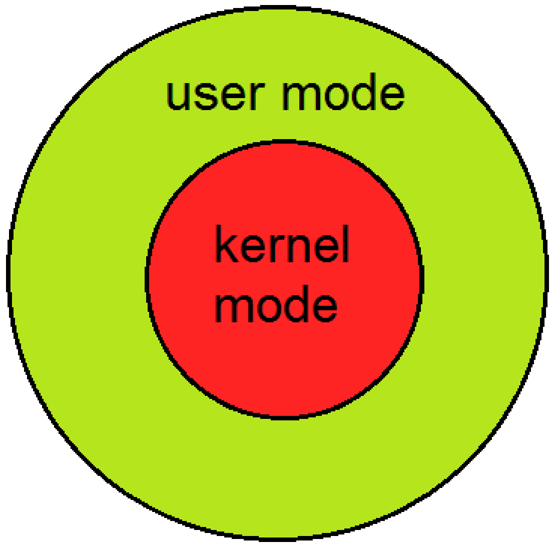
System Call and Zero copy

Sirak Kaewjamnong

517-312 Operating Systems

User mode and kernel mode



https://www.studytonight.com/operating-system/system-calls

User mode and kernel mode

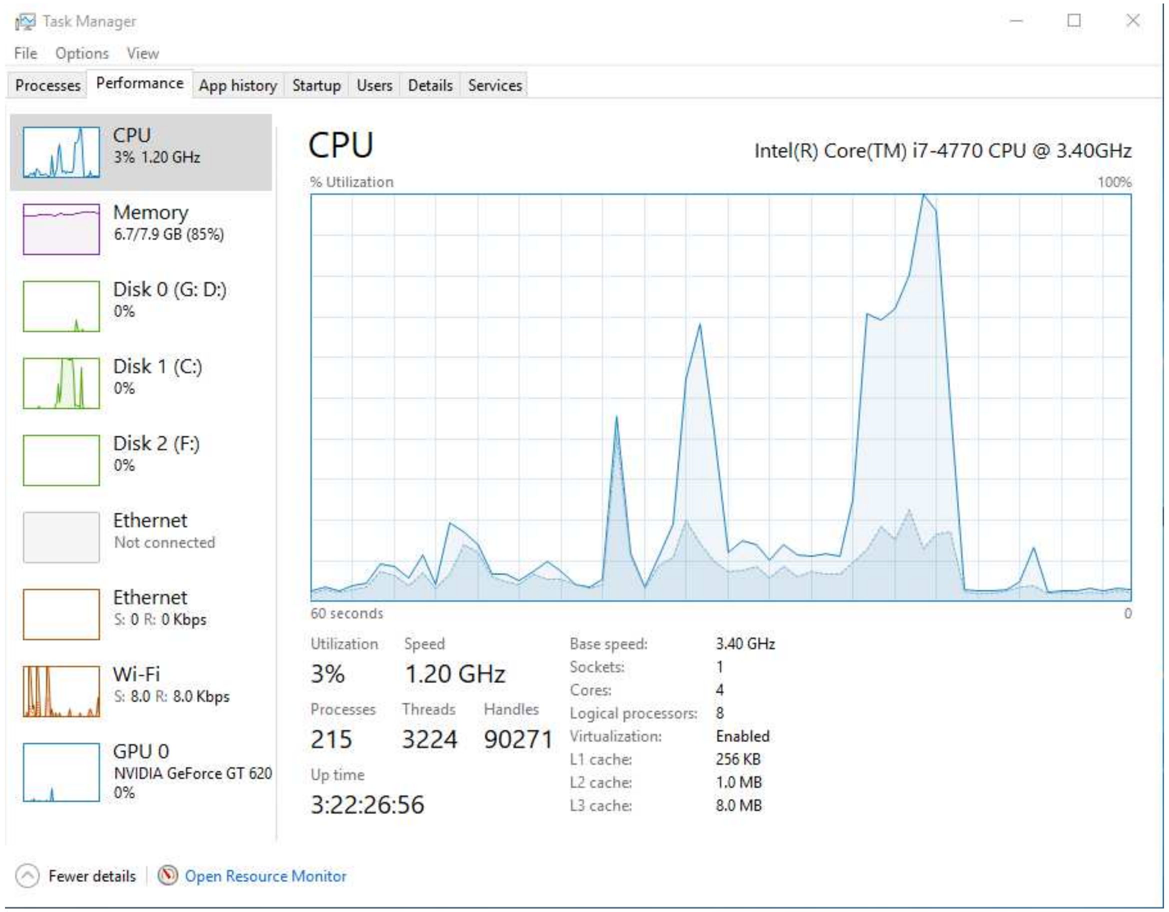
Kernel Mode

- When CPU is in **kernel mode**, the code being executed can access any memory address and any hardware resource
- Hence kernel mode is a very privileged and powerful mode
- If a program crashes in kernel mode, the entire system will be halted

User Mode

- When CPU is in **user mode**, the programs don't have direct access to memory and hardware resources
- In user mode, if any program crashes, only that particular program is halted
- That means the system will be in a safe state even if a program in user mode crashes
- Hence, most programs in an OS run in user mode

Task Manager with kernel times displayed



Utilization	Speed	
3%	1.20 GHz	
Processes	Threads	Handles
215	3224	90271
Up time		
3:22:26:56		

Base speed:	3.40 GHz
Sockets:	1
Cores:	4
Logical processors:	8
Virtualization:	Enabled
L1 cache:	256 KB
L2 cache:	1.0 MB
L3 cache:	8.0 MB

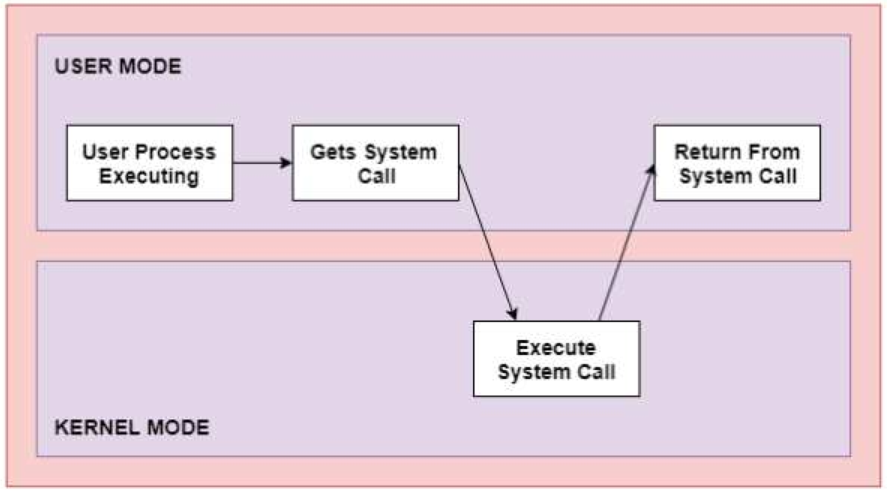
Accessing resources

- When a program in user mode requires access to RAM or a hardware resource, it must ask the kernel to provide access to that resource
- This is done via something called a system call
- When a program makes a system call, the mode is switched from user mode to kernel mode. This is called a **context switch**
- Then the kernel provides the resource which the program requested. After that, another context switch happens which results in change of mode from kernel mode back to user mode
- In a typical UNIX system, there are around 300 system calls

What is system call

- A system call is the programmatic way in which a computer program requests a service from the kernel of the OS
- A system call is a way for programs to interact with the OS and System calls are the only entry points into the kernel system
- System call provides the services of the operating system to the user programs via Application Program Interface(API)
    - It provides an interface between a process and operating system to allow user-level processes to request services of the operating system
    - All programs needing resources must use system calls
- In most systems, system calls can only be made from userspace processes

Calling a system call



https://www.tutorialspoint.com/what-are-system-calls-in-operating-system

Services Provided by System Calls

- Process control: end, abort, create, terminate, allocate and free memory
- File management: create, open, close, delete, read file etc.
- Device management
- Information maintenance
- Communication
- Protection

