Introduction to Operating Systems

G53OPS/G52OSC

Geert De Maere
(Jason Atkin – OSC)
Geert.DeMaere@Nottingham.ac.uk

University Of Nottingham
United Kingdom

2015

Goals

What and How?

- Goals:
  - Introduction to the **fundamental concepts**, **key principles** and **internals** of operating systems
  - Better understand how **application programs interact/rely with the operating system**
  - We will study **old** and **new concepts**
- How:
  - Through **lectures** (two hours per week for approximately 9-10 weeks)
  - Through **labs/coursework** for G52OSC

Goals

G52OSC Only: What and How ?

- The **labs** will teach you:
  - How to use various **concurrency primitives**
  - The basics of **Windows** and **Linux programming** in C and Java
  - Help you with (shared) **memory allocation**
  - Practical experience/insights int fundamentel **OS concepts**
- **Lectures** will **introduce** these concepts

Assessment

What Should You Expect

- A **90 minute exam** that focusses on:
  - **Knowledge**
  - **Understanding**
  - **Application**
- **Sample questions** from previous years are available on moodle and will be included in the lectures

Assessment

What Should You Expect

- A **90 minute exam** that focusses on:
  - **Knowledge**
  - **Understanding**
  - **Application**
- **Sample questions** from previous years are available on moodle and will be included in the lectures
- The **G53OPS exam** will be 3 out of 5 questions, with 100% exam for assessment
- The **G52OSC exam** will be 3 out of 5 questions, with 75% of the assessment on the exam

Assessment

G52OSC Only: What Should You Expect

- The **coursework** is worth **25%**
  - Remember that this is a **20 credit module**
  - Equivalent to **50%** of a **10 credit module**
- **Coursework** uses the **concepts from the labs**
  - **15%** using **concurrency** and operating system API
  - **10%** simulating **operating system functionality**

Assessment

What Should You Expect

An E-mail Received Evening Before the Exam

Hi Geert,

*There is a **lot of information** covered during the course, hence **making revision very challenging**.*

*Do you have any guidance as to how to break the course down in to a **list of "main topics"** that are **essential to know in detail**?*

*Also, how will these **topics be split in to the 5 optional questions** in the exam?*

Thanks

. . .

Assessment

What Should You Expect

Response

Dear . . . ,

*Unfortunately, I am **unable to provide any information** other than what was said during the lectures: the exam will try (as much as possible) to **assess all aspects covered in the course**. This is the only way in which a **fair exam** could be put together, since different students will find different topics easier/more difficult.*

*This is probably not the answer that you were hoping for, but if I would give **you a more detailed answer**, it **may be unfair to other students** which were not provided with this information.*

Best wishes,

Geert De Maere

About Me
My Background

- Contact details:
  - Name: Geert De Maere
  - Office: C76 (by appointment)
  - E-mail: Geert.DeMaere@Nottingham.ac.uk
- About me:
  - Graduated in 2000, Bsc, Msc in Engineering
  - Completed my PhD in CS in 2010 (Operational Research)
  - Member of ASAP
  - Specific interest in **airline scheduling** and **airport operations**
  - I work together with **Institute for Aerospace Technology** (IAT) and **Nottingham Geospatial Institute** (NGI)

About Me
My Background

- How does my research link in with operating systems?
  - I work on scheduling and optimisation
  - Exploit computer **architecture/design** and **common principles** in operating system design to:
    - Implement **sensible** parallelisations of algorithms
    - Speed up algorithms (caching, manipulate registers)
  - **Exploit similar principles** in my daily work (e.g. caching, parallelisation)
  - ...
- One of the learning outcomes on Saturn states: *"the ability to think independently while giving due weight to the arguments of others"*

Content

Subjects We Will Discuss

Subject	#Lectures
Introduction to operating systems/computer design	1-2
Processes, process scheduling, threading, ...	3-4
Memory management, swapping, virtual memory, ...	3-4
File Systems, file structures, management, ...	3-4
Case study: Linux	2
Revision	2

