a number of useful methods when dealing with files and directories

- **File myFile = new File("filename");**
  - Creates or reads a file based on the *path+filename* given
  - Actually connects to the location which is a **'stream of bytes'**

File Object in Java

- File myFile = new File(“output.txt”);
  - Creates a file in the same directory as that you are executing the java file from - so relative to your program
- File myfile = new File(“/Users/lionelle/output.txt”);
  - Creates a file based on the **absolute** path that is Root -> Users -> lionelle (folder) -> output.txt
- File myfile = new File(“../output.txt”);
  - What does this do?

FileOutputStream Object in Java

- Has a number of useful methods when dealing with writing binary data to a file

- FileOutputStream myFile = new FileOutputStream(“output.txt”);
  - Creates a file in the same directory as that you are executing the java file from - so relative to your program
- FileOutputStream myfile = new FileOutputStream(“/Users/lionelle/output.txt”);
  - Creates a file based on the **absolute** path that is Root -> Users -> lionelle (folder) -> output.txt
- FileOutputStream myfile = new FileOutputStream (“../output.txt”);
  - What does this do?

Print Writer

- PrintWriter is an object designed to write text to a File Stream
- What Are Streams?
  - System.out - stream to the console
  - System.in - stream *from* the console
  - System.err - stream to the error log (often console)
  - File is also a Stream
  - FileOutputStream is also a Stream
- PrintWriter uses the same interface as System.out but directs the stream
  - PrintWriter writer = new PrintWriter(new FileOutputStream("notes.txt"));
  - writer.println("#These are my notes");
  - writer.print("This is a note without the extra line");
  - writer.print(" this would append right after the one above");
  - writer.close(); //we need to close the stream after writing in the file!

Example of Writing and Reading

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class SimpleFileWritingFileOutputStream  {
   public static void main(String args[]){
      Scanner in = new Scanner(System.in);
      try {
         printSimpleFile(in);
      }catch(IOException e){
         System.out.println("Error! Could open the file to write!");
      }
      readSimpleFile();
   }
```

Discuss in your tables, how this program works.

What the throws IOException in printSimpleFile method means?

```
   public static void printSimpleFile(Scanner in) throws IOException{
      PrintWriter file = new PrintWriter(new FileOutputStream("simple.txt"));
      System.out.println("Enter a string - 'exit' to stop");
      String line = in.nextLine();
      while(!line.equals("exit")){
         file.println(line);
         System.out.println("Enter another string - 'exit' to stop");
         line = in.nextLine();
      }
      file.close();
   }
   public static void readSimpleFile(){
      System.out.println("Printing what is in simple.txt file");
      try{
         Scanner file = new Scanner(new File("simple.txt"));
         while(file.hasNext()){
            System.out.println(file.nextLine());
         }
      }catch (IOException e){
         System.out.println("Could not read the file!");
      }
   }
}
```

Example of Writing and Reading

Instead of having the try…catch block inside printSimpleFile this block is implemented where the method is called.

throws means that the method is delegating the dealing of the exception to where the method is being called.

Can we apply the same thing to readSimpleFile method?

```
public class SimpleFileWritingFileOutputStream  {
   public static void main(String args[]){
      Scanner in = new Scanner(System.in);
      try {
         printSimpleFile(in);
      }catch(IOException e){
         System.out.println("Error! Could open the file to write!");
      }
      readSimpleFile();
   }
```

```
   public static void printSimpleFile(Scanner in) throws IOException{
      PrintWriter file = new PrintWriter(new FileOutputStream("simple.txt"));
      System.out.println("Enter a string - 'exit' to stop");
      String line = in.nextLine();
      while(!line.equals("exit")){
         file.println(line);
         System.out.println("Enter another string - 'exit' to stop");
         line = in.nextLine();
      }
      file.close();
   }
   public static void readSimpleFile(){
      System.out.println("Printing what is in simple.txt file");
      try{
         Scanner file = new Scanner(new File("simple.txt"));
         while(file.hasNext()){
            System.out.println(file.nextLine());
         }
      }catch (IOException e){
         System.out.println("Could not read the file!");
      }
   }
}
```

Example of Writing and Reading

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;
public class SimpleFileWritingFileOutputStream  {
   public static void main(String args[]){
      Scanner in = new Scanner(System.in);
      try {
         printSimpleFile(in);
         readSimpleFile2();
      }catch(IOException e){
         System.out.println(e.getMessage());
      }
   }
```

Both methods here

Get the specific message that generated the exception

```
public static void printSimpleFile(Scanner in) throws IOException{
      PrintWriter file = new PrintWriter(new FileOutputStream("simple.txt"));
      System.out.println("Enter a string - 'exit' to stop");
      String line = in.nextLine();
      while(!line.equals("exit")){
         file.println(line);
         System.out.println("Enter another string - 'exit' to stop");
         line = in.nextLine();
      }
      file.close();
   }

   public static void readSimpleFile2() throws IOException{
      System.out.println("Printing what is in simple.txt file");
      Scanner file = new Scanner(new File("simple.txt"));
      while(file.hasNext()){
         System.out.println(file.nextLine());
      }
   }
}
```

throws

https://docs.oracle.com/javase/8/docs/api/java/io/IOException.html?is-external=true

In Class Practice

- Worksheet
  - Files available: https://github.com/CSU-CompSci-CS163-4/Handouts/tree/main/ClassExamples/09FileWrite
- In Class Activity on zybooks