Examples of Windows and Unix System Calls

	Windows	Unix

Process Control	CreateProcess() ExitProcess()
	WaitForSingleObject()
		fork()
		exit()
		wait()

File Manipulation	reateFile()
	ReadFile()
	WriteFile()
	CloseHandle()
		open()
		read()
		write()
		close()

Device Manipulation	SetConsoleMode()
	ReadConsole() WriteConsole()
		ioctl()
		read()
		write()

Examples of Windows and Unix System Calls

	Windows	Unix

Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid()
		alarm()
		sleep()

Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe()
		shmget()
		mmap()

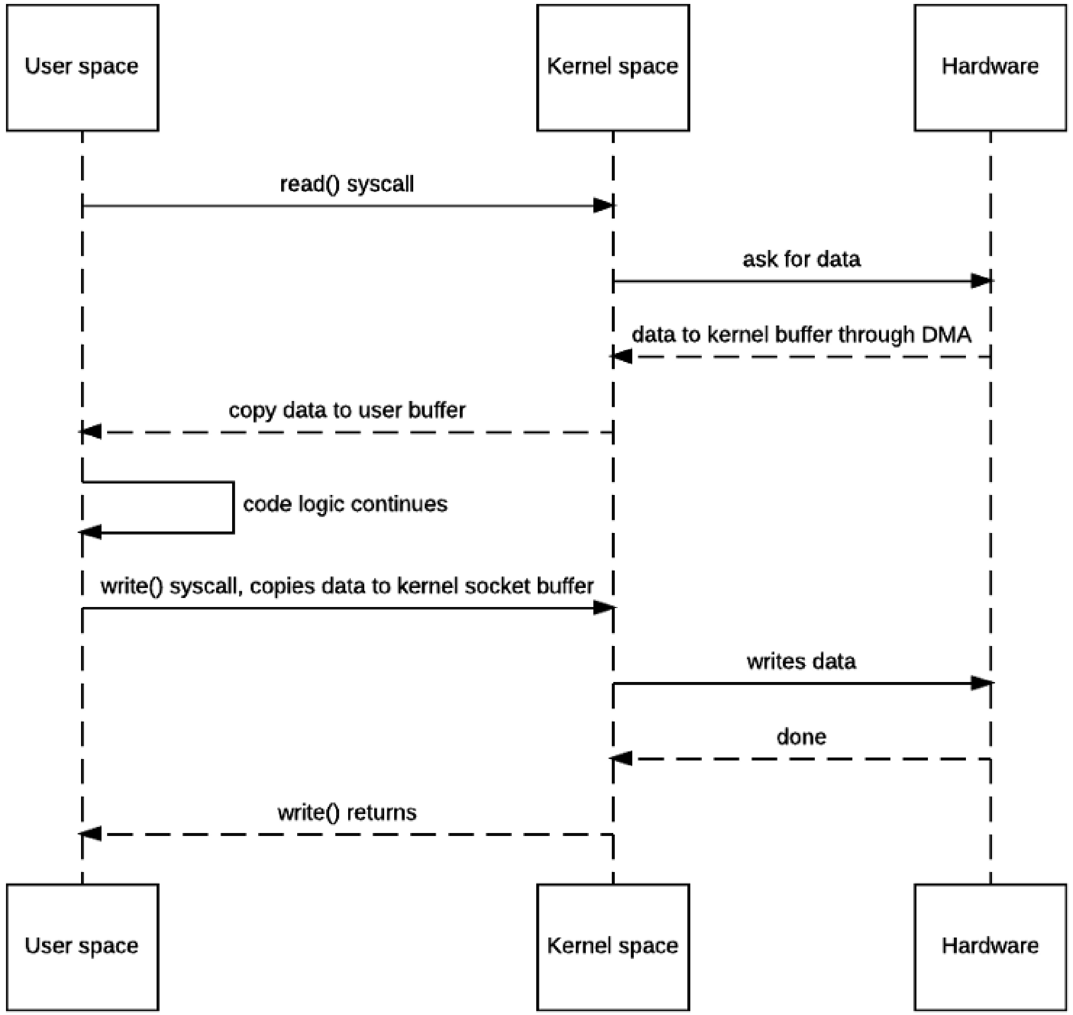
Protection	SetFileSecurity()
	InitlializeSecurityDescriptor()
	SetSecurityDescriptorGroup()
		chmod()
		umask()
		chown()