Table: Preliminary course structure

Reading Material

My Favourite Books

- Seminal books:
  - *Tanenbaum, Andrew S. 2014 Modern Operating Systems. 4th ed. Prentice Hall Press, Upper Saddle River, NJ, USA.*
  - Silberschatz, Abraham, Peter Baer Galvin, Greg Gagne. 2008. *Operating System Concepts*. 8th ed. Wiley Publishing.
  - Stallings, William. 2008. *Operating Systems: Internals and Design Principles*. 6th ed. Prentice Hall Press, Upper Saddle River, NJ, USA.
  - Thomas Anderson and Michael Dahlin. 2014 *Operating Systems, Principles & Practice*. 2nd Ed. Recursive Books, Ltd.
- Other sources:
  - Daniel P. Bovet, Marco Cesati *Understanding the Linux Kernel*. 3rd ed. O'Reilly Media, November 2005
  - Course slides will be available on Moodle

Goals for Today

Overview

- “**Defining**” operating systems
- What is **multi-programming**
- **Race conditions**, **deadlocks**, **kernel-user mode**
- Interrupts

Defining Operating Systems

What Can an OS Do For Me?

```
1  import java.io.FileWriter;
2  import java.io.IOException;
3  import java.io.PrintWriter;
4
5  public class Demo1 {
6    public static void main(String[] args) throws IOException {
7      FileWriter fw =
8        new FileWriter(("C:/Program Files (x86)/test.txt"));
9      PrintWriter pw = new PrintWriter(fw);
10     pw.close();
11   }
12 }
```

Defining Operating Systems

What Can an OS Do For Me?

- **Where** is the file physically written on the disk and how is it **retrieved** (file systems), why looks the instruction the same **independent of the device**? (abstraction)
- What if multiple programs **access the same file simultaneously**? (processes, concurrency, ...)
- Why is the **access denied**? (protection)

Defining Operating Systems

What Can an OS Do For Me?

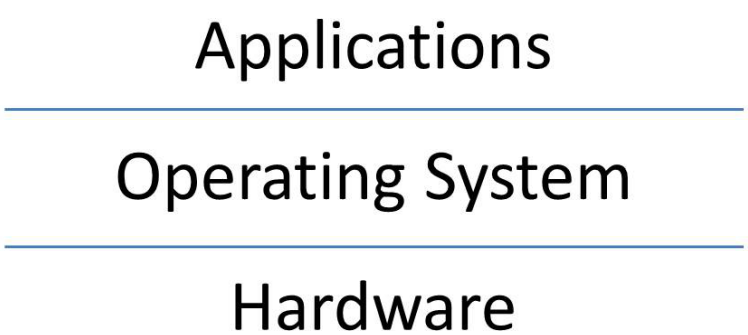
```
1 public class Demo2 {
2   public static void main(String[] args){
3     long[] aLargeArrayOfLargeNumbers
4                   = new long[Integer.MAX_VALUE];
5   }
6 }
```

- **Where** in memory will the array be stored and how is it **protected** from unauthorised access?
- What if the array requires **more memory than physically available**?
- What if an **other process starts running**?

Defining Operating Systems

A Virtual Machine Providing Abstractions

- In the early days, programmers had to **deal directly with the hardware**
  - Real computer **hardware is ugly**
  - Hardware is **extremely difficult** to manipulate/program
- An operating system is a layer of indirection on top of the hardware:
  - It provide **abstractions** for application programs (e.g., file systems)
  - It provides a **cleaner and easier interface to the hardware** and hides the complexity of **"bare metal"**
  - It allows the programmer to be lazy by using **common routines** :-)



Defining Operating Systems

Some Wisdom

David Wheeler (First PhD in Computer Science, 1951)

"All problems in computer science can be solved by another level of indirection"

Why Study Operating Systems?

