Operating Systems – Exercise 1

System Calls, Basic I/O

General Guidelines:

- Submission deadline is **Wednesday, March 18th, 23:55 Moodle server time**
- Submit your answers in the course website only as PDF and Java files, packed into a single ZIP file.
- Place your name and ID at the top of every source file, as well as in the PDF with the answers.
- No late submission will be accepted!
- Do not submit handwritten work!
- You should work on your exercise by yourself. Misconducts will be punished harshly.
- Please give concise answers, but make sure to explain any answer.
- Document your code.

Part 1 (35 points)

In this question we will examine the running time of the same program with different parameters. We will implement a Java application that copies a file. When calling the program, a buffer size (in bytes) should be given. This is the size of the buffer you should use each time you read from the source file.

A. **Read the following chapters:** from the Java Basic I/O Streams Tutorial
http://download.oracle.com/javase/tutorial/essential/io/streams.html
   - I/O Streams
   - Byte Streams (next page from I/O Streams)
   - Character Streams (next page from Byte Streams)

   These are very short, entry-level tutorials

A. Write the application TimeTest.java:
The application should receive the source file path, target file path and buffer size as command line arguments. By default, the application does not overwrite an existing file. This can be forced by adding the /force argument.

Arguments:
Force flag: will be '/verbose', and its purpose is for telling us to overwrite the existing target file.
Source file path: the original file we want to copy. (Example: c:\os\ex1\book.txt)
Target file path: the location where we want the copy to be created, should be a full path name. (Example: c:\os\ex1\book_copy.txt)
Buffer size: a number, the size (in bytes) of the buffer to use when reading the file. (Example: 1024)

Some examples of running it with and without the verbose flag:

```
java TimeTest c:\file.txt c:\copy.txt 1024
java TimeTest /force c:\file.txt c:\copy.txt 2048
```

Parameter order matters!
Usage: java TimeTest [/force] source_file target_file buffer_size

Output:

The output of the program should print out the time it took to copy the file, for example:

```
File c:\file.txt was copied to c:\copy.txt
Total time: 123ms
```

Your code:

You are expected to write tidy, documented, short code. You will need to start by parsing the variables, and then to copy the file as required or to show an error message. You will also need to measure the time it will take to copy the file, use `System.currentTimeMillis()` for that.

Your code structure should look like:

```
// Messages Constants
private static final String USAGE = "Usage: …";

public static void main(String[] args) {

    // Check arguments

    // copy file
    long startTime = System.currentTimeMillis();
    // copy logic
    long endTime = System.currentTimeMillis();

}

/**
 * Copies a file to a specific path, using the specified buffer size.
 *
 * @param srcFileName File to copy
 * @param toFileName Destination file name
 * @param bufferSize Buffer size in bytes
 * @param bOverwrite If file already exists, overwrite it
 * @return true when copy succeeds, false otherwise
 */
public static boolean copyFile(String srcFileName,
        String toFileName,
        int bufferSize,
        boolean bOverwrite) {
```

- ***copyFile*** will return true when the file was successfully copied and false if not
- You should implement it and may add anything you think is missing

Guidelines for the force flag:

If the force flag was present, the copy operation should overwrite the target file, if not, and a file exists, some error message should be displayed and the copy will not overwrite the existing file

Guidelines:

- Use the examples from the tutorial and from recitation. Do not copy the examples as they are, as they do not suit our problem.
- To read data from files, use method `read(char[] c, int off, int len)` from class `FileReader`.
- Make sure to *always* close the streams you are using. (try/catch/finally)
- Read the relevant Java API pages regarding the classes you use. Make sure you understand how to use them.

- You are not allowed to use any external / Java-provided code that copies files, if you are not sure if you are allowed to use something, ask in the forum.
- Your program must handle all exceptions and must never crash.

Your code must be neat, readable, commented and to follow the Java naming conventions:
http://www.oracle.com/technetwork/java/codeconvtoc-136057.html

Part 2: (35 points)

1. We will now examine the performance of our program from part 1.
   Find a ~5MB file on your system, we attached a 5MB file of books taken from http://onlinebooks.library.upenn.edu/ and merged into one big file

   Run your program on this file, with the following buffer sizes:
   128, 256, 512, 1024, 2048

   Then, **draw a graph** (using Excel or so) with a single series. The graph should show the time each run takes (in milliseconds) [Y axis], as a function of the buffer size [X axis].

2. Explain the graph. Why isn't it a straight line, parallel to the X axis?
   reading and writting to the disc are expensive operation since each of them cause a system call as we mention in class is an expensive oper. using the OS. as the buffer size is bigger we can reduce the number of expensive operation and to save time. and thats why the bigger the buffer is, the more time we save.

3. Hypothetically, if we change the TimeTest in part 1 to print out to the screen a '*' character each time the buffer is 'reloaded', will the running time change significantly?
   - **IMPORTANT:** do not make this change. Submission with this change will result in a 30 points penalty!

   yes, printing to the screen means we need to invoke more system calls (50% more than before, for each time we read from the file). and as mention in class, each system call cost more time spending.

Part 3: (30 points)

For each of the statements below, answer whether it is **true** or **false** and explain concisely:

1. A simple program which prints "Hello World!" to the screen does not need to use any system calls.
   True / False
   false. in order to print to the screen (screen resource) we need to use the os by a system call

2. Pressing a key on the keyboard will cause a system call.
   True / False
   false, it will cause an interrupt.

3. Interrupts are signals sent from the CPU to external devices.
   True / False
   false, interrupts reaches processes as well
   (true if it is only address hardware interrupt and not software interrupts)

4. Applications such as a web browser (e.g. Chrome/Explorer) are running in user mode and therefore are not allowed to invoke system calls.
   True / False
   false! process in user mode allowed to invoke system calls! they use the calls in order to deal with the hardware (through the OS)

5. Every program that is a part of the operating system (i.e. installed with it), like the command interpreter (the shell) and the web browser, runs in kernel mode. Programs that the user installs, like Office, run in user mode.
   True / False
   false. for example, the browser is used in user mode.

6. The operating system may disable all interrupts, thus blocking all external devices from accessing the CPU.
   True / False
   true, in some cases the OS can define to preform a sequence of instruction without interrupting by anything. it can happen while processing important process that must be finished and not be effected by anything else.

7. If a user wants to improve the throughput of his program, he should run it on a virtual machine (VM) and therefore it will run faster.
   True / False
   false. the VM is using the original resources, when creating the VM in the user mode it simulate a new level of hardware and OS, but they actually using the real hardware of the computer, but as further they are from the real resources,a system call will take more time than a call from the regular user mode.it also get less time from the CPU.

8. If a programmer wants to write an application that plays music from a CD-ROM, the program cannot be run it in user mode, as it must run in kernel mode to access the CD-ROM.
   True / False
   false. the program can run in user mode, and in order to reach to the CD ROM it can send a system call, and by the help of the OS use the CD ROM device.

9. Usually, if a program uses more system calls it will run faster.
   True / False
   false. a system call create a context switch which cause more time taken so as more system call we use, the program will cost more time.

10. External devices may access the operating system by using System Calls.
    True / False
    false. the added devices are running from the kernal mode, and not from the user mode which only from there (the user mode) can send system calls.