Problem with user/kernel context switch

When reading and writing a file in typical java program

- JVM sends read() system call
- OS context switches to kernel mode and reads data into the input socket buffer
- OS kernel then copies data into user buffer, and context switches back to user mode. read() returns
- JVM processes code logic and sends write() system call
- OS context switches to kernel mode and copies data from user buffer to output socket buffer
- OS returns to user mode and logic in JVM continues
- There are 4 context switches and 2 unnecessary copies

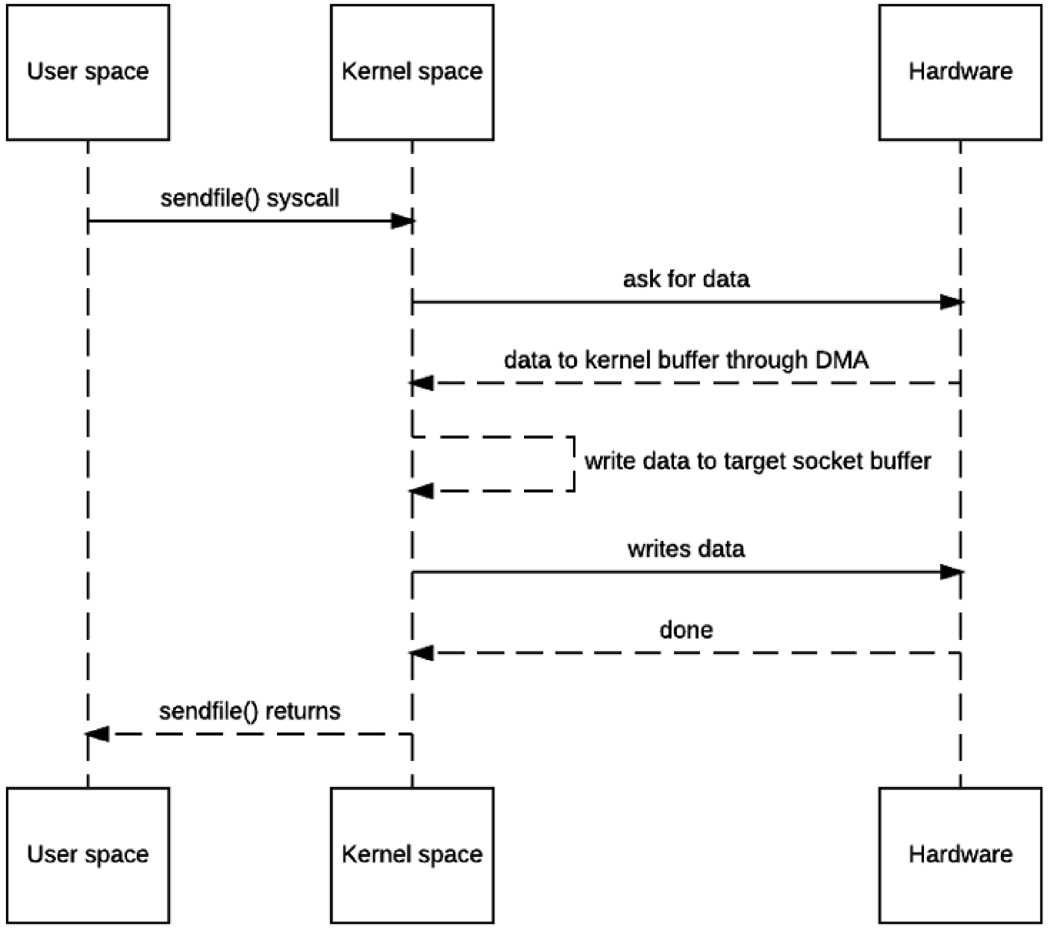
Reading/writing a file



Zero copy

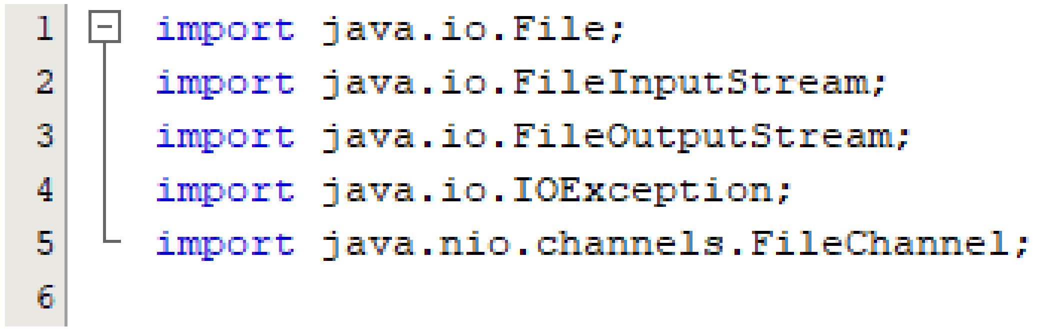
- Copy from/to user space memory is unnecessary