Motivation

- The programs that we write **rely on the operating system**
- How are the operating system's sevices **services/abstractions** implemented

Defining Operating Systems

A Resource Manager

- Many modern operating systems use **multi-programming** to **improve user experience** and **maximise resource utilisation**
  - Disks are slow: without multi-programming, CPU time is wasted while waiting for I/O requests
    - Imagine a **CPU** running at 3.2 GHz (approx. 3.2×10^9 instructions per second)
    - Imagine a **disk** rotating at 7200 RPM, taking 4.2 ms to rotate half a track
    - I/O is slow, we are **missing out on** $3.2 \times 4.2 \times 10^6$ **instructions**!
- The implementation of **multi-programming** has important **consequences** for **operating system design**

Defining Operating Systems

A Resource Manager

- The operating system must **allocate/share** resources (including CPU, memory, I/O devices) **fairly** and **safely** between **competing processes**:
  - In time, e.g. CPUs and printers
  - In space, e.g., memory and disks
- The execution of **multiple programs** (processes) needs to be **interleaved** with one another. This requires:
  - This requires **context switches** and **process scheduling** $\Rightarrow$ **mutual exclusion**, **deadlock avoidance**, **protection** and **integrity**

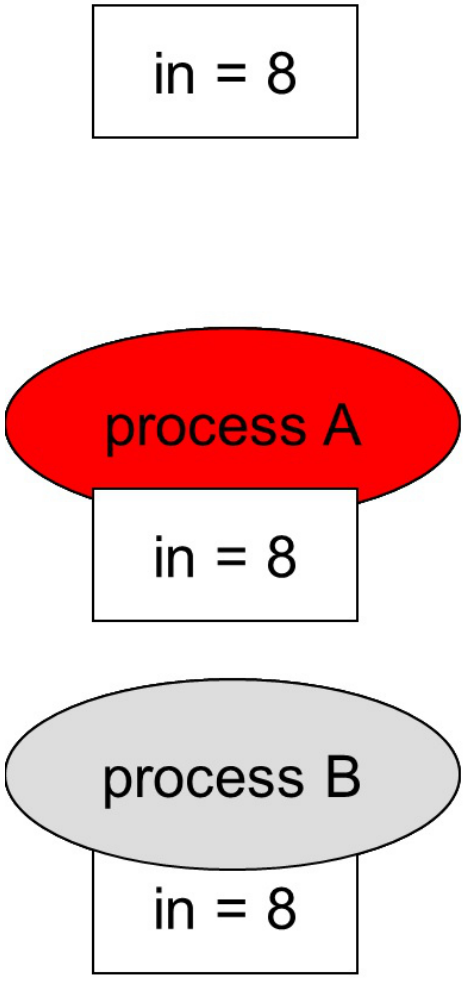
Defining Operating Systems

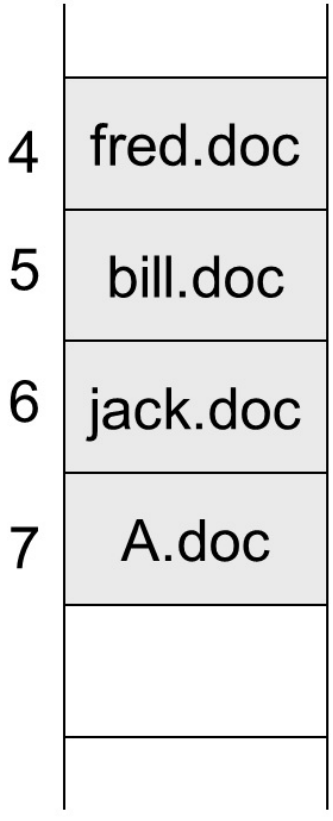
A Resource Manager (Race Conditions)

- Imagine an operating system has a **print spooler**
  - When a process wants to print a file it adds the file name in a **special spool directory** with numbered slots for files
  - The **spooler daemon** periodicaly checks to see whether files are present
  - The OS maintains two variables:
    - **IN**: the number of the next free slot
    - **OUT**: the number of the next file to be printed
- Two processes try to **add a file** to the spooler **at the same time**

Defining Operating Systems

A Resource Manager (Race Conditions)





out = 4

The operating system is now in a consistent state, but *B.doc* will never get printed

This is called a *race condition* - the result will depend on a race between the two processes

Defining Operating Systems

A Resource Manager (Deadlocks)

- Imagine **process A and B** both need **resource** R_1 **and** R_2 for execution
- Process A and B are **interleaved** and request resources R_1 and R_2 in opposite orders
- The processes may end up in **deadlock**, e.g.:

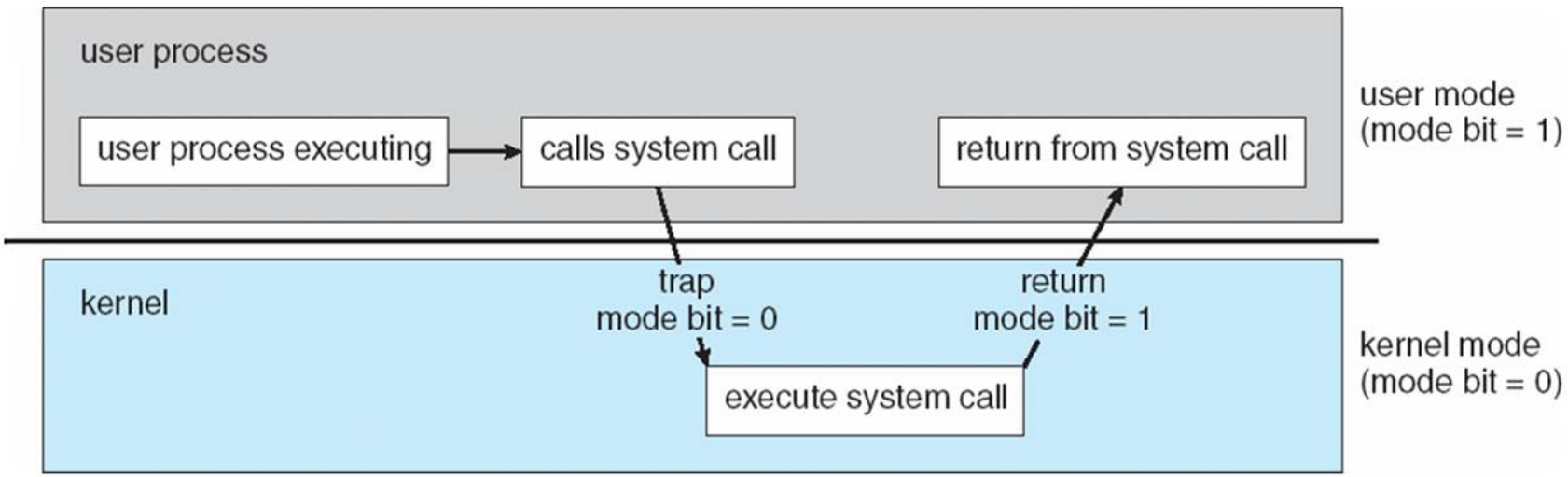
```
PROCESS A:
...
request R1
assign R1
...
...
request R2
...
```

```
PROCESS B:
...
...
...
request R2
assign R2
...
request R1
```

Defining Operating Systems

A Resource Manager (Mode Transitions)

- Modern operating systems have multiple **modes**:
  - The operating system runs in **kernel mode** and has access to **all instructions**
  - Applications run in **user mode** and have access to a **subset of instructions**
- Transitions from user mode to kernel mode happen in a controlled manner (**interrupts**, **exceptions**, **system calls**) and are **mirrored in hardware**