- Zero copy can thus be used here to save the 2 extra copies
- The actual implementation doesn’t really have a standard and is up to the OS how to achieve that
- With zero copy, no data copied from kernel space to user space
- Zero copy is possible when hardware (disk drive, network card, graphic card, sound card) supports **DMA** (Direct Memory Access)
- Java nio can help

Zero copy


User space
Kernel space
Hardware
sendfile() syscall
ask for data
data to kernel buffer through DMA
write data to target socket buffer
writes data
done
sendfile() returns
User space
Kernel space
Hardware

Simple file copy java code

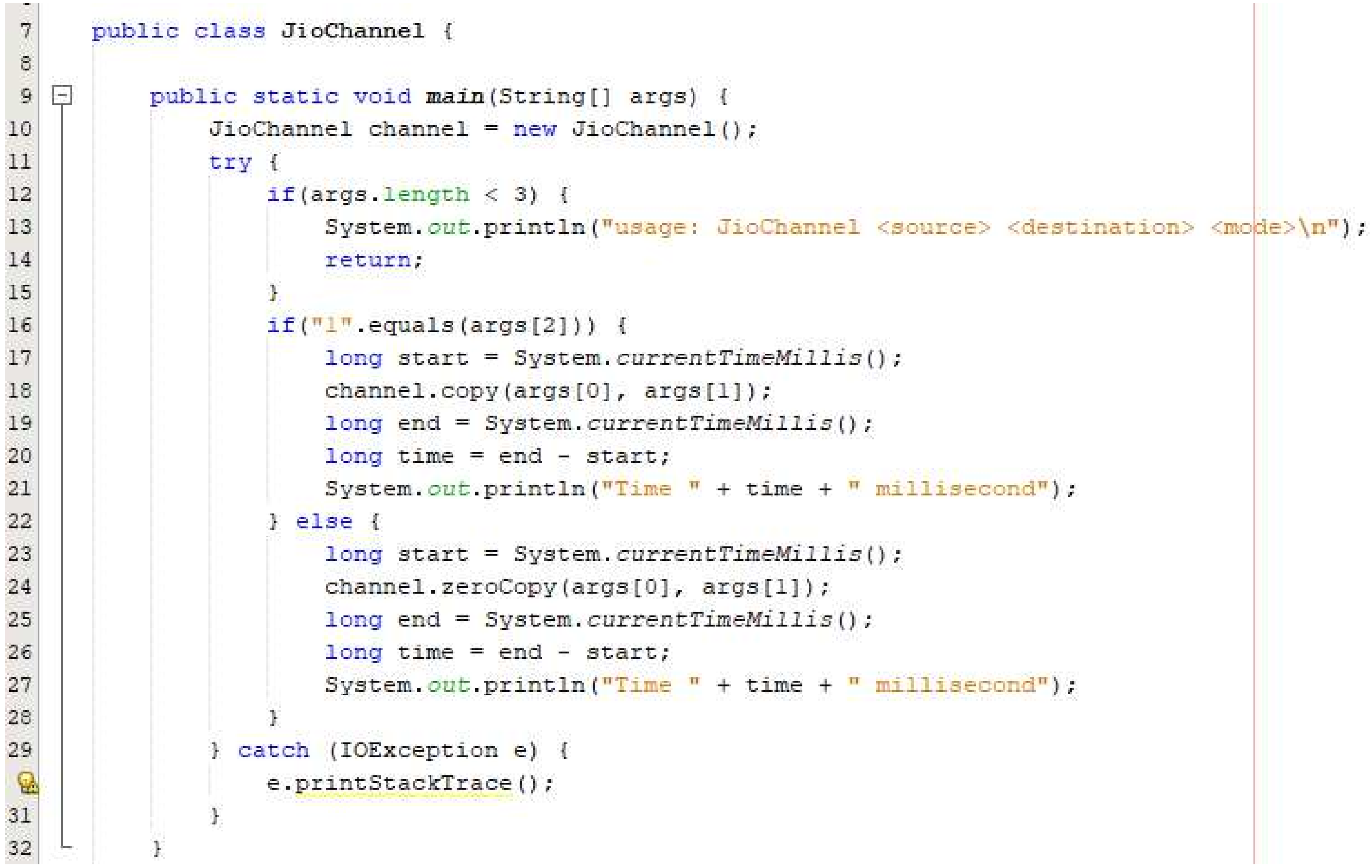
Header



```java
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.channels.FileChannel;

```

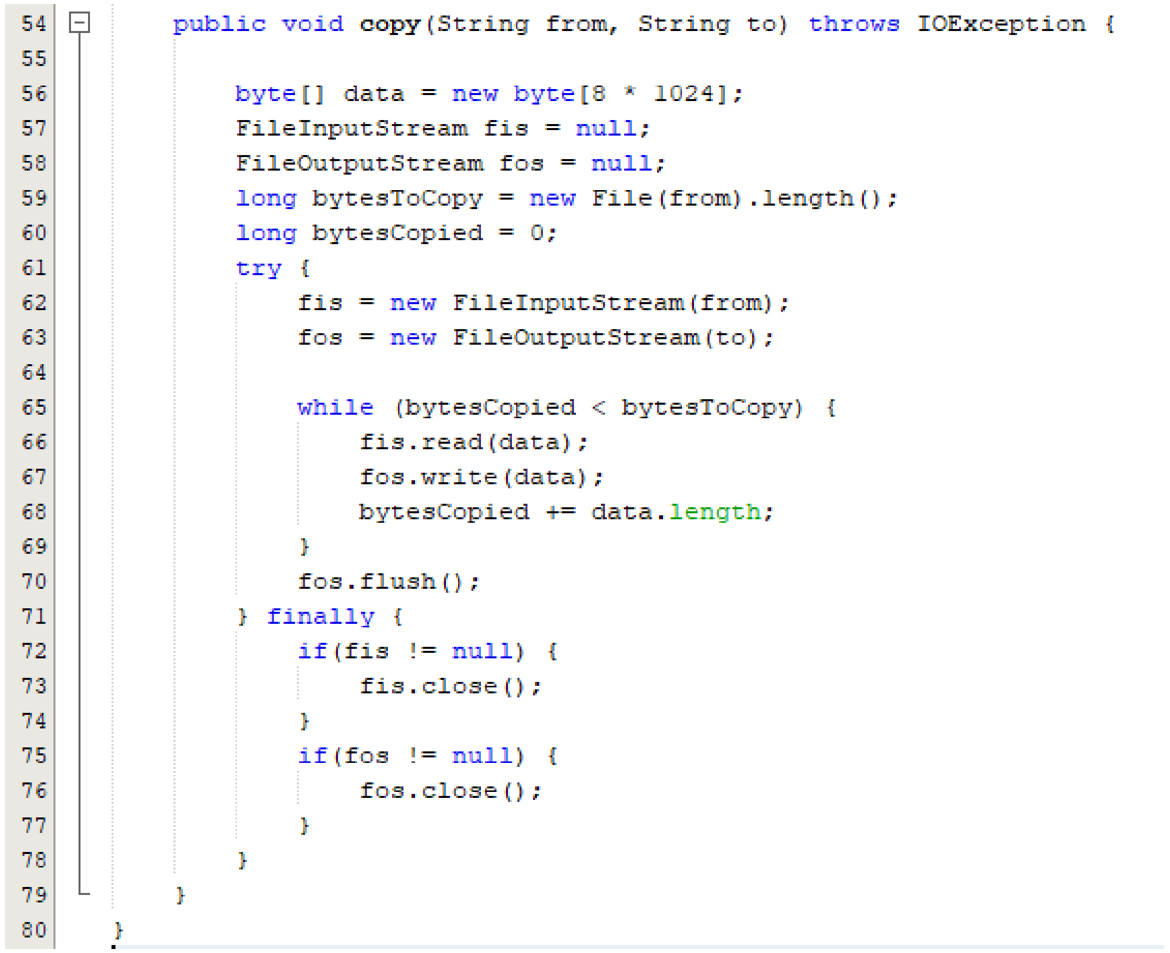
Main class



```java
public class JioChannel {

    public static void main(String[] args) {
        JioChannel channel = new JioChannel();
        try {
            if(args.length < 3) {
                System.out.println("usage: JioChannel <source> <destination> <mode>\n");
                return;
            }
            if("1".equals(args[2])) {
                long start = System.currentTimeMillis();
                channel.copy(args[0], args[1]);
                long end = System.currentTimeMillis();