Defining Operating Systems

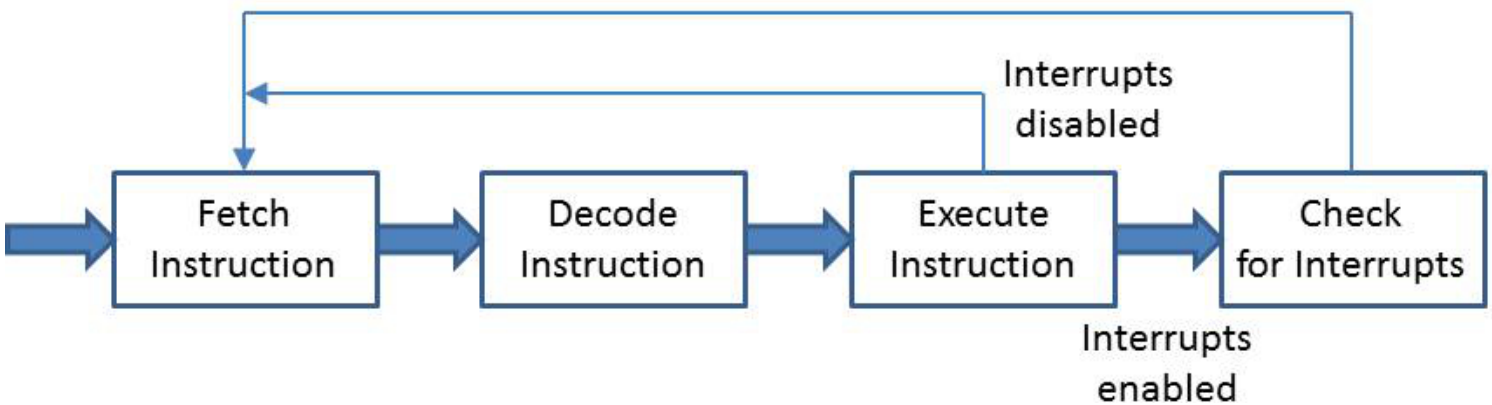
A Resource Manager (Timer Interrupts)

- Interrupts are responsible for **temporarily pausing** a process's normal operation
- Different types of interrupts exist, including:
  - Timer interrupts by **CPU clock**
  - **I/O interrupts** for **I/O completion** or error codes
  - **Software generated**, e.g. errors and exceptions
- **Context switches** can be initiated by **timer interrupts**

Defining Operating Systems

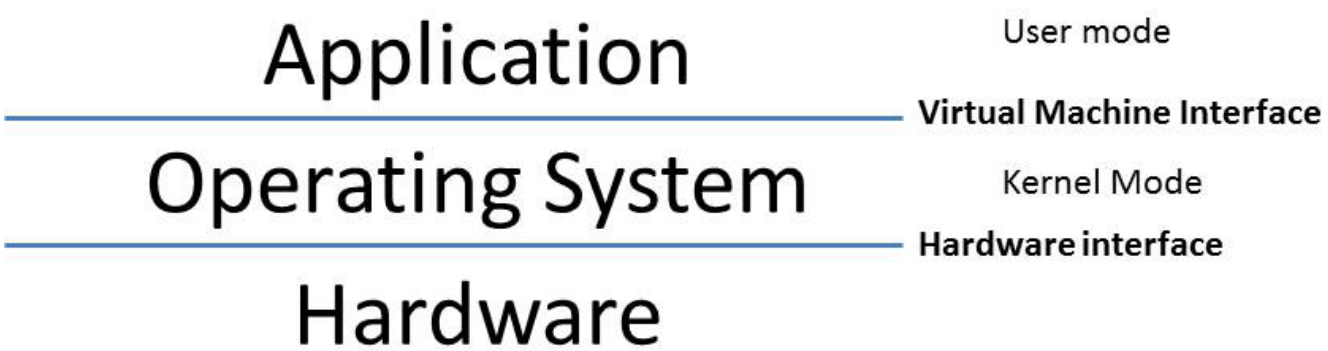
A Resource Manager (Timer Interrupts)

1. Timer generates an interrupt
2. CPU finishes current instruction and tests for interrupt
4. Transfer to interrupt service routine
   - Save current process state (PSW, program counter)
   - Set program counter to interrupt service routine
   - Save registers and other state information
6. Carry out interrupt service routine (scheduler)
7. Restore next process to run



Summary
Take-Home Message

- Some properties:
  - Sits directly on top of the hardware
  - Has access to the **full capabilities of the hardware**
  - Provides **abstractions** for the user/programmer
  - Makes sure that everything is **organised** and runs in an orderly fashion
  - Improve the **hardware interface**
- What is **part** of the **operating system**?
  - Memory management, CPU scheduling, multi-programming, file system, communication, memory management, interrupt handling, GUI, Browser



Next Lecture

Content

- Brief review of **Computer System Architecture** (CSA)
- Introduction to **processes**