                long time = end - start;
                System.out.println("Time " + time + " millisecond");
            } else {
                long start = System.currentTimeMillis();
                channel.zeroCopy(args[0], args[1]);
                long end = System.currentTimeMillis();
                long time = end - start;
                System.out.println("Time " + time + " millisecond");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Traditional copy method

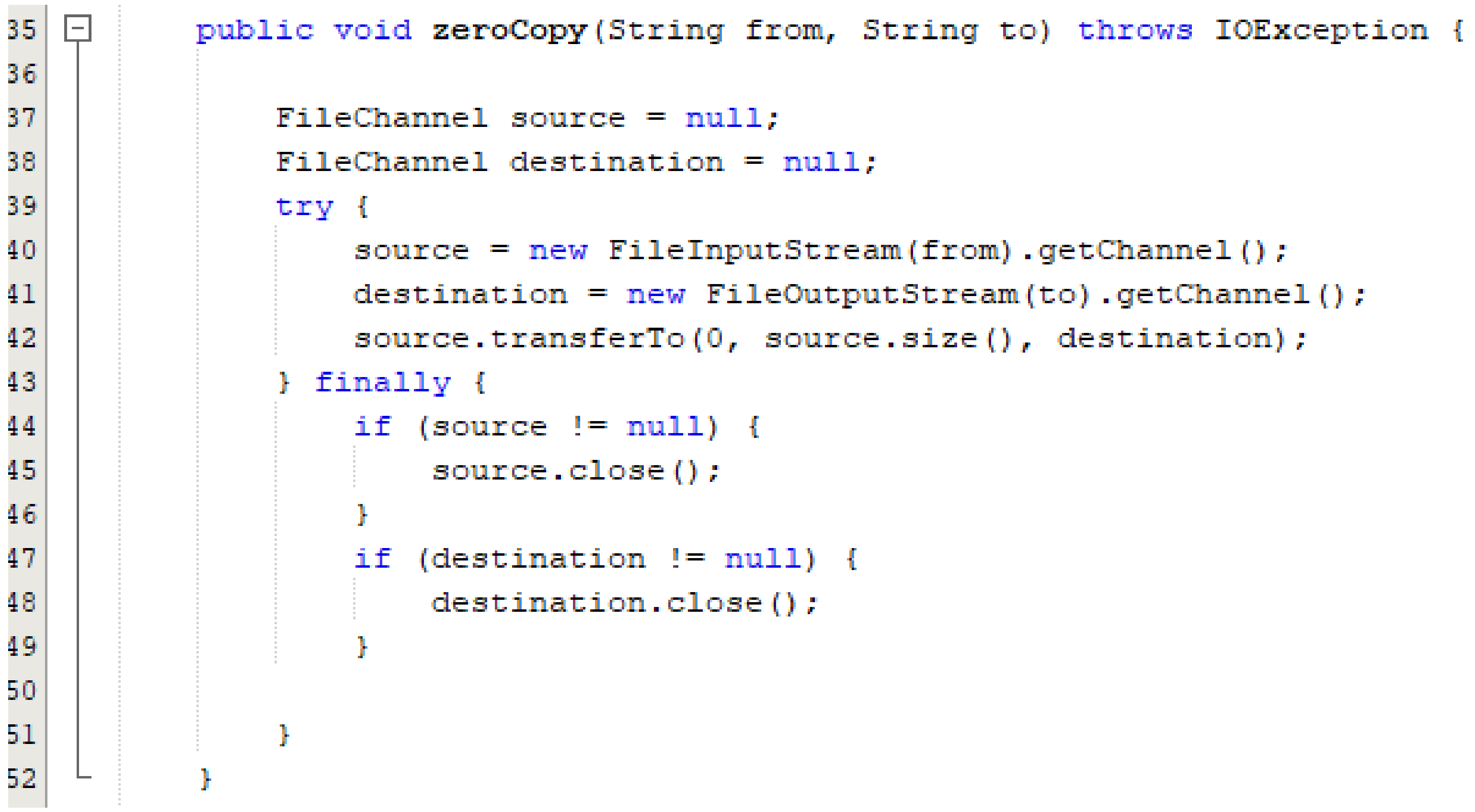


```
    public void copy(String from, String to) throws IOException {

        byte[] data = new byte[8 * 1024];
        FileInputStream fis = null;
        FileOutputStream fos = null;
        long bytesToCopy = new File(from).length();
        long bytesCopied = 0;
        try {
            fis = new FileInputStream(from);
            fos = new FileOutputStream(to);

            while (bytesCopied < bytesToCopy) {
                fis.read(data);
                fos.write(data);
                bytesCopied += data.length;
            }
            fos.flush();
        } finally {
            if(fis != null) {
                fis.close();
            }
            if(fos != null) {
                fos.close();
            }
        }
    }
}
```

Zero copy method



```
public void zeroCopy(String from, String to) throws IOException {

    FileChannel source = null;
    FileChannel destination = null;
    try {
        source = new FileInputStream(from).getChannel();
        destination = new FileOutputStream(to).getChannel();
        source.transferTo(0, source.size(), destination);
    } finally {
        if (source != null) {
            source.close();
        }
        if (destination != null) {
            destination.close();
        }

    }
}
```

Running the program

- Use zero copy
  - java JioChannel file1 file2 1
- Use traditional copy
  - java JioChannel file1 file2 2

Homework

- Install Oracle JDK 8 on your Linux VM
- Install Netbean or Eclipse on that system
- Write a multithread file server program using zero copy and client program must use zero copy as well
- Both program run on Linux in the different VMs
- Present your code in group before 15.00, 30 October 2020
- This homework worth 15%

What you need to study yourselves for this homework

- Package java.nio is an alternative I/O API for Java
- The standare I/O APIs use byte streams and character streams while nio works with channels and buffers
  - FileChannel
  - DatagramChannel
  - SocketChannel
  - ServerSocketChannel
- Java nio enables non-blocking I/O
- http://tutorials.jenkov.com/java-nio/index.html provided very useful information (but in English)

Sources and References

- https://www.studytonight.com/operating-system/system-calls
- http://faculty.salina.k-state.edu/tim/ossg/Introduction/sys_calls.html
- https://www.ibm.com/developerworks/library/j-zerocopy/index.html
- https://medium.com/@xunnan.xu/its-all-about-buffers-zero-copy-mmap-and-java-nio-50f2a1bfc05c
- https://github.com/arturmkrtchyan/zero-copy